BRUKSANVISNING OCH HANDLEDNING

MIKRONIK DIGITALKORT

INNEHÅLLSFÖRTECKNING

0 Inledning
1 Matningsspänningar. Anslutningar
2 In-utkommandon
3 Typexempel på in-utmatning. Några begrepp
4 Programmeringsuppgifter i BASIC
5 Lösningar till programmeringsuppgifter i BASIC
6 Assemblerprogrammering
7 Programmeringsuppgifter i Z80-assembler
8 Lösningar till programmeringsuppgifter i Z80-assembler
9 Beskrivning av Digitalkort-0
10 Beskrivning av Digitalkort-0-16TTL
11 Beskrivning av Digitalkort-32TTL
12 Beskrivning av Digitalkort-T och Digitalkort-T-PUR
13 Beskrivning av Digitalkort-R
14 Datablad för optokopplare och transistorutgångar
15 Kretsschemor
16 Cylinderprogram
17 Program för fräs (INTRONIC AB)
18 Program för stegmotorer
19 Program för tidmätning
20 Program för instrument med BCD-utgångar
21 Program HEXDEC

Appendix A Litteraturreferenser

0 INLEDNING

Denna Bruksanvisning och handledning till Mikronik Digitalkort innehåller först den information som behövs för att använda kortet, se punkt 1-3.

Kortet finns i sex varianter. Dessa är beskrivna i punkt 9-13. För anslutning av yttre enheter med galvanisk isolering mellan dessa och datorn finns tre varianter. Till fem varianter kan TTL-signaler anslutas.

Grunderna för assemblerprogrammering genomgås i punkt 6. Övningsuppgifter med lösningar för programmering i Basic och Z80-assembler finns i punkt 4-5 och 7-8. Handassemblerade program kan läggas in i minnet och köras med programmet HEXDEC.

De sista punkterna innehåller ett antal tillämpningar.

Litteraturreferenserna utgörs främst av böcker som har anknytning till ABC80 och Z80CPU.

1. MATNINGSSPÄNNINGAR. ANSLUTNINGAR

Innehåll sid

1.1 MATNINGSSPÄNNINGAR till Digitalkort-0 1.2
Digitalkort-0-16TTL
Digitalkort-32TTL

1.2 MATNINGSSPÄNNINGAR till Digitalkort-T 1.2
Digitalkort-T-PUR

1.3 MATNINGSSPÄNNINGAR till Digitalkort-R 1.3

1.4 INGÅNGAR FRÅN YTTRE ENHETER. Digitalkort-T 1.3
BYGLAR. Digitalkort-T-PUR
Digitalkort-R

1.5 UTGÅNGAR TILL YTTRE ENHETER. Digitalkort-T 1.5

1.6 UTGÅNGAR TILL YTTRE ENHETER. Digitalkort-T-PUR 1.5

1.7 UTGÅNGAR TILL YTTRE ENHETER. Digitalkort-R 1.6

1.8 ANSLUTNING av TTL-signaler till Digitalkort-0-16TTL 1.7

1.9 ANSLUTNING av TTL-signaler till Digitalkort-T 1.8
Digitalkort-T-PUR

1.10 ANSLUTNING av TTL-signaler till Digitalkort-R 1.8

1.11 ANSLUTNING av TTL-signaler till Digitalkort-32TTL 1.9

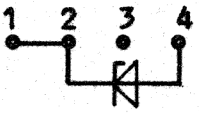
1.12 KORTADRESS 1.9

1.13 EXEMPEL PÅ ANSLUTNING AV YTTRE ENHETER. Digitalkort-T 1.10
Digitalkort-R

1. MATNINGSSPÄNNINGAR. ANSLUTNINGAR

1.1 MATNINGSSPÄNNINGAR till Digitalkort-0
Digitalkort-0-16TTL
Digitalkort-32TTL

Vid leverans är korten byglade för 5V-matning från bussen.

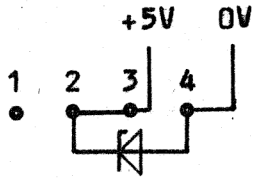


Bygling uppe till vänster.
Matning från bussen.

Om yttre 5V-matning önskas, bygla enligt vidstående figur.
5V ±5%, 300mA.

TTL-kretsarna på kortet tål max 7V matningsspänning. Zenerdioden mellan 2 och 4 skyddar dels mot för hög matningsspänning, dels mot felpolariserad spänning.

Angående byglar för de två förstnämnda kortens ingångar, se punkt 1.4 .

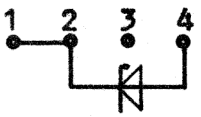


Bygling för yttre 5V-matning.

1.2 MATNINGSSPÄNNINGAR till Digitalkort-T
Digitalkort-T-PUR

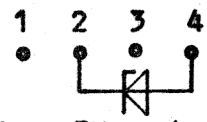
5V-matning

Vid leverans är kortet byglat för 5V-matning från bussen.



5V-matning från bussen.

Om 5V-matning till kortet önskas från yttre matningskälla, lämna 1-2 öppet. Anslut 5V-matning uppe till höger på kortet. 5V ±5%, 500 mA. Zenerdioden skyddar mot felaktiga matningsspänningar.



Yttre 5V-matning.

Extern matningsspänning (5V, 12V eller 24V)

Då yttre enheter ansluts till kortet behövs matningsspänning VEXT. Den ansluts till kopplingslist till höger på kortet. Spänningens storlek framgår av kortbeteckningen. Strömförbrukningen får vi genom att addera strömmarna till ingångar och laster. Varje ingång drar max 10 mA. Ovanför kopplingslisten för VEXT sitter en zenerdiod på 30V, 1,3W över VEXT. Zenerdioden skyddar transistorutgångarna dels mot för hög spänning, dels mot felpolariserad matningsspänning. Max VEXT se sid 1.4 och 1.5.

1.3 MATNINGSSPÄNNINGAR till Digitalkort-R

5V-matning

5V-matning ansluts till kopplingslist uppe till höger på kortet. 5V ±5%, 800 mA.
Zenerdioden mellan 2 och 4 uppe till vänster på kortet skyddar mot felaktiga spänningar, se 1.1.

Då man använder kortet utan yttre enheter och vill slippa yttre 5V-matning kan man göra följande: ta bort IC19 och IC20, löd in en bygel mellan 1 och 2 samt sätt tejp över kopplingslist för 5V så att inga misstag sker. 5V-matning erhålles på så sätt från bussen.

Extern matningsspänning (5V, 12V eller 24V)

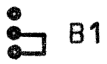
Då yttre enheter ansluts till kortet behövs matningsspänning VEXT. Den ansluts till kopplingslist till höger på kortet. Spänningens storlek framgår av kortbeteckningen. Strömförbrukningen får vi genom att addera strömmarna till ingångar och laster. Varje ingång drar max 10 mA.
Max VEXT se sid 1.4 och 1.6.

1.4 INGÅNGAR FRÅN YTTRE ENHETER. BYGLAR.

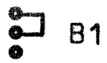
Digitalkort-T
Digitalkort-T-PUR
Digitalkort-R

Byglar B1-B8

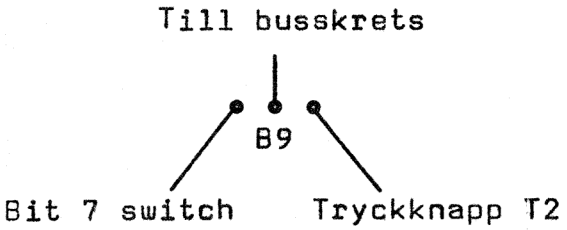
Med byglar B1-B8 i nedre läget fås ingångar IN0 från switchar på kortet.

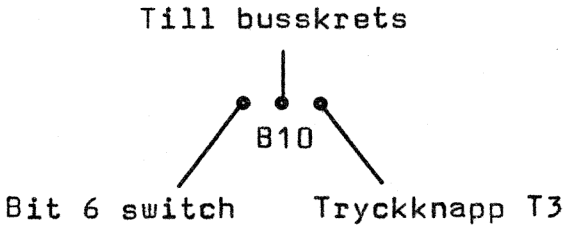


Med byglar B1-B8 i övre läget fås ingångar IN0 från yttre enheter.



Byglar B9 och B10





Med byglar B9 och B10 i vänstra läget fås IN1 (8 bitar) från switchar.
Med B9 och B10 i högra läget tas bit 7 i IN1 från tryckknapp T2 och bit 6 i IN1 från tryckknapp T3. T2 och T3 kan t ex användas som START- och STOPP-knappar.

Ingångar från yttre enheter

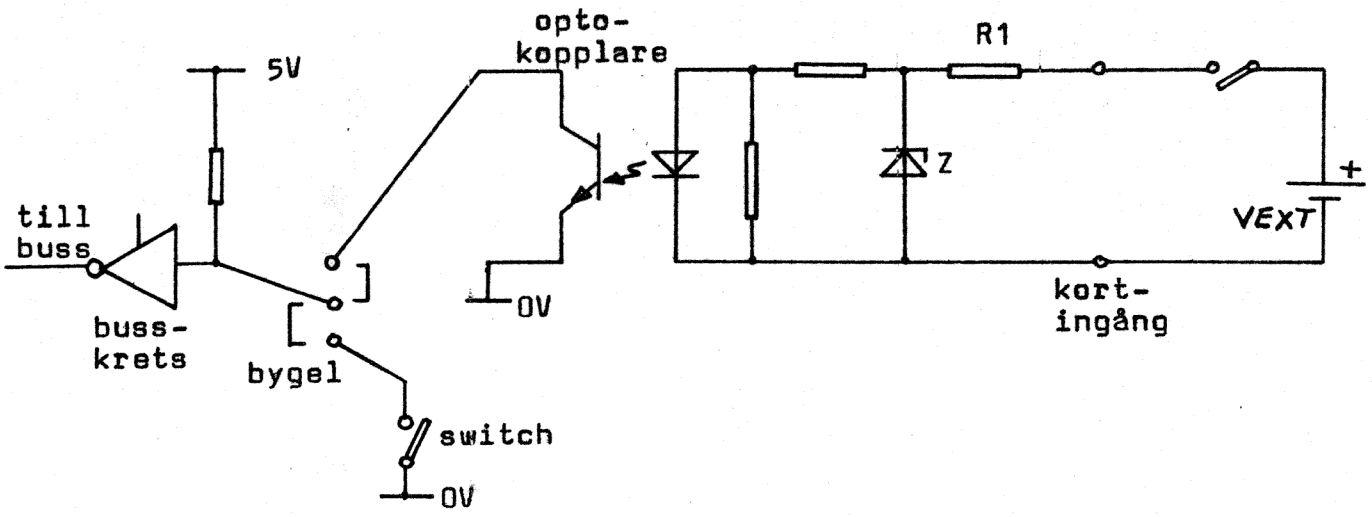


Fig 1 visar en av de 8 optoisolerade ingångarna.

När den yttre kontakten är sluten erhålles en "1" på bussen vid inläsning. Anpassning till olika yttre VEXT göres med R1. Zenerdioden skyddar optokopplaren dels mot för höga inströmmar, dels mot felpolariserade spänningar.

Tekniska data
Inströmmen för nominell inspänning är mindre än 10 mA (typ 8,5 mA).

Nominell inspänning VEXT (V)	Min inspänning för "1" typ (V)	svåraste fallet(V)	R1 ohm
5	3,5	4	100
12	6,5	10	820
24	12	20	2200

För zenerdioden gäller 5,6V och 0,4W. Resistorn R1 tål 0,5W. Därför får följande VEXT inte överskridas för ingångarna.

Nominell inspänning VEXT (V)	Max tillåten inspänning (V)
5	10
12	20
24	30

1.5 UTGÅNGAR TILL YTTRE ENHETER. Digitalkort-T.

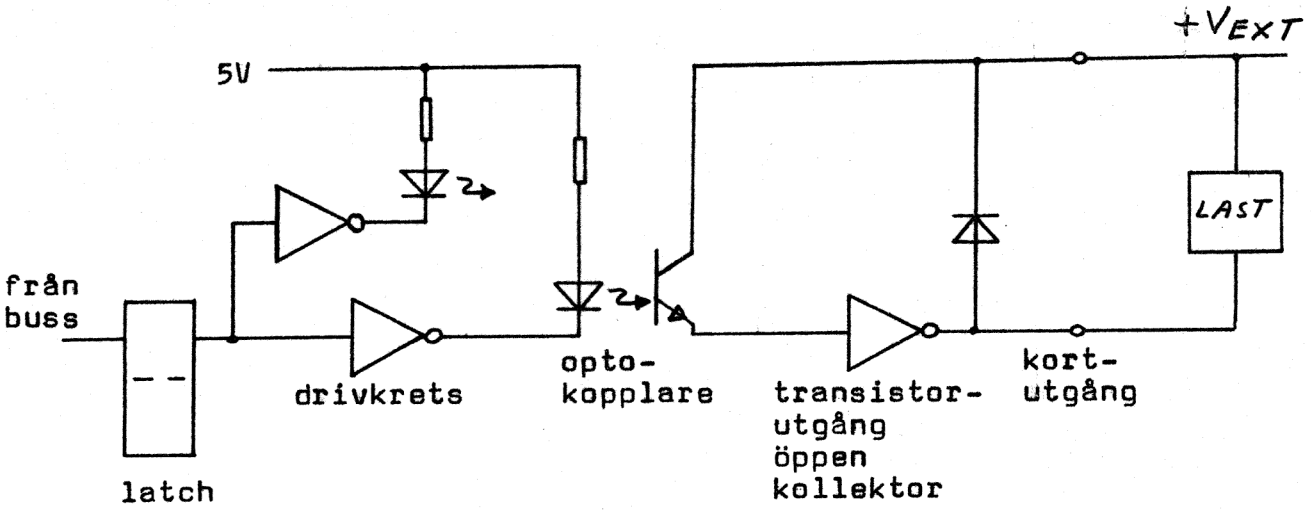


Fig 2 visar en av de 8 optoisolerade strömsänkande transistorutgångarna.

En "1" från bussen medför att lasten får ström. Dioder skyddar transistorerna då induktiva laster ansluts.

Tekniska data
Transistorutgångarna utgörs av ULN-kretsar (2 kapslar).
I vardera kapseln används 4 utgångar.
Utgångarna kan samtidigt sänka 240 mA. De kan parallellkopplas.
Utgångarna finns för 5V, 12V, 24V. Max VEXT=30V för utgångar.
Max VEXT för ingångar, se sid 1.4.

1.6 UTGÅNGAR TILL YTTRE ENHETER. Digitalkort-T-PUR.

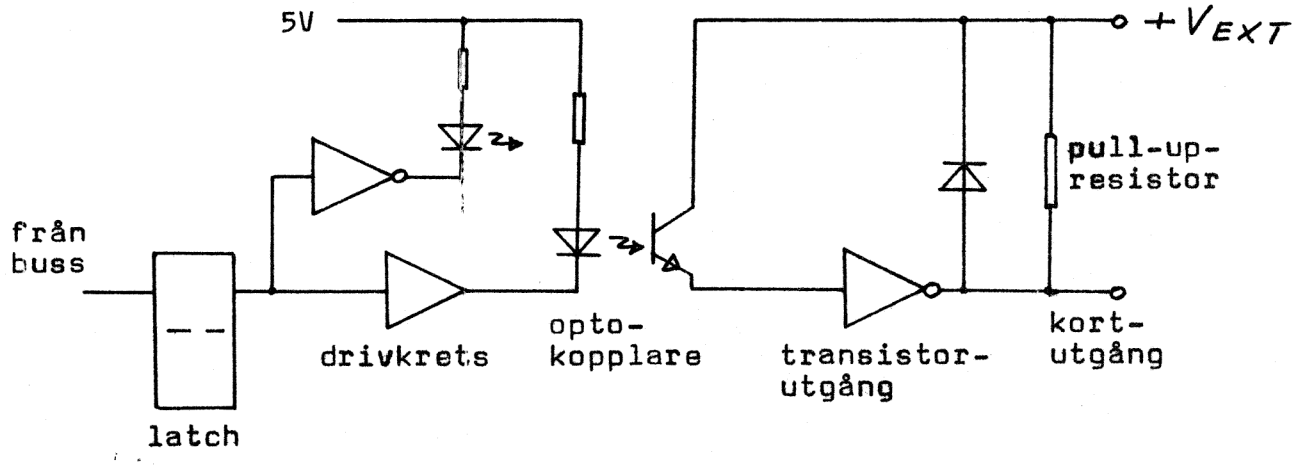


Fig 3 visar en av de 8 transistorutgångarna med "pull-up"-resistor.

Utgångarna har försetts med "pull-up"-resistorer. Genom att välja en drivkrets utan invertering erhålles hög utgång för en "1" från bussen. Om inget annat anges i beställningen är resistorerna på 10 kohm.
Max VEXT=30V för utgångar. Max VEXT för ingångar, se sid 1.4.

1.7 UTGÅNGAR TILL YTTRE ENHETER. Digitalkort-R.

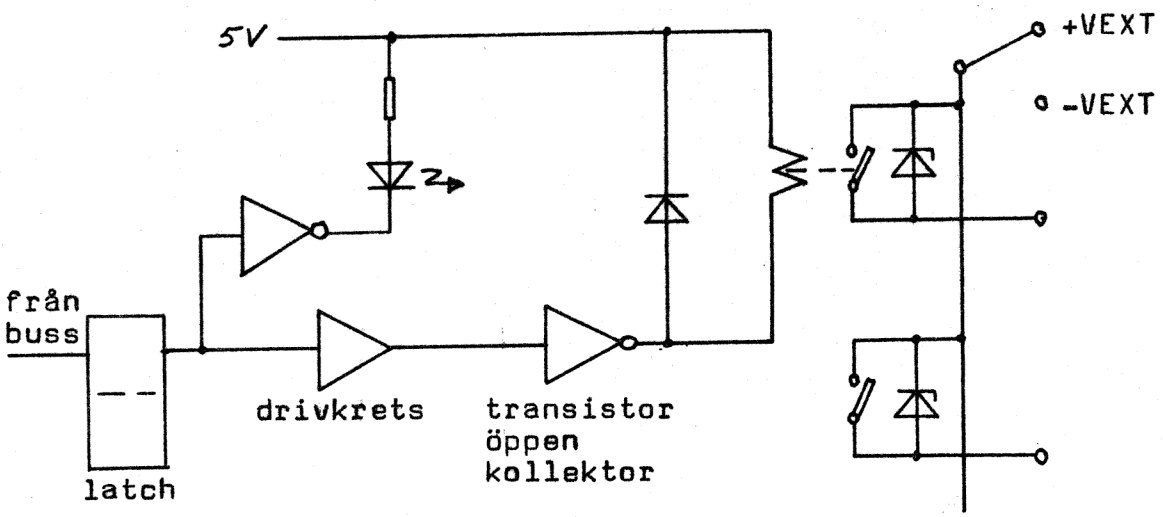
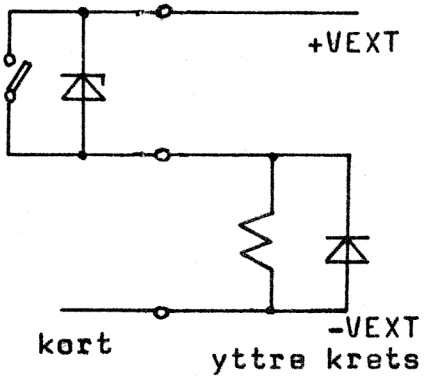


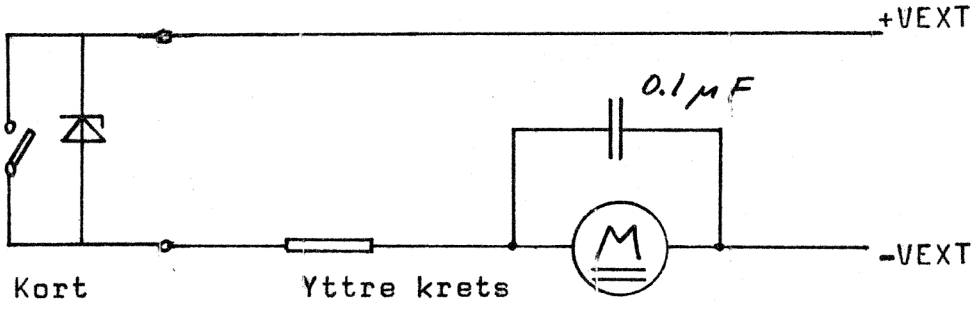
Fig 4 visar en av de 8 reläutgångarna.

En "1" från bussen medför att kontakten sluts. Kontakternas ena sida är hopkopplade och ansluts till + eller - på VEXT. Vid leverans är anslutning gjord till +VEXT. Denna anslutning sitter ovanför kopplingslist för VEXT. Om man ändrar denna anslutning till -VEXT måste zenerdioderna vändas. De sitter på kortets undersida.

Reläkontakterna är skyddade med zenerdioder på 30V, 1,3W. Om reläkontakterna skall driva induktiva laster med strömmar mindre än 0,3-0,4 A räcker detta skydd. Skall däremot reläutgångarna driva induktiva laster med större strömmar än 0,3-0,4 A skall lasterna förses med frihjulsdioder så att störningar på elektronikkretsarna undvikes. Vid frånslag kommer strömmen att klinga av till noll genom dioden.



Små likströmsmotorer kan anslutas direkt till reläutgångarna om de kopplas enligt figuren.



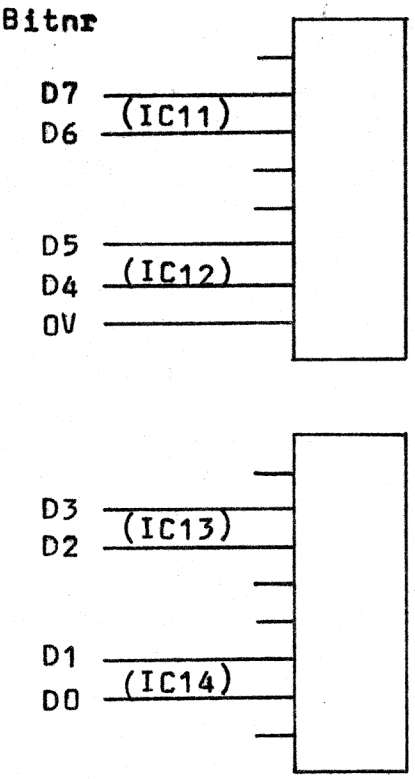
Tekniska data
Kontakterna kan bryta 3A/30V DC.
Omslagstid: 10 ms.
Max VEXT=30V för utgångar.
Max VEXT för ingångar, se sid 1.4.

1.8 ANSLUTNING av TTL-signaler till Digitalkort-0-16TTL

Se även punkt 10.

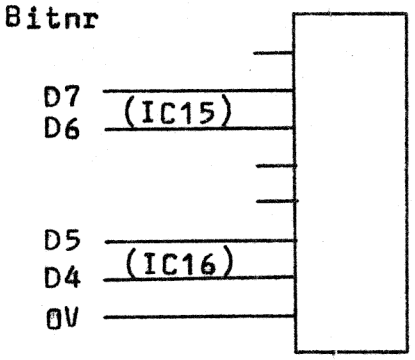
Utgångar UT0

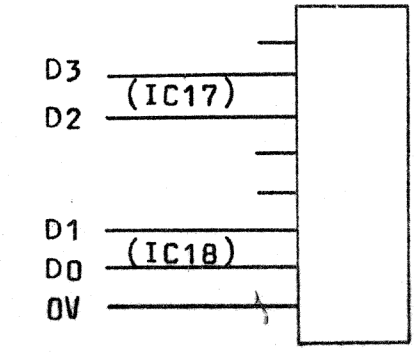
8 TTL-utgångar, UT0, tas från buffertkrets IC10.
De två 16-poliga hållarna, den övre avsedd för IC11 och IC12, den nedre avsedd för IC13 och IC14, utnyttjas. I hållarna placeras komponent-adaptrar. På dessa löds anslutningstrådarna fast. Tilledningarna till hållarna syns på kortets översida.



Ingångar IN0

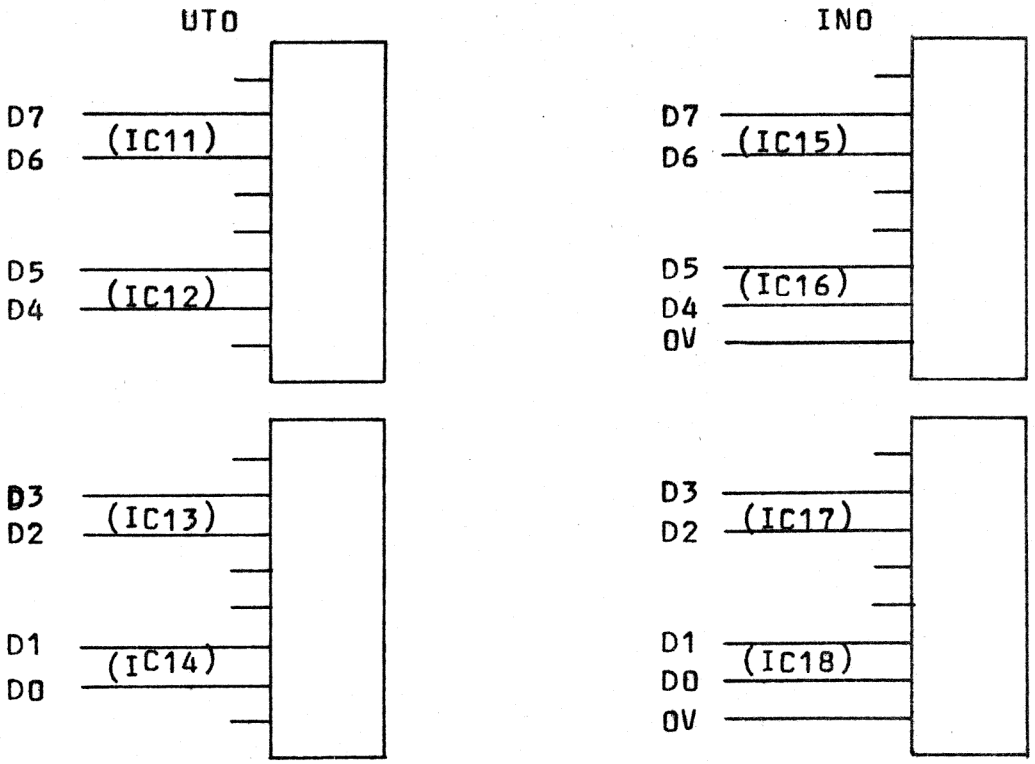
8 TTL-ingångar, IN0, tas in via byglar B1-B8. Byglarna skall placeras i övre läget.
De två 16-poliga hållarna, den övre avsedd för IC15 och IC16, den nedre avsedd för IC17 och IC18, utnyttjas. I hållarna placeras komponentadaptrar. På dessa löds anslutnings-trådar fast.





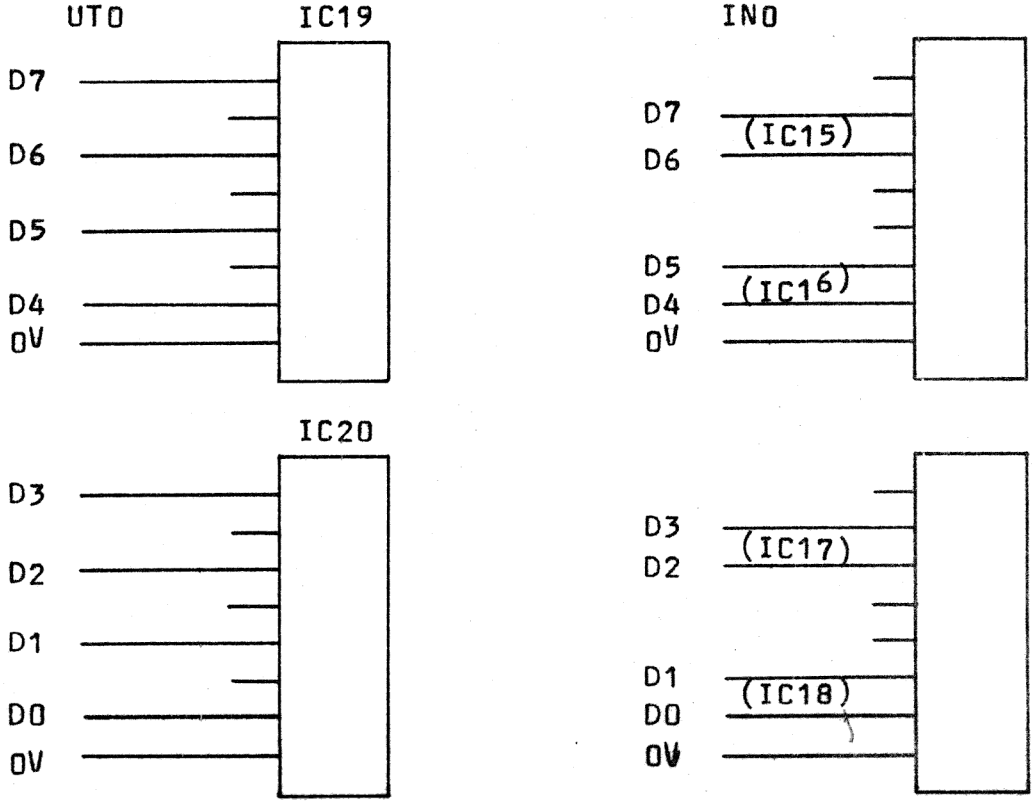
1.9 ANSLUTNING av TTL-signaler till Digitalkort-T
Digitalkort-T-PUR

Ta bort IC-kretsarna IC11-IC18. Placera komponentadaptrar i hållarna och löd fast anslutningstrådar på dessa. Se även 1.8. Se även punkt 12.4.



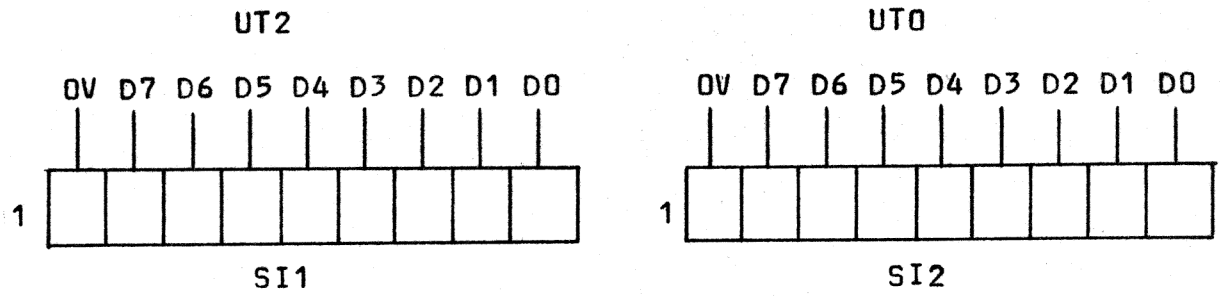
1.10 ANSLUTNING av TTL-signaler till Digitalkort-R

Ta bort IC-kretsarna IC19, IC20, IC15-IC18. Placera komponentadaptrar i hållarna och löd fast anslutningstrådar på dessa. Se även punkt 13.

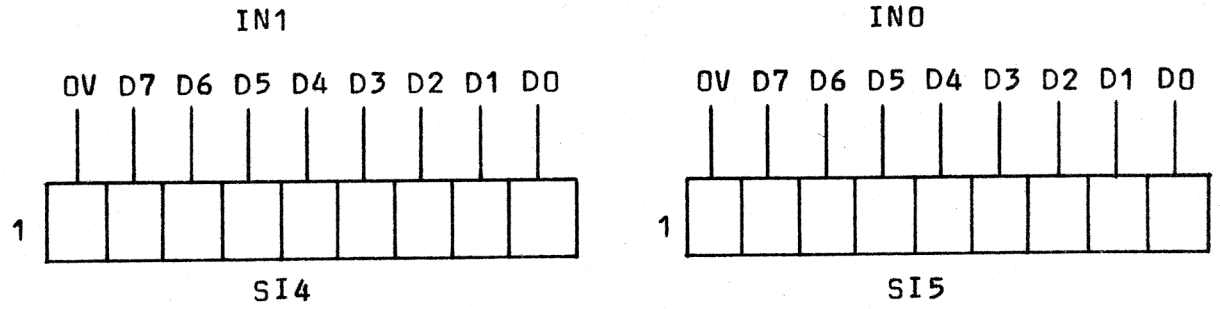


1.11 ANSLUTNING av TTL-signaler till Digitalkort-32TTL

Utgångar_



Ingångar_



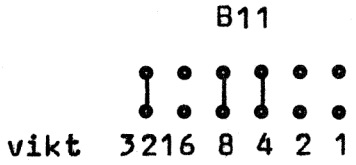
Montering

Skala av några mm av isoleringen på tråden.
Kläm fast hylsan på tråden.
Löd eventuellt försiktigt.
Trä in hylsan i hylsblocket så att
hylsan låses fast.

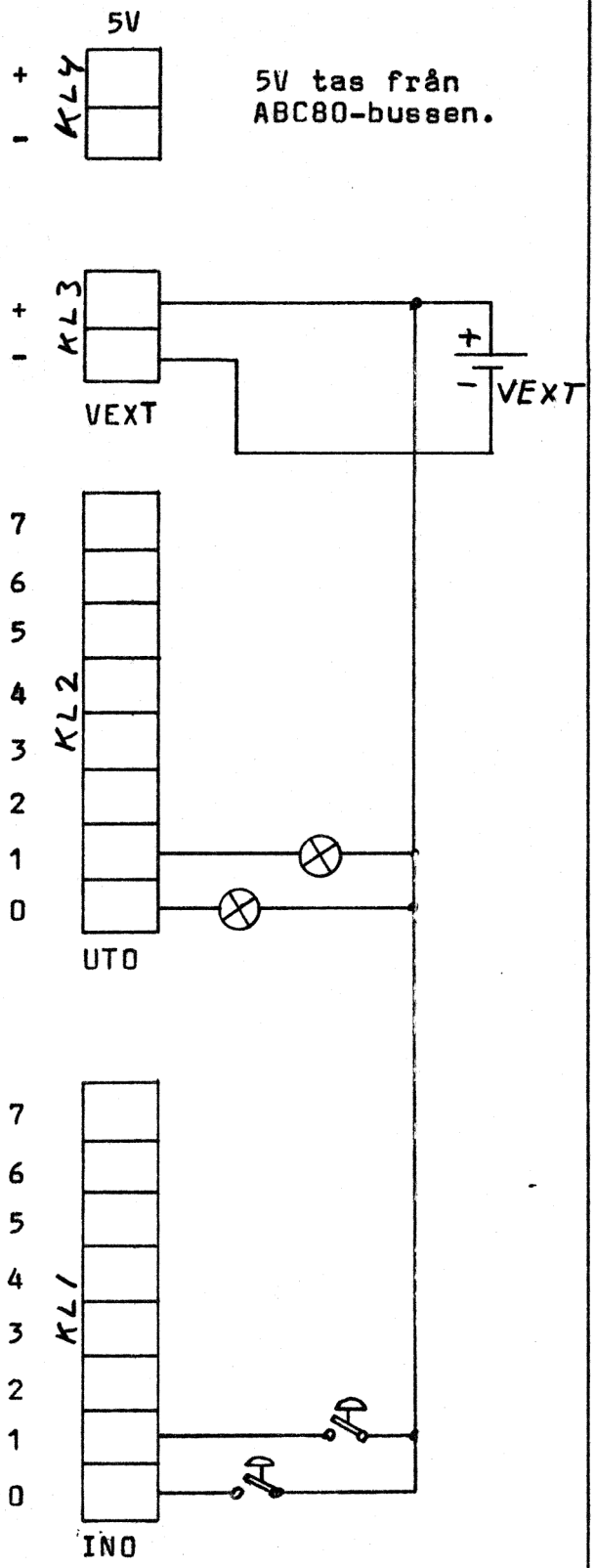
Se även punkt 11.

1.12 KORTADRESS

Varje kort ges en kortadress
mellan 0 och 63D genom
inlödning av byglar vid B11.
Inlödd bygel ger logisk 0.
Vidstående exempel ger
kortadress 19D.
Flera kort kan anslutas
till ABC80 via bussbox.
Korten skall därvid ges
olika adresser. Yttre
5V-matning krävs.
Sedan ett kort adresserats
kan det nås av datorn ända
tills nästa kort adresseras.

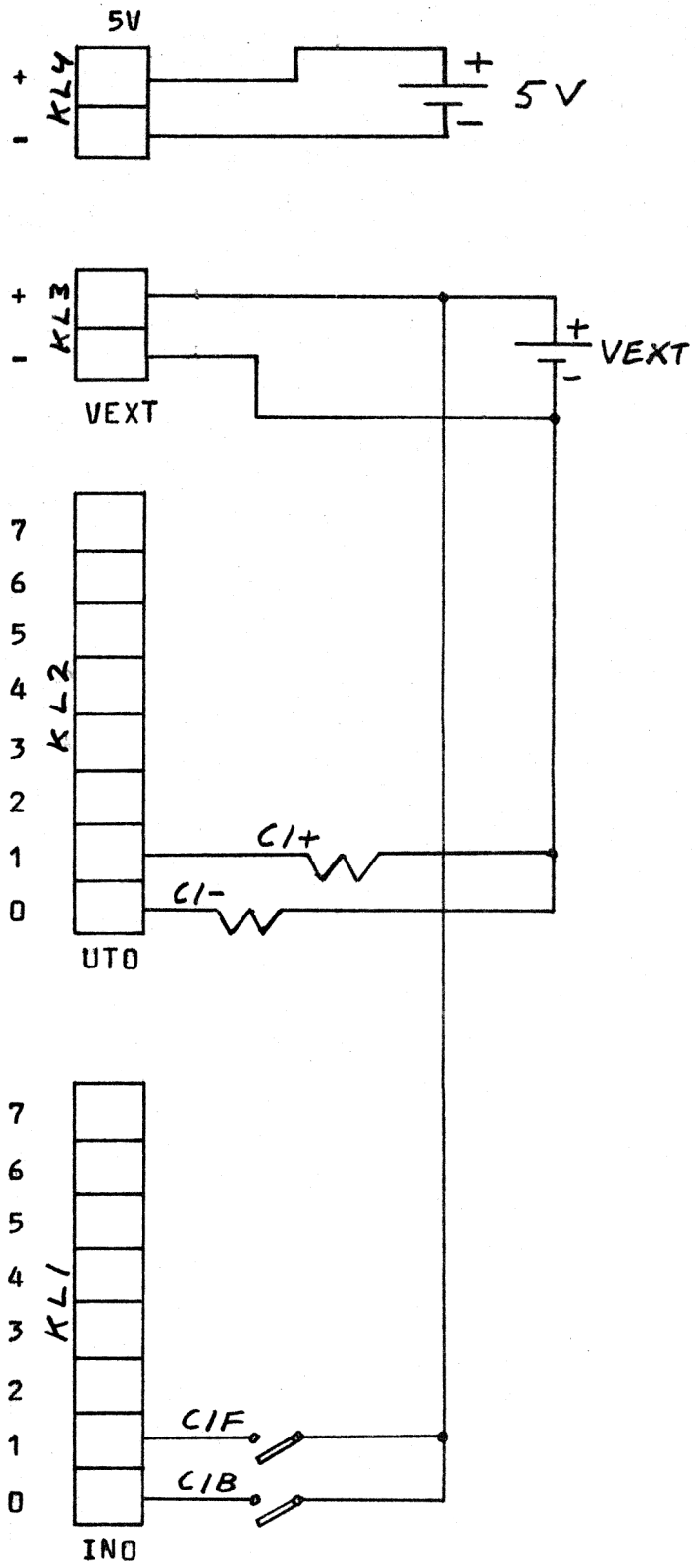


1.13 EXEMPEL PÅ ANSLUTNING AV YTTRE ENHETER. Digitalkort-T Digitalkort-R



Digitalkort-T

Två lampor är anslutna till bit 0 och 1. Två tryckknappar är anslutna till bit 0 och 1.



Digitalkort-R

En cylinder utan fjäderretur är ansluten till bit 0 och 1. Lägesgivarna är anslutna till bit 0 och 1.

2. IN-UT-KOMMANDON

Vid överföring mellan datorn och kortet är det de 8 minst signifikanta bitarna (den minst signifikanta byten) som överförs.

BASIC

OUT	kan användas både som kommando och i program.
INP	kan användas både i direktmod (;INP(0)) och i program som funktion.
OUT 1,A	Kortet med adress A kopplas in. Lysdioden på kortet tänds vid satsens utförande. Kortets adress anges på kortet.
OUT 0,D	D överförs och lagras i utgång UT0.
OUT 2,D	D överförs och lagras i utgång UT2.
X=INP(0)	Ingång IN0 överförs till X.
Y=INP(1)	Ingång IN1 överförs till Y.

Om heltalsvariabler används snabbas programmen upp.

Ex:

```
10 OUT 1%,A%
20 X%=INP(0%)
```

Z80 ASSEMBLER

Överföringen sker mellan accumulatorn (Acc) i Z80 CPU och yttre enhet i nedanstående exempel. Inverkan på yttre enheter, se ovan.

OUT (1),A	Adresserar ett kort.
OUT (0),A	Innehållet i Acc överförs till utgång UT0.
OUT (2),A	Innehållet i Acc överförs till utgång UT2.
IN A,(0)	Ingång IN0 överförs till Acc.
IN A,(1)	Ingång IN1 överförs till Acc.

3. TYPEXEMPEL PÅ IN-UTMATNING. NÅGRA BEGREPP

3.1 Typexempel

Följande exempel visar

- kortadressering
- läsning av ingångar IN0 och IN1
- skrivning på utgångar UT0 och UT2.

```
TYPEXEMPEL DEM1  DIGITALKORT
20 OUT 1,19 : REM KORT NR 19 ADRESSERAS
100 REM INLÄSNING **********************
110 X=INP(0) : REM IN0 öVERFöRS TILL X
120 Y=INP(1) : REM IN1 öVERFöRS TILL Y
200 REM ********************************
210 REM HÄR KAN BERäKNINGAR UTFöRAS
500 REM DATA SKICKAS UT ***************
510 OUT 0,X : REM X öVERFöRS TILL UT0
520 OUT 2,Y : REM Y öVERFöRS TILL UT2
530 GOTO 110
```

Genom att använda heltalsvariabler snabbas programmen upp. Dessutom kan man skriva flera instruktioner på varje radnummer, se ref 1 sid 31.

```
10 REM TYPEXEMPEL DEM2 DIGITALKORT
20 OUT 1%,19%
100 REM INLÄSNING **********************
110 X%=INP(0%)
120 Y%=INP(1%)
200 REM ********************************
210 REM HÄR KAN BERäKNINGAR UTFöRAS
500 REM DATA SKICKAS UT ***************
510 OUT 0%,X%,2%,Y%
530 GOTO 110
```

```
10 REM DEM3 DIGITALKORT
20 OUT 1,19
30 OUT 0%,INP(0%),2%,INP(1%) : GOTO 30
```

Lösningarna till exemplen nedan är i allmänhet skrivna med en instruktion per radnummer för att underlätta programmens läsbarhet.

3.2 Några begrepp

Den minsta informationsenheten, en bit, 0 eller 1, kan lagras i en vippa. Flera bitar kan lagras i ett register. 8 bitar kallar vi för en byte. 1 byte kan i ABC80 lagras i en minnescell med en adress om 16 bitar.

Med 8 bitar kallar vi bitarna D0-D7.

Bitvikt	128	64	32	16	8	4	2	1
	D7	D6	D5	D4	D3	D2	D1	D0
Bitnr	7	6	5	4	3	2	1	0
	MSB							LSB

MSB=mest signifikanta biten (most significant bit)
LSB=minst signifikanta biten (least significant bit).

Bitnr N har bitvikt 2^N. De binära talen ligger mellan 00000000B och 11111111B. Med B på slutet ser vi att det är binärt. Decimala tal skrivs på vanligt sätt.

4. PROGRAMMERINGSUPPGIFTER I BASIC

Nedanstående uppgifter kan köras på Digitalkort-0.
Man behöver alltså inte ansluta yttre enheter.
Uppgifterna är lämpliga som förövningar till tillämpnings-exempel såsom cylinderprogram.

4.1 Exempel på koder

ASCII-koden finns i ref 1 sid 53. Heltalsvariabler, se ref 1 sid 23. Exemplen avser att ge förtrogenhet med ASCII-koden, binärkoden samt tvåkomplement. Skriv program för nedanstående uppgifter.

A1
Då en tangent på ABC80:s tangentbord nedtryckes vill vi se motsvarande karaktär på bildskärmen och karaktärens ASCII-kod på UT0.

A2
Vi låter IN0 vara ASCII-koden för en karaktär. Visa karaktären på skärmen. För att få en stillastående bild på skärmen kan vi göra så att IN0 läses in varje gång en tangent nedtryckes.

A3
Presentera det binära innehållet i IN0 i decimal form på skärmen varje gång en tangent nedtryckes.

A4
Presentera det binära innehållet i IN1 (high byte) och IN0 (low byte), 16 bitar, i decimal form på skärmen varje gång en tangent nedtryckes.

A5
Ge ett tal mellan 0 och 255 på tangentbordet. Presentera motsvarande binära tal (8 bitar) på UT0.

A6
Ge ett tal mellan 0 och 65535 på tangentbordet. Presentera motsvarande binära tal (16 bitar) på UT2 (high byte) och UT0 (low byte).

A7
Vi vill räkna med både positiva och negativa heltal och använder därför tvåkomplement. Ge ett tal mellan -32768 och 32767 på tangentbordet. Presentera motsvarande binära tal (16 bitar) i tvåkomplement på UT2 (high byte) och UT0 (low byte).

A8
Vi vill räkna med positiva och negativa heltal inom talområdet -128 till 127 (8 bitar).
Ge ett tal mellan -128 och 127 på tangentbordet. Presentera motsvarande binära tal (8 bitar) i tvåkomplement på UT0.

4.2 Exempel på aritmetik

B1
Skriv ett program som ger summan av IN0 och IN1 på UT2 och UT0.
Ställ upp IN0 och IN1 i binär form och utför kontrollberäkning för hand.

B2
Skriv ett program som ger IN0 minus IN1 på UT2 och UT0. Resultatet ges i tvåkomplement för 16 bitar. Utför även beräkningarna för hand.

4.3 Exempel på logiska funktioner

För att kunna manipulera enstaka bitar måste man behärska de logiska funktionerna.

C1
Skriv ett program som utför den logiska operationen OCH mellan IN0 och IN1 (positionsvis).
Resultatet visas på UT0.
Räkna för hand och kontrollera.
Ex:

```
IN0: 1100 0101
IN1: 0101 1100
-------------
UT0: 0100 0100
```

C2
Ändra operationen i uppg C1 till ELLER.
Ex på handräkning:

```
IN0: 1100 0101
IN1: 0101 1100
-------------
UT0: 1101 1101
```

C3
Ändra operationen i uppg C1 till EXKLUSIVT-ELLER.
Ex på handräkning:

```
IN0: 1100 0101
IN1: 0101 1100
-------------
UT0: 1001 1001
```

C4
Läs in IN0. Invertera IN0 och visa resultatet på UT0.

C5
Skriv ett program som utför följande.
Bit 0 och 3 i IN0 skall överföras till UT0.
Övriga bitar i UT0 skall vara 0.

C6
Skriv ett program som utför följande.
Bit 1 och 4 i IN0 skall överföras till UT0.
Övriga bitar i UT0 skall vara 1.

C7
Skriv ett program som läser in IN0.
Invertera bit 4 och 6 och låt övriga bitar vara oförändrade.
Visa resultatet på UT0.

C8
Skriv ett program som utför följande.
Bit 0, 2 och 4 i IN0 och bit 1, 3 och 5 i IN1 skall överföras till motsvarande positioner i UT0.
Övriga bitar i UT0 skall vara 0.

C9
Vi vill från tangentbordet ändra en bit i UT0 utan att övriga bitar påverkas.
På tangentbordet anger vi först bitnr (0-7) och sedan 0 eller 1 för 0-ställning respektive 1-ställning av den bit vi vill påverka.

C10
Skriv ett program som läser in IN0.
Visa på bildskärmen de 8 bitarnas tillstånd (0 eller 1).

C11
Skriv ett program som räknar antalet gånger vi trycker ned tryckknapp T2.
Antalet nedtryckningar visas på UT0.
Inkoppling av T2, se punkt 1.4.

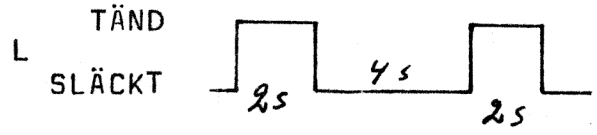
C12
Beskriv hur operationen OCH används för maskning.

C13
Beskriv hur operationen ELLER används för maskning.

C14
Beskriv hur operationen EXKLUSIVT-ELLER används för bitmanipulering.

4.4 Exempel på tidfördröjningar

D1
En lampa L skall tändas och släckas enligt nedanstående sekvens.
Använd programvarumässig fördröjning.

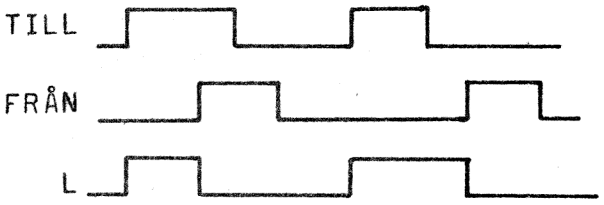


D2
Skriv program för samma sekvens som i D1 men använd realtidsklockan för tidfördröjningarna.

4.5 Blandade uppgifter

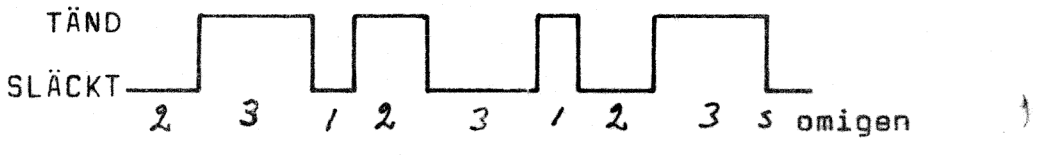
Nedanstående uppgifter är lämpliga som förövning till uppgifter av typ cylinderprogram.

E1
En lampa L skall tändas av en switch TILL och släckas av en switch FRÅN.
Då TILL går från 0 till 1 skall L tändas.
Då FRÅN går från 0 till 1 skall L släckas.
Se även nedanstående figur.
Rita flödesschema.



E2
Samma som uppg E1 men nu har vi 8 lampor L.
Till varje L hör en switch TILL och en switch FRÅN.

E3
Vi vill ha ett program som medger att vi kan programmera in en viss sekvens för en lampa, exempel se nedan.
Tiderna lägger vi lämpligen i en lista (indexerad variabel).
Hitta även på andra sekvenser och provkör dessa.



E4
Skriv program för samma problem som i uppg E3 men lägg in tiderna i en tabell i minnet med POKE-satsen, se ref 1 sid 44.
Vi bör kunna göra ett menyval mellan programmering av sekvens och körning av sekvens.
Vid körning hämtas tiderna i minnet med PEEK-funktionen.
Om körningen avbryts med CTRL-C bör vi sedan kunna fortsätta körningen igen i den fas där sekvensen avbröts.

E5
Vi vill kunna styra utgångarna på UTO efter ett visst mönster, exempel se nedan.
Både koden för utgångarna och tidfördröjningarna lägger vi i en lista.

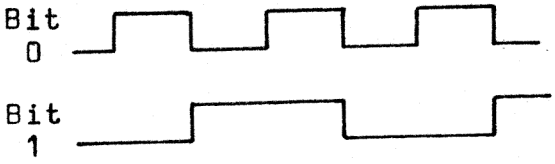
Exempel:

Tid	D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	0	0	0	1
0,5	0	1	0	0	0	0	1	0
2	0	0	1	0	0	1	0	0
3	0	0	0	1	1	0	0	0

omigen

E6
Skriv om programmet i uppg E5 så att både koder och tider läggs in i minnet med POKE-satsen.
Vi bör kunna göra menyval mellan programmering och körning. Om körningen avbryts bör vi sedan kunna fortsätta körningen i den fas där avbrottet skedde.

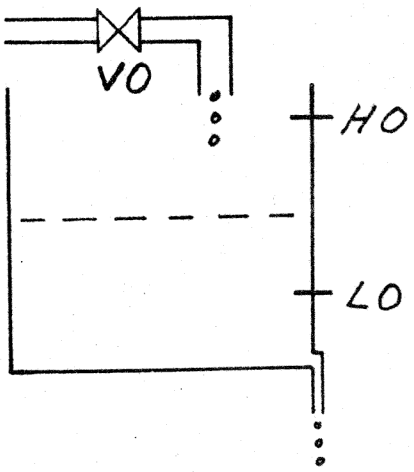
E7
Skriv ett program som får lamporna i UTO att blinka på följande sätt.
Lampa i bit nr 0, 2, 4, 6 skall tändas och släckas med 1 sek mellanrum.
Lampa i bit nr 1, 3, 5, 7 skall tändas och släckas med halva frekvensen jämfört med de först nämnda lamporna.



E8
Modifiera uppg E7 så att frekvensdelningen blir 1:M och M ges i början av programmet.

E9
Vi har 8 fönster som skall övervakas. Till varje fönster finns en switch (IN0) som indikerar öppet fönster = 1 och stängt fönster = 0.
Vi har två lamptablåer för fönstren, UT2 och UT0.
Då ett fönster öppnas skall motsvarande lampa i UT2 blinka snabbt ända tills tryckknapp T2 påverkas. Då T2 påverkas slocknar lampan i UT2.
Om fönstret fortfarande är öppet då T2 påverkas börjar en lampa i UT0 att blinka långsamt.
Även när fönstret stängts skall lampan i UT0 blinka tills T2 påverkas.
T2 är alltså en kvittens-signal. Lampor kan enbart släckas då T2 påverkas.

E10



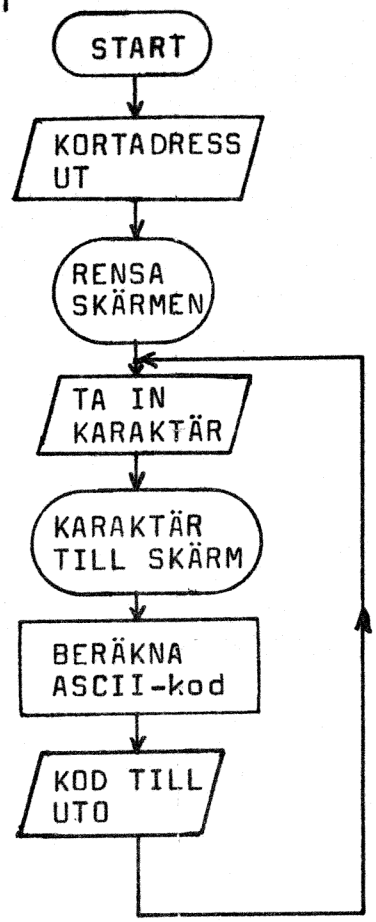
Nivån i tanken regleras genom TILL-FRÅN-reglering.
VO=1 öppnar ventilen (bit 0 UT0).
För hög nivå indikeras av att HO (bit 0 IN1) blir 1.
För låg nivå indikeras av att LO (bit0 IN0) blir 1.
Då nivån blir för låg öppnas ventilen. Då nivån blir för hög stängs ventilen.
Observera att HO och LO inte kan bli 1 samtidigt.

E11
Vi har 8 tankar som skall nivåregleras samtidigt.
För L, H och V tar vi lämpligen samma bitnummer i IN0, IN1 och UT0 till en tank med samma nummer. Till tank nr 4 hör L4, H4 och V4 osv.

5. LÖSNINGAR TILL PROGRAMMERINGSUPPGIFTER I BASIC

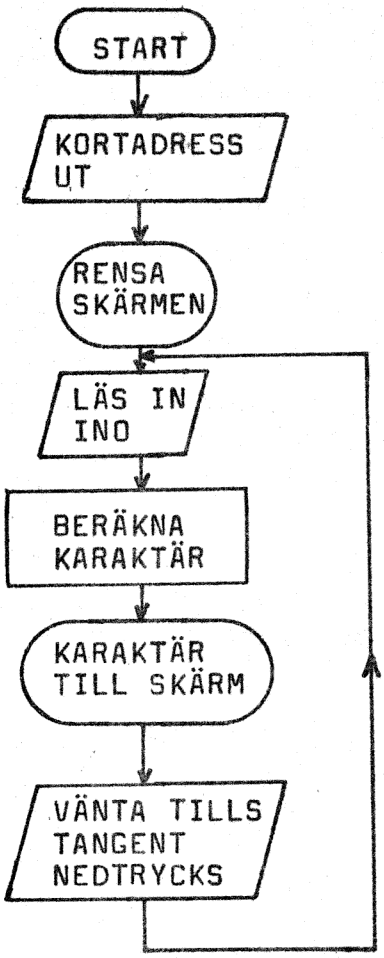
En programmeringsuppgift kan lösas på många sätt. Nedanstående förslag skall därför ses som en av många lösningar.

A1




```
10 REM UPPG A1 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM **********************************
60 ; CHR$(12)
70 GET A$ : ; A$
80 X=ASC(A$)
90 OUT 0,X
100 GOTO 70
```

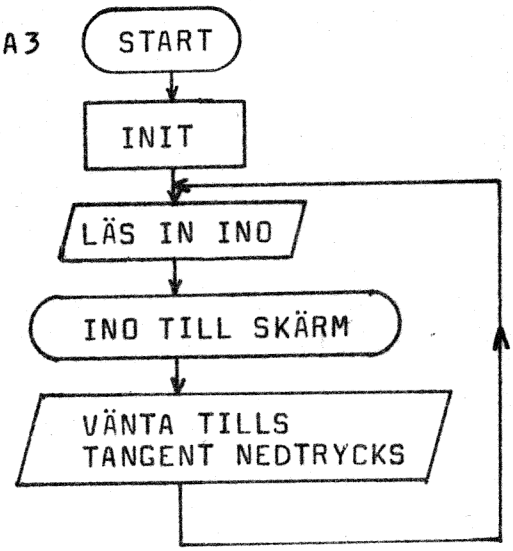
A2




```
10 REM UPPG A2 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM **********************************
60 ; CHR$(12)
70 Y=INP(0)
80 ; CHR$(Y)
90 GET A$
100 GOTO 70
```

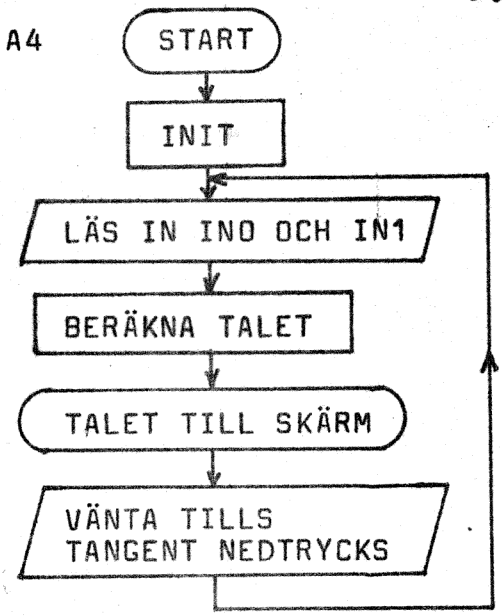
I fortsättningen ritar vi en ruta med INIT (initiering) för KORTADRESS UT och RENSA SKÄRM.

A3



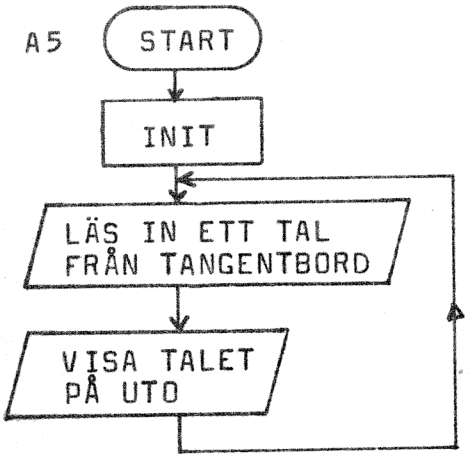

```
10 REM UPPG A3 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; INP(0)
80 GET A$
90 GOTO 70
```

A4



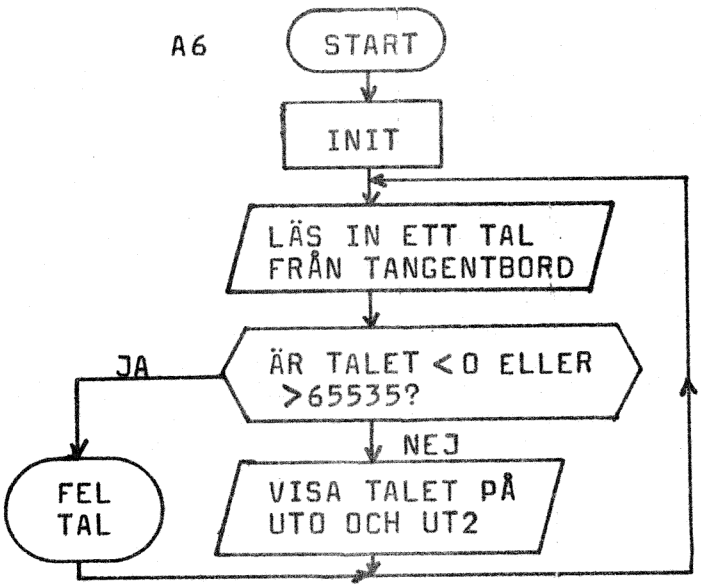

```
10 REM UPPG A4 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 X%=INP(0)
80 Y%=INP(1)
90 ; Y%*256+X%
100 GET A$
110 GOTO 70
```

A5




```
10 REM UPPG A5 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; "GE ETT TAL MELLAN 0 OCH 255 ";
80 INPUT X
90 OUT 0,X
100 GOTO 70
```

A6




```
10 REM UPPG A6 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; "GE ETT TAL MELLAN 0 OCH 65535 ";
80 INPUT X
90 IF X<0 THEN 130
100 IF X>65535 THEN 130
110 OUT 0,X,2,X/256
120 GOTO 70
130 ; "FEL TAL" : GOTO 70
```

A7

```
10 REM UPPG A7 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; "GE ETT TAL MEL. -32768 OCH 32767 ";
80 INPUT X
90 IF X<-32768 THEN 140
100 IF X>32767 THEN 140
110 X%=X
120 OUT 0%,X%,2%,SWAP%(X%)
130 GOTO 70
140 ; "FEL TAL" : GOTO 70
```

A8

```
10 REM UPPG A8 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; "GE ETT TAL MELLLAN -128 OCH +127 ";
80 INPUT X%
90 IF X%<-128% OR X%>127% THEN 120
100 OUT 0%,X%
110 GOTO 70
120 ; "FEL TAL" : GOTO 70
```

B1

```
10 REM UPPG B1 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Y%=INP(1%)
80 Z%=X%+Y%
90 OUT 0%,Z%,2%,SWAP%(Z%)
100 GOTO 60
```

B2

```
10 REM UPPG B2 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Y%=INP(1%)
80 Z%=X%-Y%
90 OUT 0%,Z%,2%,SWAP%(Z%)
100 GOTO 60
```

C1

```
10 REM UPPG C1 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Y%=INP(1%)
80 Z%=X% AND Y%
90 OUT 0%,Z%
100 GOTO 60
```

C2

```
10 REM UPPG C2 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Y%=INP(1%)
80 Z%=X% OR Y%
90 OUT 0%,Z%
100 GOTO 60
```

Vi observerar att i detta fall fungerar ABC80 och kortet som 8 st 2-ingångars OCH-grindar. Fördröjningen från ingång till utgång rör sig om millisekunder med program skrivna i BASIC. För en TTL-grind är fördröjningen 10 nanosekunder.

C3

```
10 REM UPPG C3 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Y%=INP(1%)
80 Z%=X% XOR Y%
90 OUT 0%,Z%
100 GOTO 60
```

C4

```
10 REM UPPG C4 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 OUT 0%, NOT X%
80 GOTO 60
```

C5

```
10 REM UPPG C5 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Z%=X% AND 9%
80 OUT 0%,Z%
90 GOTO 60
```

C6

```
10 REM UPPG C6 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Z%=X% OR NOT 18%
80 OUT 0%,Z%
90 GOTO 60
```

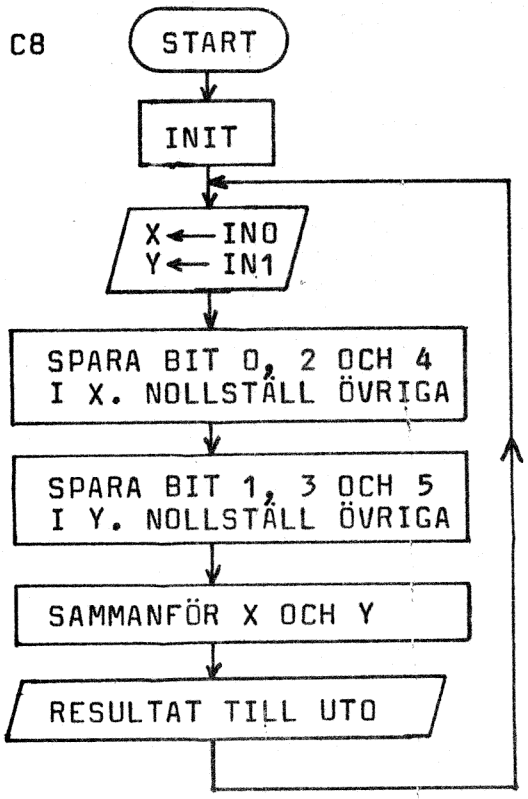
C7

```
10 REM UPPG C7 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Z%=X% XOR 80%
80 OUT 0%,Z%
90 GOTO 60
```

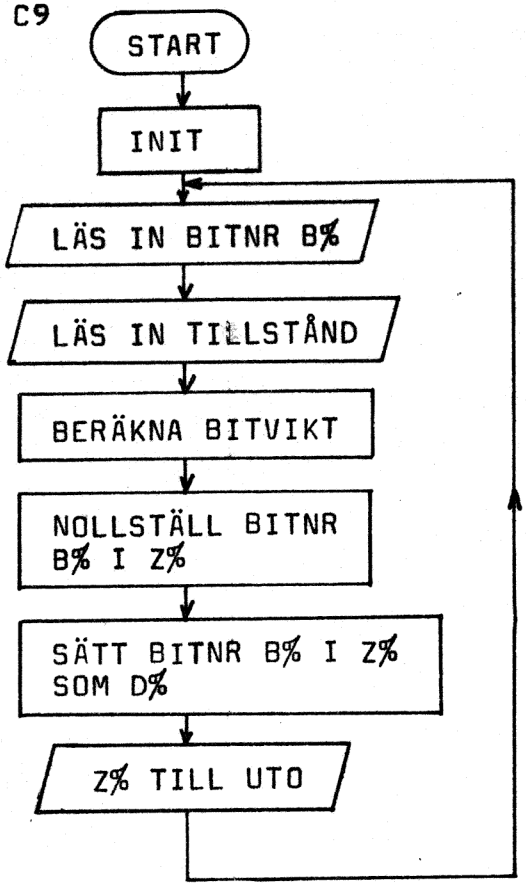
C8

```
10 REM UPPG C8 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 X%=INP(0%)
70 Y%=INP(1%)
80 X%=X% AND 21%
90 Y%=Y% AND 42%
100 Z%=X% OR Y%
110 OUT 0%,Z%
120 GOTO 60
```

C8



C9




```
10 REM UPPG C9 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; "VILKEN BIT SKALL PÅVERKAS (0-7) "
80 GET A$
90 B%=VAL(A$)
100 IF B%<0% OR B%>7% THEN 70
110 ; "SKALL BIT";B%;" VARA 0 ELLER 1 "
120 GET A$
130 D%=VAL(A$)
140 IF D%<0% OR D%>1% THEN 110
145 Y%=2%^B%
150 Z%=Z% AND NOT Y%
160 Z%=Z% OR (D%*Y%)
170 OUT 0%,Z% : ; : ;
180 GOTO 70
```

C10

Med X%=INP(0%)

N%=bitnr

D%=tillståndet för bit nr N%

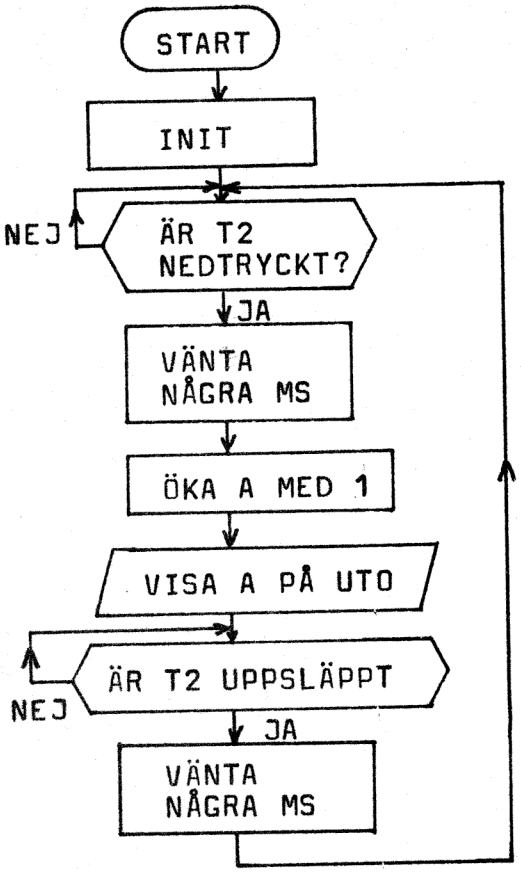
får vi

D%=(X% AND 2%↑N%)/2%↑N% .

```
10 REM UPPG C10 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 ; CHR$(12)
70 ; "INGÅNGARNAS TILLSTÅND FÖR IN0"
80 ; CUR(8,0);"BIT NR  D7  D6  D5  D4  D3  D2  D1  D0"
90 X%=INP(0%)
100 FOR N%=7% TO 0% STEP -1%
110 ; CUR(10%,8%+(7%-N%)*4%);(X% AND 2%^N%)/2%^N%
120 NEXT N%
130 GOTO 90
```

C11 Lösning A

I denna lösning väntar vi först på att T2 skall nedtryckas. Då T2 nedtrycks väntar vi några millisekunder på grund av studs innan vi fortsätter. Sedan väntar vi på att T2 släppe.
Denna lösning medför att vi låser datorn i en loop.
Datorn kan bara behandla en enda tryckknapp.




```
10 REM UPPG C11A DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 A%=0% : REM NOLLSTÄLL RÄKNARE
70 REM VÄNTA PÅ NEDTRYCKNING *********
80 IF (INP(1%) AND 128%)=0% THEN 80
90 REM ANTISTUDS
100 FOR N=1 TO 20 : NEXT N
110 A%=A%+1%
120 OUT 0%,A%
130 REM VÄNTA TILLS T2 SLÄPPS
140 IF (INP(1%) AND 128%)=128% THEN 140
150 REM ANTISTUDS
160 FOR N=1 TO 20 : NEXT N
170 GOTO 80
```

C11 Lösning B

Vi vill inte låsa upp datorn på en enda tryckknapp och söker därför en algoritm.
Sätt A1= sista inläsningen av IN1 och
B1= näst sista inläsningen av IN1 (gamla värden).
Vi antar värden på A1 och B1 och räknar.

EXEMPEL:

```
B1:   1011 0110
A1:   1101 0011
      ---------
XOR:  0110 0101
```

Tar vi XOR mellan NYA och GAMLA värden får vi 1 i de positioner som ändrats.
Genom att sedan ta logiskt OCH mellan ÄNDRINGAR och NYA värden får vi 1 i de positioner som ändrats från 0 till 1.

```
ÄNDRINGAR:  0110 0101
NYA (A1) :  1101 0011
            ---------
OCH         0100 0001
```

Vi får fram ändringarna för alla 8 positionerna samtidigt. I exempel C11 är vi bara intresserade av bitnr 7. Om vi vill ha 1 i de positioner som ändrats från 1 till 0 skall vi ta logiskt OCH mellan ÄNDRINGAR och GAMLA värden (B1) i andra beräkningen.

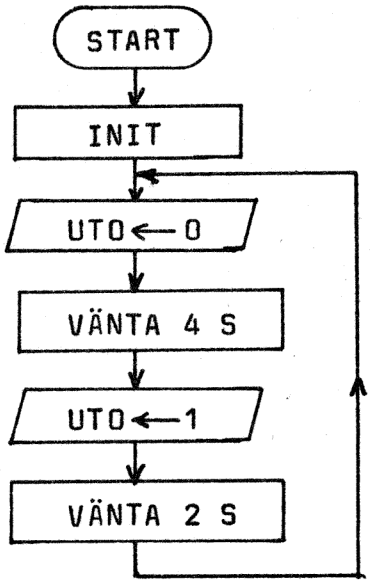
```
10 REM UPPG C11B DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 A%=0% : B1%=0%
70 A1%=INP(1%) AND 128%
80 X%=A1% XOR B1%
90 Y%=X% AND A1%
100 IF Y%=0% THEN 130
110 A%=A%+1%
120 OUT 0%,A%
130 B1%=A1%
140 FOR N=1 TO 20 : NEXT N : REM ANTISTUDS
150 GOTO 70
```

C12
Med operationen AND kan vi maska bort, lägga mask över, ointressanta bitar. De bortmaskade bitarna nollställs.

C13
Med operationen OR ettställs de bortmaskade bitarna. OR kan även användas då man vill sammanställa flera variabler, se uppg C8.

C14
Med operationen XOR kan vi invertera vissa bitpositioner och låta andra vara oförändrade. Vi kan även använda XOR för att upptäcka ändringar. Lösning C11A kan vi jämföra med nivåtriggning medan C11B kan jämföras med flanktriggning. Vi kan känna av positiv eller negativ flank. Jämför triggning av vippor.

D1

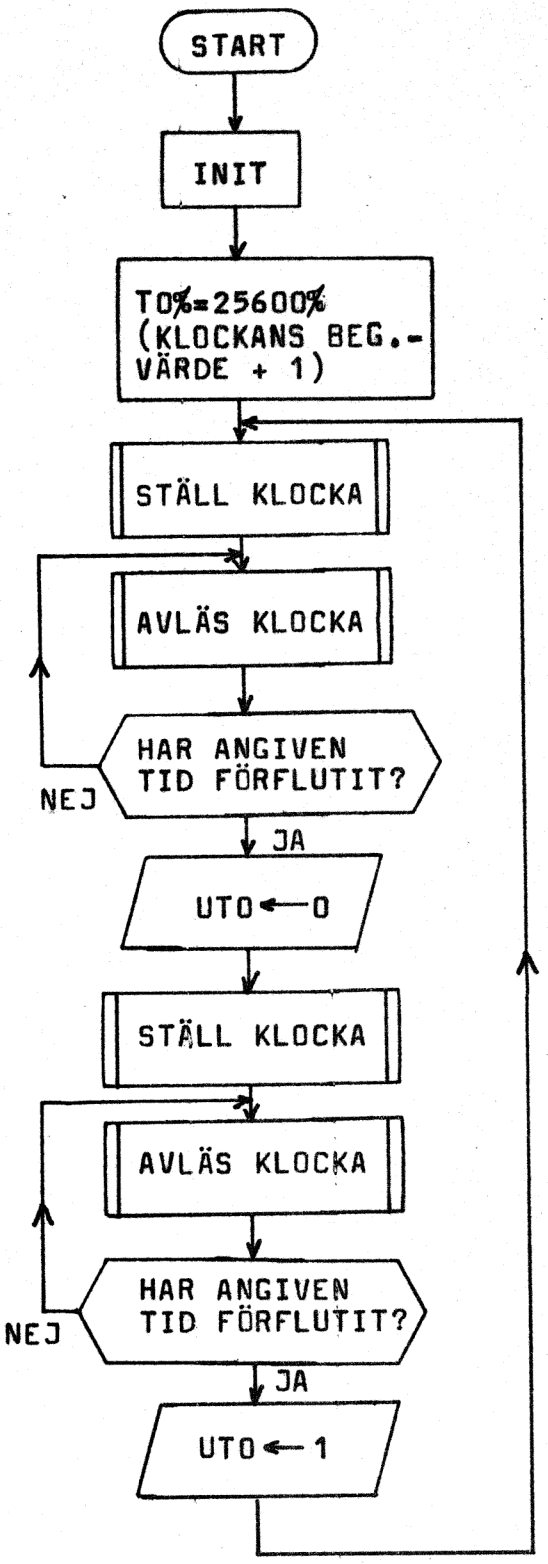



```
10 REM UPPG D1 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ********************************
60 OUT 0,0
70 FOR N=0 TO 4000 : NEXT N
80 OUT 0,1
90 FOR N=0 TO 2000 : NEXT N
100 GOTO 60
```

Vi ser att datorn blir låst i två olika loopar. Jämför lösning C11A. Eftersom datorn kan användas till bättre uppgifter än att "springa runt" i en loop bör vi undvika sådana loopar i de flesta fall.

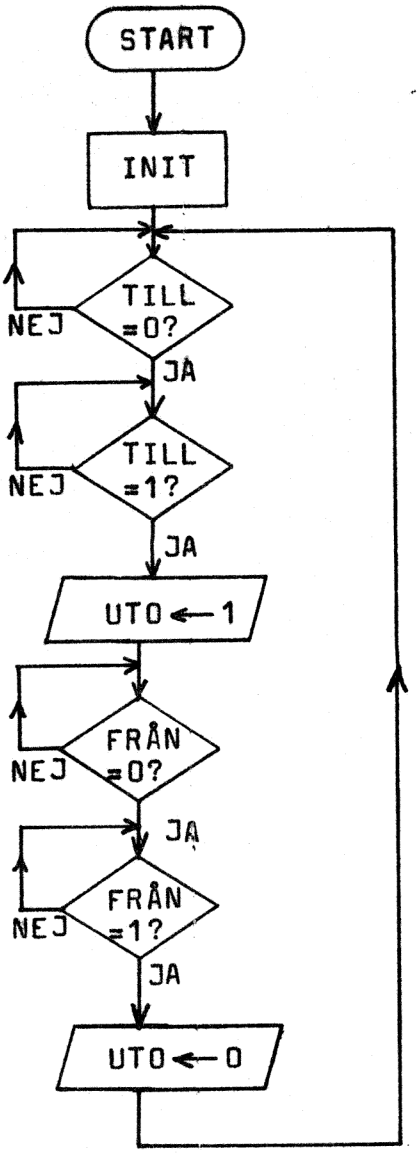
D2
I detta fall använder vi realtidsklockan i ABC80. Ett register om 24 bitar (3 bytes i minnet) räknas ned var 20:e millisekund. Angående nedräkningen, se ref 5 sid 124.
Om vi använder byte 1 och 2 och i dessa laddar in 25599 tar det över 8 minuter innan båda är nedräknade till 0. 8 minuter räcker för många av våra tillämpningar.
Vi ställer klockan genom att ladda in 25599 i de två byten. Vi avläser klockan genom att avläsa innehållet i de två byten i minnet. Genom att beräkna hur långt nedräkningen har kommit kan vi se om den tid vi angivit i programmet har förflutit.
Programmet för tidfördröjningen ger en tid som är 40 ms kortare än angivet värde.
Programdelarna STÄLL KLOCKA och AVLÄS KLOCKA är skrivna som subrutiner så att de lätt kan användas i andra program.

```
10 REM UPPG D2 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
55 REM SEKVENS **************************
60 T0%=25600%
70 GOSUB 800
80 GOSUB 900
90 IF T0%-100%<T1% THEN 80
100 OUT 0,0
110 GOSUB 800
120 GOSUB 900
130 IF T0%-200%<T1% THEN 120
140 OUT 0,1
150 GOTO 70
780 REM ++++++++++++++++++++++++++++++++
790 REM STÄLL KLOCKA ********************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLÄS KLOCKA ********************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```



Även i ovanstående flödesschema finns loopar. Det är dock ingenting som hindrar att vi lägger in andra programdelar i looparna. Det medför då att datorn mellan avläsningarna av klockan exekverar dessa programdelar.

E1
Lösningen är gjord med
UTO bit 0 som L
IN0 bit 0 som TILL
IN1 bit 0 som FRÅN.




```
10 REM UPPG E1 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 IF (INP(0%) AND 1%)=0% THEN 70 ELSE 60
70 IF (INP(0%) AND 1%)=1% THEN 80 ELSE 70
80 OUT 0%,1%
90 IF (INP(1%) AND 1%)=0% THEN 100 ELSE 90
100 IF (INP(1%) AND 1%)=1% THEN 110 ELSE 100
110 OUT 0%,0%
120 GOTO 60
```

Med denna lösning låser vi datorn i en loop, jämför C11A.
Dessutom kan datorn bara behandla en lampa.

E2
Vi använder UTO för L
INO för TILL och
IN1 för FRÅN och parar ihop switchar med lampa bitvis.
I detta fall använder vi samma metod som i C11B.
Beteckningar:
B0%= GAMLA VÄRDEN för TILL
A0%= NYA VÄRDEN för TILL
B1%= GAMLA VÄRDEN för FRÅN
A1%= NYA VÄRDEN för FRÅN
X1%=ÄNDRINGAR till 1 för TILL
Y1%=ÄNDRINGAR till 1 för FRÅN
Z%= lamptillstånd L

Med nedanstående lösningsprincip kan lampor styras via flera kort.

```
10 REM UPPG E2 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 B0%=INP(0%) : REM BEGYNNELSEVÄRDEN
70 B1%=INP(1%)
75 REM **********************************
80 A0%=INP(0%)
90 A1%=INP(1%)
100 X%=A0% XOR B0% : REM ÄNDRINGAR
110 X1%=X% AND A0% : REM ÄNDRINGAR TILL 1
120 Y%=A1% XOR B1% : REM ÄNDRINGAR
130 Y1%=Y% AND A1% : REM ÄNDRINGAR TILL 1
140 Z%=Z% OR X1%
150 Z%=Z% AND NOT Y1%
160 OUT 0%,Z%
170 B0%=A0% : REM GAMLA
180 B1%=A1% : REM GAMLA
190 GOTO 80
```

```
10 REM UPPG E2KOLL DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 B0%=INP(0%) : REM BEGYNNELSEVÄRDEN
70 B1%=INP(1%)
72 ; "A0";"B0";"X1","A1";"B1";"Y1"
75 REM **********************************
80 A0%=INP(0%)
90 A1%=INP(1%)
100 X%=A0% XOR B0% : REM ÄNDRINGAR
110 X1%=X% AND A0% : REM ÄNDRINGAR TILL 1
120 Y%=A1% XOR B1% : REM ÄNDRINGAR
130 Y1%=Y% AND A1% : REM ÄNDRINGAR TILL 1
140 Z%=Z% OR X1%
150 Z%=Z% AND NOT Y1%
160 OUT 0%,Z%
165 ; A0%;B0%;X1%,A1%;B1%;Y1%
170 B0%=A0% : REM GAMLA
180 B1%=A1% : REM GAMLA
186 GET A$
190 GOTO 80
```

Av C11B framgår hur vi får 1 i de bitpositioner som ändrats till 1. För att kunna handräkna och komma fram till lämplig algoritm antar vi att

Z%= 1001 0001
X1%=0000 0111
Y1%=0001 0011

Vi ser att lampor i pos 0, 1 och 2 skall tändas utan att övriga påverkas.

Med Z% = Z% OR X1% fås

Z%	1001 0001
X1%	0000 0111
Z% efter OR	1001 0111

Sedan skall vi släcka lampor i pos 0, 1 och 4, dvs där Y1% är 1.

Invertera först Y1% och tag sedan OCH med Z%.

NOT Y1%	1110 1100
Z%	1001 0111
Z% efter AND	1000 0100

Vi ser att lampan i pos 7 förblir ettställd, lampan i pos 4 släcks, lampan i pos 1 förblir släckt samt lampan i pos 0 släcks. Lampan i pos 2 tänds.
Om TILL och FRÅN ettställs under samma loop kommer i detta fall FRÅN att dominera, se pos 0 och 1.
Vill vi att TILL skall dominera får vi kasta om raderna 140 och 150.
För att kunna rita flödesschema behöver vi först lämpliga algoritmer, dvs vi måste ta reda på hur vi skall gå tillväga för att uppnå ett visst resultat.
För att få fram algoritmer får man som ovan anta vissa värden på instorheter och handräkna.
Man kan sedan vid testkörning printa vissa storheter på skärmen och dra ned beräkningshastigheten, se E2KOLL rad 72, 165 och 186.
I E2KOLL körs programmet en gång varje gång vi trycker på en tangent på tangentbordet.

E3
Vi delar in programmet i PROGRAMMERING AV SEKVENS och KÖRNING samt utnyttjar subrutinerna för klockan från uppg 2.
För sekvensen i uppgiften skall vi ha följande tabell.

	TID I S
SLÄCKT	2
TÄND	3
SLÄCKT	1
TÄND	2
SLÄCKT	3
TÄND	1
OMIGEN	0

Sista raden visar att vi skall börja om från början igen. 0 utgör alltså slutmarkering.

Kommentarer till programmet:

Radnr	Kommentar
60	Klockans startvärde. DIM ger max 26 faser.
80-100	Ta in tid för släckt lampa i tabell.
105	N% för nästa fas.
110	Var sista inläsningen en slutmarkering?
120- 140	Ta in tid för tänd lampa i tabell.
145	N% för nästa fas.
150	Börja om inläsningen igen.
160	Fas 0.
170	Släck lampa.
180	Ställ klocka.
190	Avläs klocka.
200	Har angiven tid förflutit? Om ja fortsätt.
210	Tänd lampa.
220	N% för nästa fas.
230-250	Jämför 180-200.
260	N% för nästa fas.

Radnr	Kommentar
270	Är det slutmarkering? Om ja, börja om från början i tabellen.

```
10 REM UPPG E3 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++
55 REM PROGRAMMERING AV SEKVENS ******
60 T0%=25600% : DIM T%(25%)
70 ; CHR$(12)
80 ; "LAMPA SLÄCKT, GE TID I SEK "
90 ; "OM SEKVENS SLUT, GE 0"
95 INPUT T : ; : ;
100 T%(N%)=T*50%
105 N%=N%+1%
110 IF T=0 THEN 160 ELSE 120
120 ; " LAMPA TÄND, GE TID I SEK"
130 INPUT T : ; : ;
140 T%(N%)=T*50%
145 N%=N%+1%
150 GOTO 70
154 REM ++++++++++++++++++++++++++++++
155 REM KÖRNING **********************
160 N%=0%
170 OUT 0%,0%
180 GOSUB 800
190 GOSUB 900
200 IF T0%-T%(N%)<T1% THEN 190
210 OUT 0%,1%
220 N%=N%+1%
230 GOSUB 800
240 GOSUB 900
250 IF T0%-T%(N%)<T1% THEN 240
260 N%=N%+1%
270 IF T%(N%)>0% THEN 170 ELSE 160
300 REM ++++++++++++++++++++++++++++++
790 REM STÄLL KLOCKA *****************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLÄS KLOCKA *****************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```

E4
I detta fall lägger vi tabellen direkt i minnet med POKE. För att kunna fortsätta en avbruten körning lägger vi adressen till den fas som kör överst i tabellen.

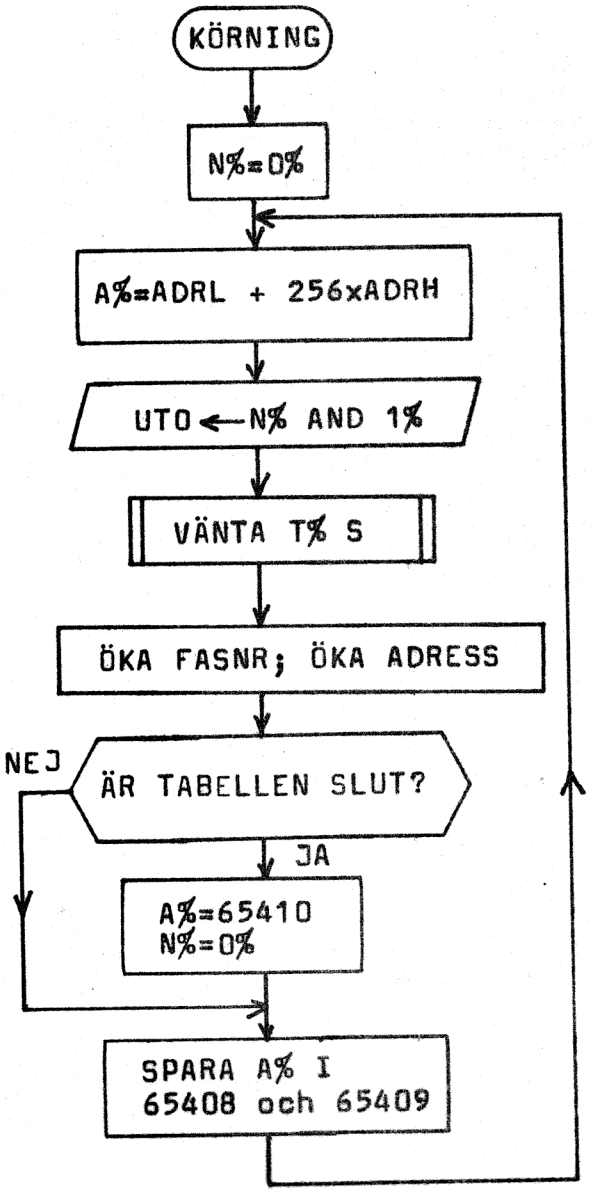
ADRESS	TABELL
65408	ADRL
65409	ADRH
65410	2
65411	3
65412	1
65413	2
65414	3
65415	1
65416	255

I detta fall utgör 255 slutmarkering. ADRH,ADRL innehåller adressen till den fas som för tillfället exekveras.

Programmet består av
INITIERING
MENYVAL
PROGRAMMERING AV SEKVENS
KÖRNING samt
subrutinerna för klockan.

Kommentarer till programmet:

Radnr	Kommentarer
90-120	Menyval.
130	Kontroll av menyval.
140	Hopp till programdel enligt menyval.
170	Fas 0 med adress.
180	Spara adress till fas 0.
200-230	Ta in tid för släckt lampa.
240	Spara tid i minnet.
250	Om det var slutmarkering, hoppa till körning.
260	Öka fasnr och adress.
270-290	Ta in tid för tänd lampa.
300	Spara tid i minnet.
310	Öka fasnr och adress.
350	Klockans startvärde.
360	Börja med fas 0.
370	Ta fram adress till fasen.
380	Släck lampa för jämna fasnr och tänd lampa för udda fasnr.
390	Ställ klocka.
400	Ta fram tid och beräkna antal steg för klockan.
410-420	Har tiden förflutit? Om ja, fortsätt.
430	Öka fasnr och adress.
440	Slutmarkering? Om ja 450, annars 460.
450	Fas 0 med adress.
460	Spara adress.
470	Hoppa till 370.



E4 forts.
Om vi avbryter körning med CTRL-C och sedan börjar direkt med KÖRNING igen kommer sista fasen att upprepas.

```
10 REM UPPG E4 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++
60 REM MENYVAL **********************
80 ; CHR$(12)
90 ; "VÄLJ MELLAN" : ; : ;
100 ; "1  PROGRAMMERING AV SEKVENS" : ;
110 ; "2  KÖRNING" : ; : ;
120 INPUT V
130 IF V<1 OR V>2 THEN 90
140 ON V GOTO 150,330
150 REM ++++++++++++++++++++++++++++++
160 REM PROGRAMMERING ****************
170 N%=0% : A%=65410%
180 POKE 65408%,A%,SWAP%(A%)
190 ; CHR$(12)
200 ; "FAS NR";N% : ;
210 ; "ANGE FASSLUT MED 255" : ; : ;
220 ; "GE TID I SEK FÖR SLÄCKT LAMPA"
230 INPUT T%
240 POKE A%,T%
250 IF T%=255% THEN 330
260 N%=N%+1% : A%=A%+1% : ; : ; : ;
270 ; "FAS NR";N% : ;
280 ; "GE TID I SEK FÖR TÄND LAMPA"
290 INPUT T%
300 POKE A%,T%
310 N%=N%+1% : A%=A%+1%
320 GOTO 190
330 REM ++++++++++++++++++++++++++++++
340 REM KÖRNING **********************
350 T0%=25600%
360 N%=0%
370 A%=PEEK(65408%)+256%*PEEK(65409%)
380 OUT 0%,N% AND 1%
390 GOSUB 800
400 T%=50%*PEEK(A%)
410 GOSUB 900
420 IF T0%-T%<T1% THEN 410
430 N%=N%+1% : A%=A%+1%
440 IF PEEK(A%)=255% THEN 450 ELSE 460
450 A%=65410% : N%=0%
460 POKE 65408%,A%,SWAP%(A%)
470 GOTO 370
790 REM STÄLL KLOCKA *****************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLÄS KLOCKA *****************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```

E5
I detta exempel lägger vi koderna för utgångarna i en tabell, A%(N%), och tiderna T%(N%) i en tabell. N% anger fasnr.
Lösningen till denna uppgift påminner om lösningen till uppg E3.

```
10 REM UPPG E5 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++
60 REM INMATNING AV SEKVENS **********
70 T0%=25600% : N%=0%
80 DIM A%(25%),T%(25%)
90 ; CHR$(12)
100 ; "GE KOD FÖR UTSIGNALER"
110 ; "OM SEKVENS SLUT, GE 256"
120 INPUT A%(N%) : ; : ; : ;
130 IF A%(N%)=256% THEN 300 ELSE 140
140 ; "GE TID I SEK"
150 INPUT T : T%(N%)=T*50%
160 N%=N%+1%
170 GOTO 90
200 REM ++++++++++++++++++++++++++++++
300 REM KÖRNING **********************
310 N%=0%
320 OUT 0%,A%(N%)
330 GOSUB 800
340 GOSUB 900
350 IF T0%-T%(N%)<T1% THEN 340
360 N%=N%+1%
370 IF A%(N%)=256% THEN 310 ELSE 320
400 REM ++++++++++++++++++++++++++++++
790 REM STÄLL KLOCKA *****************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLÄS KLOCKA *****************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```

E6
Vid programmeringen av en sekvens lägger vi in en tabell i minnet, se uppg E4. Överst i tabellen lägger vi adressen till den fas som "körs".
Därefter kommer i varannan minnescell KOD FÖR UTSIGNALER och i varannan TID.

TABELLUPPLÄGGNING

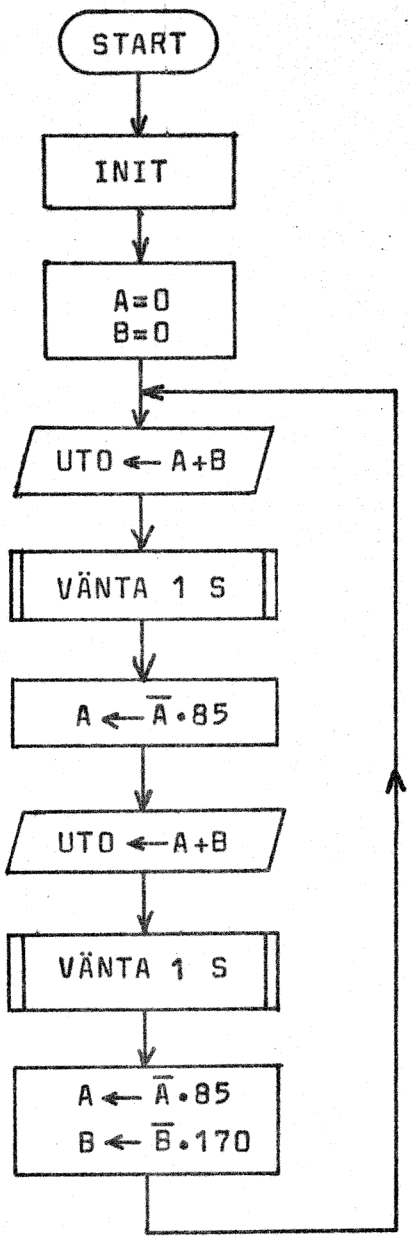
Fasnr	Adress	Innehåll
	65408	ADRL
	65409	ADRH
0	65410	KOD0
	65411	TID0
1	65412	KOD1
	65413	TID1
2	65414	KOD2
	.	.
	.	.
OMIGEN		255

Som slutmarkering används 255. Det betyder att vi inte kan ha alla lamporna lysande samtidigt.
Annars får vi hitta på någon annan kod som slutmarkering.

```
10 REM UPPG E6 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++
60 REM MENYVAL **************************
70 ; CHR$(12)
80 ; "VÄLJ MELLAN" : ; : ;
90 ; "1  PROGRAMMERING AV SEKVENS" : ;
100 ; "2  KÖRNING" : ; : ;
110 INPUT V
120 IF V<1 OR V>2 THEN 80
130 ON V GOTO 150,320
140 REM ++++++++++++++++++++++++++++++
150 REM PROGRAMMERING ********************
160 T0%=25600% : N%=0%
170 A%=65410%
180 POKE 65408%,A%,SWAP%(A%)
190 ; CHR$(12)
200 ; CUR(2,20);"FAS NR";N% : ; : ; : ;
210 ; "GE KOD FÖR UTSIGNALER"
220 ; "OM SEKVENS SLUT, GE 255"
230 INPUT K% : ; : ; : ;
240 POKE A%,K% : A%=A%+1%
250 IF K%=255% THEN 320
260 ; "GE TID I SEK"
270 INPUT T : T%=T*50%
280 POKE A%,T% : A%=A%+1%
290 N%=N%+1%
300 GOTO 190
310 REM ++++++++++++++++++++++++++++++
320 REM KÖRNING **************************
325 T0%=25600%
330 A%=PEEK(65408%)+256%*PEEK(65409%)
340 OUT 0%,PEEK(A%)
350 A%=A%+1%
360 T%=PEEK(A%)
370 GOSUB 800
380 GOSUB 900
390 IF T0%-T%<T1% THEN 380
400 A%=A%+1%
410 IF PEEK(A%)=255% THEN A%=65410%
420 POKE 65408%,A%,SWAP%(A%)
430 GOTO 340
440 REM ++++++++++++++++++++++++++++++
790 REM STÄLL KLOCKA **********************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLÄS KLOCKA **********************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```

E7
Först observerar vi att bit 0,2,4 och 6 har vikten 85. Med 85 kan vi maska bort övriga bitar.
Bit 1,3,5 och 7 har vikten 170. Med 170 kan vi maska bort övriga bitar.
Låt A vara tillståndet för jämna bitnummer och B tillståndet för udda bitnummer. Variablerna skrivs här utan %-tecknet.
Genom att invertera A varje sekund och B varannan sekund får vi fram det önskade mönstret.

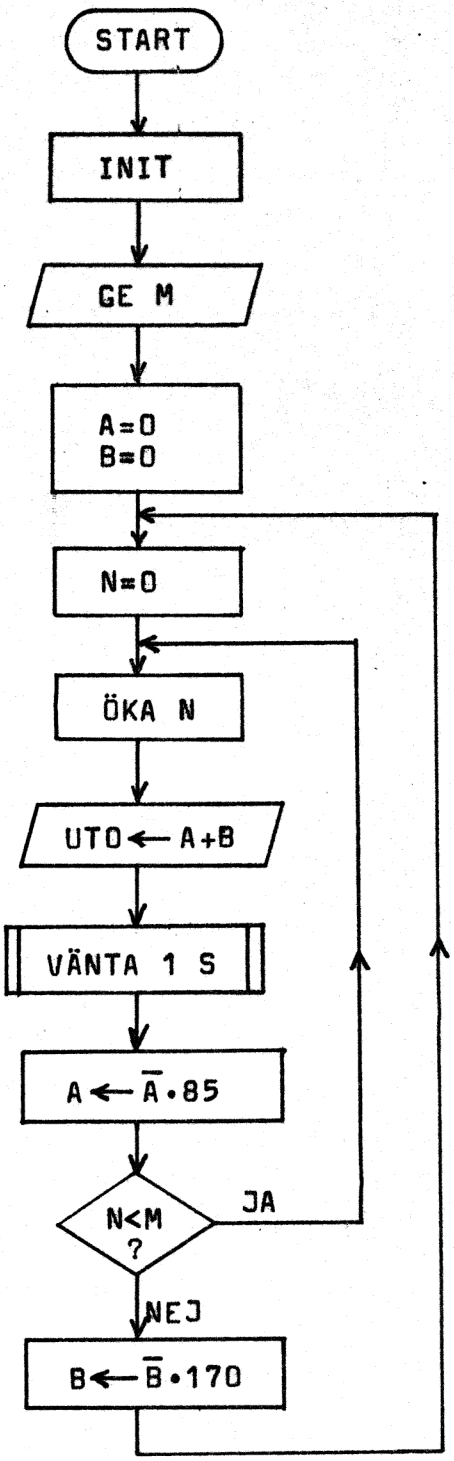
```
10 REM UPPG E7 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTALL UTGANGAR
50 REM ++++++++++++++++++++++++++++++++
55 REM ********************************
60 T0%=25600%
70 A%=0% : B%=0%
75 REM SEKVENS ***********************
80 OUT 0%,A% OR B%
90 GOSUB 300
100 A%=( NOT A%) AND 85%
110 OUT 0%,A% OR B%
120 GOSUB 300
130 A%=( NOT A%) AND 85%
140 B%=( NOT B%) AND 170%
150 GOTO 80
160 REM VANTA ************************
300 GOSUB 800
310 GOSUB 900
320 IF T0%-50%<T1% THEN 310
330 RETURN
400 REM ++++++++++++++++++++++++++++++++
790 REM STALL KLOCKA *****************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLAS KLOCKA *****************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ANDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```



Subrutinerna för VÄNTA tar vi från uppg D2.

E8
Frekvensdelningen M
ges i början av programmet.
Vi modifierar uppg E7.
Varje gång A inverteras
räknar vi upp N ett steg.
Då N=M inverteras även B.
N är en varvräknare.
I flödesschemat skrivs
variablerna utan %-tecken.

```
10 REM UPPG E8 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 REM ********************************
70 T0%=25600%
80 A%=0% : B%=0%
90 ; CHR$(12)
100 ; "GE FREKVENSDELNING 1:M "
110 INPUT M%
120 REM SEKVENS ************************
130 N%=0%
140 N%=N%+1%
150 OUT 0%,A% OR B%
160 GOSUB 250
170 GOSUB 280
180 IF T0%-50%<T1% THEN 170
190 A%=( NOT A%) AND 85%
200 IF N%<M% THEN 140
210 B%=( NOT B%) AND 170%
220 GOTO 130
230 REM ++++++++++++++++++++++++++++++++
240 REM STÄLL KLOCKA ********************
250 POKE 65008%,255%,99%
260 RETURN
270 REM AVLÄS KLOCKA ********************
280 K0%=(PEEK(65008%)-1%) AND 255%
290 K1%=PEEK(65009%) AND 255%
300 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
310 IF PEEK(65008%)<>K0% THEN 280
320 T1%=K1%*256%+K0%
330 RETURN
```



E9
Beteckningar.
%-tecknet utelämnas i beskrivningen.
M=4 ger frekvensdelning.
A visar vilka fönster som öppnats före sista kvittensen på T2. Vi har alltså meddelat datorn att vi vet om att dessa är öppna. A visas på UT0 med blinkmask M0.
B visar vilka fönster som öppnats efter sista kvittensen på T2. Vi vet alltså inte om att dessa öppnats. B visas på UT2 med blinkmask M2.
X visar fönstrens "momentana" tillstånd.
B0 visar fönstrens tillstånd då T2 nedtrycks.

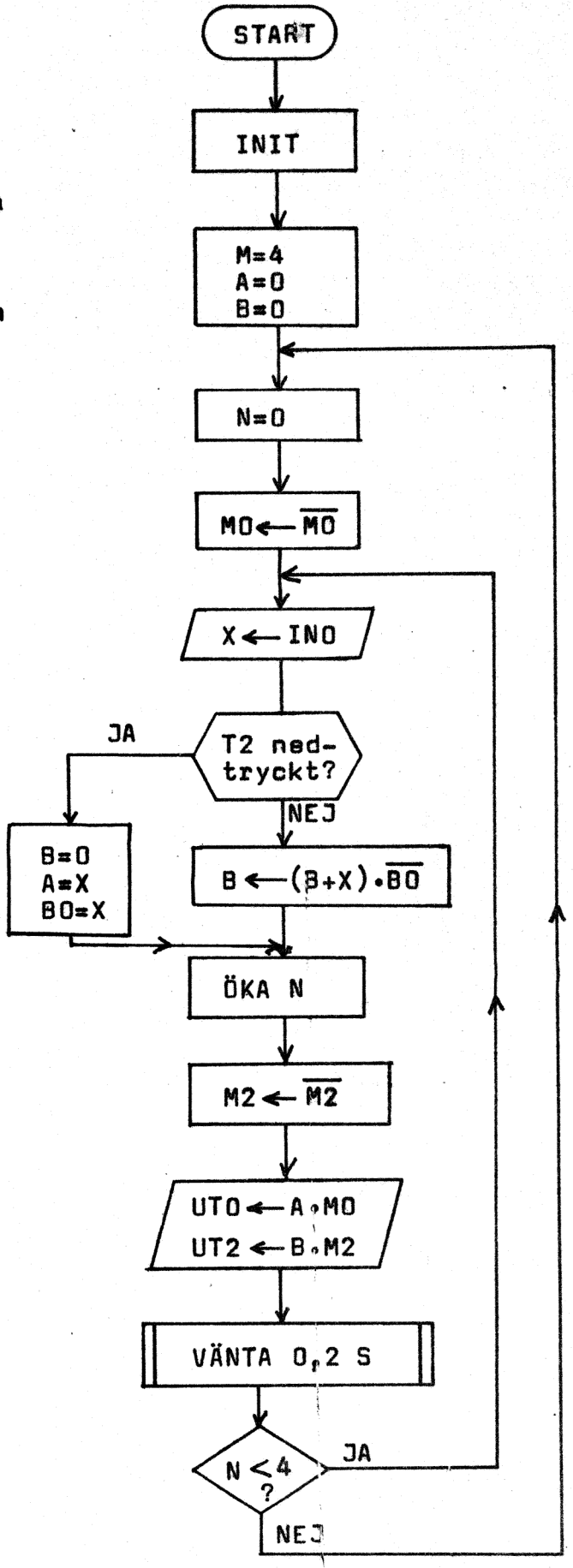
Beräkningen av B tillgår på följande sätt.
Med B+X ettställs de positioner för vilka ett fönster öppnats. Observera att vi även får med de fall då ett fönster öppnats ett kort ögonblick.
Vi skall dock maska bort de fönster som stått öppna sedan före sista kvittensen. Detta gör vi genom att ta

logiskt OCH med $\overline{B0}$.

Således $B \leftarrow (B+X) \cdot \overline{B0}$.

Ovan och i flödesschemat avser
+ logiskt ELLER och
· logiskt OCH.

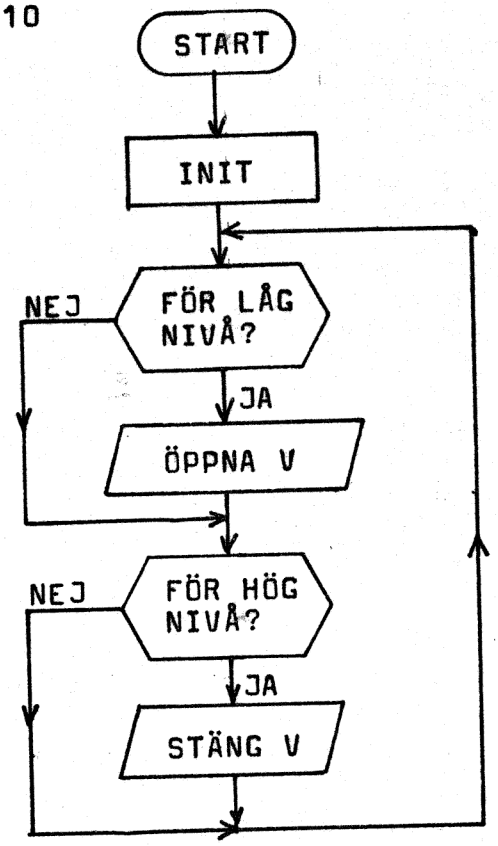
När det gäller blinkningen är uppg E8 en lämplig förövning.



E9 forts

```
10 REM UPPG E9 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 T0%=25600% : M%=4%
70 A%=0% : B%=0%
80 N%=0%
90 M0%= NOT M0%
100 X%=INP(0%)
110 K%=INP(1%) AND 128%
120 IF K%=128% THEN 200
130 B%=(B% OR X%) AND NOT B0%
140 GOTO 300
200 B%=0%
210 A%=X%
220 B0%=X%
300 N%=N%+1%
310 M2%= NOT M2%
320 OUT 0%,A% AND M0%
330 OUT 2%,B% AND M2%
340 GOSUB 800
350 GOSUB 900
360 IF T0%-10%<T1% THEN 350
370 IF N%<4% THEN 100 ELSE 80
790 REM STÄLL KLOCKA ******************
800 POKE 65008%,255%,99%
810 RETURN
890 REM AVLÄS KLOCKA ******************
900 K0%=(PEEK(65008%)-1%) AND 255%
910 K1%=PEEK(65009%) AND 255%
920 REM KONTROLLERA ATT INTE (65008) ÄNDRATS
930 IF PEEK(65008%)<>K0% THEN 900
940 T1%=K1%*256%+K0%
950 RETURN
```

E10




```
10 REM UPPG E10 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 IF (INP(0) AND 1)=0 THEN 80
70 OUT 0,1
80 IF (INP(1) AND 1)=0 THEN 100
90 OUT 0,0
100 GOTO 60
```

E11
Nu skall vi klara av 8 st nivåregleringar samtidigt. Vi kan ta en nivåreglering i taget och upprepa programmet i uppg E10 8 gånger. Med FOR-NEXT behöver vi inte skriva raderna mer än en gång. Vi tittar på en annan lösning nedan där alla positioner beräknas samtidigt.

Beteckningar:
V1= GAMLA VENTILTILLSTÅND
V0= NYA VENTILTILLSTÅND
L= FÖR LÅG NIVÅ
H= FÖR HÖG NIVÅ

För att komma underfund om hur nya ventiltillstånd skall beräknas tittar vi på ett exempel.

Antag

L = 0000 0011

H = 1100 0000

V1= 0111 0001

Ventiler 0, 4, 5 och 6 är öppna.
För LÅG nivå har nr 0 och 1.
För HÖG nivå har nr 6 och 7.
En nivå mellan LÅG och HÖG har nr 2, 3, 4 och 5. Nivån för nr 2 och 3 sjunker. Nivån för nr 4 och 5 stiger.
Sedan förra beräkningen har nivån för nr 6 blivit för HÖG, och nivån för nr 1 för LÅG.
De positioner som har för låg nivå skall ettställas.
Beräkna V1+L:

V1	0111 0001
L	0000 0011
V1+L	0111 0011

Nollställ sedan de positioner som har för hög nivå.

V1+L	0111 0011
$\overline{H}$	0011 1111
$(V1+L)\cdot\overline{H}$	0011 0011

Ventiler nr 0, 1, 4 och 5 blir öppna vilket stämmer.
Ovan betyder
+ logiskt ELLER och
• logiskt OCH.

Med denna lösningsmetod är det möjligt att ansluta ventiler via flera kort. Beräkningarna göres för ett kort i taget.

```
10 REM UPPG E11 DIGITALKORT
20 REM MIKRONIK SUNDSVALL
30 OUT 1,19 : REM KORTADRESS 19
40 OUT 0,0,2,0 : REM NOLLSTÄLL UTGÅNGAR
50 REM ++++++++++++++++++++++++++++++++
60 V1=0 : REM GAMLA VÄRDEN
70 L=INP(0)
80 H=INP(1)
90 V0=(V1 OR L) AND NOT H
100 OUT 0,V0
105 OUT 2,L AND H : REM INDIKERAR FEL
110 V1=V0
120 GOTO 70
```

Eftersom en nivå inte kan vara både för HÖG och för LÅG samtidigt kan man som på rad 105 ovan ha felindikering om detta skulle inträffa på grund av något fel.

6. ASSEMBLERPROGRAMMERING

INNEHÅLL

		sid
6.1	Allmän datormodell	6.1
6.2	Orientering om Z80 CPU	6.3
6.3	In- och utinstruktioner	6.6
6.4	Hur skriver vi program?	6.6
6.5	Hur får vi in programmet i minnet?	6.7
6.6	Sammanställning av ovan nämnda instruktioner	6.8
6.7	Dataöverföring av 8 bitar	6.9
6.8	Adresseringsmetoder. I	6.10
6.9	Dataöverföring av 16 bitar	6.12
6.10	Att ange operander	6.15
6.11	Flaggor	6.16
6.12	Aritmetiska instruktioner för 8 bitar	6.17
6.13	Logiska instruktioner	6.18
6.14	Specialinstruktioner för ackumulator och carry-flagga	6.19
6.15	Instruktioner för rotation och skift	6.20
6.16	Aritmetiska instruktioner för 16 bitar	6.21
6.17	Hopp	6.22
6.18	Subrutiner	6.25
6.19	Instruktioner för bitmanipulering	6.29
6.20	Indexregister	6.29
6.21	Adresseringsmetoder. Översikt	6.30
6.22	Övriga instruktioner	6.31

Denna punkt riktar sig främst till de som vill komplettera program i BASIC med rutiner i assembler. Då snabba yttre enheter skall styras kan program i BASIC vara för långsamma.
Denna punkt kompletteras med övningsuppgifter med lösningar i punkt 7. och 8. .

6. ASSEMBLERPROGRAMMERING

Under denna punkt tar vi upp grunderna för assemblerprogrammering.

6.1 Allmän datormodell

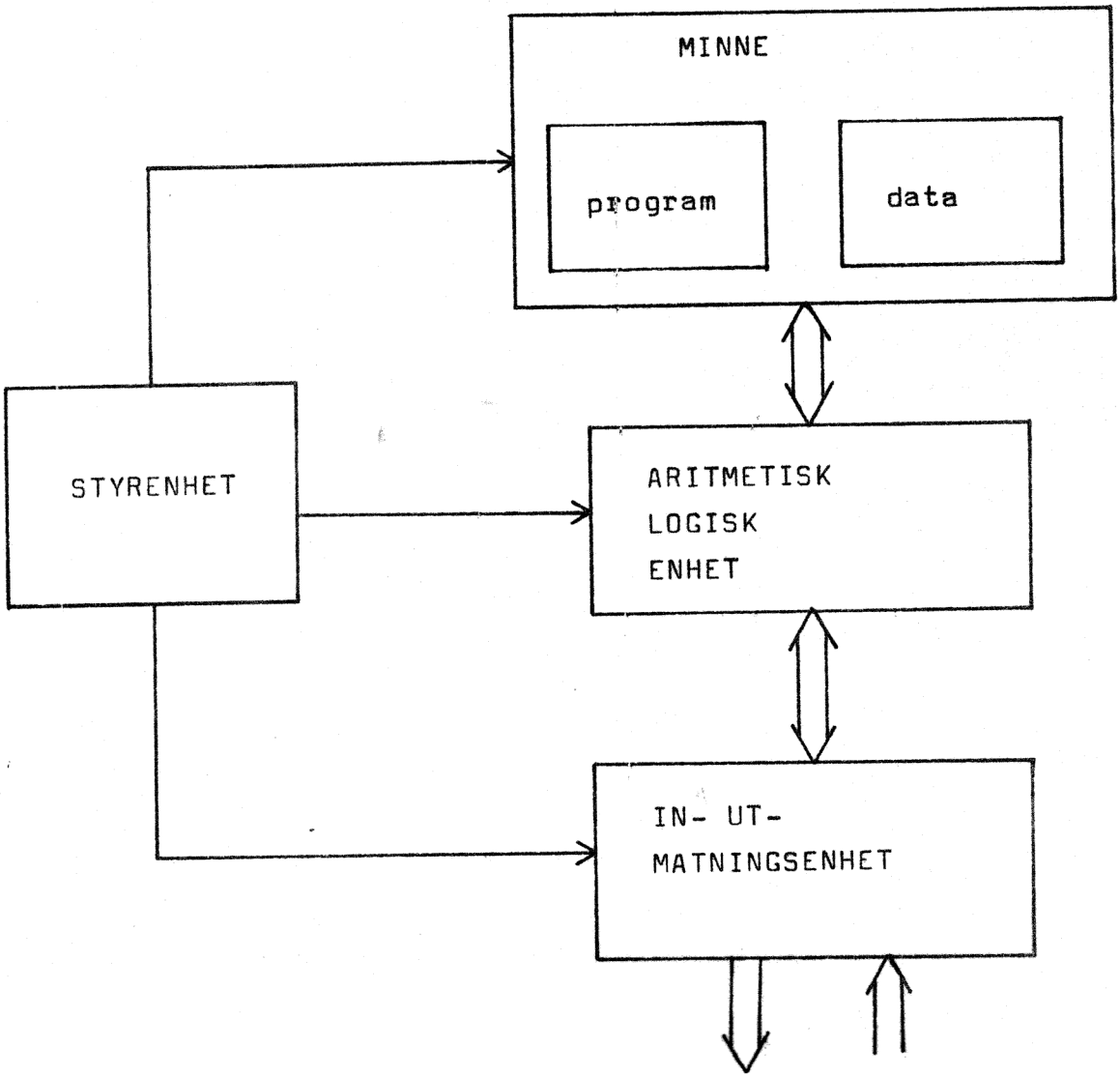
En dator skall kunna

1. Mottaga information
2. Lagra information
3. Behandla information
4. Presentera resultatet av den behandlade informationen

De fyra kraven uppfylles av en vanlig kalkylator. En dator kännetecknas av en femte punkt.

5. Datorn delges hur och när punkterna 1-4 skall utföras genom ett program, som lagras i datorns minne.

På 1940-talet angav John von Neuman de byggblock som fordras för att uppfylla de fem kraven.



I minnet lagras information i form av instruktioner och data. Aritmetisk-logiska enheten utför alla beräkningar. Till denna hör ett eller flera register som lagrar resultaten från operationerna. Ett sådant register kallas ackumulator. In-utmatningsenheten sköter kommunikationen med yttre enheter. Styrenheten styr informationsflödet.

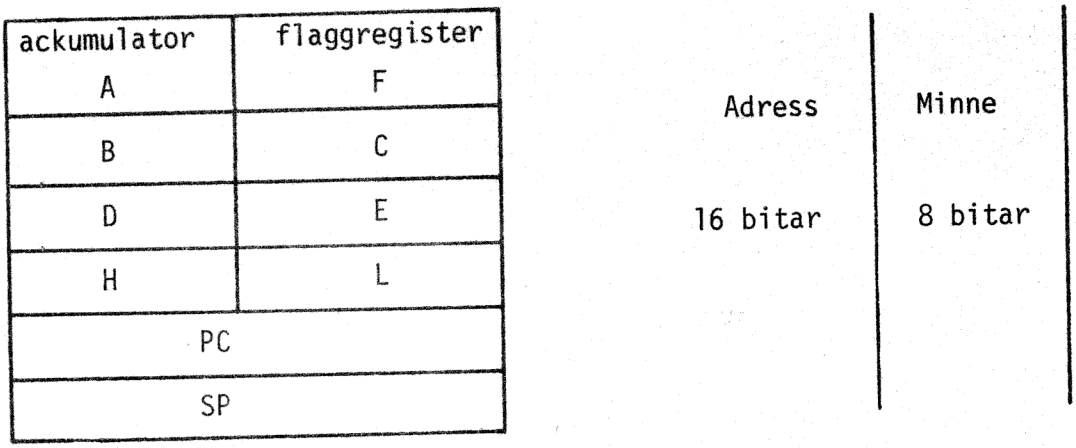
Programmet består av ett antal instruktioner som ligger efter varandra i minnet. Styrenheten hämtar nästa instruktion och avkodar denna. Därefter skickas styrorder ut så att instruktionen utförs på avsett sätt. Så länge ingen hoppinstruktion förekommer avverkas instruktionerna efter varandra.

En beskrivning av hur ABC80 är uppbyggd samt hur en instruktion utförs i Z80 finns i ref 3.

6.2 Orientering om Z80 CPU.

I denna punkt tar vi upp de delar av Z80 som vi behöver känna till för att skriva enkla program.

Z80 CPU



Registren A, F, B, C, D, E, H, L innehåller 8 bitar.

Ackumulator A:
A hör till ALU:n. Resultaten av aritmetiska och logiska operationer hamnar i A.

Flaggregister F :
Flaggorna indikerar resultatet av vissa operationer.
Ofta är vi intresserade av en enda flagga som finns i en vippa i flaggregistret.

6 allmänna register:
Registren B, C, D, E, H, L kan vi använda som 6 st 8-bitars register för t ex mellanlagring av data eller varvräknare.
Det är lättare att adressera ett register än en minnescell.
Registren kan även användas som 3 st 16-bitars registerpar BC, DE och HL. De används för operationer på 16-bitars data och som minnespekare. 16 bitar kan vi dela in i 2 bytes.
Ex: 5AC3H

5A	C3	
hög byte hög adress	låg byte låg adress	eller

I registerparen innehåller B, D, H hög byte och C, E, L låg byte.

Programräknare PC :
Programräknaren innehåller adressen till nästa byte som skall hämtas i programminnet. Efter varje hämtning av en byte ökas PC med 1. Därför utförs intruktionerna efter varandra såsom de ligger i minnet. Vill vi hoppa till en annan programdel i minnet måste alltså PC laddas med startadressen till denna programdel.

Stackpekare SP :
SP innehåller adressen till den del av minnet som kallas för stacken. På stacken lagras återhoppsadressen till huvudprogrammet då subrutiner anropas. Man kan även lagra data på stacken.

Minne:
I minnet lagras 8 bitar på varje adress.
Adresserna anges med 16 bitar.

Instruktionsformat:
Instruktionerna består av 1-4 bytes för Z80.
Med 8 bitar får vi 256 kombinationer. Intel 8080 använder 244 av 256 för operationskoder (OPKOD). Dessa 244 OPKODER har sina identiska motsvarigheter i Z80. För Z80 används även de 12 återstående kombinationerna för OPKODER. För att öka antalet instruktioner används i några fall en andra byte för OPKODEN. För Z80 finns nästan 700 olika instruktionskoder.

Nedan visas hur instruktionerna kan vara uppbyggda.

a) Instruktion omfattande 1 byte

	byte 1	OPKOD

b) Instruktion omfattande 2 bytes

Typ 1:	byte 1	OPKOD
	byte 2	DATA
Typ 2:	byte 1	OPKOD
	byte 2	I/O-ADRESS
Typ 3:	byte 1	OPKOD
	byte 2	OPKOD
Typ 4:	byte 1	OPKOD
	byte 2	RELATIVADRESS

c) Instruktion omfattande 3 bytes

Typ 1:	byte 1	OPKOD
	byte 2	LÅG ADRESS
	byte 3	HÖG ADRESS
Typ 2:	byte 1	OPKOD
	byte 2	DATA
	byte 3	DATA
Typ 3:	byte 1	OPKOD
	byte 2	OPKOD
	byte 3	DATA
Typ 4:	byte 1	OPKOD
	byte 2	OPKOD
	byte 3	INDEX

d) Instruktion omfattande 4 bytes

Typ 1:	byte 1	OPKOD
	byte 2	OPKOD
	byte 3	LÅG ADRESS
	byte 4	HÖG ADRESS
Typ 2:	byte 1	OPKOD
	byte 2	OPKOD
	byte 3	INDEX
	byte 4	DATA
Typ 3:	byte 1	OPKOD
	byte 2	OPKOD
	byte 3	INDEX
	byte 4	OPKOD

Vilka grupper av instruktioner har Z80?

För att få en allmän uppfattning vad vi kan göra tittar vi i ref (6) sid 23-37, speciellt på de tabeller som finns.

a) Dataöverföring
Data, 8 bitar eller 16 bitar, kan flyttas inom CPU:n eller mellan CPU och minne.
I BASIC överför vi data i t ex
```
80 A=17
90 B=A
```

b) Aritmetiska och logiska funktioner
Det finns aritmetiska för 8 bitar och 16 bitar. De logiska opererar på 8 bitar.
Vi ser att instruktioner för multiplikation och division saknas.

c) Instruktioner för rotation och skift
Dessa ger möjlighet att operera med enskilda bitar. De används även för program som utför multiplikation och division.

d) Instruktioner för bitmanipulering
Med dessa kan man operera på enskilda bitar.

e) Hoppinstruktioner
Med dessa kan hopp i programmet utföras.
I BASIC finns GOTO samt GOSUB-RETURN.
Vi har även villkorliga hopp som IF-THEN.

f) In-utinstruktioner
Med dessa kan data överföras mellan CPU:n och yttre enheter.

g) Övriga
Ytterligare några grupper finns.

I BASIC kan vi t ex skriva

```
  :
120 INPUT A,B
130 X=A+B
  :
```

Vi kan använda variabler (A,B,X). Vi bryr oss inte om var någonstans i minnet variablerna (data) finns lagrade.
När vi nu skall skriva program i assembler kan vi inte använda variabler utan vi måste hålla reda på var i minnet vi lagrar data.

6.3 In- och utinstruktioner

Vi börjar med de enklaste in- och utinstruktionerna.

INSTRUKTIONSKOD	MNEMONIC	BETYDELSE
DB n	IN A,(n)	A ← Ingång nr n
D3 n	OUT (n),A	Utgång nr n ← A

n är ett tal mellan 00H och FFH. Vi kan alltså adressera 256 ingångar och 256 utgångar med dessa instruktioner.
Instruktionskoden består av 2 bytes, operationskod och portadress.

OPKOD	byte 1
ADRESS	byte 2

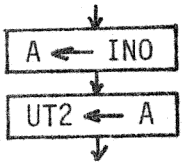
Om vi vill hämta data från ingång IN1 på digitalkortet skall vi lägga in
11011011 (=DBH)
00000001 (=01H)
i minnet. Elektronikkretsarna arbetar med låga och höga spänningar som vi kallar för 0 respektive 1. Eftersom det är besvärligt för oss att skriva en massa nollor och ettor, skriver vi koderna i hexadecimal form.
Varje instruktion representeras av en bokstavskombination, som kallas symbolisk kod eller mnemonic. Den symboliska koden är så vald att man kan förstå betydelsen av densamma (på engelska). Koden består av operationskod och operand.
Ett program skrivet med symbolisk kod, assemblerkod, måste översättas, assembleras, till maskinkod.

6.4 Hur skriver vi program?

Vi vill skriva ett kort program med de genomgångna instruktionerna. Avsikten är att visa hur programskrivning kan gå till och hur in- och utinstruktionerna används.

Exempel: Överför IN0 till accumulatorn A. Överför sedan A till UT2.

Vi ritar först flödesschema:



Programmen skriver vi enligt följande mönster.

ASSEMBLERAD KOD		KÄLLKOD			
ADRESS	OBJEKTKOD	LABEL	OPKOD	OPERAND	KOMMENTAR
			IN	A,(0)	A ← Ingång nr 0
			OUT	(2),A	Utgång nr 2 ← A

Vi fyller först i källkoden med mnemonics. För att förtydliga intruktionerna använder vi kommentarfältet.
Källkoden består av tre fält. I fältet LABEL eller LÄGE skriver vi symboliska adresser. Vi återkommer till detta längre fram.
Instruktionerna består av operationskod och operander vilka skrivs i varsitt fält.

Vi översätter nu källkoden till objektkod och lägger programmet i minnet med början i adress 65408D=FF80H.
Objektkoden kallas även maskinkod.
I ref (6) eller (8) hittar vi instruktionernas operationskoder i hexadecimal form.

ASSEMBLERAD KOD		KÄLLKOD			
ADRESS	OBJEKTKOD	LABEL	OPKOD	OPERAND	KOMMENTAR
FF80	DB00		IN	A,(0)	A←Ingång nr 0
FF82	D302		OUT	(2),A	Utgång nr 2←A

I minnet skall vi alltså lägga in följande.

adress	minne
FF80	DB
81	00
82	D3
83	02
	..

6.5 Hur får vi in programmet i minnet?

Vi kan välja något av de följande sätten.

a) Vi kan använda programmen EDITOR och ASSEMBLER som finns på kassett och diskett. Därvid utgår vi från källkoden. Vi behöver alltså inte handassemblera programmen (slå upp maskinkoden för hand).

b) Vi kan lägga in maskinkoden med programmet HEXMON som finns på samma kassett eller diskett som programmen EDITOR och ASSEMBLER.

c) Vi kan själva räkna om maskinkoden i hexadecimal form till decimal form och sedan lägga in koden i minnet med en POKE-sats. Då missar vi emellertid poängen med att använda datorn, nämligen att låta den räkna åt oss.

d) Vi kan använda programmet HEXDEC, se punkt 21.

För korta program kan vi i början använda något av alternativen b) eller d).

I ABC80 kör vi maskinspråksprogrammen genom att i BASIC anropa dem med CALL. Maskinspråksprogrammet är en subrutin till BASIC-programmet. Vårt program ovan måste avslutas på något sätt. Som det står ovan kommer processorn först att avverka de två första instruktionerna IN och OUT. Därefter tar processorn in koden i adress FF84H och betraktar den som nästa operationskod. Eftersom den är obekant och kan vara vad som helst mellan 00H och FFH spårar programmet troligen ur.
Därför bör programmen avslutas på något av följande sätt.

a) Avsluta programmet med HALT-instruktionen HALT. Därvid "stoppar" processorn och enda möjligheten att komma tillbaka till BASIC är att använda RESET-knappen på ABC80. Därvid förlorar vi tyvärr vårt hjälpprogram HEXMON eller HEXDEC.

b) Avsluta programmet med instruktionen JP FF80H. Denna hoppinstruktion medför att vi får en loop som genomlöps i det oändliga ända tills vi använder RESET-knappen på ABC80. Nackdel, se a).

c) Avsluta programmet med återhoppsinstruktionen RET. Denna ger återhopp från subrutinen.

Om vi använder BASIC-programmet HEXDEC kan vi först lägga in hexadecimala koden för vårt maskinspråksprogram, därefter lista koden och kontrollera att den är rätt. Slutligen kan vi köra vårt maskinspråksprogram genom att välja "körning". I HEXDEC hoppar vi därvid till rad 610.

```
610 X=CALL(A)
620 GOTO 610
```

Vi kommer att anropa subrutinen som börjar i adress A upprepade gånger. Körningen kan brytas med CTRL-C.

Om vi väljer alternativ c) ovan skall vårt program se ut så här.

ASSEMBLERAD KOD		KÄLLKOD			
ADRESS	OBJEKTKOD	LABEL	OPKOD	OPERAND	KOMMENTAR
FF80	DB00		IN	A,(0)	A←Ingång nr 0
82	D302		OUT	(2),A	Utgång nr 2←A
84	C9		RET		Åter till BASIC

6.6 Sammanställning av ovan nämnda instruktioner

INSTRUKTIONSKOD	MNEMONIC	BETYDELSE
DB n	IN A,(n)	A←Ingång nr n. 2 bytes
D3 n	OUT (n),A	Utgång nr n←A. 2 bytes
C9	RET	Ger återhopp från subrutin. 1 byte
76	HALT	Haltinstruktion. 1 byte
C3 nn	JP nn	Ger hopp till adress nn. 3 bytes

Beteckningar:
n avser 8 bitar.
nn avser 16 bitar.

Öva på uppgifter under punkt 7.1.

6.7 Dataöverföring av 8 bitar.

Med instruktionen LD kan vi överföra 8 bitar mellan två register eller mellan register och minne. Överföringen sker från en källa (source) till en destination (destination). Överföringen är egentligen en kopiering så källans innehåll ändras inte. För nedanstående instruktioner finns identiska till 8080/8085.

MNEMONIC	ANTAL BYTES	BETYDELSE	KOMMENTAR
LD r,r'	1	r ← r'	Överför innehållet i r' till r
LD r,(HL)	1	r ← (HL)	Överför innehållet i den minnescell vars adress finns i HL till r
LD (HL),r	1	(HL) ← r	Överför innehållet i r till den minnescell vars adress finns i HL
LD r,n	2	r ← n	Ladda r med n. n finns i instruktionens andra byte.
LD (HL),n	2	(HL) ← n	Ladda den minnescell vars adress finns i (HL) med n.
LD A,(nn)	3	A ← (nn)	Överför innehållet i den minnescell vars adress är nn till A. nn finns i instruktionens andra och tredje byte.
LD (nn),A	3	(nn) ← A	Överför innehållet i A till den minnescell vars adress är nn.
LD A,(BC)	1	A ← (BC)	Överför innehållet i den minnescell vars adress finns i BC till A.
LD A,(DE)	1	A ← (DE)	Överför innehållet i den minnescell vars adress finns i DE till A.
LD (BC),A	1	(BC) ← A	Överför innehållet i A till den minnescell vars adress finns i BC.
LD (DE),A	1	(DE) ← A	Överför innehållet i A till den minnescell vars adress finns i DE.

Ovanstående instruktioner påverkar icke flaggorna.

Beteckningar:

r och r' avser något av registren A, B, C, D, E, H, L.

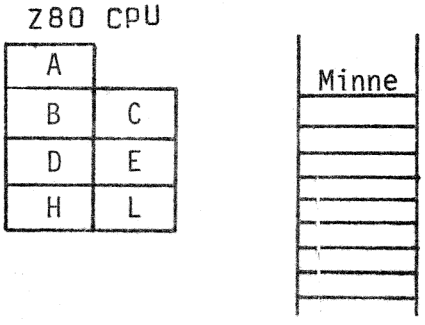
Då parentes finns med avses en minnescell.

n avser 8 bitar.

nn avser 16 bitar.

Vi ser av tabellen ovan att överföringen av en byte från källa till destination sker mellan de platser vidstående figur visar.

Hur anges källa och destination i instruktionen?



6.8 Adresseringsmetoder. I.

När vi vill överföra data från en källa till en destination måste adresserna till dessa på något sätt anges i instruktionen. Vi börjar med nedanstående adresseringsmetoder som även finns i 8080/8085.

a) Register-adressering (Register Addressing)

Adressen till registret eller registren ingår i operationskoden.

Ex: LD A,C Innehållet i C överförs till A.
Både källa och destination adresseras medelst register-adressering.

b) Indirekt register-adressering (Register Indirect Addressing)

I operationskoden anges ett registerpar som innehåller adressen till den aktuella minnescellen. I registerparen BC, DE, HL innehåller därvid B, D och H de 8 mest signifikanta bitarna (hög adress) av adressen. C, E och L innehåller de 8 minst signifikanta bitarna (låg adress) av adressen.

Ex: LD C,(HL).
Antag H innehåller C1H och
L innehåller 07H. Innehållet i
minnescell C107H är A7H.
Då instruktionen i exemplet
utförts innehåller C data A7H.

adress	minne
	.
	.
C107	A7
	.

I detta fall adresseras källan medelst indirekt register-adressering. Innan vi använder ett registerpar som minnespekare måste vi se till att registerparet innehåller den avsedda adressen.

c) Omedelbar adressering

(Immediate)

Instruktionens andra byte innehåller operanden.
Ex: LD E,79H Ladda E med 79H.

Då instruktionen utförts innehåller E data 79H.

Adress	minne	
FF90	1E	OPKOD
91	79	OPERAND

(Immediate extended)

I detta fall består data av 2 bytes, ofta en adress.
Andra byten innehåller låg adress och tredje byten hög adress.

Ex: LD HL,0FF95H Ladda registerparet HL med FF95H. (Se punkt 6.9)

Efter instruktionens utförande innehåller H data FFH och L data 95H.

adress	minne	
FF85	21	OPKOD
86	95	LÅG ADRESS
87	FF	HÖG ADRESS

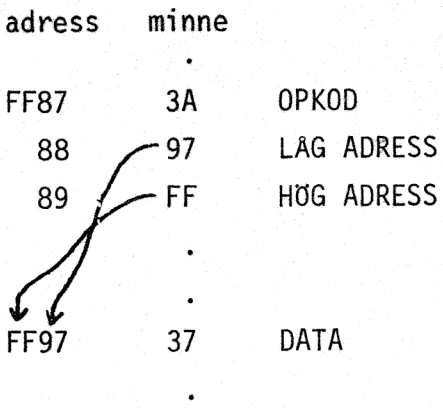
d) Direkt adressering (Extended Addressing)

Instruktionens andra och tredje byte innehåller adressen till den aktuella minnescellen. Andra byten innehåller låg adress och tredje byten hög adress.

Ex: LD A,(0FF97H) Överför innehållet i minnescell FF97H till A.

Källan adresseras medelst direkt adressering.

Efter instruktionens utförande innehåller A data 37H.



e) Adresseringsmetoder. Sammanfattning

I åtskilliga instruktioner anges källa och destination med olika adresseringsmetoder.

Ex: LD C,(HL)
Källan adresseras medelst indirekt register-adressering.
Destinationen adresseras medelst registeradressering.

Vid omedelbar adressering ligger data i programmet. Ändringar kräver därför ändring i programmet. Metoden används därför enbart för konstanter.

Instruktionerna för direkt adressering upptar 3 bytes medan motsvarande för indirekt register-adressering upptar 1 byte. Då man skall behandla flera bytes och dessa ligger efter varandra i minnet är den senare adresseringsmetoden att föredra.

Öva på uppgifter under punkt 7.2.

6.9 Dataöverföring av 16 bitar

INSTRUK-TIONSKOD	MNEMONIC	ANTAL BYTES	BETYDELSE	KOMMENTAR
01nn	LD BC,nn	3	BC ← nn	Ladda BC med nn
11nn	LD DE,nn	3	DE ← nn	
21nn	LD HL,nn	3	HL ← nn	
31nn	LD SP,nn	3	SP ← nn	
ED4Bnn	LD BC,(nn)	4	BC ← (nn)	Överför innehållet i minnes-cell nn till C och inne-hållet i minnescell nn+1 till B
ED5Bnn	LD DE,(nn)	4	DE ← (nn)	
2Ann	LD HL,(nn)	3	HL ← (nn)	
ED7Bnn	LD SP,(nn)	4	SP ← (nn)	
ED43nn	LD (nn),BC	4	(nn) ← BC	Överför innehållet i C till den minnescell vars adress är nn och i B till den minnescell vars adress är nn+1
ED53nn	LD (nn),DE	4	(nn) ← DE	
22nn	LD (nn),HL	3	(nn) ← HL	
ED73nn	LD (nn),SP	4	(nn) ← SP	
F9	LD SP,HL	1	SP ← HL	
F5	PUSH AF	1		Spara AF på stacken
C5	PUSH BC	1		
D5	PUSH DE	1		
E5	PUSH HL	1		
F1	POP AF	1		Återställ AF
C1	POP BC	1		
D1	POP DE	1		
E1	POP HL	1		
EB	EX DE,HL	1	DE ↔ HL	Byt innehåll mellan DE och HL

Ovanstående instruktioner påverkar icke flaggorna (POP AF återställer flaggorna).

Beteckningar:
nn avser 16 bitar.
Då parentes finns avses innehållet i en minnescell.

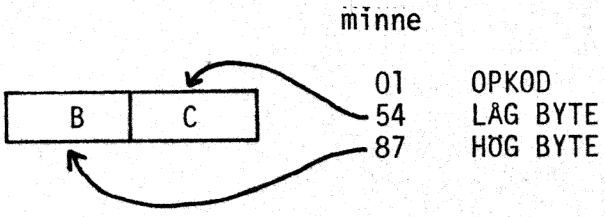
Eftersom två bytes överförs måste vi tänka på i vilken ordning de kommer.

Vi ser ovan att instruktionerna kan indelas i grupper om 4.
Två instruktioner, LD SP,HL och EX DE,HL utgör undantag.
Med ett exempel för varje grupp visas i vilken ordning överföringen sker.

Exempel för första gruppen:

LD BC,8754H

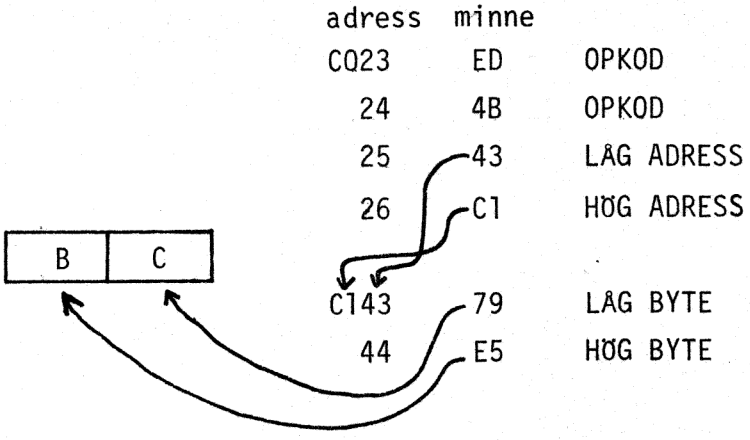
Då instruktionen utförts innehåller C data 54H och B data 87H.



Exempel för andra gruppen:

LD BC,(0C143H)

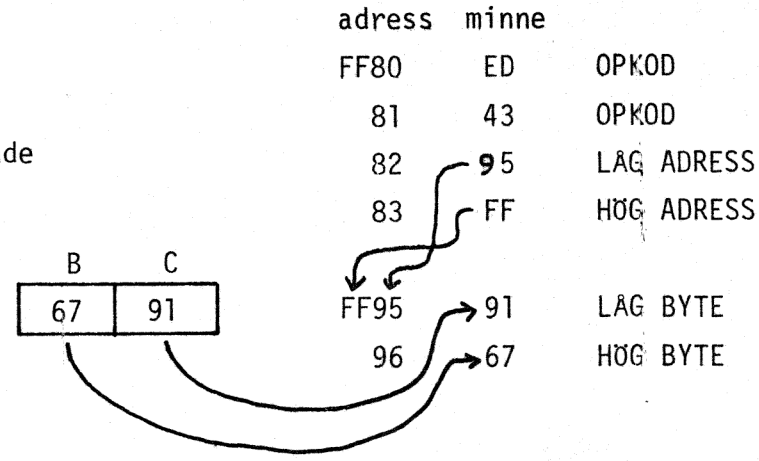
Då instruktionen utförts innehåller C data 79H och B data E5H.



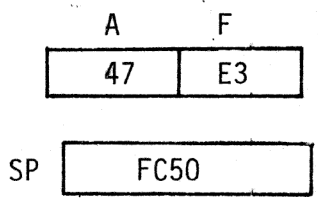
Exempel för tredje gruppen:

LD (0FF95),BC

Antag B innehåller 67H och C innehåller 91H. Efter instruktionens utförande innehåller minnescell FF95H data 91H och minnescell FF96H data 67H.



Stackinstruktionerna PUSH och POP finns med trots att vi tar upp subrutiner längre fram.
Ex: PUSH AF. A och F överförs till stacken.



adress	minnes-stack
FC4B	
FC4C	
FC4D	
FC4E	E3
FC4F	47
FC50	

Antag AF och SP har
ovanstående innehåll
före instruktionens utförande.

Då instruktionen utförts har SP innehållet FC4EH.
SP pekar på toppen av stacken.

EX: POP AF återställ AF.
Antag SP innehåller FC4EH. Stackens innehåll, se ovan.
Då instruktionen utförts har A och F återställts och
SP innehåller FC50H.

Programexempel på dataöverföring för illustration av adresseringsmetoder.
Överför INO till minnescell FFAOH med HL som minnespekare.
Överför sedan (FFAO) till (FFCO) med registerparen BC och DE
som minnespekare.

ASSEMBLERAD KOD		KÄLLKOD			
ADRESS	OBJEKTKOD	LABEL	OPKOD	OPERAND	KOMMENTAR
FF80	21AOFF		LD	HL,OFFAOH	
	DBOO		IN	A,(O)	
	77		LD	(HL),A	
	01AOFF		LD	BC,OFFAOH	
	11COFF		LD	DE,OFFCOH	
	OA		LD	A,(BC)	
	12		LD	(DE),A	
	C9		RET		

Öva på uppgifter under punkt 7.3.

6.10 Att ange operander

I BASIC kan vi skriva t ex

170 INPUT A,B

180 C=A+B

Vi behöver inte fundera över var i minnet operanderna A, B och C finns lagrade.

Vi skall nu titta på aritmetiska och logiska operationer. Instruktionerna är oftast uppbyggda som

Z=X⊕Y där

X, Y och Z är operander och ⊕ en aritmetisk eller logisk operation. Tecknet = uppfattar vi som "tilldelas". I instruktionerna måste alla dessa fyra enheter anges. Typ av operation anges i operationskoden. Operanderna kan anges på olika sätt.

a) Implicerad operand (Inbyggd operand)
I instruktionen öka A med 1 är Y=1 och ligger implicerad i operationskoden.

b) Implicerad adress till operand
För de flesta aritmetiska och logiska instruktionerna utgör Z och X ackumulatorn A. Adressen till Z och X är implicerad, nämligen A.

Ex: ADD A,C addera C till A och lägg resultatet i A.

c) Adresserad operand
Adressen till Y i A=A⊕Y anges med någon adresseringsmetod, se punkt 6.8.

Ex: ADD A,C registeradressering

ADD A,(HL) indirekt register-adressering

ADD A,n omedelbar adressering

Då vi vill utföra en operation på två bytes, t ex addition, måste vi först överföra X till A, sedan utföra operationen och därvid använda lämplig adresseringsmetod, samt slutligen ta hand om resultatet i A, t ex spara det i minnet.

6.11 Flaggor

Flaggregistret innehåller 6 flaggor.
Dessa påverkas av vissa instruktioner.
Det är med hjälp av flaggorna vi kan undersöka "tillstånd" och ställa ja/nej-villkor.
Därför är det väsentligt att förstå vad flaggornas tillstånd innebär och ta reda på hur flaggorna påverkas av olika instruktioner.

Vi kan "känna av" följande 4 flaggor.

C (Carry, minnessiffra) (betecknas ibland med CY)
Vid addition sätts C om det blir minnessiffra från mest signifikanta biten.
Vid subtraktion sätts C (C=1) för att visa lån (borrow), d v s negativt resultat.
Vissa skift- och rotationsinstruktioner påverkar C.
Vi får inte blanda ihop Carry och register C!

Z (Zero, noll)
Z sätts om en operation ger ett resultat som är noll (alla bitarna = noll).

S (Sign, tecken)
Flaggan används för "signed numbers" och sätts om resultatet är negativt. Eftersom bit 7 utgör teckenbit sätts S=D7 i resultatet.

P/V (Parity/Overflow)
För logiska operationer anger flaggan resultatets paritet.
Flaggan sätts för jämn paritet (jämnt antal ettor i resultatet).

För aritmetiska operationer anger flaggan overflow i resultatet för tvåkomplementtal. Overflow föreligger om resultatet skall bli större än 127D eller mindre än -128D.
A innehåller i dessa fall ett felaktigt resultat.

Ingen LD-instruktion påverkar flaggorna.
(Undantag: LD A,I och LD A,R men dessa berör vi inte).

Flaggregistret har följande innehåll:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	X	H	X	P/V	N	C

X-positionerna används inte och är obestämda.

H-flaggan (Half carry) och N-flaggan (Subtract Flag) används för BCD-aritmetik men är icke testbara.

6.12 Aritmetiska instruktioner för 8 bitar

Nedan tar vi upp de instruktioner i vilka ena operanden adresseras enligt

register-adressering
indirekt register-adressering
omedelbar adressering (ej för INC och DEC).

MNEMONIC	ANTAL BYTES	BETYDELSE	KOMMENTAR
ADD A,r	1	A←A+r	r adderas till A och resultatet finns i A
ADD A,(HL)	1	A←A+(HL)	
ADD A,n	2	A←A+n	
ADC A,r	1	A←A+r+CY	r och CY adderas till A och resultatet finns i A.
ADC A,(HL)	1	A←A+(HL)+CY	
ADC A,n	2	A←A+n+CY	
SUB r	1	A←A-r	r subtraheras från A och resultatet finns i A
SUB (HL)	1	A←A-(HL)	
SUB n	2	A←A-n	
SBC A,r	1	A←A-r-CY	r och CY subtraheras från A och resultatet finns i A
SBC A,(HL)	1	A←A-(HL)-CY	
SBC A,n	2	A←A-n-CY	
CP r	1		operationen A-r utförs och flaggorna sätts enligt punkt 6.11. Innehållet i A påverkas icke.
CP (HL)	1		
CP n	2		
INC r	1	r←r+1	S-flaggan och Z-flaggan påverkas. C-flaggan påverkas icke.
INC (HL)	1	(HL)←(HL)+1	
DEC r	1	r←r-1	S-flaggan och Z-flaggan påverkas. C-flaggan påverkas icke.
DEC (HL)	1	(HL)←(HL)-1	

Beteckningar:

r avser något av registren A, B, C, D, E, H, L.

Då parentes finns med avses minnescell.

n avser 8 bitar.

CY avser Carry-flaggan.

Vi ser att för ADD, ADC och SBC finns A med som andra operand i operandfältet för mnemonics, dock icke i CP eller SUB. SUB A medför att A nollställs.

Öva på uppgifter under punkt 7.4.

6.13 Logiska instruktioner

Logiska instruktioner finns för 8 bitar. Vi tar nedan upp de instruktioner i vilka ena operanden adresseras enligt

register-adressering
indirekt register-adressering
omedelbar adressering

MNEMONIC	ANTAL BYTES	BETYDELSE	KOMMENTAR
AND r	1	A←A OCH r	Bitvis OCH mellan A och r. Resultatet finns i A.
AND (HL)	1	A←A OCH (HL)	
AND n	2	A←A OCH n	
OR r	1	A←A ELLER r	Bitvis ELLER mellan A och r. Resultatet finns i A.
OR (HL)	1	A←A ELLER (HL)	
OR n	2	A←A ELLER n	
XOR r	1	A←A XOR r	Bitvis EXKLUSIVT ELLER mellan A och r. Resultatet finns i A.
XOR (HL)	1	A←A XOR (HL)	
XOR n	2	A←A XOR n	

Beteckningar:

r avser något av registren A, B, C, D, E, H, L.

Då parentes finns med avses minnescell

n avser 8 bitar.

Flaggorna påverkas enligt följande:

S sätts lika med D7 i A.

Z sätts om resultatet är noll, annars nollställs den.

P/V sätts för jämn paritet, nollställs för udda.

C nollställs.

I ovanstående mnemonics ingår icke A som andra operand i operandfältet.

XOR A innebär att A nollställts.

OR A och AND A påverkar flaggor utan att A ändras.

Öva på uppgifter under punkt 7.5.

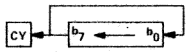
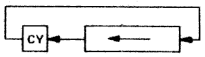
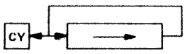
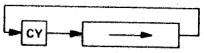
6.14 Specialinstruktioner för ackumulator och carry-flagga

INSTRUKTIONSKOD	MNEMONIC	BETYDELSE
27	DAA	Används för BCD-aritmetik efter ADD, ADC, INC, SUB, SBC, DEC och NEG. DAA justerar resultatet så att korrekt BCD-representation erhålls. Flaggor påverkas.
2F	CPL	A ← $\overline{A}$. Bitvis invertering. (1-komplement). Flaggor S, Z, P/V, C påverkas icke.
ED44	NEG	A ← 0-A. (2-komplement). Flaggor påverkas.
3F	CCF	CY ← $\overline{CY}$. Invertera carry. Flaggor S, Z, P/V påverkas icke.
37	SCF	CY ← 1. Flaggor S, Z, P/V påverkas icke.

Öva på uppgifter under punkt 7.6.

6.15 Instruktioner för rotation och skift

Med dessa instruktioner kan vi manipulera enskilda bitar. Högerskift innebär division med 2 och vänsterskift multiplikation med 2.

Instruktions-kod	Mnemonic	Betydelse	Kommentar
07	RLCA		Flaggor S, Z, P/V påverkas icke.
17	RLA		"
0F	RRCA		"
1F	RRA		"
	RLC r		Samtliga flaggor påverkas.
CB06	RLC (HL)		"
	RL r		"
CB16	RL (HL)		"
	RRC r		"
CB0E	RRC (HL)		"
	RR r		"
CB1E	RR (HL)		"
	SLA r		"
CB26	SLA (HL)		"
	SRA r		"
CB2E	SRA (HL)		"
	SRL r		"
CB3E	SRL (HL)		"

Beteckningar:

r avser något av registren A, B, C, D, E, H, L.
Då parentes finns med avses minnescell.

Vi ser att de fyra första instruktionerna avviker från de övriga när det gäller påverkan på flaggorna. De fyra första opererar enbart på A och identiskt lika instruktioner finns för 8080.

Öva på uppgifter under punkt 7.7.

6.16 Aritmetiska instruktioner för 16 bitar

Det finns några aritmetiska instruktioner som använder registerpar. Registerparet HL fungerar här som en 16-bitars ackumulator. Vi tar här upp följande instruktioner.

INSTRUK-TIONSKOD	MNEMONIC		ANTAL BYTES	BETYDELSE	KOMMENTAR
09	ADD	HL,BC	1	HL←HL+BC	Flaggor S,Z, P/V påverkas icke. Flagga CY sätts av carry från bit 15.
19	ADD	HL,DE	1	HL←HL+DE	"
29	ADD	HL,HL	1	HL←HL+HL	"
39	ADD	HL,SP	1	HL←HL+SP	"
ED4A	ADC	HL,BC	2	HL←HL+BC+CY	Flaggor S, Z, P/V och CY (C) påverkas.
ED5A	ADC	HL,DE	2	HL←HL+DE+CY	"
ED6A	ADC	HL,HL	2	HL←HL+HL+CY	"
ED7A	ADC	HL,SP	2	HL←HL+SP+CY	"
ED42	SBC	HL,BC	2	HL←HL-BC-CY	Flaggor S, Z, P/V och CY påverkas.
ED52	SBC	HL,DE	2	HL←HL-DE-CY	"
ED62	SBC	HL,HL,	2	HL←HL-HL-CY	"
ED72	SBC	HL,SP	2	HL←HL-SP-CY	"
03	INC	BC	1	BC←BC+1	Inga flaggor påverkas
13	INC	DE	1	DE←DE+1	"
23	INC	HL	1	HL←HL+1	"
33	INC	SP	1	SP←SP+L	"
0B	DEC	BC	1	BC←BC-1	Inga flaggor påverkas
1B	DEC	DE	1	DE←DE-1	"
2B	DEC	HL	1	HL←HL-1	"
3B	DEC	SP	1	SP←SP-1	"

Beteckningar: CY avser carryflaggan.

Vi observerar att
- INC för ett register påverkar Z-flaggan (punkt 6.12)
- INC för ett registerpar påverkar ICKE Z-flaggan,

Vill vi använda en varvräknare bör vi använda ett 8-bitars register så att Z-flaggan sätts då vi kommit ned till noll i varvräknaren.
Då vi använder ett registerpar som adresspekare och ökar eller minskar denna med 1 påverkas alltså inte Z-flaggan.
De instruktioner ovan som består av en byte har identiska motsvarigheter för 8080.

6.17 Hopp

Hoppen indelas i två grupper, villkorliga och ovillkorliga.
Ett ovillkorligt hopp utförs alltid. Jämför GOTO i BASIC.

De ovillkorliga hoppen är

JP nn (omedelbar adressering) nn=16-bitars adress

JR e (relativ adressering) e=relativ adress i tvåkompl.

De villkorliga hoppen medger möjlighet att ställa JA/NEJ-frågor.
Jämför BASIC-satsen IF P=0 THEN 570.

De villkorliga hoppen är

JP CC,nn (omedelbar adressering) CC avser flaggvillkor

JR CC,e (relativ adressering)

Hoppen utförs på så sätt att programräknaren laddas med den nya adressen då hoppinstruktionen utförs.

Hopp med omedelbar adressering

INSTRUKTIONSKOD	MNEMONIC	KOMMENTAR
C3nn	JP nn	Hopp till adress nn
C2nn	JP NZ,nn	Hopp till adress nn om resultatet av en föregående operation är icke noll, dvs Z=0.
CAnn	JP Z,nn	Hopp till adress nn om resultatet av en föregående operation är noll, dvs Z=1.
D2nn	JP NC,nn	Hopp till adress nn om resultatet av en föregående operation är CY=0.
DAnn	JP C,nn	Hopp till adress nn om resultatet av en föregående operation är CY=1.
E2nn	JP PO,nn	Hopp till adress nn om resultatet av en föregående operation är udda paritet, dvs P/V=0.
EAnn	JP PE,nn	Hopp till adress nn om resultatet av en föregående operation är jämn paritet, dvs P/V=1.
F2nn	JP P,nn	Hopp till adress nn om resultatet av en föregående operation är positivt, dvs S=0. (Resultatet > =0)
FAnn	JP M,nn	Hopp till adress nn om resultatet av en föregående operation är negativt, dvs S=1.
E9	JP (HL)	Hoppa till den adress som finns i HL, dvs PC←HL. Indirekt register-adresser.

Uttrycket "en föregående operation" ovan avser den sista operationen före hoppet som påverkar den aktuella flaggan.
För ovanstående instruktioner finns identiskt lika i 8080.

Angående omedelbar adressering, se punkt 6.8c.

Ovanstående instruktioner består av 3 bytes utom den sista som består av 1 byte.

Hopp med relativ adressering

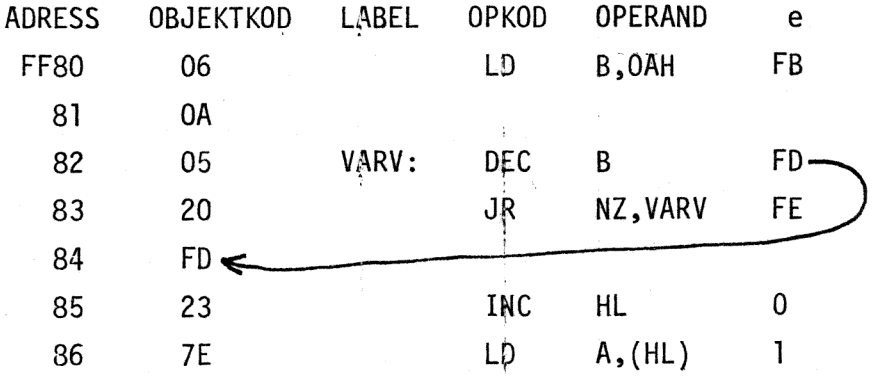
Relativ adressering är en adresseringsmetod som vi inte nämnt tidigare. Vi tittar först på instruktionerna.

INSTRUK-TIONSKOD	MNEMONIC	KOMMENTAR
18e	JR e	Hopp till adress e relativt nästa OPKOD.
38e	JR C,e	Hopp till adress e relativt nästa OPKOD om resultatet av en föregående operation är CY=1.
30e	JR NC,e	Hopp till adress e relativt nästa OPKOD om resultatet av en föregående operation är CY=0.
28e	JR Z,e	Hopp till adress e relativt nästa OPKOD om resultatet av en föregående operation är noll, dvs Z=1.
20e	JR NZ,e	Hopp till adress e relativt nästa OPKOD om resultatet av en föregående operation är icke noll, dvs Z=0.
10e	DJNZ,e	Se manual

Uttrycket "en föregående operation" ovan avser den sista operationen före hoppet som påverkar den aktuella flaggan.

Hoppadressen anges från adressen för nästa OPKOD.
e anges i tvåkomplement och har området -128 till +127.
PC←PC+e.
Med två exempel visar vi hur hoppen utförs.

Exempel 1.



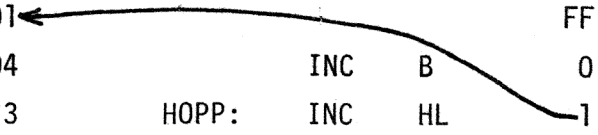
ADRESS	OBJEKTKOD	LABEL	OPKOD	OPERAND	e
FF80	06		LD	B,0AH	FB
81	0A				
82	05	VARV:	DEC	B	FD
83	20		JR	NZ,VARV	FE
84	FD				
85	23		INC	HL	0
86	7E		LD	A,(HL)	1

I ovanstående exempel laddar vi in 10D i B, räknar sedan ned B till 0 innan vi fortsätter i programmet. Adressen dit vi vill hoppa kallar vi för VARV och anger den i fältet LABEL. Använder vi programmet ASSEMBLER skall VARV följas av:. I källkoden använder vi alltså symboliska adressen. Om vi sedan handassemblerar måste vi räkna fram relativadressen e. Om e=0 så fortsätter vi med nästa instruktion även om villkoret är uppfyllt. Vill vi hoppa bakåt som i detta fall är e negativt och anges i tvåkomplement. -1 i 2-kompl. är FFH, -2 är FEH osv. I detta fall vill vi hoppa tre steg bakåt som är FDH. Andra byten i hoppinstruktionen blir alltså i detta fall FDH. Om vi handassemblerar måste vi räkna ut hoppadresser för hand när vi fyller i OBJEKTKODEN. Till höger om programmet ovan finns de e-värden angivna som gör att vi kommer till en viss rad.

Instruktionen JR OFEH medför att vi alltid hoppar tillbaka till denna instruktion och orsakar en loop som vi inte kommer ur på annat sätt än med RESET-knappen.

Exempel 2.

ADRESS	OBJEKTKOD	LABEL	OPKOD	OPERAND	e
FF90	82		ADD	A,D	FD
91	30		JR	NC,HOPP	FE
92	01				FF
93	04		INC	B	0
94	23	HOPP:	INC	HL	1



Programdelen avser bara att visa hur hoppinstruktionen fungerar.
I detta fall vill vi hoppa över en instruktion om CY=0.
Således e=1 i detta fall.
Fördelen med relativ adressering är dels att det åtgår en byte mindre än med omedelbar adressering, dels att programmet kan flyttas i minnet utan att man behöver ändra någon hoppadress.

Sammanställning av villkorliga hopp och subrutinanrop.
Angående subrutiner, se nästa punkt.

FLAGGA	HOPP	HOPP	SUBRUTINANROP	R=RESULTAT
Z=0	JP NZ,nn	JR NZ,e	CALL NZ,nn	R≠0
Z=1	JP Z,nn	JR Z,e	CALL Z,nn	R=0
CY=0	JP NC,nn	JR NC,e	CALL NC,nn	C=0
CY=1	JP C,nn	JR C,e	CALL C,nn	C=1
P/V=0	JP PO,nn		CALL PO,nn	R har udda par.
P/V=1	JP PE,nn		CALL PE,nn	R har jämn par.
S=0	JP P,nn		CALL P,nn	R ⩾ 0
S=1	JP M,nn		CALL M,nn	R < 0

R avser resultatet av den sista instruktionen som påverkar den aktuella flaggan.

En jämförelse mellan hopp och subrutinanrop finns i nästa punkt.

Observera att ett resultat räknas som positivt om det är noll eller större än noll, se näst sista raden.

6.18 SUBRUTINER

Först visas instruktionerna. Se även sammanställningen i föregående punkt.

INSTRUKTIONSKOD	MNEMONIC
CDnn	CALL nn
C4nn	CALL NZ,nn
CCnn	CALL Z,nn
D4nn	CALL NC,nn
DCnn	CALL C,nn
E4nn	CALL PO,nn
ECnn	CALL PE,nn
F4nn	CALL P,nn
FCnn	CALL M,nn
C9	RET

Beteckningar: nn avser 16-bitars adress.

Den första och sista instruktionen, CALL nn resp RET , är ovillkorliga. De andra instruktionerna är villkorliga på samma sätt som hoppinstruktionerna. Man tittar lämpligen i sammanställningstabellen i föregående punkt för att finna lämplig instruktion för villkorliga subrutinanrop.
Det finns även villkorliga återhopp från subrutiner men dessa bör icke användas och återges därför icke. En subrutin bör ha en ingång och en utgång och vid behov kan man ha ett villkorligt hopp till sista delen av subrutinen.

I BASIC kan vi anropa en programdel som vi vill använda flera gånger med instruktionen GOSUB ... . Subrutinen avslutas med RETURN i BASIC varvid programexekveringen fortsätter med instruktionen på radnumret efter GOSUB ... .

Även i assembler kan vi skriva subrutiner för programdelar som används flera gånger. En subrutin kan betraktas som ett program i programmet. Som vi sett ovan kan vi ha både ovillkorligt och villkorliga anrop på subrutiner. Subrutiner avslutas med instruktionen RET som ger återhopp till huvudprogrammet. Subrutinanrop är ett "intelligent hopp" därför att datorn håller rätt på återhoppsadressen. På samma sätt är hoppen (JP och JR) "dumma hopp" eftersom datorn inte håller rätt på återhoppsadresserna.
På de två följande sidorna visas skillnaden mellan hopp och subrutinanrop med hjälp av figurer.

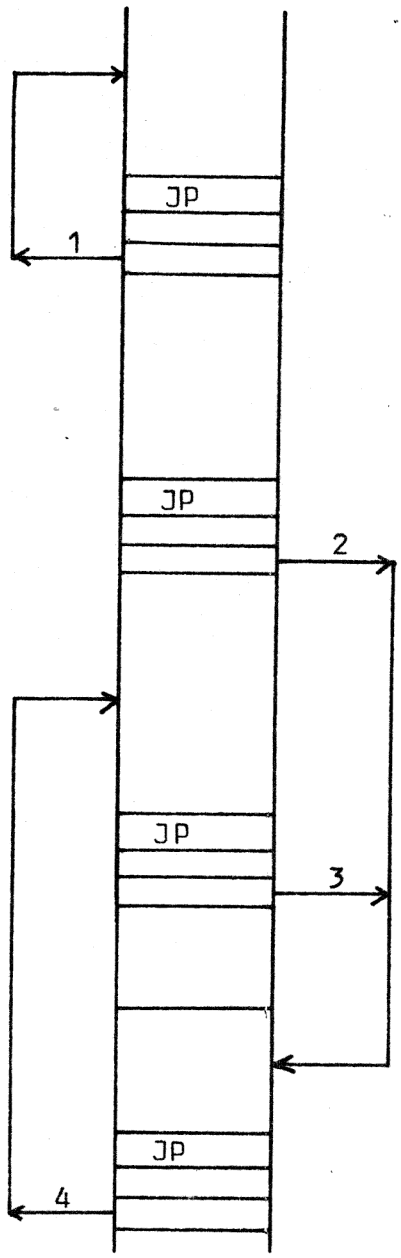


Fig 6.18.1

Hoppinstruktionen 1 skapar en loop. Om man startar programmet från början, fastnar man i denna loop. Låt oss anta att vi kommer förbi hoppinstruktion 1. Först hoppar vi från 2 till 4 och tillbaka.
Sedan kommer vi att fastna i en loop som orsakas av hoppinstruktionerna 3 och 4 .

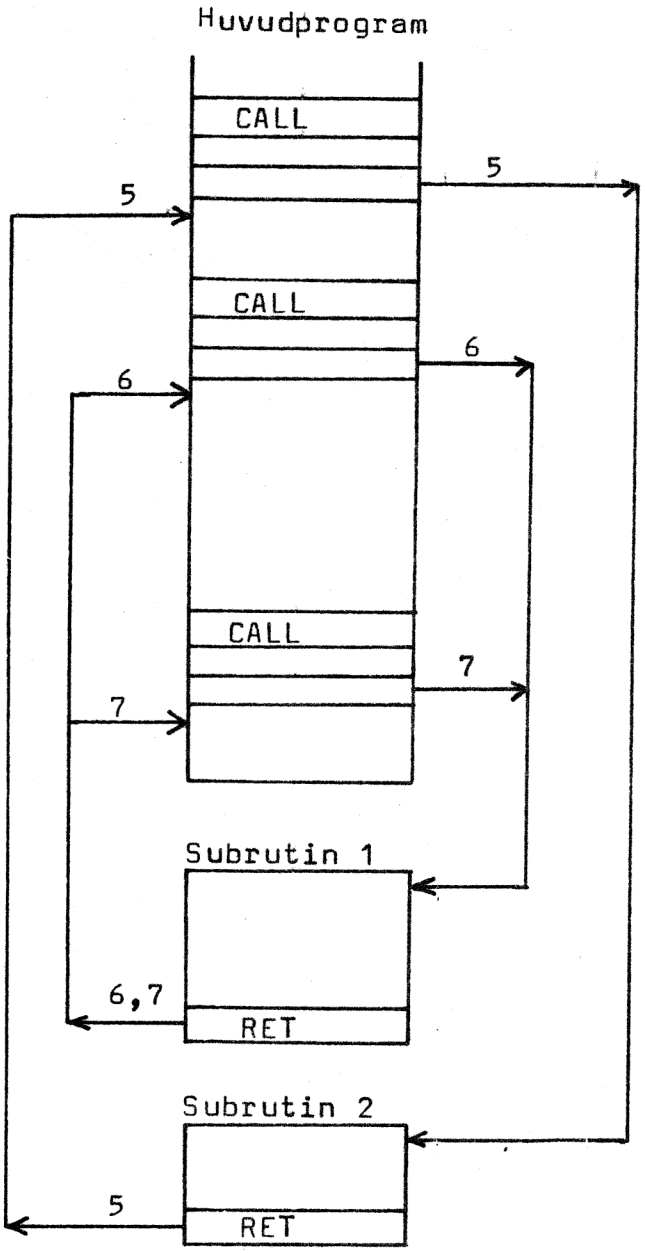


Fig 6.18.2

Subrutinanrop 5 anropar subrutin 2. Därefter fortsätter vi med den instruktion som följer efter subrutinanrop 5 .
Sedan anropas subrutin 1 av subrutinanrop 6 och 7 i huvudprogrammet.
Återhoppet sker alltid till den instruktion som följer efter subrutinanropet.

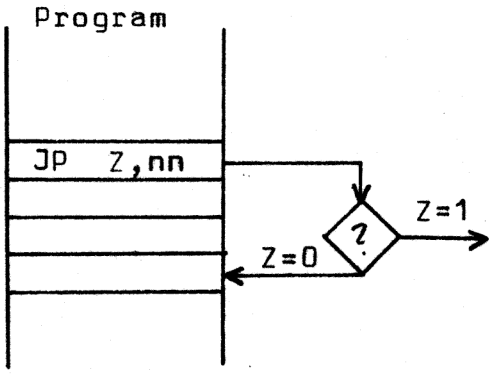


Fig 6.18.3

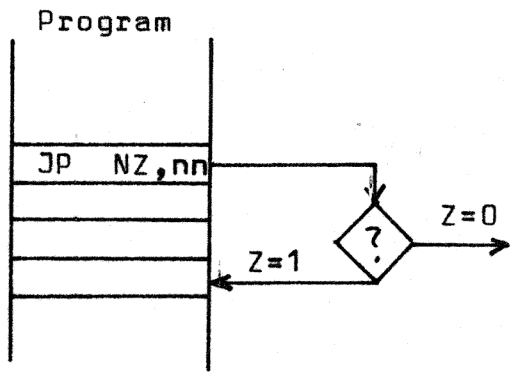


Fig 6.18.4

Figurerna ovan visar att om villkoret är uppfyllt sker hopp till angiven adress. Om villkoret icke är uppfyllt fortsätter datorn med instruktionen närmast efter den villkorliga hoppinstruktionen.

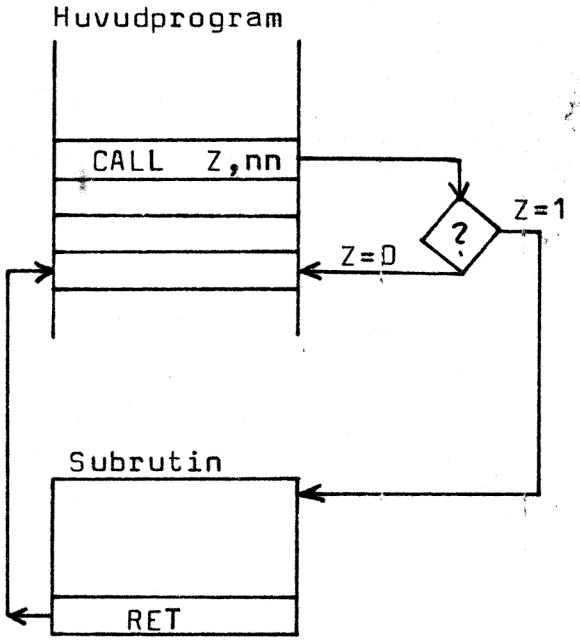


Fig 6.18.5

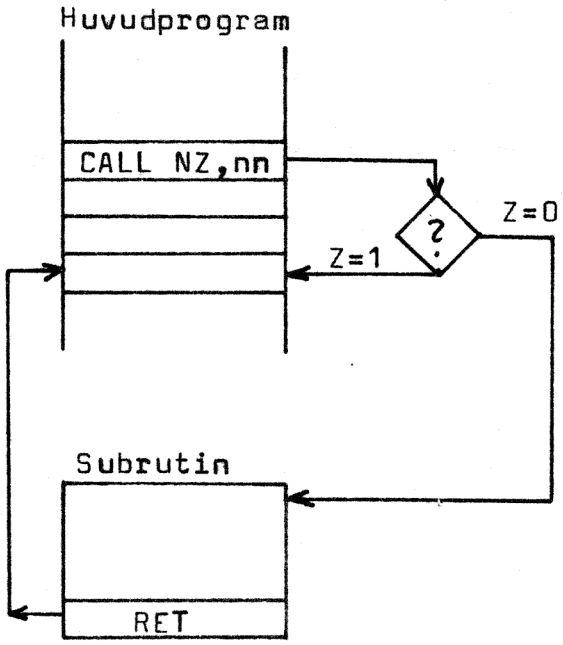
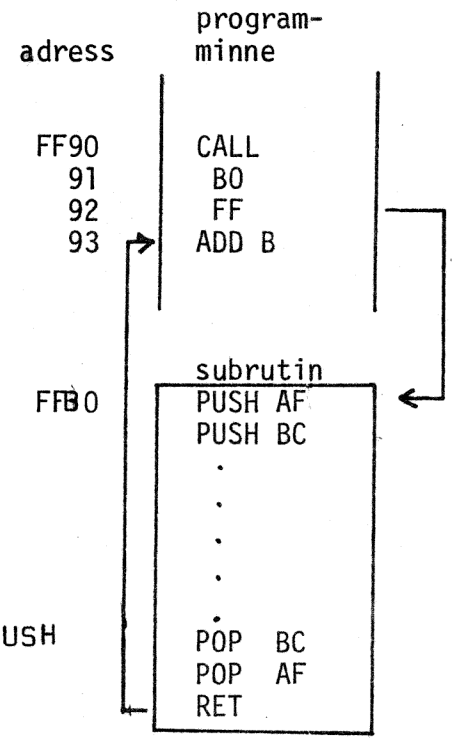


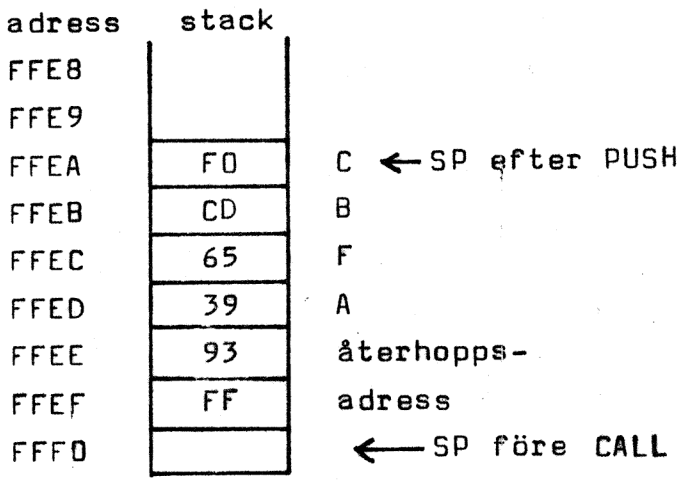
Fig 6.18.6

Figurerna ovan visar att om villkoret är uppfyllt sker subrutinanrop till angiven subrutinadress.
Om villkoret icke är uppfyllt fortsätter datorn med den instruktion som följer efter det villkorliga subrutinanropet.

När datorn kommer till ett subrutinanrop lagras programräknarens (PC) innehåll undan på stacken. Vid undanlagringen innehåller PC adressen till instruktionen efter CALL.
Stacken är en del av minnet. Stackpekaren(SP) innehåller adressen till stackens topp. Lagringen på stacken sker enligt principen sist in först ut (LIFO). (Tänk på en trave tallrikar).
Vi kan ha subrutinanrop från en subrutin osv.
Vi kan även lagra data på stacken. Då vi anropar en subrutin har vi kanske data i registren som inte får förstöras.
Subrutinen börjar då med undanlagring av registren medelst PUSH-instruktioner. I slutet av subrutinen sker återställande av registren medelst POP-instruktioner.
Stackens storlek bestäms av hur mycket utrymme vi har reserverat för den. Observera att varje gång något lagras på stacken så minskas SP (gäller Z80). I ett program för en mikrodator med Z80 laddas därför SP i programmets början lämpligen med adressen till den sista minnescellen i skriv/läsminnet.
I ABC80 har SP getts en adress i BASIC-tolken så vi bör inte ändra på denna.
För att studera vad som händer vid subrutinanrop tittar vi på följande exempel.

Vi har ett program som innehåller ett subrutinanrop på adress FF90H. Subrutinen börjar på adress FFB0. Vi antar att SP=FFF0H. I programmet använder vi registren A,B,C så deras innehåll får inte förstöras.
Vi antar att A=39H
flaggregistret F=65H
B=CDH
C=F0H.
Stackens innehåll då vi är mitt i subrutinen framgår nedan.





Först lagras återhoppsadressen (FF93H) undan. I subrutinen sparar vi först undan AF och sedan BC. SP pekar på toppen av stacken (SP=FFEA). När subrutinprogrammet har utförts återställer vi registren. Eftersom BC ligger överst börjar vi med BC och därefter återställs AF. Registerparen skall alltså återställas i omvänd ordning jämfört med undanlagringen. Instruktionen RET medför att återhoppsadressen överförs till PC.

Hur överförs data till subrutiner?

Det enklaste sättet är att överföra data via något register.
Det går även att överföra data via minnet.

6.19 INSTRUKTIONER FÖR BITMANIPULERING

I Z80 kan vi manipulera enstaka bitar med 2-bytes eller 4-bytes instruktioner. Med manipulera menar vi här testa, ettställa eller nollställa. Nedan visas 2-bytesinstruktioner.

MNEMONIC	BETYDELSE	KOMMENTAR
BIT b,r	$Z \leftarrow \overline{r_b}$	Testa bit nr b i register r. Z-flaggan påverkas.
BIT b,(HL)	$Z \leftarrow \overline{(HL)_b}$	Testa bit nr b i den minnescell vars adress finns i HL. Z-flaggan påverkas.
SET b,r	$r_b \leftarrow 1$	Sätt bit nr b i register r.
SET b,(HL)	$(HL)_b \leftarrow 1$	Sätt bit nr b i den minnescell vars adress finns i HL.
RES b,r	$r_b \leftarrow 0$	Nollställ bit nr b i register r.
RES b,(HL)	$(HL)_b \leftarrow 0$	Nollställ bit nr b i den minnescell vars adress finns i HL.

Beteckningar:
b anger bitnr i en byte (b= 0 - 7).
r avser något av registren A, B, C, D, E, H, L.
Då parentes finns avses minnescell.
Sätt = ettställ.

Instruktionerna SET och RES påverkar inga flaggor.

6.20 INDEXREGISTER

Z80 innehåller 2 indexregister om 16 bitar, IX och IY.
Med dessa kan vi adressera minnesceller.
Många av de instruktioner vi gått igenom har även en adresseringsmod med indexregister. Adressering med indexregister är ett effektivt sätt när man skall behandla data i tabeller eller datablock.
IX laddas med en basadress. I instruktionskoden anges "displacement", index d. d anges i 2-komplement och ligger mellan -128 och +127.

Ex: LD A,(IX+d)
Ladda A med innehållet i den minnescell vars adress är IX+d.
Om vi tittar på tabellerna över instruktioner på sid 23-37 i ref 6 ser vi att överallt där vi har adresseringsmoden indirekt register-adressering med (HL), kan vi även adressera med indexregister, (IX+d) eller (IY+d).
För instruktioner för dataöverföring av 16 bitar får vi nya instruktioner genom att byta HL mot IX eller IY.
Detsamma gäller för aritmetiska instruktioner för 16 bitar.

6.21 ADRESSERINGSMETODER. ÖVERSIKT

A. Omedelbar adressering (se 6.8.c)

Immediate
En byte, källan, som finns i instruktionen, laddas in i ett register eller en minnescell. Destinationen adresseras med någon annan metod.

Immediate extended
2 bytes, källan, som finns i instruktionen, laddas in i ett registerpar, BC, DE, HL, SP, IX, IY.

B. Modified Page Zero Addressing

Med Page 0 (sida 0) avses de 256 första byten i minnet. Z80 (och 8080) har 8 st speciella enbytes CALL-instruktioner till 8 olika adresser i sida 0.

Instruktion	ger subrutinanrop till adress
RST 0H	0000H
RST 8H	0008H
RST 10H	0010H
RST 18H	0018H
RST 20H	0020H
RST 28H	0028H
RST 30H	0030H
RST 38H	0038H

Instruktionerna består av en byte. Ofta förekommande subrutiner kan placeras vid dessa adresser. RST 10H ger ett subrutinanrop på samma sätt som CALL 0010H. För Z80 kan de även användas för avbrottsrutiner för maskerbara avbrott i mod 0.
RST står för restart.

C. Relativ adressering (se punkt 6.17)

Finns för hopp. Hoppet sker relativt den egna instruktionen.

D. Direkt adressering (se punkt 6.8.d)

Instruktionens adressdel innehåller adressen till operanden i minnet.

E. Adressering med indexregister (se punkt 6.20)

Till instruktionens adressdel adderas innehållet i ett indexregister.

F. Register-adressering (se punkt 6.8.a)

Adressen till ett register ingår i OPKOD:en.

G. Implicerad adress till operand (se punkt 6.10)

I vissa instruktioner, t ex aritmetiska och logiska, är adressen till en operand, A, implicerad (inbyggd).

H. Indirekt register-adressering (se punkt 6.8.b)

Adressen till en operand i minnet finns i ett registerpar, oftast HL.

I. Bitadressering (se punkt 6.19)

Enstaka bitar i ett register eller minnet kan sättas, nollställas eller testas.

Då en instruktion behandlar mer än en operand, kan operanderna adresseras med olika adresseringsmetoder.

6.22 ÖVRIGA INSTRUKTIONER

A. Bytesinstruktioner (Exchanges)

Vi har inte nämnt tidigare att det finns två uppsättningar register i Z80.
Vi har A, F, B, C, D, E, H och L samt

A', F', B', C', D', E', H' och L'.

I en subrutin kan vi byta register i stället för att spara registrens innehåll på stacken.

B. Överföring av block (Block transfer group)

Instruktionerna används för flyttning av datablock i minnet. De är lämpliga vid manipulering av strängar.

C. Instruktioner för sökning (Block search group)

Med dessa instruktioner kan man söka efter en byte i ett block. De är lämpliga vid sökning i en sträng.

D. In-utinstruktioner

Förutom de som nämnts tidigare finns två typer till.
För båda typerna skall portadressen finnas i C.
Den ena typen överför data mellan yttre enhet och ett av registren.
Den andra typen överför ett helt block mellan yttre enhet och minnet.

E. Instruktioner för CPU-kontroll

Det finns några instruktioner för avbrott.

Instruktionerna under 6.22 behövs knappast av dem som skriver program för styrning av yttre enheter.

7. PROGRAMMERINGSUPPGIFTER I Z80-ASSEMBLER

De första uppgifterna nedan har som uppgift att ge förståelse för hur instruktionerna fungerar. De är därför korta. Först när vi gått igenom några instruktionsgrupper kan vi skriva meningsfulla program.

7.1 Exempel på in-ut-instruktioner

K1
Skriv ett program som överför IN0 till UT0 och IN1 till UT2.

7.2 Dataöverföring av 8 bitar

L1
Skriv ett program som överför IN0 till minnescell FFFFH. När vi kört programmet kan vi med ;PEEK titta i minnescellen.

L2
Lägg in talet 37H i A. Överför A till minnescell FFFEH. Överför även A till UT2. Kontrollera resultatet efter körning.

7.3 Dataöverföring av 16 bitar

M1
Lägg in talet 3CH i minnescell FFF0H medelst indirekt register-adressering. Överför sedan innehållet i FFF0H till UT0.

7.4 Aritmetiska instruktioner för 8 bitar

N1
Skriv ett program som ger summan av IN0 och IN1 på UT0.

N2
Skriv ett program som ger summan av IN0 och IN1 på UT0. Dessutom vill vi se flaggornas innehåll på UT2. Detta kan vi åstadkomma genom att först överföra AF till stacken. Sedan flyttar vi tillbaka toppen av stacken till DE. Se även punkt 6.11. Utför även beräkningarna för hand. Studera flaggorna. Kör igenom ett antal fall med olika IN0 och IN1.

N3
Använd uppg N2 och byt operation så att vi får IN0-IN1. Utför beräkningarna för hand. Undersök fall då resultatet blir positivt respektive negativt. Studera flaggorna.

N4
Byt operation i uppg N2 till CP. Undersök resultat och flaggor för ett antal fall.

N5
Överför IN0 till A, öka med 1 och överför resultatet till UT0 och flaggregistret till UT2. Studera några olika fall.

N6
Samma som N5 men minska
med 1 i stället.

7.5 Logiska instruktioner

P1
Överför IN1 till minnescell
FFA0H. Överför IN0 till
A. Utför operationen OCH
mellan A och innehållet i
minnescell FFA0H.
Visa resultatet på UT0
och flaggregistret på UT2.
Utför även beräkningar
för hand. Studera flaggorna.

P2
Samma som P1 men ändra
operationen till ELLER.

P3
Samma som P1 men ändra
operationen till XOR.

P4
Skriv ett program som
utför följande.
Bit 0 och 3 i IN0 skall
överföras till UT0.
Övriga bitar i UT0
skall vara 0.

P5
Skriv ett program som
utför följande.
Bit 1 och 4 i IN0 skall
överföras till UT0.
Övriga bitar i UT0
skall vara 1.

P6
Skriv ett program som
läser in IN1.
Invertera bit 4 och 6
och låt övriga bitar
vara oförändrade.
Visa resultatet på UT2.

P7
Skriv ett program som
utför följande.
Bit 0, 2 och 4 i IN0 och
bit 1, 3 och 5 i IN1 skall
överföras till motsvarande
positioner i UT0. Övriga
bitar i UT0 skall vara 0.

P8
Beskriv hur operationen AND
används för maskning.

P9
Beskriv hur operationen OR
används för maskning.

P10
Beskriv hur operationen
XOR används för
bitmanipulering.

7.6 Specialinstruktioner för ackumulatorn

Q1
Undersök operationen
CPL genom att utgå
från uppg N5.
Byt operation.
Studera flaggorna.

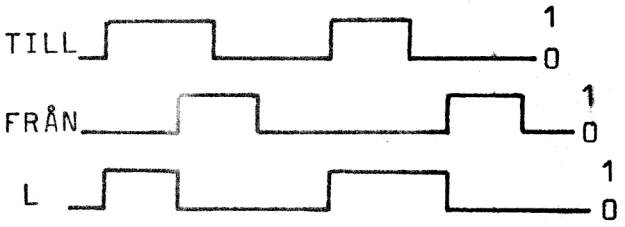
Q2
Undersök operationen NEG
genom att utgå från uppg Q1.
Studera flaggorna.

7.7 Rotationer och skift

R1
Utgå från uppg Q1.
Byt CPL mot olika
instruktioner för
rotation och skift.
Studera flaggorna.

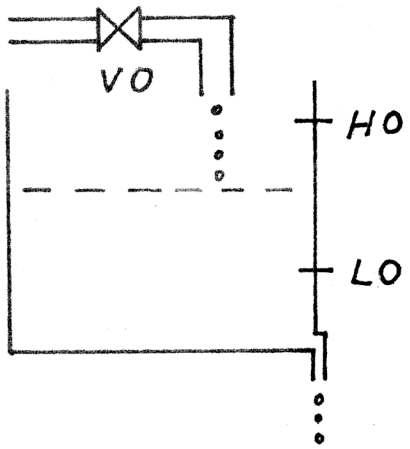
7.8 Allmänna uppgifter

S1
En lampa L skall tändas av en switch TILL och släckas av en switch FRÅN.
Då TILL går från 0 till 1 skall L tändas.
Då FRÅN går från 0 till 1 skall L släckas.
Se även nedanstående figur.
Rita flödesschema.



S2
Samma uppgift som S1 men nu har vi 8 lampor L.
Till varje lampa hör en switch TILL och en switch FRÅN. Rita flödesschema.

S3



Nivån i tanken regleras genom TILL-FRÅN-reglering.
VO=1 öppnar ventilen (bit0 i UT0).
För hög nivå indikeras av att HO (bit0 i IN1) blir 1.
För låg nivå indikeras av att LO (bit0 i IN0) blir 1.
Då nivån blir för låg öppnas ventilen. Då nivån blir för hög stängs ventilen. Observera att HO och LO inte kan bli 1 samtidigt.

S4
Vi har 8 tankar som skall nivåregleras samtidigt.
För L, H och V tar vi lämpligen samma bitnummer i IN0, IN1 och UT0 till en tank med samma nummer.
Till tank nr 4 hör L4, H4 och V4 osv.

S5
Skriv ett program som räknar upp innehållet i UT0 en gång per 0,5s.
Använd programvarumässig fördröjning.
För fördröjningar betydligt kortare än en sekund måste programvarumässig fördröjning användas i ABC80.

S6
Hur känner man av om en tryckknapp är 1-ställd (påverkad)?
Skriv ett program som tänder en lampa (bit 0 i UT0) då tryckknapp T2 påverkas.
Skriv programdelen som känner av T2 på fem olika sätt.

8. LÖSNINGAR TILL PROGRAMMERINGSUPPGIFTER I Z80-ASSEMBLER

De förslag till lösningar som ges nedan utgöres av källkoder erhållna med ABC80 ASSEMBLER från Luxor.

För uppgifterna K1-R1 ligger kortadresseringen i Basicprogrammet.

Demonstrationsprogram på sid 6.8. Källkod.

```
10              ORG 65408
20 DEMON: IN   A,(0)  ; IN0 TILL A
30        OUT  (2),A  ; A TILL UT2
40        RET         ; ÅTER TILL BASIC
```

Basicprogram för demonstrationsprogram på sid 6.8.

```
5 REM DEMONA
6 REM MIKRONIK
10 POKE -128%,219%,0%,211%,2%,201%
30 OUT 1,19
40 X=CALL(65408)
50 GOTO 40
```

K1 Källkod

```
10              ORG 65408
20 K1:    IN   A,(0)  ; IN0 TILL A
30        OUT  (0),A  ; A TILL UT0
40        IN   A,(1)  ; IN1 TILL A
50        OUT  (2),A  ; A TILL UT2
60        RET
```

K1 Basicprogram

```
1 REM K1A
10 POKE -128%,219%,0%,211%,0%,219%,1%,211%,2%,201%
20 OUT 1,19
30 X=CALL(65408)
40 GOTO 30
```

L1

```
10              ORG 65408
20 L1:    IN   A,(0)          ; IN0 TILL A
30        LD   (0FFFFH),A     ; A TILL MINNET
40        RET
```

N1

```
10              ORG 65408
20 N1:    IN   A,(0) ;IN0 TILL A
30        LD   B,A   ;SPARA A I B
40        IN   A,(1) ;IN1 TILL A
50        ADD  A,B   ;A PLUS B TILL A
60        OUT  (0),A ;A TILL UT0
70        RET
```

L2

```
10              ORG 65408
20 L2:    LD  A,37H          ;LADDA A MED 37H
30        LD  (0FFFEH),A     ;A TILL MINNET
40        OUT (2),A          ;A TILL UT2
50        RET
```

N2

```
10              ORG 65408
20 N2:    IN    A,(1) ;IN1 TILL A
30        LD    B,A   ;SPARA A I B
40        IN    A,(0) ;IN0 TILL A
50        ADD   A,B   ;A PLUS B TILL A
60        OUT   (0),A ;A TILL UT0
70        PUSH  AF    ;SPARA AF PÅ STACKEN
80        POP   DE    ;STACKENS TOPP TILL DE
90        LD    A,E   ;E TILL A (FLAGGORNA)
100       OUT   (2),A ;A TILL UT0
110       RET
```

M1

```
10              ORG 65408
20 M1:    LD    HL,0FFF0H ;LADDA HL MED ADRESS
30        LD    (HL),3CH  ;LADDA (HL) MED 3CH
40        LD    A,(HL)    ;ÖVERFÖR (HL) TILL A
50        OUT   (0),A     ;A TILL UT0
60        RET
```

N3

```
10                ORG 65408
20  N3:    IN     A,(1)
30         LD     B,A
40         IN     A,(0)
50         SUB    B
60         OUT    (0),A
70         PUSH   AF
80         POP    DE
90         LD     A,E
100        OUT    (2),A
110        RET
```

N4

```
10                ORG 65408
20  N4:    IN     A,(1)
30         LD     B,A
40         IN     A,(0)
50         CP     B
60         OUT    (0),A
70         PUSH   AF
80         POP    DE
90         LD     A,E
100         OUT    (2),A
110         RET
```

N5

```
10                ORG 65408
20  N5:    IN     A,(0)
30         INC    A
40         OUT    (0),A
50         PUSH   AF
60         POP    DE
70         LD     A,E
80         OUT    (2),A
90         RET
```

N6

```
10                ORG 65408
20  N6:    IN     A,(0)
30         DEC    A
40         OUT    (0),A
50         PUSH   AF
60         POP    DE
70         LD     A,E
80         OUT    (2),A
90         RET
```

P1

```
10                ORG 65408
20  P1:    LD     HL,0FFA0H ;LADDA HL MED ADRESS
30         IN     A,(1)     ;IN1 TILL A
40         LD     (HL),A    ;A TILL MINNET
50         IN     A,(0)     ;IN0 TILL A
60         AND    (HL)      ;A OCH (HL) TILL A
70         OUT    (0),A     ;A TILL UT0
80         PUSH   AF        ;SPARA AF PÅ STACKEN
90         POP    DE        ;STACKENS TOPP TILL DE
100         LD    A,E       ;E TILL A
110         OUT   (2),A     ;A TILL UT2
120         RET
```

P2

```
10                ORG 65408
20  P2:    LD     HL,0FFA0H
30         IN     A,(1)
40         LD     (HL),A
50         IN     A,(0)
60         OR     (HL)
70         OUT    (0),A
80         PUSH   AF
90         POP    DE
100         LD    A,E
110         OUT   (2),A
120         RET
```

P3

```
10                ORG 65408
20  P3:    LD     HL,0FFA0H
30         IN     A,(1)
40         LD     (HL),A
50         IN     A,(0)
60         XOR    (HL)
70         OUT    (0),A
80         PUSH   AF
90         POP    DE
100         LD    A,E
110         OUT   (2),A
120         RET
```

P4

```
10                ORG 65408
20  P4:    IN     A,(0)
30         AND    09H
40         OUT    (0),A
50         RET
```

P5

```
10              ORG 65408
20  P5:   IN    A,(0)
30        OR    0EDH
40        OUT   (0),A
50        RET
```

P6

```
10              ORG 65408
20  P6:   IN    A,(1)
30        XOR   50H
40        OUT   (2),A
50        RET
```

P7

Jämför lösning till uppg C8 sid 5.4.

```
10              ORG 65408
20  P7:   IN    A,(1)
30        AND   2AH
40        LD    B,A     ; SPARA I B
50        IN    A,(0)
60        AND   15H
70        OR    B
80        OUT   (0),A
90        RET
```

P8-P10

Se uppg C12-C14 sid 5.7.

Q1

```
10              ORG 65408
20  Q1:   IN    A,(0)
30        CPL
40        OUT   (0),A
50        PUSH  AF
60        POP   DE
70        LD    A,E
80        OUT   (2),A
90        RET
```

Q2

```
10              ORG 65408
20  Q2:   IN    A,(0)
30        NEG
40        OUT   (0),A
50        PUSH  AF
60        POP   DE
70        LD    A,E
80        OUT   (2),A
90        RET
```

R1

```
10              ORG 65408
20  R1:   IN    A,(0)
30        RLCA
40        OUT   (0),A
50        PUSH  AF
60        POP   DE
70        LD    A,E
80        OUT   (2),A
90        RET
```

S1

Se även uppg E1 sid 5.9.

```
10              ORG 65408
20  ;TILL:IN0 BIT 0, FRÅN:IN1 BIT 0
30  S1:   LD    A,13H   ;ADRESSERA
40        OUT   (1),A   ;KORT
50                      ;
60  V1:   IN    A,(0)   ;IN0 TILL A
70        AND   1       ;KÄNN AV BIT 0
80        JP    NZ,V1   ;(TILL)
90        CALL  TID     ;ANTISTUDS
100                     ;
110 V2:   IN    A,(0)
120       AND   1
130       JP    Z,V2
140       CALL  TID
150                     ;
160       LD    A,1     ;TÄND
170       OUT   (0),A   ;LAMPA
180                     ;
190 V3:   IN    A,(1)   ;IN1 TILL A
200       AND   1       ;KÄNN AV BIT 0
210       JP    NZ,V3   ;(FRÅN)
220       CALL  TID     ;ANTISTUDS
230                     ;
240 V4:   IN    A,(1)
250       AND   1
260       JP    Z,V4
270       CALL  TID
280                     ;
290       SUB   A       ;SLÄCK
300       OUT   (0),A   ;LAMPA
310                     ;
320       JP    V1
```

```
330                     ;
340 TID:  LD    C,8H    ;ANTISTUDS
350 T1:   LD    B,0FFH  ;TAR 17 MS
360 HOPP: IN    A,(7)   ;TAR TID
370       DEC   B
380       JP    NZ,HOPP
390       DEC   C
400       JP    NZ,T1
410       RET           ;
```

S2
Lösningsmetod framgår av uppg E2 sid 5.10.
I lösningen nedan anges att registren B och D används för att lagra gamla värden på TILL respektive FRÅN. De används även till att lagra mellanresultat (rad 250 och 300) efter det att gamla TILL och FRÅN använts.

```
10              ORG 65408
20 S2:
30    ;B GAMLA TILL
40    ;C NYA TILL
50    ;D GAMLA FRÅN
60    ;E NYA FRÅN
70    ;L LAMPTILLSTÅND
80      LD   A,13H    ;ADRESSERA
90      OUT  (1),A    ;KORT
100                   ;
110     SUB  A        ;SLÄCK
120     OUT  (0),A    ;LAMPOR
130     OUT  (2),A
140     IN   A,(0)    ;KÄNN AV
150     LD   B,A      ;GAMLA TILL
160     IN   A,(1)    ;KÄNN AV
170     LD   D,A      ;GAMLA FRÅN
180                   ;
190 V1: IN   A,(1)    ;KÄNN AV NYA FRÅN
200     LD   E,A      ;SPARA I E
210     IN   A,(0)    ;KÄNN AV NYA TILL
220     LD   C,A      ;SPARA I C
230     XOR  B        ;ÄNDRING I TILL
240     AND  C        ;ÄNDRING 0-1 I TILL
250     LD   B,A      ;SPARA I B
260     LD   A,E      ;NYA FRÅN I A
270     XOR  D        ;ÄNDRING I FRÅN
280     AND  E        ;ÄNDRING 0-1 I FRÅN
290     CPL           ;INVERTERA
300     LD   D,A      ;SPARA I D
310     LD   A,L      ;LAMPTILLSTÅND
320     OR   B        ;TÄND FÖR TILL 0-1
330     AND  D        ;SLÄCK FÖR FRÅN 0-1
340     OUT  (0),A    ;A TILL UT0
350     LD   L,A      ;SPARA LAMPTILLSTÅND
360     LD   B,C      ;NYA TILL BLIR GAMLA
370     LD   D,E      ;NYA FRÅN BLIR GAMLA
380                   ;
390 T1: LD   H,0FFH   ;ANTISTUDS
400 T2: IN   A,(7)
410     DEC  H
420     JP   NZ,T2    ;TAR 2.1 MS
430     JP   V1
```

S3
Lösningsmetod framgår av uppg E10 sid 5.18.

```
10              ORG 65408
20  ;LÅG:IN0 BIT 0, HÖG:IN1 BIT 0
30 S3:    LD   A,13H  ;ADRESSERA
40        OUT  (1),A  ;KORT
50 VARV:  IN   A,(0)  ;KÄNN AV
60        AND  1      ;LÅG NIVÅ
70        JP   Z,H1
80                    ;
90        LD   A,1    ;ÖPPNA
100       OUT  (0),A  ;VENTIL
110                   ;
120 H1:   IN   A,(1)  ;KÄNN AV
130       AND  1      ;HÖG NIVÅ
140       JP   Z,H2
150                   ;
160       SUB  A      ;STÄNG
170       OUT  (0),A  ;VENTIL
180                   ;
190 H2:   JP   VARV
```

S4
Lösningsmetod framgår av uppg E11 sid 5.19.

```
10              ORG 65408
20 S4:    LD   A,13H  ;ADRESSERA
30        OUT  (1),A  ;KORT
40        SUB  A      ;STÄNG
50        OUT  (0),A  ;VENTILER
60        LD   B,A    ;GAMLA VENTILTILLSTÅND
70                    ;
80 VARV:  IN   A,(1)  ;KÄNN AV HÖG NIVÅ
90        CPL         ;INVERTERA
100       LD   C,A    ;SPARA I C
110                   ;
120       IN   A,(0)  ;KÄNN AV LÅG NIVÅ
130       OR   B      ;ÖPPNA VID LÅG NIVÅ
140       AND  C      ;STÄNG VID HÖG NIVÅ
150       OUT  (0),A  ;A TILL UT0 (VENTILER)
160       LD   B,A    ;SPARA VENTILTILLSTÅND
170                   ;
180 T1:   LD   D,0FFH ;ANTISTUDS
190 T2:   IN   A,(7)
200       DEC  D
210       JP   NZ,T2  ;TAR 2.1 MS
220                   ;
230       JP   VARV
240 LIST
```

S5

```
10              ORG 65408
20 S5:    LD    A,13H  ;ADRESSERA
30        OUT   (1),A  ;KORT
40                     ;
50 VARV:  SUB   A      ;SLÄCK
60        OUT   (0),A  ;LAMPA
70        LD    B,A    ;NOLLSTÄLL
80                     ;RÄKNEVERK
90                     ;
100 TID:  LD    HL,0A724H ;42788D VARV
110 VÄNT: IN    A,(7)  ;11 STATES
120       DEC   HL     ; 6   "
130       LD    A,H    ; 4   "
140       OR    L      ; 4   "
150       JP    NZ,VÄNT;10   "
160   ;ETT VARV TAR 35 STATES
170   ;42788 VARV TAR 1497580 STATES
180   ;ETT STATE TAR 0,29952 MIKROSEK
190   ;TID TAR 0.5 SEK
200                    ;
210       INC   B      ;ÖKA RÄKNEVERK
220       LD    A,B    ;RÄKNEVERK
230       OUT   (0),A  ;TILL UT0
240       JP    TID
```

S61

Sätt alla bitar utom nr 7 till 0. Hoppa på Z-flaggan.

```
10              ORG 65408
20 S61:   LD    A,13H  ;ADRESSERA
30        OUT   (1),A  ;KORT
40        SUB   A      ;SLÄCK
50        OUT   (0),A  ;LAMPA
60                     ;
70 H1:    IN    A,(1)  ;KÄNN
80        AND   80H    ;AV
90        JP    Z,H1   ;T2
100                    ;
110        LD    A,1   ;TÄND
120       OUT   (0),A  ;LAMPA
130       RET
```

S62

Rad 70-90 ovan ersättes med nedanstående rader. Rotera bit 7 till CY.

```
70 H1:    IN    A,(1)  ;KÄNN
80        RLCA         ;AV
90        JP    NC,H1  ;T2
```

S63

I tredje varianten sätts S-flaggan=bit 7 genom instruktionen OR A.

```
70 H1:    IN    A,(1)  ;KÄNN
80        OR    A      ;AV
90        JP    P,H1   ;T2
```

S64

I fjärde varianten används instruktionen BIT 7,A så att Z-flaggan påverkas.

```
70 H1:    IN    A,(1)  ;KÄNN
80        BIT   7,A    ;AV
90        JP    Z,H1   ;T2
```

S65

I de fyra varianterna ovan känns nivån av. I femte varianten känns övergången 0-1 av.

```
10              ORG 65408
20      ;C NYA, D GAMLA
30 S65:   LD    A,13H  ;ADRESSERA
40        OUT   (1),A  ;KORT
50        SUB   A      ;SLÄCK
60        OUT   (0),A  ;LAMPA
70        LD    D,A    ;GAMLA=0
80                     ;
90 H1:    IN    A,(1)  ;KÄNN
100       AND   80H    ;AV
110       LD    C,A    ;SPARA NYA
120       XOR   D      ;ÄNDRINGAR
130       AND   C      ;ÄNDRING TILL 1
140       LD    D,C    ;NYA BLIR GAMLA
150       JP    Z,H1   ;T2
160                    ;
170       LD    A,1    ;TÄND
180       OUT   (0),A  ;LAMPA
190       RET
```

9. BESKRIVNING AV DIGITALKORT-0

Kretsschemat finns på sid 15.2 och 15.3.
Kortet adresseras med kommandot OUT 1,A. Adressen A läggs ut på bussen av datorn. I busskretsen IC7 jämförs adressen A med en adress angiven av byglar B11, se sid 1.9.
Om adresserna är lika 1-ställs en vippa i IC7 då datorn ger en puls på stift 7 i IC7 via A23. Den 1-ställda vippan orsakar att stift 9 i IC7 blir 0. Härvid får ett antal kretsar frisignal. Lysdioden L0 indikerar att kortet är inkopplat. Sedan ett kort adresserats kan det nås av datorn ända tills nästa kort adresseras.
I IC3 och IC4 finns 8 flanktriggade D-vipper.
Med kommandot OUT 0,X lägger datorn ut 8 databitar på databussen. Under denna tid ger datorn en puls via A22 till stift 11 på IC4 så att data överförs till vipporna i denna krets.
IC1 och IC2 används som drivkretsar till lysdioderna, se fig 9.1. Då en vippa 1-ställs blir buffertkretsens utgång 0 så att ström flyter genom lysdioden som därigenom lyser.
Antag att spänningen på buffertkretsens utgång är 0,2V för logisk nolla samt spänningen över lysdioden 1,5V. Spänningen över R blir då 3,3V och strömmen genom lysdioden 7 mA. Testvärden för lysdioder brukar anges för 16 mA. Vi har valt 470 ohm för att begränsa strömmen från 5V-matningen då vi tycker lysdioderna lyser tillräckligt starkt med 7 mA.

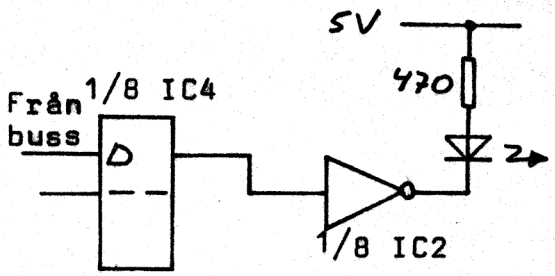


Fig 9.1

IC5 och IC6 är buffertkretsar med TRI-STATE-utgångar. Detta innebär att utgångarna som är anslutna till bussen normalt är flytande eller högimpediva.
Med instruktionen X=INP(0) läses data in. Datorn ger via A17 en signal till kontrollingångarna på IC6 så att data kommer ut på bussen. Datorn passar på att läsa in data.
Buffertkretsen innehåller invertering eftersom en ettställd switch ger en nolla på ingången.

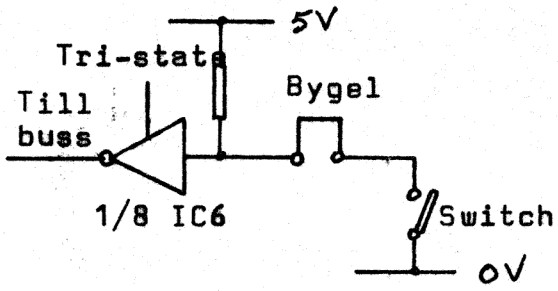


Fig 9.2

Som alternativ till switchar kan två tryckknappar kopplas in, se sid 1.3.

Kortet är lämpligt för övning på in-utrutiner.
Många program avsedda för styrning av yttre enheter kan simuleras på kortet.

10. BESKRIVNING AV DIGITALKORT-0-16TTL

Detta kort utgör en förlängning av Digitalkort-0.
Förutom det som finns på Digitalkort-0 finns
8 TTL-ingångar och 8 TTL-utgångar.

TTL-utgångar

Kretsarna IC2 och IC10 ligger parallellt. Data i IC4 (UTO) visas på lysdioderna och kan även tas ut via IC10 som är en buffertkrets utan invertering.
Anslutning framgår av sid 1.7.

Data för utgångarna:

Utgång LÅG: $V_{OL} < 0,5V$ Utgång HÖG: $V_{OH} > 2,7V$

$I_{OL} > 16$ mA $|I_{OH}| > 2,6$ mA

Överföringen från datorn sker utan invertering.
En logisk ETTA ut är större än 2,7V och
en logisk NOLLA ut är mindre än 0,5V.
En utgång kan sänka 16 mA.

TTL-ingångar

IN0 kan med byglar kopplas till 8 TTL-ingångar i stället för switchar. Anslutning till yttre enhet framgår av sid 1.7.

Data för ingångarna:

Ingång LÅG: $V_{IL} < 0,8V$

$|I_{IL}| < 0,36$ mA för IC6 plus 0,5 mA genom pull-up-resistor i SI5

Ingång HÖG: $V_{IH} > 2,0V$

$I_{IH} < 20\ \mu A$

En logisk NOLLA in skall alltså vara mindre än 0,8V och en logisk ETTA in skall vara större än 2,0V.
Till en yttre utgång rinner strömmen 0,36+0,5=0,86 mA i tillstånd LÅG.

Signaltillstånd

Signal från yttre enhet	Signal till datorn
LÅG	HÖG
HÖG	LÅG

Buffertkretsen IC6 ger invertering.

11. BESKRIVNING AV DIGITALKORT-32TTL

Kretsschemat framgår av sid 15.2 och 15.6.

På detta kort finns 2X8 TTL-ingångar och 2X8 TTL-utgångar.

Utgångar

På utgångssidan är IC3 och IC4 samma som på Digitalkort-0. Buffertkretsarna IC1 och IC2 är utan invertering. Utgången från en vippa skall inte anslutas till en ledning. Eventuella reflexioner på ledningen skulle kunna få vippan att slå om. Därför finns buffertkretsarna med. Anslutning framgår av sid 1.9.

Data för utgångarna:

Utgång LÅG: $V_{OL} < 0,5V$ Utgång HÖG: $V_{OH} > 2,7V$

$I_{OL} > 16$ mA $|I_{OH}| > 2,6$ mA

Överföringen från datorn sker utan invertering. En logisk ETTA ut är större än 2,7V och en logisk NOLLA ut är mindre än 0,5V. En utgång kan sänka 16 mA.

Ingångar

På ingångssidan är ingångarna anslutna direkt till IC5 och IC6. Dessa buffertkretsar är utan invertering.

Data för ingångarna:

Ingång LÅG: $V_{IL} < 0,8V$ Ingång HÖG: $V_{IH} > 2,0V$

$|I_{IL}| < 0,36$ mA $I_{IH} < 20\ \mu A$

En logisk NOLLA in skall alltså vara mindre än 0,8V och en logisk ETTA in skall vara större än 2,0V. Överföringen till datorn sker utan invertering.

12. BESKRIVNING AV DIGITALKORT-T OCH DIGITALKORT-T-PUR

Kretsschemat finns på sid 15.2-15.5.

Kortets vänstra del är samma som Digitalkort-0.

12.1 Ingångar

INO kan med byglar kopplas till 8 optoisolerade ingångar. Nedan ges exempel på beräkningar på sådana ingångar.

Problem 1.

Hur skall R1, R2 och R3 i fig 12.1 väljas så att säker funktion erhålles men strömmen in ändå blir måttlig?

Lösning.

Beräkningarna göres här för VEXT=24V nominellt. Denna spänning skall ge en logisk ETTA till datorn på bussen. Låg spänning skall ge en logisk NOLLA på bussen.

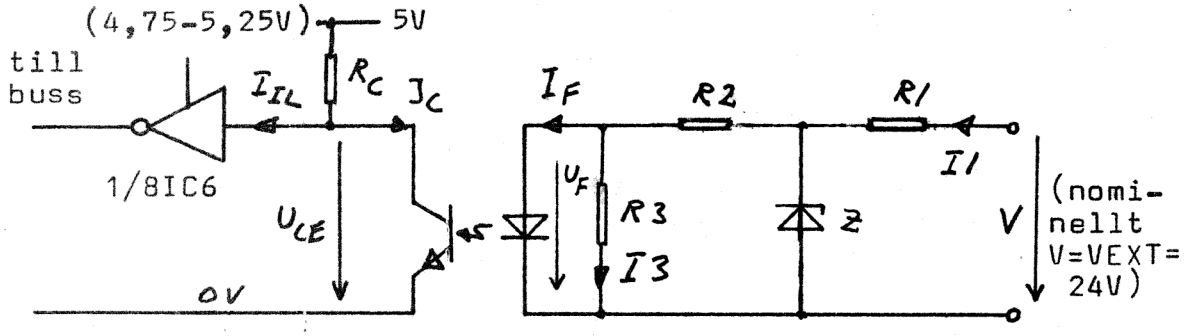


Fig 12.1 Optoisolerad ingång

För optokopplaren finns datablad i punkt 14.
En ström genom lysdioden ger upphov till en ström genom fototransistorn. Optokopplare används bland annat då man vill ha galvanisk isolering mellan ingång och utgång.

För optokopplare definieras CTR = $\frac{I_C}{I_F}$ (Current Transfer Ratio)[1].

Av formeln framgår att ju större I_C är desto större I_F krävs för att bottna transistorn. För att hålla I_F låg väljes R_C ganska hög. Vi väljer R_C = 10 kohm.

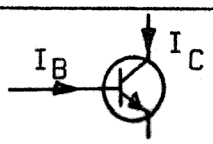
För logisk 0 väljer vi $U_{CE} \leq 0{,}4V$. För TTL-kretsen IC6 gäller max$|I_{IL}|$ = 0,36 mA.

Vi beräknar först max I_C vid bottning och sedan det I_F som krävs för att bottna transistorn.

Max $I_C = \frac{5{,}25-0{,}4}{10} + 0{,}36 = 0{,}85$ mA. (5,25V är max tillåten spänning för TTL-kretsar i drift).

---

[1] Jämför strömförstärkningen $B=\frac{I_C}{I_B}$.



Databladet för optokopplaren ger CTR = 0,2 - 1,5, (typ 0,5), således ganska vida toleranser.
Vi måste räkna med sämsta fallet CTR=0,2.

För I_C = 0,85 mA krävs I_F = 0,85/0,2 = 4,25 mA för säker bottning.

I_F = 4,25 mA ger U_F = 1,15V ur databladet.

För att inte små strömmar på ingången skall ge felfunktion kopplas R3 in. Se problem 2 samt "Förslag på mätningar" nedan.

Strömmen I_F är försumbart liten för $U_F < 0,9V$.

Välj R3 = 470 ohm. (330 ohm är också vanligt).

Med U_F = 1,15V blir I3 = 1,15/0,47 = 2,44 mA.

Med I_C = 0,85 mA krävs alltså I1 = 4,25 + 2,44 = 6,7 mA för säker bottning.

Vi väljer att VEXT = 20V skall räcka för att ge $U_{CE} \leq 0,4V$.

$$R1 + R2 = \frac{20-1,15}{6,7} = 2,8 \text{ kohm.}$$

Av databladet för lysdioden framgår att dioden tål en viss framström och en viss backspänning.
Därför kopplas en zenerdiod in. Vi väljer R2 och zenerspänning så att ingen ström går genom zenerdioden för nominellt VEXT.
Zenerdioden skyddar lysdioden för både för hög framspänning och för hög backspänning.

Välj R2 = 470 ohm. Med R1 + R2 = 2,8 kohm kan vi välja R1 = 2,2 kohm. Tolerans: 5%.

Med R_C = 10 kohm, R1 = 2,2 kohm, R2 = 470 ohm och R3 = 470 ohm blir enligt ovanstående beräkningar $U_{CE} \leq 0,4V$ då VEXT $\geq$ 20V med sämsta optokopplaren med CTR = 0,2.

Problem 2.

För vilken inspänning V erhålles logisk ETTA på bussen med typvärden på komponenterna, bl a CTR = 0,5?
Resistorvärden enligt ovan.

Lösning.

Antag TTL-kretsen slår om för U_{CE} = 1,4V och typ $|I_{IL}|$ = 0,3 mA.

$$I_C = \frac{5-1,4}{10} + 0,3 = 0,66 \text{ mA.}$$

I_F = 0,66/0,5 = 1,32 mA krävs. Detta ger U_F = 1,1V.

I3 = 1,1/0,47 = 2,34 mA. I1 = 1,32 + 2,34 = 3,66 mA.

V = 1,1 + 3,66(0,47+2,2) = 10,9V.

För komponenter med typvärden kan man vänta sig en logisk ETTA på bussen för inspänningar större än 11V då nominellt VEXT = 24V.
Utan R3 skulle en logisk ETTA erhållas för inspänningar större än 4,6V.

Problem 3.

Hur stor blir inströmmen för nominell inspänning?
Hur skall zenerdioden väljas?
Räkna med nominella värden på komponenterna.

Lösning.

$U_F = 1,2V.$ $I1 = \frac{24-1,2}{0,47+2,2} = 8,5$ mA.

Inströmmen är 8,5 mA.

Spänningen över zenerdioden blir 1,2 + 8,5·0,47 = 5,2V.

Zenerdiod 5,6V 0,4W väljes. Den leder ström först vid inspänningar större än nominellt värde.

Kommentar.

Vid leveranstestning accepteras bara kretsar som ger logisk ETTA på bussen för inspänningar från och med 16V för VEXT = 24V nominellt.
På sid 1.4 anges max VEXT. Dessa är valda så att effektförlusten är något lägre än vad komponenterna tål.

Förslag på mätningar.

Anslut en variabel inspänning V till en optoisolerad ingång.
Mät
① U_{CE} för V = 0V (IC18 stift 7 för bit 0)
② U_{CE} för V = 24V (V=VEXT nominellt)
③ V för $U_{CE} = V_{OH} - 0,1$ V (övre knäpunkt)
④ V för $U_{CE} = 0,4$ V (nedre knäpunkt)

Skriv ett program som visar ingångens tillstånd på skärmen. Uppsök det värde på V som ger en logisk ETTA på skärmen.
Skissa överföringsdiagram enligt fig 12.2.

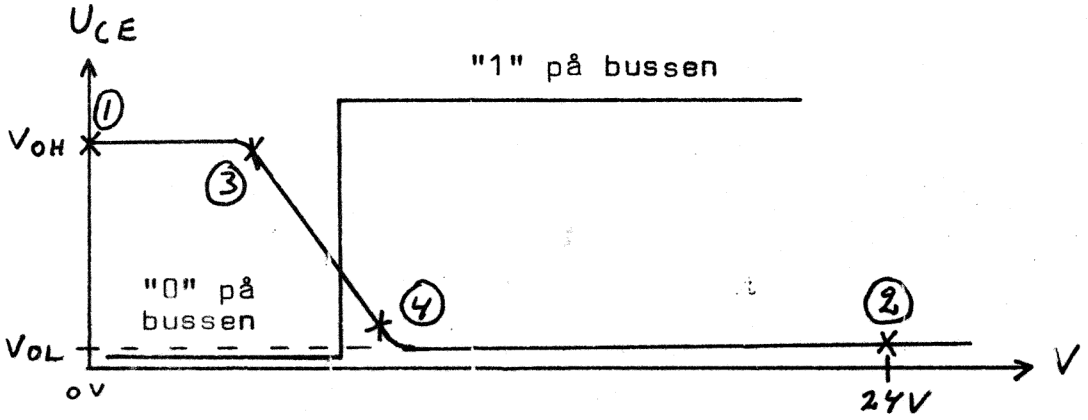


Fig 12.2 Överföringsdiagram för optoisolerad ingång.
Beteckningar se fig 12.1.

Avståndet mellan ① och ③ kan ses som en störmarginal.
Utan R3 är denna betydligt mindre.

Den spänningsändring som behövs i V för att komma från ③ till ④ beror på CTR.

12.2 Transistorutgång för Digitalkort-I

En logisk ETTA från bussen medför att lasten får ström.

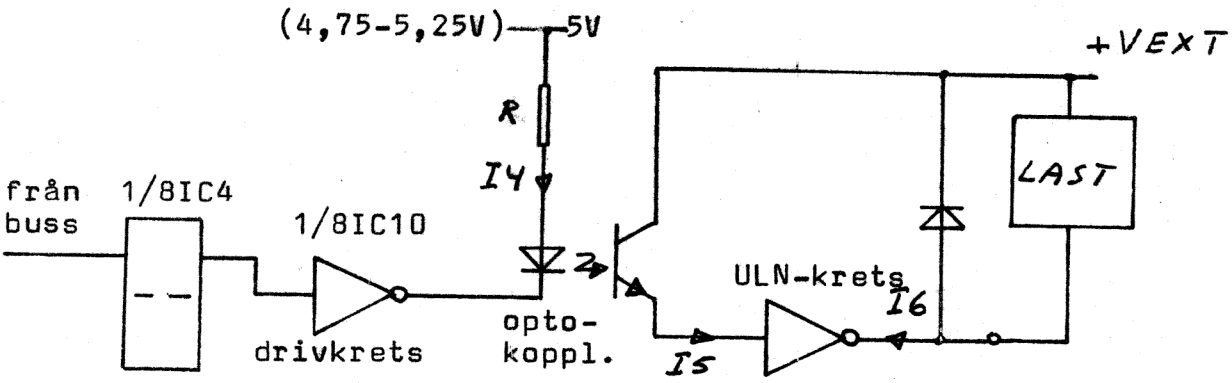


Fig 12.3 Optoisolerad strömsänkande transistorutgång.

Problem.

Hur skall R väljas?

Lösning.

Datablad för ULN-kretsen och optokopplaren finns i punkt 14. För ULN-kretsen gäller att I5 = 0,5 mA på ingången räcker för att bottna transistorn på utgången då I6 = 350 mA.

En logisk ETTA från bussen ger en logisk NOLLA på utgången av drivkretsen IC10, som ger invertering.
Antag spänningen på drivkretsens utgång är 0,4V för logisk NOLLA. Vidare antar vi att matningsspänningen är 4,75V (undre gränsen för TTL) samt spänningen över lysdioden 1,15V, se datablad och punkt 12.1.
Med sämsta optokopplaren, CTR = 0,2, behövs strömmen I4 = 0,5/0,2 = 2,5 mA, då strömmen genom fototransistorn är 0,5 mA.

Vi får $R = \frac{4,75-1,15-0,4}{2,5} = 1,28$ kohm för sämsta fallet.

Välj R = 470 ohm, 5%.

För sämsta fallet blir $I4 = \frac{4,75-1,1-0,4}{0,47 \cdot 1,05} = 6,5$ mA.

Detta räcker till en ström genom fototransistorn I5 = 6,5·0,2 = 1,3 mA. I svåraste fallet är strömmen in på ULN-kretsen 1,3 mA vilket räcker för säker bottning av utgångstransistorn.

Den i fig 12.3 utritade dioden finns i ULN-kretsen. Om lasten är induktiv klingar strömmen av till noll genom denna diod vid frånslag.

ULN-kretsen innehåller ett darlingtonpar på utgången. Utgångstransistorn kan därför inte bottnas till en så låg spänning som en enstaka transistor.

ULN-kretsen tål max 50V på utgången och 30V på ingången. Fototransistorn tål max 30V. Optokopplaren har valts med hänsyn till detta samt min CTR och pris. Även stift-konfigurationen är av intresse. (4 stift per optokopplare).

12.3 Transistorutgång för Digitalkort-T-PUR

Jämfört med Digitalkort-T finns en pull-up-resistor på utgången. En logisk ETTA från bussen ger därför HÖG utgång. Detta åstadkommes genom att drivkretsen IC10 är utan invertering. Jämför fig 2 och fig 3 på sid 1.5.

12.4 Anslutning av TTL-signaler till yttre enheter

På sid 1.8 visas hur man kan få 8 TTL-utgångar och 8 TTL-ingångar.

Data för utgångarna:

Utgång HÖG: $V_{OH} > 2,7V$ Utgång LÅG: $V_{OL} < 0,5V$

$|I_{OH}| > 2,6$ mA $I_{OL} > 16$ mA

Detta innebär att för en logisk ETTA ut är spänningen större än 2,7V. Utgången klarar därvid av en ström ut om minst 2,6 mA.
För en logisk NOLLA ut är spänningen mindre än 0,5V och utgången klarar att sänka minst 16 mA.

Data för ingångarna:

Ingång LÅG: $V_{IL} < 0,8V$

$|I_{IL}| < 0,36$ mA för IC6 plus 0,5 mA genom pull-up-resistorn.

Ingång HÖG: $V_{IH} > 2,0V$

$I_{IH} < 20\ \mu A$

En yttre utgång ansluten till en TTL-ingång på kortet får i läge LÅG sänka strömmen 0,86 mA.

Signaltillstånd

Yttre TTL-signal	Bussignal Digitalkort-T	Bussignal Digitalkort-T-PUR
INGÅNGAR		
LÅG	HÖG	HÖG
HÖG	LÅG	LÅG
UTGÅNGAR		
LÅG	HÖG	LÅG
HÖG	LÅG	HÖG

TTL-utgångarna tas från drivkrets (IC10), se fig 2 och 3 sid 1.5.

13. BESKRIVNING AV DIGITALKORT-R

Kretsschemat finns på sid 15.2-15.4.

Kortets vänstra del är samma som Digitalkort-O.

13.1 Ingångar

Digitalkort-R har samma optoisolerade ingångar som Digitalkort-T, se punkt 12.1.

13.2 Utgångar

På kortet finns 8 reläutgångar. Reläerna drivs av de transistorutgångar som finns på Digitalkort-T. Reläerna har drivspänningen 5V. Reläkontakterna ger galvanisk isolering mellan 5V-matning för elektronikkretsarna och VEXT. Därför behövs inte optokopplare. Jämför fig 2 på sid 1.5 och fig 4 på sid 1.6. Vid frånslag klingar strömmen genom en reläspole av till noll genom diod i ULN-kretsen. Ett relä slår till för en drivspänning om c:a 3V. Spänningsfallet över ULN-kretsen är c:a 1V. Minsta acceptabla matningsspänning, 4,75V för TTL-kretsar, räcker alltså väl till för säkert tillslag. Reläspolen drar vid 5V-matning 65 mA. ULN-kretsen drivs direkt från drivkretsen IC10.

Över reläkontakterna sitter zenerdioder, se fig 4 sid 1.6. Dessa skyddar mot störningar vid vanliga induktiva laster såsom magnetventiler. Prov har visat att en last utgörande spole med järnkärna med strömmen 1A inte stör elektronik-kretsarna. Däremot uppstår gnistbildning över relä-kontakterna vid frånslag.
På sid 1.6 anges att induktiva laster som drar strömmar mer än 0,3-0,4 A skall förses med frihjulsdioder. Denna gräns har satts med hänsyn till reläkontakterna.
En likströmsmotor skall avstöras med kondensator, se sid 1.6.

13.3 Anslutning av TTL-signaler till yttre enheter

På sid 1.8 visas hur man kan få 8 TTL-utgångar och 8 TTL-ingångar.

Data för utgångarna:

Utgång HÖG: $V_{OH} > 2,7V$ Utgång LÅG: $V_{OL} < 0,5V$

$|I_{OH}| > 2,6$ mA $I_{OL} > 16$ mA

Detta innebär att för en logisk ETTA ut är spänningen större än 2,7V. Utgången klarar därvid av en ström ut om minst 2,6 mA.
För en logisk NOLLA ut är spänningen mindre än 0,5V och utgången klarar att sänka minst 16 mA.

Data för ingångarna:

Ingång LÅG: $V_{IL} < 0,8V$

$|I_{IL}| < 0,36$ mA för IC6 plus 0,5 mA genom pull-up-resistorn.

Ingång HÖG: $V_{IH} > 2,0V$

$I_{IH} < 20$ µA

En yttre utgång ansluten till en TTL-ingång på kortet får i läge LÅG sänka strömmen 0,86 mA.

Signaltillstånd:

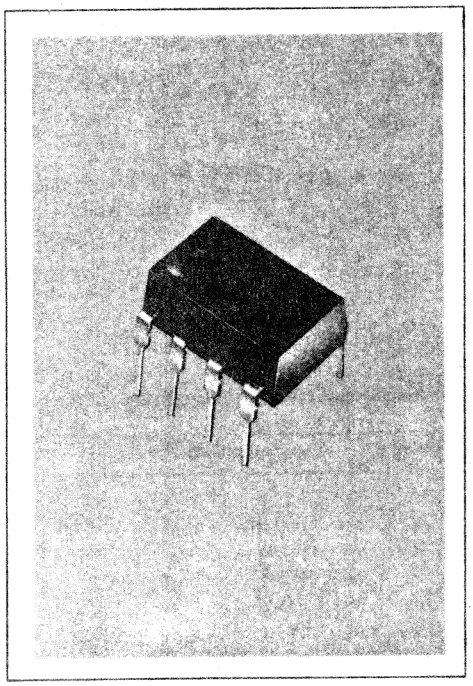
Yttre TTL-signal	Bussignal
INGÅNG	
LÅG	HÖG
HÖG	LÅG
UTGÅNG	
LÅG	LÅG
HÖG	HÖG

TTL-utgångarna tas från drivkrets (IC10), se fig 4 sid 1.6.

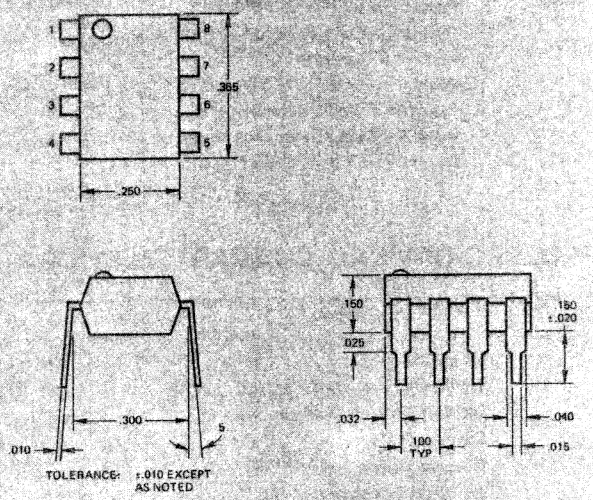
litronix

Iso-Lit CT6

PHOTOTRANSISTOR OPTO-ISOLATOR



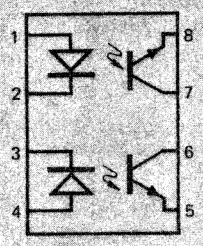
Package Dimensions (in inches)



FEATURES

- Two Isolated Channels Per Package
- 1500V Isolation
- 50% Typical Current Transfer Ratio
- 1 nA Typical Leakage Current
- Direct Replacement For MCT6

Pin Configuration



LED CHIPS ON PINS 2 AND 3
PT CHIPS ON PINS 6 AND 7

PIN NO.	FUNCTION
1	ANODE
2	CATHODE
3	CATHODE
4	ANODE
5	EMITTER
6	COLLECTOR
7	COLLECTOR
8	EMITTER

DESCRIPTION

The IL-CT6 is a two channel opto isolator for high density applications. Each channel consists of an optically coupled pair employing a Gallium Arsenide infrared LED and a silicon NPN phototransistor. Signal information, including a DC level, can be transmitted by the device while maintaining a high degree of electrical isolation between input and output. The IL-CT6 is especially designed for driving medium-speed logic where it may be used to eliminate troublesome ground loop and noise problems. It can also be used to replace relays and transformers in many digital interface applications, as well as analog applications such as CRT modulation.

European Headquarters:
Litronix, Bevan House, Bancroft Court, Hitchin, Herts. SG5 1LW, England. Telephone: Hitchin 2676 Telex: 825497.
LITRONIX INC. 19000 HOMESTEAD ROAD / VALLCO PARK / CUPERTINO, CALIF. 95014 / (408) 257-7910 / TWX 910-338-0022

MAXIMUM RATINGS

Maximum Temperatures
- Storage Temperature ..... −55°C to +150°C
- Operating Temperature ..... −55°C to +100°C
- Lead Temperature (Soldering, 10 seconds) ..... 250°C

Input Diode (each channel)
- Rated Forward Current, DC ..... 60 mA
- Peak Forward Current (1μs pulse, 300 pps) ..... 3 A
- Power Dissipation at 25°C Ambient ..... 100 mW
- Derate Linearly From 50°C ..... 2 mW/°C

Output Transistor (each channel)
- Power Dissipation @ 25°C Ambient ..... 150 mW
- Derate Linearly From 25°C ..... 2 mW/°C
- Collector Current ..... 30 mA

Coupled
- Input to Output Breakdown Voltage ..... 1500 Volts DC
- Total Package Power Dissipation @ 25°C Ambient ..... 400 mW
- Derate Linearly From 25°C ..... 5.33 mW/°C

ELECTRO-OPTICAL CHARACTERISTICS (25°C Free Air Temperature Unless Otherwise Specified)

Parameter	Min	Typ	Max	Units	Test Conditions
Input Diode					
Threshold Forward Voltage		0.7		V	ΔI_C = 1.0 nA
Rated Forward Voltage		1.25	1.50	V	I_F = 60 mA
Reverse Voltage	3.0	5.0		V	I_R = 10 μA
Reverse Current		0.001	10	μA	V_R = 3.0 V
Junction Capacitance		100		pF	V_F = 0V
Rise Time, Fall Time		10		ns	I_F = 50 mA, 50Ω System
Output Transistor (I_F = 0)					
Breakdown Voltage, Collector to Emitter	30	65		V	I_C = 1.0 mA
Emitter to Collector	7.0	13		V	I_C = 100 μA
Leakage Current, Collector to Emitter		1.0	100	nA	V_{CE} = 10V
Capacitance Collector to Emitter		8.0		pF	V_{CE} = 0V
Coupled					
DC Current Transfer Ratio (I_C/I_F)	20	50	150	%	V_{CE} = 10 V, I_F = 10 mA
Isolation Voltage	1500	2500		V	
Isolation Resistance		10^{11}		Ω	V_{I-O} = 500 V
Isolation Capacitance		0.5		pF	f = 1.0 MHz
Breakdown Voltage — Channel-to-Channel		1500		V	Relative Humidity = 40%
Capacitance Between Channels		0.4		pF	f = 1.0 MHz
Saturation Voltage — Collector to Emitter			0.40	V	I_C = 2.0 mA, I_F = 16 mA
Bandwidth		150		KHz	I_C = 2.0 mA, V_{CC} = 10V, R_L = 100 Ω
Switching Times, Output Transistor					
Non-Saturated Rise Time, Fall Time		2.4		μs	I_C = 2.0 mA, V_{CE} = 10V, R_L = 100 Ω
Non-Saturated Rise Time, Fall Time		15		μs	I_C = 2.0 mA, V_{CE} = 10V, R_L = 1.0 KΩ
Saturated Turn-On Time (From 5.0 V to 0.8 V)		5.0		μs	R_L = 2.0 kΩ, I_F = 15 mA
Saturated Turn-Off Time (From Saturation to 2.0V)		25		μs	R_L = 2.0 KΩ, I_F = 15 mA

TYPICAL OPTO-ELECTRONIC CHARACTERISTIC CURVES FOR EACH CHANNEL

FIGURE 1. I-V CURVE OF PHOTOTRANSISTOR

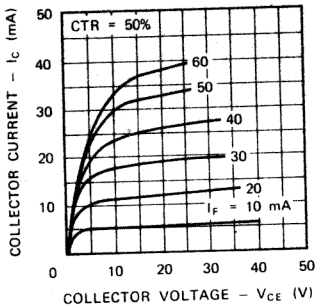


FIGURE 2. I-V CURVE IN SATURATION

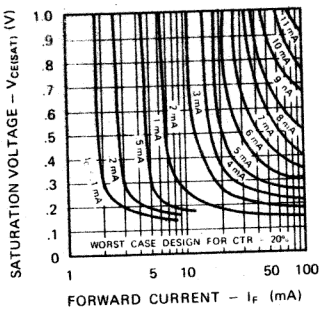


FIGURE 3. CTR VS FORWARD CURRENT

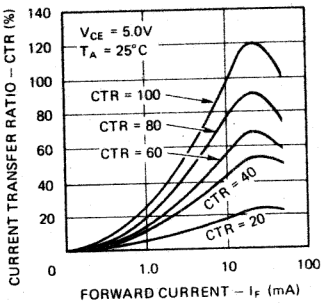


FIGURE 4. CURRENT TRANSFER RATIO VS TEMPERATURE

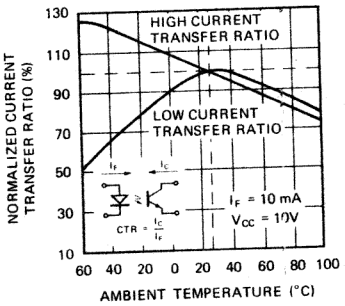


FIGURE 5. I-V CURVE OF LED VS TEMPERATURE

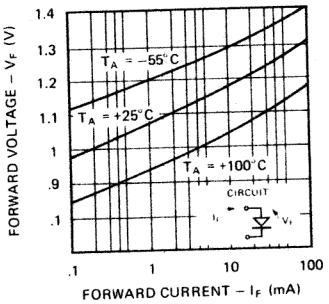


FIGURE 6. LEAKAGE CURRENT VS TEMPERATURE VS COLLECTOR VOLTAGE

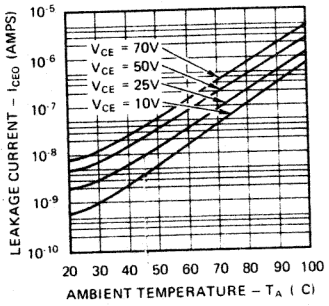
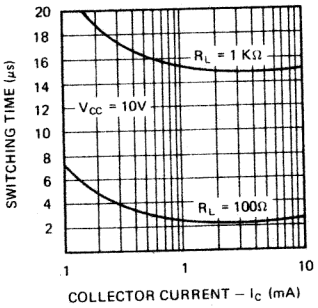


FIGURE 7. SWITCHING TIME VS COLLECTOR CURRENT



INTERIM BULLETIN

Subject to Revision Without Notice

JANUARY 10, 1977

INTEGRATED CIRCUIT ENGINEERING BULLETIN

ULN-2000A

SPRAGUE
THE MARK OF RELIABILITY

SERIES ULN-2000A, ULN-2010A, and ULN-2020A HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS

These high-voltage, high-current Darlington arrays are comprised of seven silicon NPN Darlington pairs on a common monolithic substrate. All units feature open collector outputs and integral suppression diodes for inductive loads. Peak inrush currents to 600mA (Series ULN-2000A and ULN-2020A) or 750mA (Series ULN-2010A) are allowable, making them ideal for driving tungsten filament lamp loads.

Types ULN-2001A, ULN-2011A, & ULN-2021A are general purpose arrays which may be used with DTL and TTL using external current limiting, or with most PMOS and CMOS directly. They are pinned with outputs opposite inputs to facilitate ease of circuit board layout and are priced to compete directly with discrete transistor alternatives.

Types ULN-2002A, ULN-2012A, & ULN-2022A were specifically designed for use with 14 to 25V PMOS devices. Each input has a Zener diode and resistor in series to limit the input current to a safe value in that application.

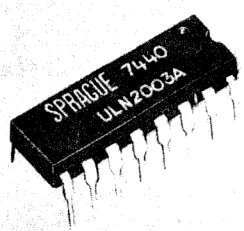
Types ULN-2003A, ULN-2013A, & ULN-2023A have a 2.7kΩ series base resistor for each Darlington pair, and thus allow operation directly with TTL or CMOS operating at a supply voltage of 5V. These devices will handle a great many interface needs——particularly those beyond the capabilities of standard logic buffers.

Types ULN-2004A, ULN-2014A, & ULN-2024A feature a 10,5kΩ series input resistor to allow their operation direcly from CMOS or PMOS outputs utilizing supply voltages of 6 to 15V. The required input current is below that of Types ULN-2003A, ULN-2013A, & ULN-2023A while the required input voltage is less than that required by Types ULN-2002A, ULN-2012A, & ULN-2022A.

Types ULN-2005A, ULN-2015A, & ULN-2025A were especially designed for use with standard and Schottky TTL where higher output currents are required and loading down of the logic output is not a concern. These devices will sink a minimum of 350mA when driven from a standard TTL totem pole output. Typical voltage and current levels for the ULN-2003A, ULN-2013A, & ULN-2023A devices and the ULN-2005A, ULN-2015A, & ULN-2025A are shown in the graphs on pages 7 and 8.

SERIES ULN-2000A, ULN-2010A, and ULN-2020A HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS

Bulletin Z-29304B



CONDUCTOR DIVISION
SPRAGUE ELECTRIC COMPANY
115 Northeast Cutoff, WORCESTER, MASS. 01606

SERIES ULN-2000A, ULN-2010A, and ULN-2020A HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS
ENGINEERING BULLETIN Z-29304B

SERIES ULN-2000A, ULN-2010A, & ULN-2020A
HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS

The Series ULN-2000A is the original high-voltage, high-current Darlington array. The output transistors are capable of sinking 500mA and will sustain at least 50V in the OFF state. Outputs may be paralleled for high load current capability. The Series ULN-2010A devices are similar except that they will sink 600mA. The Series ULN-2020A will sustain 95V in the OFF state. A table showing the specific type numbers available for the various applications is given below.

All Series ULN-2000A, ULN-2010A, & ULN-2020A Darlington arrays are furnished in a 16-pin dual-in-line plastic package. These devices, as shown in Engineering Bulletin No. 29304.1, can also be supplied in a hermetic dual-in-line package for use in military and aeropsace applications (with a slightly reduced power handling capability).

Device Type Number Designation

$V_{CE(MAX.)}$ =	50V	50V	95V
$I_{C(MAX.)}$ =	500mA	600mA	500mA
	Type Number		
General Purpose PMOS, CMOS	ULN-2001A	ULN-2011A	ULN-2021A
15 – 25V PMOS	ULN-2002A	ULN-2012A	ULN-2022A
5V TTL, CMOS	ULN-2003A	ULN-2013A	ULN-2023A
6 – 15V CMOS, PMOS	ULN-2004A	ULN-2014A	ULN-2024A
High Output TTL	ULN-2005A	ULN-2015A	ULN-2025A

Absolute Maximum Ratings at +25°C Free - Air Temperature for any one Darlington pair (unless otherwise noted)

Output Voltage V_{CE}
- Series ULN-2000A & ULN-2010A 50V
- Series ULN-2010A 95V

Input Voltage, V_{IN}
- Types ULN-2002A, ULN-2003A, & ULN-2004A 30V
- Type ULN-2005A 15V

Continuous Collector Current, I_C
- Series ULN-2000A & ULN-2020A 500mA
- Series ULN-2010A 600mA

Continuous Base Current, I_B 25mA

Power Dissipation, P_D
- One Darlington Pair 1.0W*
- Total Package ..... 2.0W†

Operating Ambient Temperature Range, T_A ..... 0°C to +85°C

Storage Temperature Range, T_S ..... –55°C to +150°C

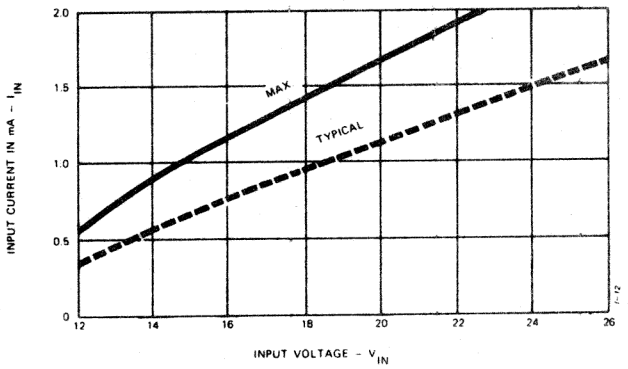
*Derate at the rate of 16.67mW/°C above +25°C.

†Under normal operating conditions, these devices will sustain 350mA per output with $V_{CE(SAT)}$ = 1.6V at +70°C with a pulse width of 20ms and a duty cycle of 34%. Other allowable combinations of output current, number of outputs conducting, and duty cycle are shown on page 8.

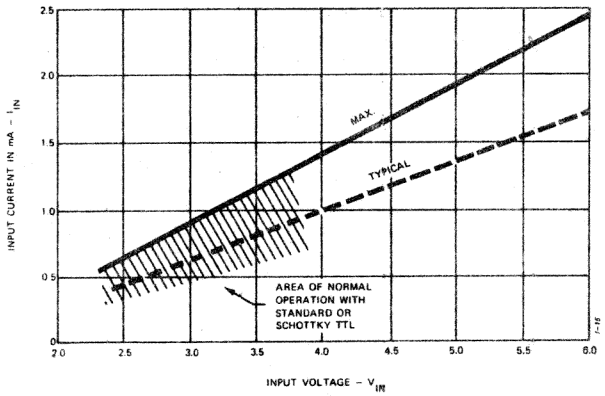
SERIES ULN-2000A

Electrical Characteristics at +25°C (unless otherwise noted)

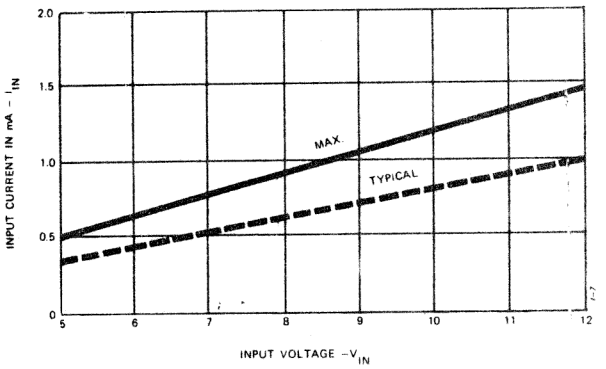
Characteristic	Symbol	Test Fig.	Applicable Devices	Test Conditions	Limits Min.	Typ.	Max.	Units
Output Leakage Current	I_{CEX}	1A	All	V_{CE} = 50V, T_A = +25°C	—	—	50	µA
				V_{CE} = 50V, T_A = +70°C	—	—	100	µA
		1B	ULN-2002A	V_{CE} = 50V, T_A = +70°C, V_{IN} = 6.0V	—	—	500	µA
			ULN-2004A	V_{CE} = 50V, T_A = +70°C, V_{IN} = 1.0V	—	—	500	µA
Collector-Emitter Saturation Voltage	$V_{CE(SAT)}$	2	All	I_C = 100mA, I_B = 250µA	—	0.9	1.1	V
				I_C = 200mA, I_B = 350µA	—	1.1	1.3	V
				I_C = 350mA, I_B = 500µA	—	1.30	1.6	V
Input Current	$I_{IN(ON)}$	3	ULN-2002A	V_{IN} = 17V	—	0.85	1.25	mA
			ULN-2003A	V_{IN} = 3.85V	—	0.93	1.35	mA
			ULN-2004A	V_{IN} = 5.0V	—	0.35	0.5	mA
				V_{IN} = 12V	—	1.0	1.45	mA
			ULN-2005A	V_{IN} = 3.0V	—	1.5	2.4	mA
	$I_{IN(OFF)}$	4	All	I_C = 500µA, T_A = +70°C	50	65	—	µA
Input Voltage	$V_{IN(ON)}$	5	ULN-2002A	V_{CE} = 2.0V, I_C = 300mA	—	—	13	V
			ULN-2003A	V_{CE} = 2.0V, I_C = 200mA	—	—	2.4	V
				V_{CE} = 2.0V, I_C = 250mA	—	—	2.7	V
				V_{CE} = 2.0V, I_C = 300mA	—	—	3.0	V
			ULN-2004A	V_{CE} = 2.0V, I_C = 125mA	—	—	5.0	V
				V_{CE} = 2.0V, I_C = 200mA	—	—	6.0	V
				V_{CE} = 2.0V, I_C = 275mA	—	—	7.0	V
				V_{CE} = 2.0V, I_C = 350mA	—	—	8.0	V
			ULN-2005A	V_{CE} = 2.0V, I_C = 350mA	—	—	2.0	V
DC Forward Current Transfer Ratio	h_{FE}	2	ULN-2001A	V_{CE} = 2.0V, I_C = 350mA	1000	—	—	
Input Capacitance	C_{IN}	—	All		—	15	25	pF
Turn-On Delay	t_{PLH}	—	All	0.5 E_{in} to 0.5 E_{out}	—	0.25	1.0	µs
Turn-Off Delay	t_{PHL}	—	All	0.5 E_{in} to 0.5 E_{out}	—	0.25	1.0	µs
Clamp Diode Leakage Current	I_R	6	All	V_R = 50V, T_A = +25°C	—	—	50	µA
				V_R = 50V, T_A = +70°C	—	—	100	µA
Clamp Diode Forward Voltage	V_F	7	All	I_F = 350mA	—	1.7	2.0	V



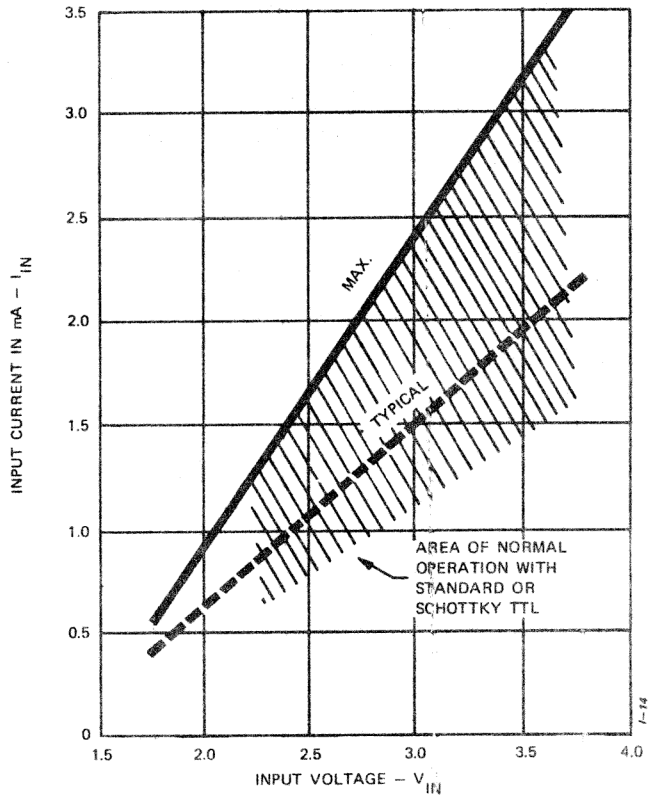
INPUT CURRENT AS A FUNCTION OF INPUT VOLTAGE FOR TYPE ULN-2002A



INPUT CURRENT AS A FUNCTION OF INPUT VOLTAGE FOR TYPE ULN-2003A

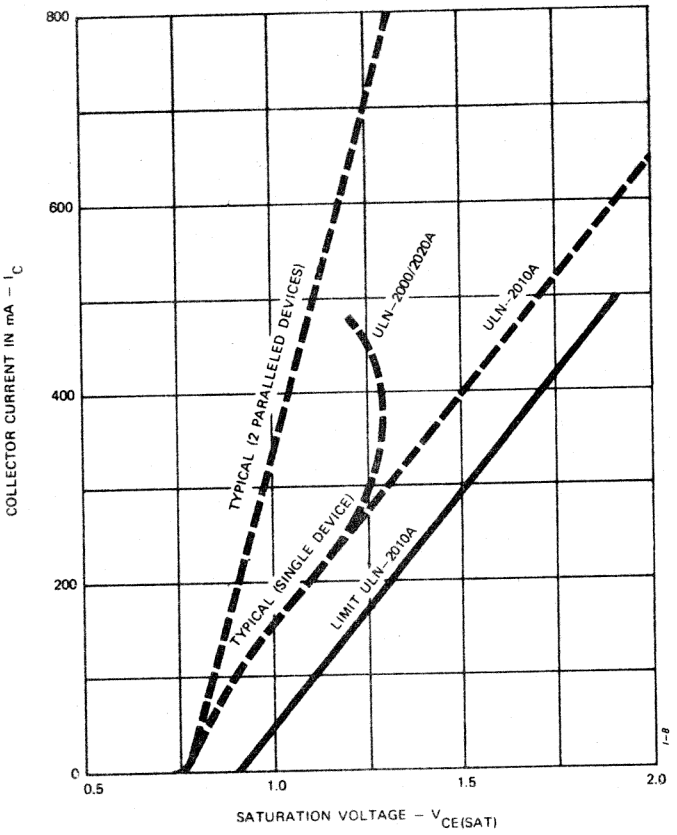


INPUT CURRENT AS A FUNCTION OF INPUT VOLTAGE FOR TYPE ULN-2004A

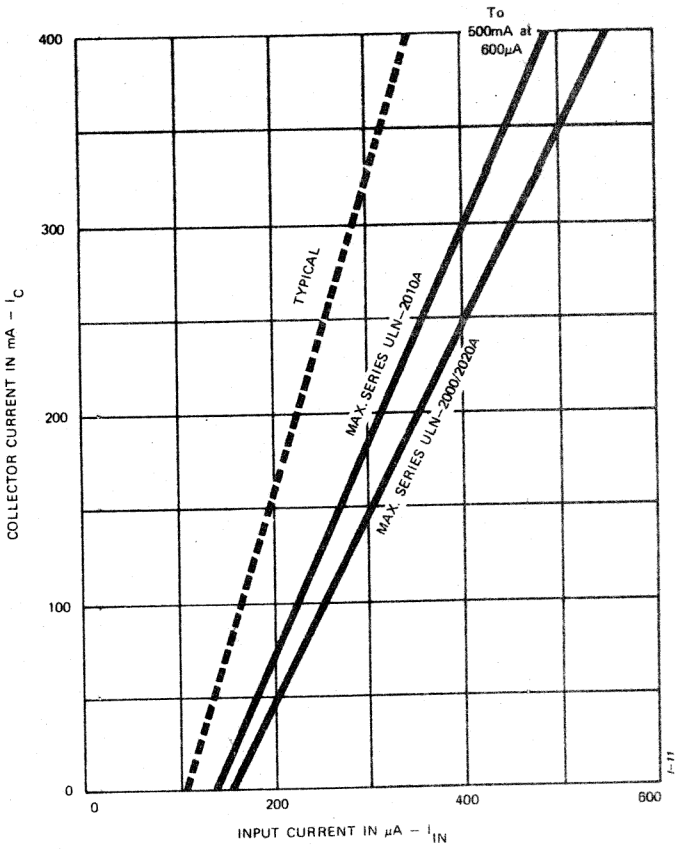


INPUT CURRENT AS A FUNCTION OF INPUT VOLTAGE FOR TYPE ULN-2005A

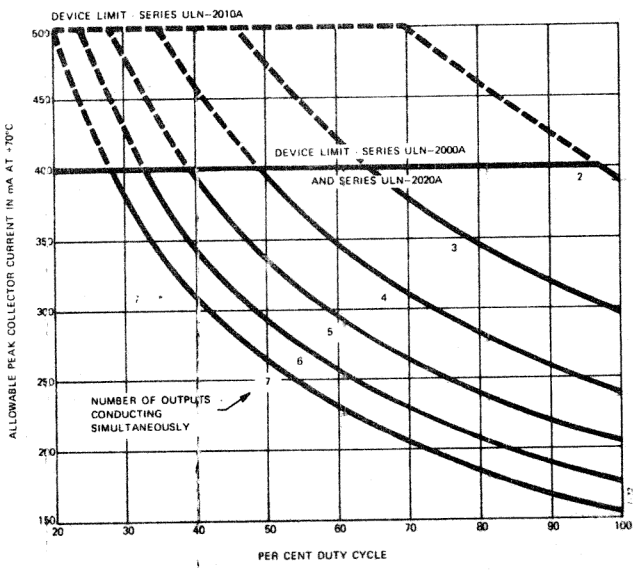
SERIES ULN-2000A, ULN-2010A, & ULN-2020A
HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS



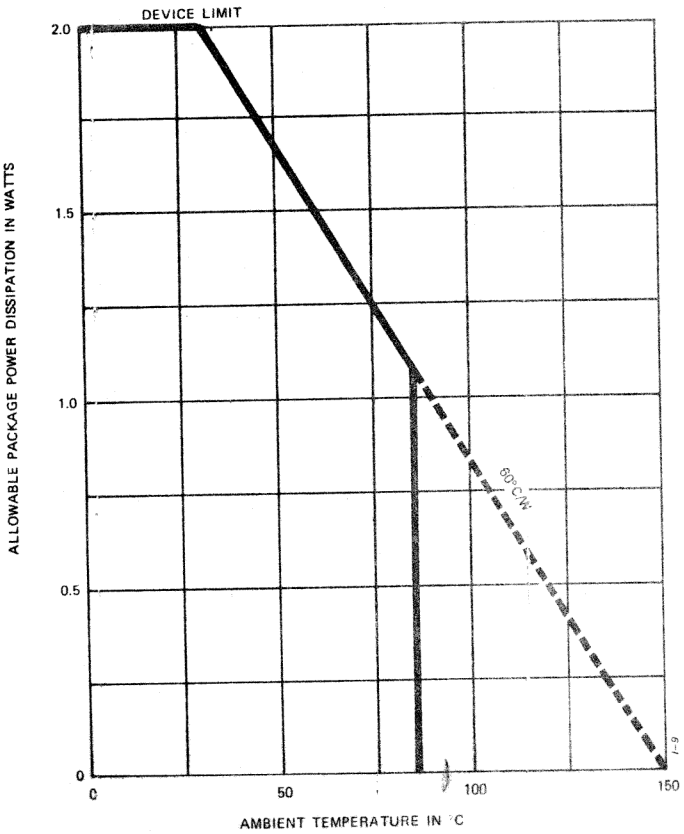
COLLECTOR CURRENT AS A FUNCTION OF SATURATION VOLTAGE



COLLECTOR CURRENT AS A FUNCTION OF INPUT CURRENT

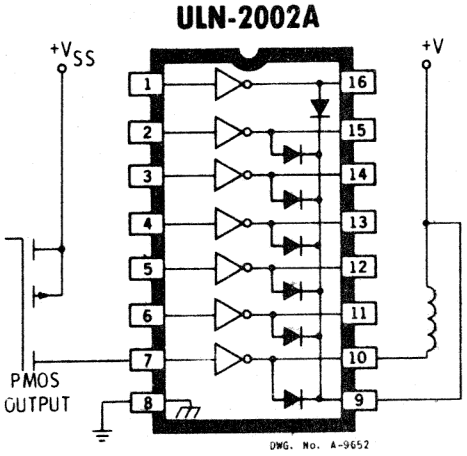


PEAK COLLECTOR CURRENT AS A FUNCTION OF DUTY CYCLE AND NUMBER OF OUTPUTS

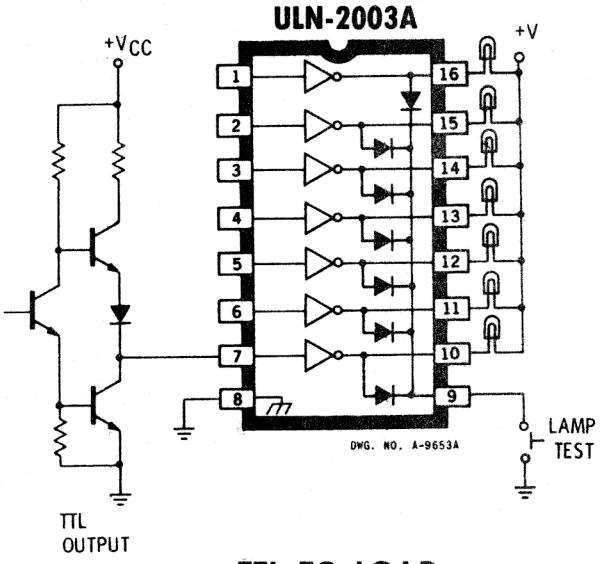


ALLOWABLE AVERAGE PACKAGE POWER DISSIPATION AS A FUNCTION OF AMBIENT TEMPERATURE

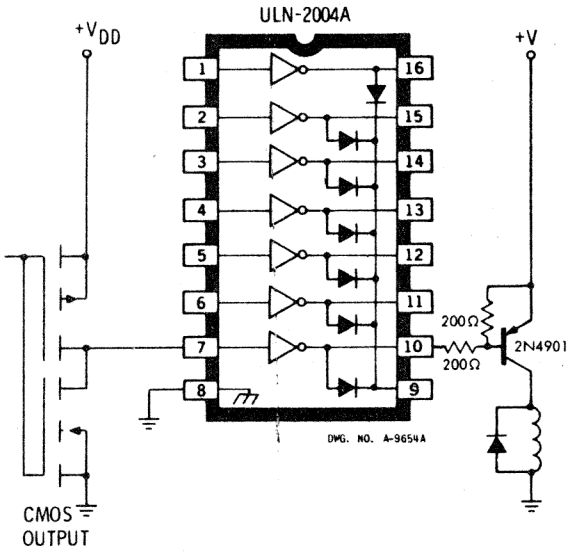
TYPICAL APPLICATIONS



PMOS TO LOAD

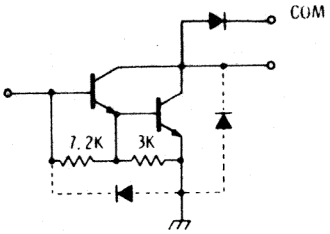


TTL TO LOAD

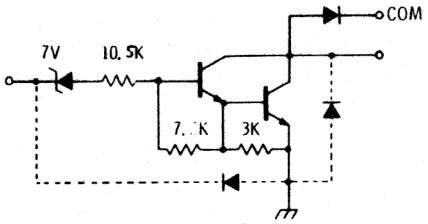


BUFFER FOR HIGHER CURRENT LOADS

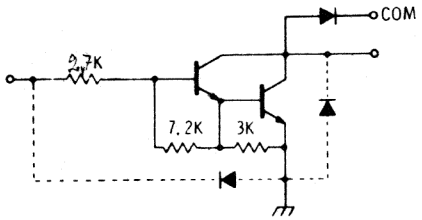
PARTIAL SCHEMATICS



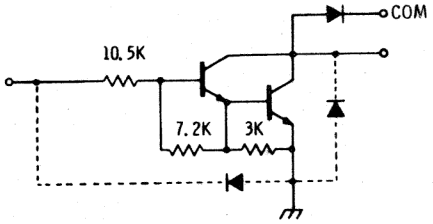
Type ULN-2001A A
(each driver)



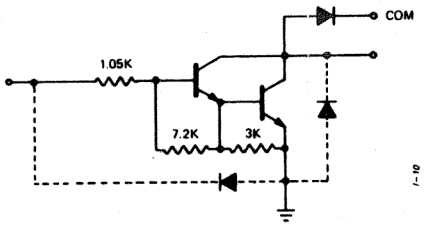
Type ULN-2002A
(each driver)



Type ULN-2003A
(each driver)

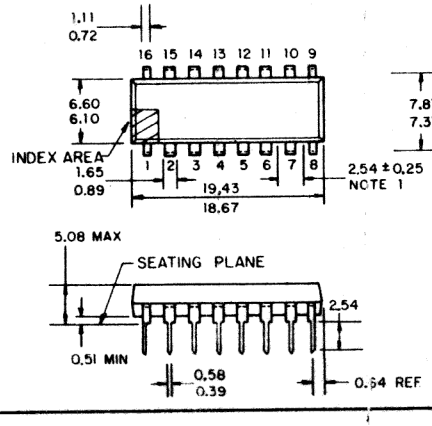


Type ULN-2004A
(each driver)



Type ULN-2005A
(each driver)

METRIC DIMENSIONS



DWG NO A-64028-M

NOTES:
1. LEAD SPACING TOLERANCE IS NON-CUMULATIVE.
2. EXACT BODY AND LEAD CONFIGURATION AT VENDOR'S OPTION WITHIN LIMITS SHOWN.
3. LEADS MISSING FROM THEIR DESIGNATED POSITIONS SHALL ALSO BE COUNTED WHEN NUMBERING LEADS.
4. TERMINAL LEAD STANDOFFS MAY BE OMITTED AND REPLACED BY BODY STANDOFFS.
5. LEAD GAUGE PLANE IS 0.76 MAX. BELOW SEATING PLANE.

15. KRETSSCHEMOR

Sid 15.2 Styrsignaler.
15.3 Digitalkort-0.
15.4 Reläutgångar. Optoisolerade ingångar.
15.5 Digitalkort-T(-PUR), transistorutgångar.
15.6 Digitalkort-32TTL
15.7-8 Komponentförteckning.

Kretsschema för Digitalkort-0 finns på sid 15.2 och 15.3.

Kretsschema för Digitalkort-0-16TTL finns på sid 15.2 och 15.3. För anslutning av TTL-signaler se sid 1.7.

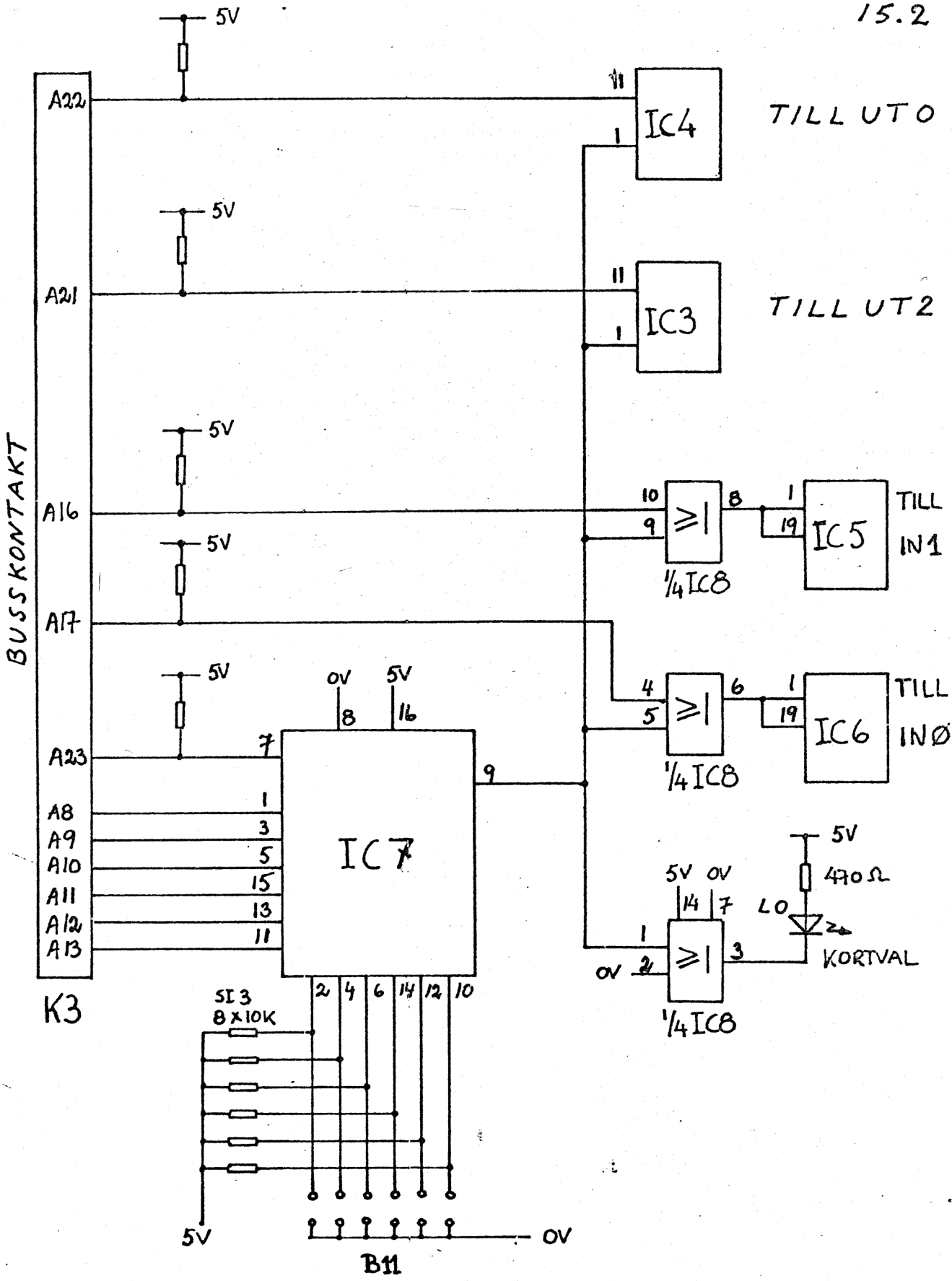
Kretsschema för Digitalkort-R finns på sid 15.2-15.4. För anslutning av TTL-signaler se sid 1.8.

Kretsschema för Digitalkort-T och Digitalkort-T-PUR finns på sid 15.2-15.5. För anslutning av TTL-signaler se sid 1.8.

Kretsschema för Digitalkort-32TTL finns på sid 15.2 och 15.6. Anslutning av yttre enheter framgår även av sid 1.9.

Nr	Ant.	Ändring och/eller medd.-nr	Datum	Inf.	Godk.

15.2



GEMENSAM DEL FÖR ALLA DIGITALKORT

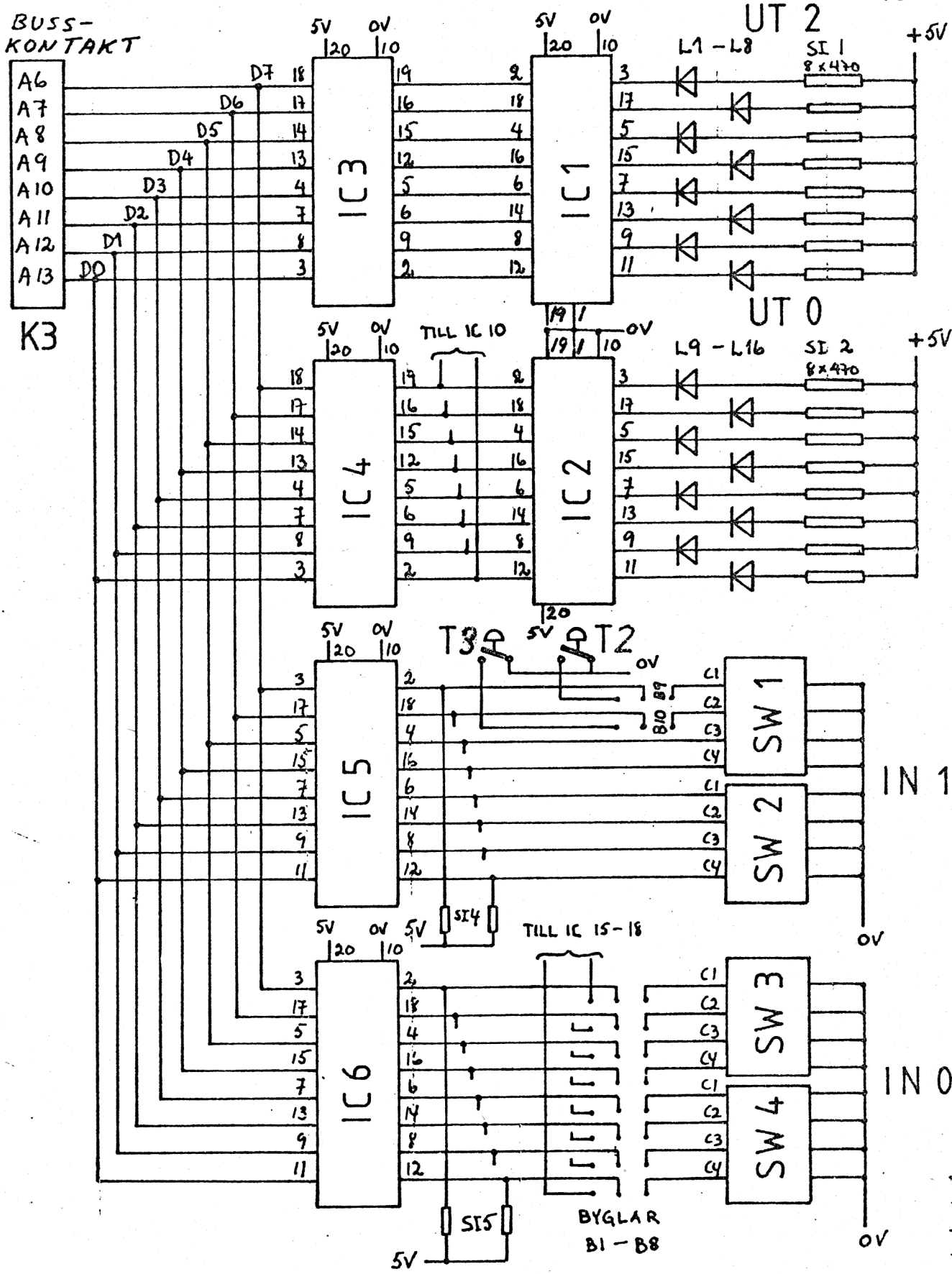
Det.-nr	Ant.	Benämning	Material	Mod.-nr Ämne Dimension	Anm.
Konstr.	Ritad	Kop.	Kontr.	Stand.	Godk.	Skala	Ersätter	Ersatt av
MIKRONIK SUNDSVALL		STYRSIGNALER DIGITALKORT			Dat 801127 Ritn.-nr

Beckers RITMATERIEL

BB A.4
SMS 687.

Nr	Ant.	Ändring och/eller medd.-nr	Datum	Inf.	Godk.

15.3



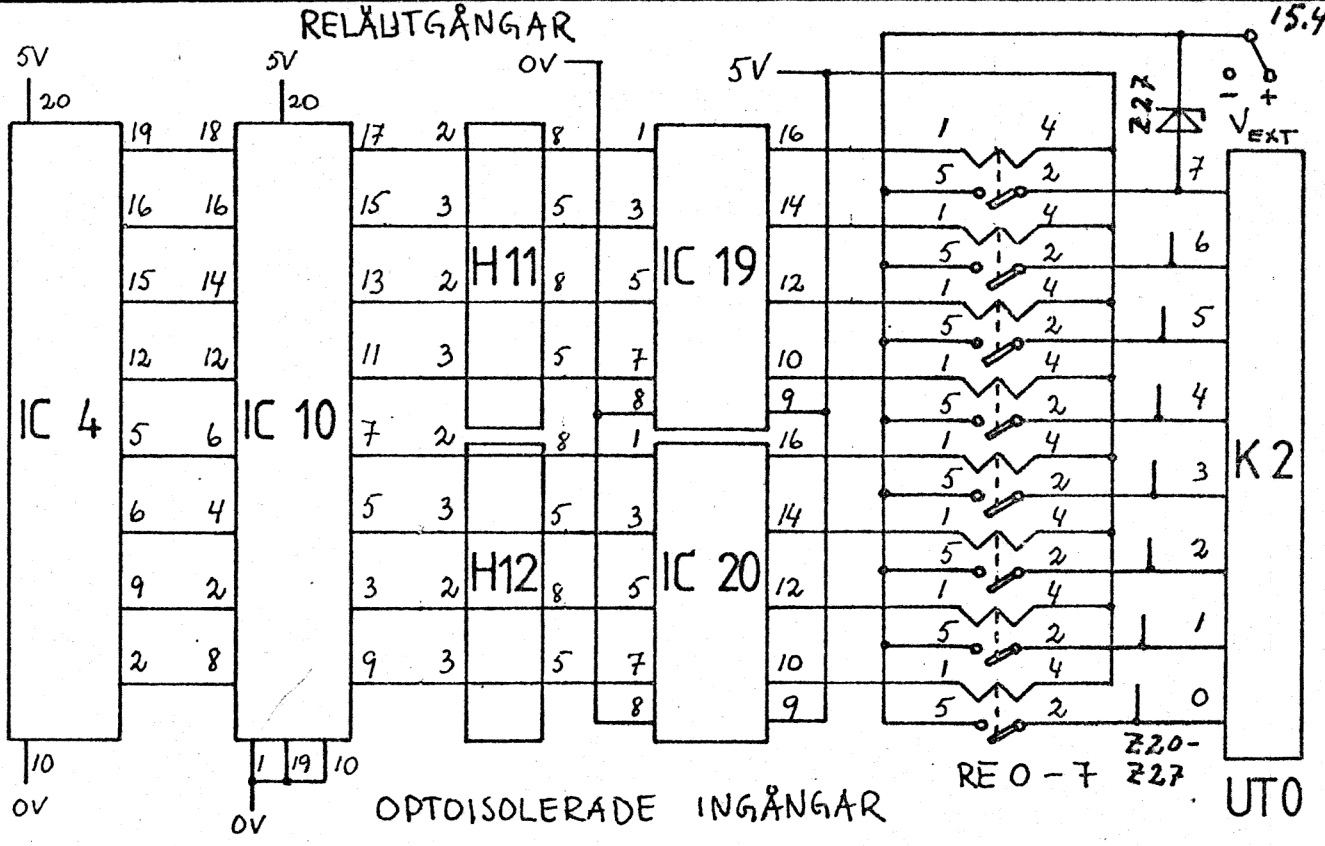
GEMENSAM DEL FÖR ALLA DIGITALKORT UTOM DKORT-32 TTL

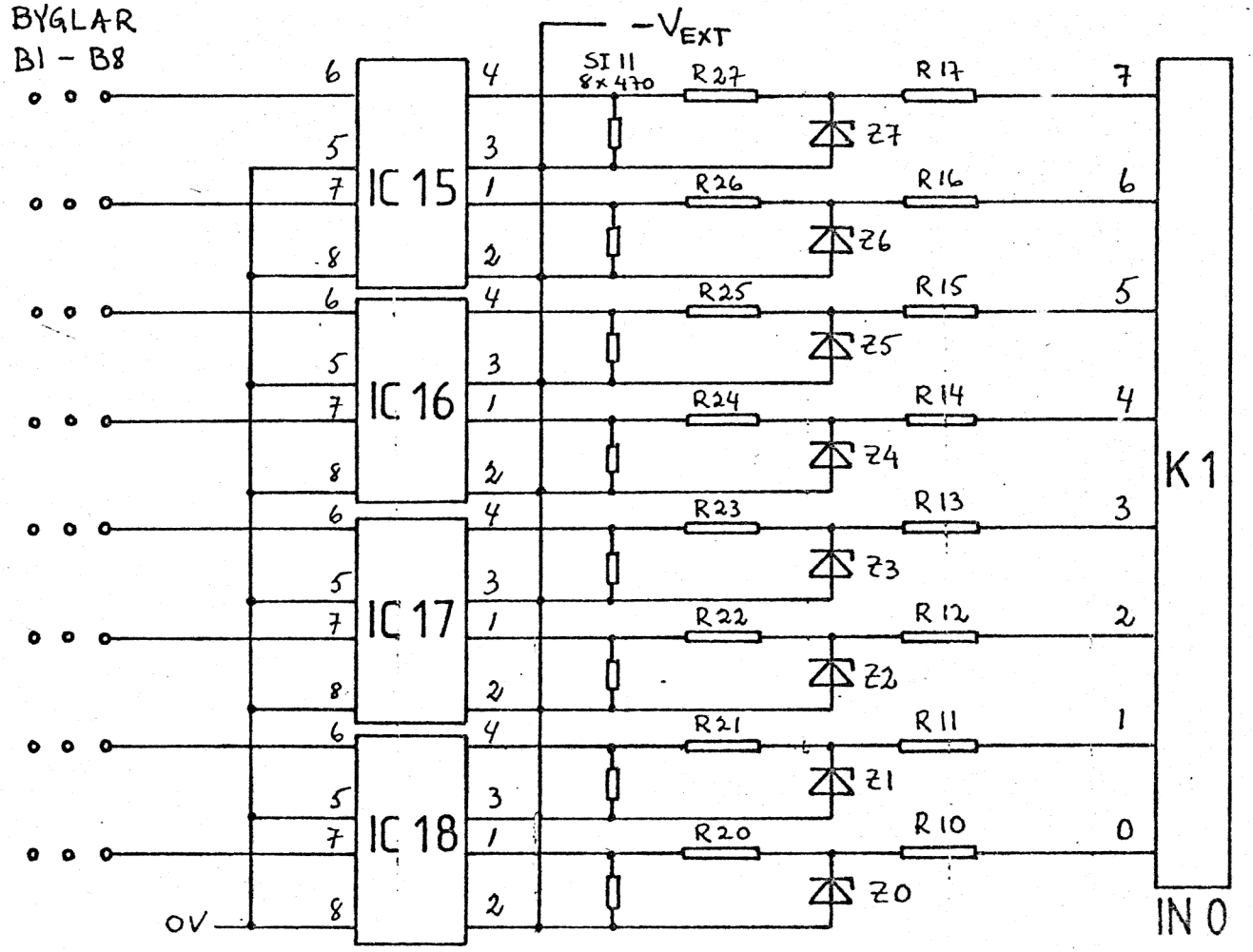
Det.-nr	Ant,	Benämning	Material	Mod.-nr Ämne Dimension	Anm.
Konstr.	Ritad	Kop.	Kontr.	Stand.	Godk.	Skala	Ersätter	Ersatt av
MIKRONIK SUNDSVALL			DIGITALKORT-0				Dat. 800512
							Ritn. nr

Beckers RITMATERIEL

BB A.4
SMS 687.

Nr	Ant.	Ändring och/eller medd.-nr	Datum	Inf.	Godk.





Det.-nr	Ant.	Benämning	Material	Mod.-nr Ämne Dimension	Anm.
Konstr.	Ritad	Kop.	Kontr.	Stand.	Godk.	Skala	Ersätter	Ersatt av

MIKRONIK SUNDSVALL

RELÄUTGÅNGAR OPTOISOLERADE INGÅNGAR

Dat. 800512

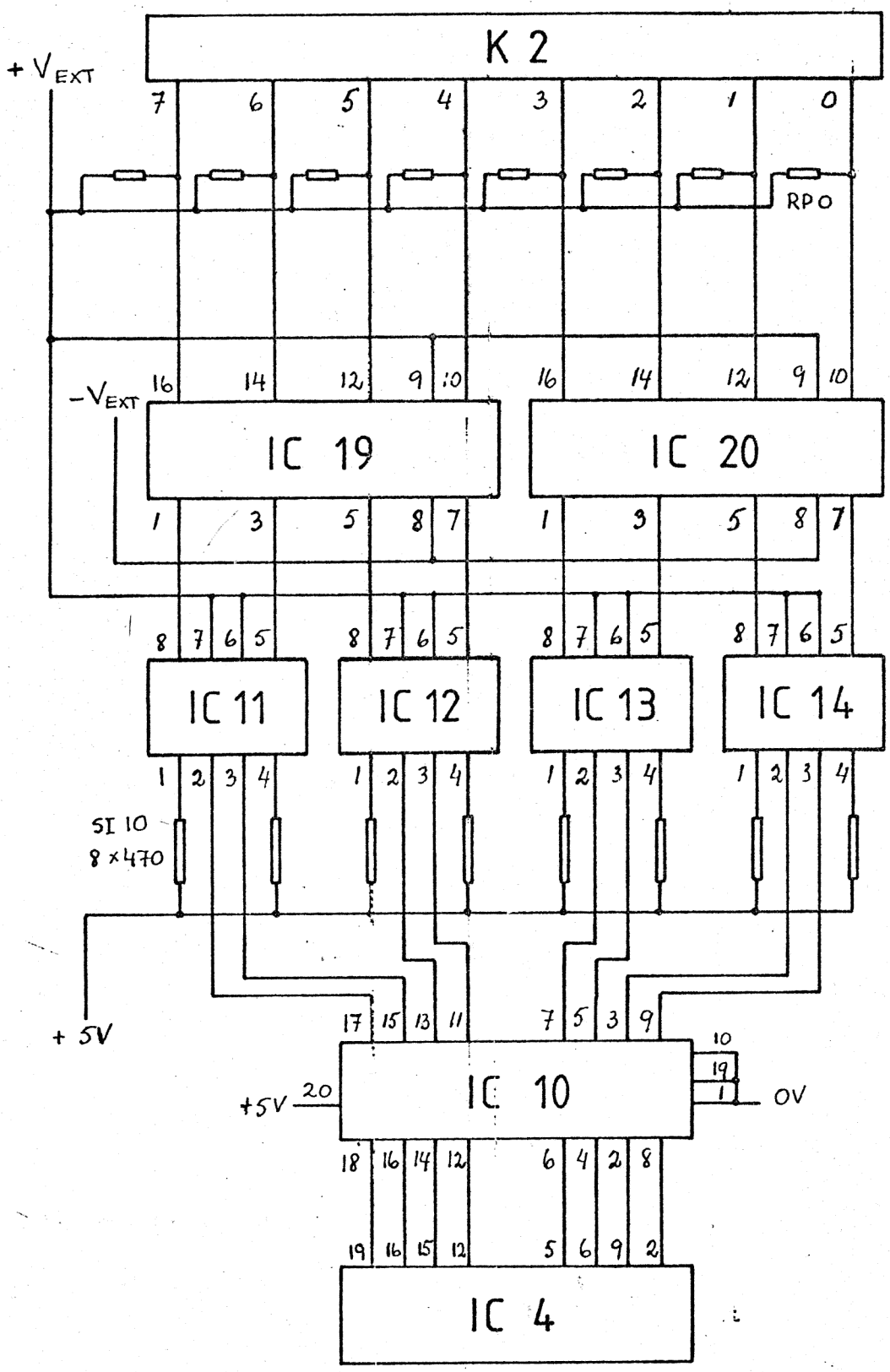
Ritn.-nr

Beckers RITMATERIEL

BB A.4 SMS 687.

Nr	Ant.	Ändring och/eller medd.-nr	Datum	Inf.	Godk.

15.5



RP 0-7
ENBART DIGI-
TALKORT -T-PUR

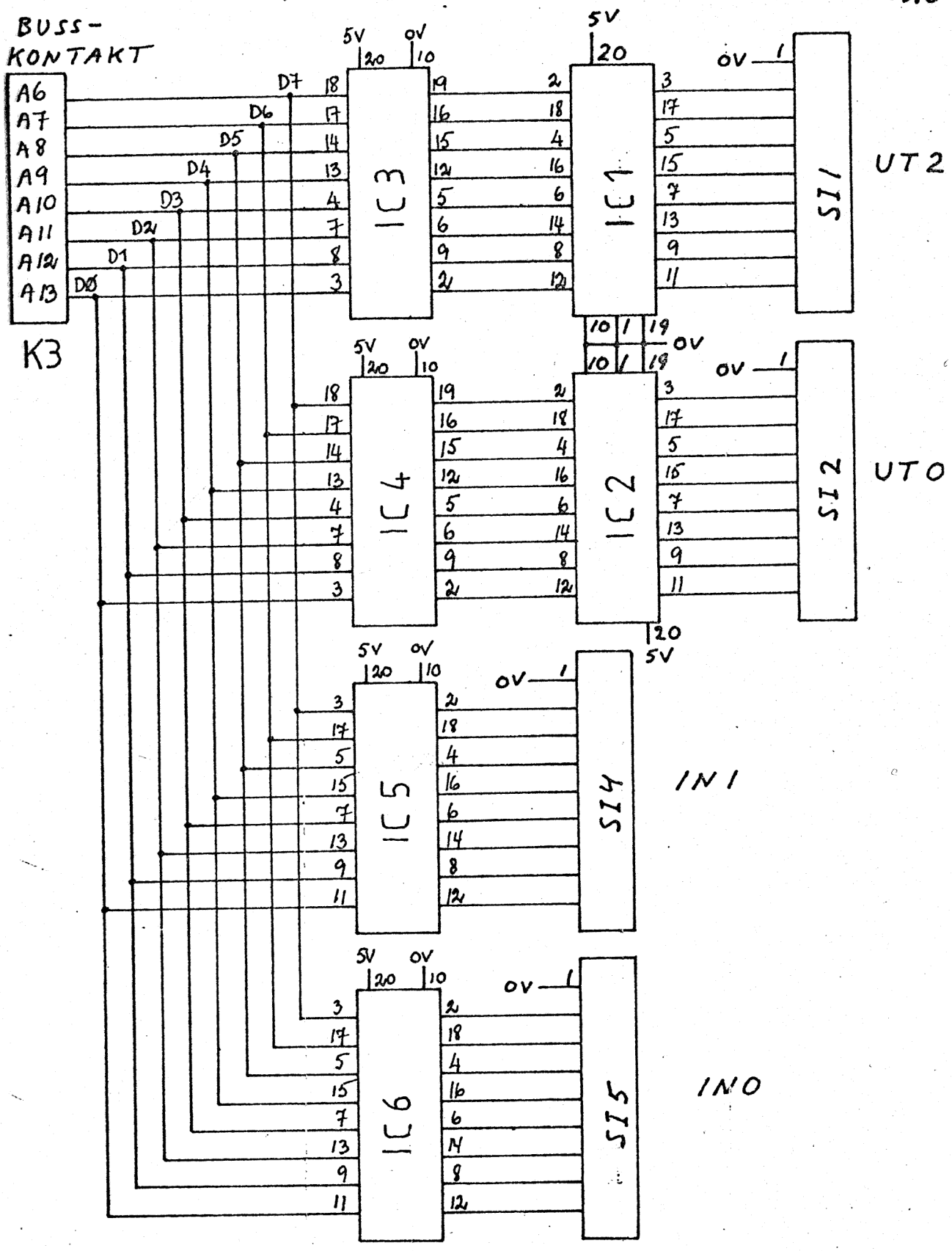
Det.-nr	Ant.	Benämning	Material	Mod.-nr Ämne Dimension	Anm.
Konstr.	Ritad	Kop.	Kontr.	Stand.	Godk.	Skala	Ersätter	Ersatt av
MIKRONIK SUNDSVALL		DIGITALKORT -T (-PUR) TRANSISTORUTGÅNGAR			Dat. 800512
					Ritn.-nr

Beckers RITMATERIEL

BB A.4
SMS 687.

Nr	Ant.	Ändring och/eller medd.-nr	Datum	Int.	Godk.

15.6



Det.-nr	Ant.	Benämning	Material	Mod.-nr Ämne Dimension	Anm.
Konstr.	Ritad	Kop.	Kontr.	Stand.	Godk.	Skala	Ersätter	Ersatt av
MIKRONIK SUNDSVALL	DIGITALKORT - 32 TTL				Dat. 80/127 Ritn.-nr

Beckers RITMATERIEL

BB A.4
SMS 687.

Komponentförteckning

Digitalkort-0

IC1-2	DM81LS96
IC3-4	74LS377
IC5-6	DM81LS96
IC7	DM8131
IC8	74LS32
SI1-2	SILR 470 ohm (single in line resistor)
SI3-5	SILR 10 kohm
L0,1-16	lysdiod
T2-3	tryckknapp Isostat D6
SW1-4	dipswitch 76TC04
C1-6	kondensator keramik 22 nF
C7	" tantal 10 µF
C8	" elektrolyt 100 µF/16V
R1-4,6	resistor kolfilm 1,5 kohm
R5	" " 470 ohm
B1-8	byglar inlödda
B9,10	" 3-poliga
B11	" inlödda
Z10	zenerdiod 6,2V 1,3W

Digitalkort-0-16TTL

Som Digitalkort-0 plus följande:

B1-8	byglar 3-poliga
IC10	DM81LS95
H11-12,15-16	hållare 16-poliga

Digitalkort-T

Som Digitalkort-0 plus följande:

B1-8	byglar 3-poliga
IC10	DM81LS96
IC11-18	optokopplare ILCT6
IC19-20	ULN2003A för Digitalkort-T-5V
	ULN2004A för Digitalkort-T-12V
	ULN2002A för Digitalkort-T-24V
SI10-11	SILR 470 ohm
R10-17	resistor kolfilm 100 ohm, 0,5W för Dkort-T-5V
	" " 820 ohm, 0,5W för Dkort-T-12V
	" " 2200 ohm, 1W för Dkort-T-24V
R20-27	" " 470 ohm, 0,25W
Z0-7	zenerdiod 5,6V 0,4W
KL1-2	klämlist 8-polig
KL3-4	klämlist 2-polig
C10	kondensator tantal 10 µF
Z11	zenerdiod 30V 1,3W

Digitalkort-T-PUR

Avviker från Digitalkort-T med följande:

IC10	DM81LS95
RP0-7	resistor kolfilm 10 kohm

Digitalkort-R

Som Digitalkort-0 plus följande:

B1-8	byglar 3-poliga
IC10	DM81LS95
IC15-18	optokopplare ILCT6
IC19-20	ULN2003A
SI11	SILR 470 ohm
R10-17	resistor kolfilm 100 ohm, 0,5W för Dkort-R-5V
	" " 820 ohm, 0,5W för Dkort-R-12V
	" " 2200 ohm, 1W för Dkort-R-24V
R20-27	" " 470 ohm, 0,25W
Z0-7	zenerdiod 5,6V 0,4W
KL1-2	klämlist 8-polig
KL3-4	klämlist 2-polig
C10	kondensator tantal 10 µF
RE0-7	relä National HA 1 -5V (5V drivspänning)
Z20-27	zenerdiod 30V 1,3W

Digitalkort-32TTL

IC1-2	DM81LS95
IC3-4	74LS377
IC5-6	DM81LS95
IC7	DM8131
IC8	74LS32
L0	lysdiod
C1-5	kondensator keramik 22nF
C7	" tantal 10 uF
C8	" elektrolyt 100 µF/16V
R1-4,6	resistor kolfilm 1,5 kohm
R5	" " 470 ohm
SI3	SILR 10 kohm
B11	byglar inlödda
SI1-2,4-5	stiftrad 9-polig
Z10	zenerdiod 6,2V 1,3W

Vi förbehåller oss rätten till ändringar.

16. CYLINDERPROGRAM

16.1 Cylindertyper

Inom styrtekniken används pneumatiska cylindrar. Vi ger nedan exempel på hur dessa kan styras efter ett visst mönster. Vi behandlar här cylindrar av två typer, med fjäderretur och utan fjäderretur.

Cylindrar med fjäderretur

En cylinder har två ändlägen kallade FRAM och BACK. Spänning på arbetsventilen VC1 gör att cylinder C1 går till läge FRAM. Då spänningen försvinner går C1 till läge BACK. Två ändlägesgivare för varje cylinder kvitterar kolvrörelsen.
C1F sluts då C1 är i läge FRAM.
C1B sluts då C1 är i läge BACK.

Cylindrar utan fjäderretur

För denna cylinder finns en arbetsventil för fram och en för back.
Spänning på VC1+ (C1+=logisk 1) gör att C1 går till läge FRAM.
Spänning på VC1- (C1-=logisk 1) gör att C1 går till läge BACK.
Då cylindern inte skall ändra tillstånd skall både VC1+ och VC1- vara spänningslösa. Man skall inte lägga spänning på VC1+ och VC1- samtidigt.
Kvittenssignaler som ovan.

Anslutning till kort

1-4 cylindrar kan anslutas till ett kort.
Lämpligt kort är Digitalkort-R. På vissa cylinderenheter för undervisning finns förstärkare (TERCO). Då kan även Digitalkort-T eller Digitalkort-T-PUR användas.
Inkoppling till kortet framgår av sid 1.10.
Den i programmen använda ordningsföljden mellan cylindrarna framgår nedan.

Cylindrar utan fjäderretur

```
160 ; "NÄSTA CYLINDERLÄGE"; : ; "  NR";N% : ;
170 ; "128  64  32  16   8   4   2   1"
180 ; " D7  D6  D5  D4  D3  D2  D1  D0"
190 ; " C4+ C4- C3+ C3- C2+ C2- C1+ C1-" : ;
200 ; "KOD=253 : CYKEL SLUT, KOD=192 :TIDFöRD:"
210 ; "C1+=1 : C1 FRAM, C1-=1 :C1 BACK " : ;
220 ; "NÄSTA SEKVENSKOD"; : ; : ;
```

```
370 ; : ; "KVITTENSKOD" : ;
380 ; "128  64  32  16   8   4   2   1"
390 ; " D7  D6  D5  D4  D3  D2  D1  D0"
400 ; "C4F C4B C3F C3B C2F C2B C1F C1B"
```

Cylindrar med fjäderretur

```
160 ; "NÄSTA CYLINDERLÄGE"; : ; "  NR";N% : ;
170 ; "128  64  32  16   8   4   2   1"
180 ; " D7  D6  D5  D4  D3  D2  D1  D0"
190 ; "                 C4  C3  C2  C1" : ;
200 ; "KOD=253 : CYKEL SLUT, KOD=192 :TIDFöRD:
210 ; "C1=1 : C1 FRAM, C1=0 :C1 BACK " : ;
220 ; "NÄSTA SEKVENSKOD"; : ; : ;
```

```
370 ; : ; "KVITTENSKOD" : ;
380 ; "128  64  32  16   8   4   2   1"
390 ; " D7  D6  D5  D4  D3  D2  D1  D0"
400 ; "C4F C4B C3F C3B C2F C2B C1F C1B"
```

16.2 Enkla cylinderprogram

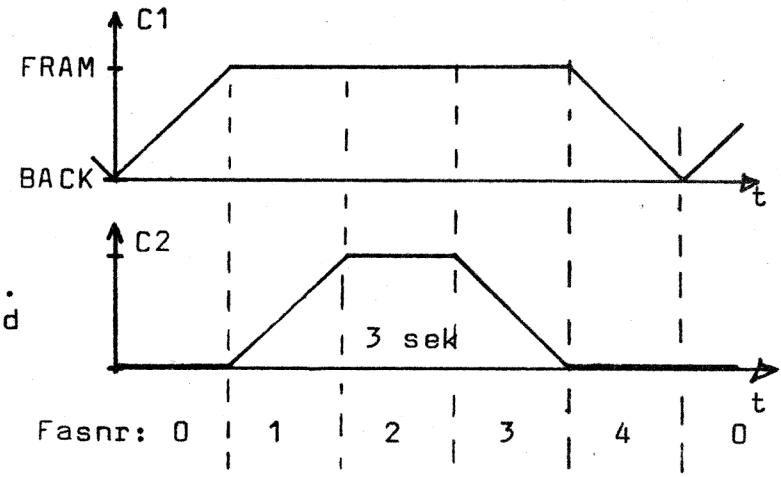
Krav på programmet

Vi vill ha ett allmänt program som medger att ett rörelseschema läggs in i datorn utgående från ett följddiagram. Vi skall alltså inte behöva skriva om programmet för varje nytt följddiagram.

Förslag på lösning

Vi tittar först på en lösning för cylindrar utan fjäderretur. Koderna för ett följddiagram lägges in i två listor med indexerade variabler.

Som exempel tar vi vidstående följddiagram.



Följddiagrammet indelas i faser med numrering från 0.
Med sekvenskod anges en av följande tre händelser.
1. Cylinderrörelse, varvid koden anger vilken eller vilka arbetsventiler som skall ha spänning.
2. Tidfördröjning.
3. Slut på cykeln.
För 2 och 3 väljes sådana koder som inte kan förekomma för 1.

För ovanstående följddiagram skall nedanstående data matas in.

Fasnr	Sekvenskod	Kvittenskod/Tid	Kommentar
N%	A%(N%)	B%(N%)	
0	2	86	C1 fram
1	8	90	C2 fram
2	192	3	Vänta 3 sek
3	4	86	C2 back
4	1	85	C1 back
5	253		Slut på cykel

Det förutsättes här att fyra cylindrar är anslutna och att C3 och C4 är i läge BACK.

På sid 16.3 och 16.4 visas flödesschema för program CYL1A.

Principen att lagra tabeller med indexerade variabler finns upptagen i uppgift E3 och E5 i punkt 4 och 5.

13.3 Anslutning av TTL-signaler till yttre enheter

På sid 1.8 visas hur man kan få 8 TTL-utgångar och 8 TTL-ingångar.

Data för utgångarna:

Utgång HÖG: $V_{OH} > 2,7V$ Utgång LÅG: $V_{OL} < 0,5V$

$|I_{OH}| > 2,6$ mA $I_{OL} > 16$ mA

Detta innebär att för en logisk ETTA ut är spänningen större än 2,7V. Utgången klarar därvid av en ström ut om minst 2,6 mA.
För en logisk NOLLA ut är spänningen mindre än 0,5V och utgången klarar att sänka minst 16 mA.

Data för ingångarna:

Ingång LÅG: $V_{IL} < 0,8V$

$|I_{IL}| < 0,36$ mA för IC6 plus 0,5 mA genom pull-up-resistorn.

Ingång HÖG: $V_{IH} > 2,0V$

$I_{IH} < 20$ µA

En yttre utgång ansluten till en TTL-ingång på kortet får i läge LÅG sänka strömmen 0,86 mA.

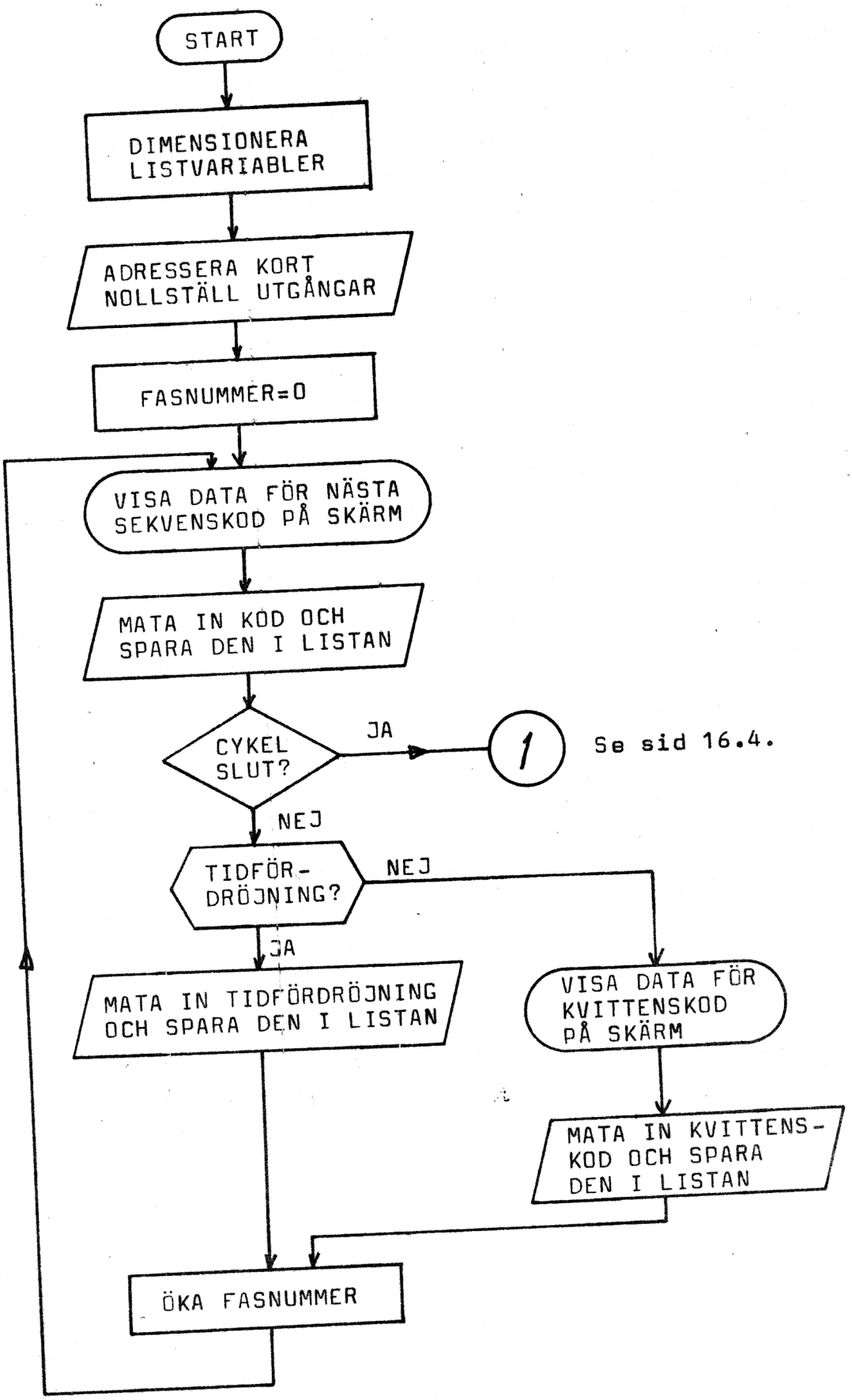
Signaltillstånd:

Yttre TTL-signal	Bussignal
INGÅNG	
LÅG	HÖG
HÖG	LÅG
UTGÅNG	
LÅG	LÅG
HÖG	HÖG

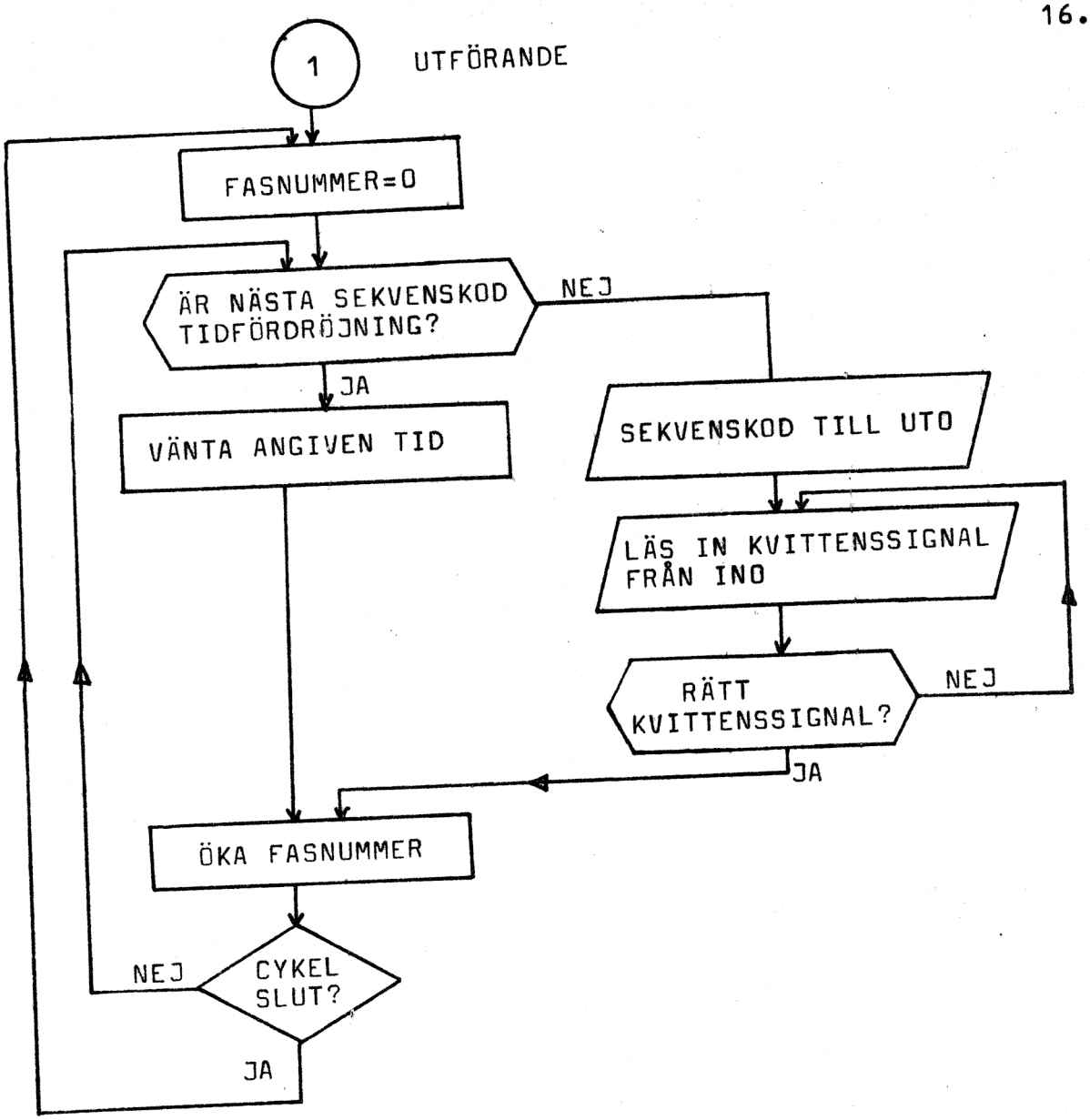
TTL-utgångarna tas från drivkrets (IC10), se fig 4 sid 1.6.

Flödesschema för program CYL1A.

Programmet består av tre delar, initiering, inmatning av koder för följddiagram samt utförande då cylindrarna körs.



Se sid 16.4.



På sid 16.5 visas programmet.

Kommentarer:

Radnr	Kommentar
10-100	Initiering
60	Dimensionering för 25 faser plus fasslut.
110-440	Inmatning av koder. Felhanteringen finns inte med i flödesschemat.
150,250,255	Exempel på felhantering. Ett pip signalerar felaktig kod.
450-580	Körning av cylindrar.

Programmet är skrivet för flercykelförlopp och avbrytes med CTRL-C. En svaghet med programmet är att när en körning har avbrutits måste koderna matas in igen.

Program CYL1B är skrivet för 1-4 cylindrar med fjäderretur. I övrigt är det samma som CYL1A.

Angående provning se sid 16.7.

```
10 REM CYL1A 810130
20 REM MIKRONIK  SUNDSVALL
30 ; CHR$(12)
40 ; "SEKVENSPROGRAM FÖR 1-4 CYLINDRAR"
50 ; "UTAN FJÄDERRETUR" : ;
55 ; "FLERCYKELFÖRLOPP" : ; : ; : ;
60 DIM A%(25%),B%(25%)
62 REM N%=FASNUMMER
64 REM A%(N%)=SEKVENSKOD
66 REM B%(N%)=TID ELLER KVITTENSKOD
68 REM T=TID I SEK
70 ; "TRYCK PÅ TANGENT"
80 GET A$
90 REM KORTADRESS OCH NOLLSTÄLLNING AV UTGÅNGAR
100 OUT 1,19,0,0,2,0
110 REM ++++++++++++++++++++++++++++++++
120 REM INMATNING **********************
130 N%=0%
140 ; CHR$(12)
150 ONERRORGOTO 250
160 ; "NÄSTA CYLINDERLÄGE"; : ; "  NR";N% : ;
170 ; "128  64  32  16   8   4   2   1"
180 ; " D7  D6  D5  D4  D3  D2  D1  D0"
190 ; " C4+ C4- C3+ C3- C2+ C2- C1+ C1-" : ;
200 ; "KOD=253 : CYKEL SLUT, KOD=192 :TIDFÖRD:"
210 ; "C1+=1 : C1 FRAM, C1-=1 :C1 BACK " : ;
220 ; "NÄSTA SEKVENSKOD"; : ; : ;
230 INPUT A$ : ; : ;
240 A%(N%)=VAL(A$) : GOTO 260
250 ; CHR$(12) : ; CHR$(7)
255 ; "FELAKTIG KOD, OMIGEN" : ; : GOTO 150
260 IF A%(N%)>=253% THEN 470
270 IF A%(N%)=192% THEN 290 ELSE 360
280 REM ********************************
290 ONERRORGOTO 330
300 ; : ; "GE TIDFÖRDRÖJNING I SEK"
310 INPUT A$ : ;
320 B%(N%)=VAL(A$) : GOTO 440
330 ; CHR$(7) : ; "FELAKTIG KOD, OMIGEN" : GOTO 290
350 REM ********************************
360 ONERRORGOTO 430
370 ; : ; "KVITTENSKOD" : ;
380 ; "128  64  32  16   8   4   2   1"
390 ; " D7  D6  D5  D4  D3  D2  D1  D0"
400 ; "C4F C4B C3F C3B C2F C2B C1F C1B"
410 ; : INPUT A$
420 B%(N%)=VAL(A$) : GOTO 440
430 ; CHR$(7) : ; CHR$(12)
435 ; "FELAKTIG KOD, OMIGEN" : GOTO 360
440 N%=N%+1% : GOTO 140
450 REM ++++++++++++++++++++++++++++++++
460 REM UTFÖRANDE **********************
470 N%=0%
480 REM TID ELLER CYL-RÖRELSE
490 IF A%(N%)=192% THEN 500 ELSE 530
500 T=1000*B%(N%) : REM TIDFÖRDRÖJNING
510 FOR P=1 TO T : NEXT P
520 GOTO 560
530 OUT 0%,A%(N%) : REM CYL-RÖRELSE
540 X%=INP(0%) : REM KVITTENSSIGNAL
550 IF X%=B%(N%) THEN 560 ELSE 540
560 OUT 0%,0% : N%=N%+1%
570 REM ÄR CYKELN SLUT?
580 IF A%(N%)=253% THEN 470 ELSE 490
```

```
10 REM CYL1B 810130
20 REM MIKRONIK  SUNDSVALL
30 ; CHR$(12)
40 ; "SEKVENSPROGRAM FÖR 1-4 CYLINDRAR"
50 ; "MED FJÄDERRETUR" : ;
55 ; "FLERCYKELFÖRLOPP" : ; : ; : ;
60 DIM A%(25%),B%(25%)
62 REM N%=FASNUMMER
64 REM A%(N%)=SEKVENSKOD
66 REM B%(N%)=TID ELLER KVITTENSKOD
68 REM T=TID I SEK
70 ; "TRYCK PÅ TANGENT"
80 GET A$
90 REM KORTADRESS OCH NOLLSTÄLLNING AV UTGÅNGAR
100 OUT 1,19,0,0,2,0
110 REM ++++++++++++++++++++++++++++++++
120 REM INMATNING **********************
130 N%=0%
140 ; CHR$(12)
150 ONERRORGOTO 250
160 ; "NÄSTA CYLINDERLÄGE"; : ; "  NR";N% : ;
170 ; "128  64  32  16   8   4   2   1"
180 ; " D7  D6  D5  D4  D3  D2  D1  D0"
190 ; "                 C4  C3  C2  C1" : ;
200 ; "KOD=253 : CYKEL SLUT, KOD=192 :TIDFÖRD:"
210 ; "C1=1 : C1 FRAM, C1=0 :C1 BACK " : ;
220 ; "NÄSTA SEKVENSKOD"; : ; : ;
230 INPUT A$ : ; : ;
240 A%(N%)=VAL(A$) : GOTO 260
250 ; CHR$(12) : ; CHR$(7)
255 ; "FELAKTIG KOD, OMIGEN" : ; : GOTO 150
260 IF A%(N%)>=253% THEN 470
270 IF A%(N%)=192% THEN 290 ELSE 360
280 REM ********************************
290 ONERRORGOTO 330
300 ; : ; "GE TIDFÖRDRÖJNING I SEK"
310 INPUT A$ : ;
320 B%(N%)=VAL(A$) : GOTO 440
330 ; CHR$(7) : ; "FELAKTIG KOD, OMIGEN" : GOTO 29
350 REM ********************************
360 ONERRORGOTO 430
370 ; : ; "KVITTENSKOD" : ;
380 ; "128  64  32  16   8   4   2   1"
390 ; " D7  D6  D5  D4  D3  D2  D1  D0"
400 ; "C4F C4B C3F C3B C2F C2B C1F C1B"
410 ; : INPUT A$
420 B%(N%)=VAL(A$) : GOTO 440
430 ; CHR$(7) : ; CHR$(12)
435 ; "FELAKTIG KOD, OMIGEN" : GOTO 360
440 N%=N%+1% : GOTO 140
450 REM ++++++++++++++++++++++++++++++++
460 REM UTFÖRANDE **********************
470 N%=0%
480 REM TID ELLER CYL-RÖRELSE
490 IF A%(N%)=192% THEN 500 ELSE 530
500 T=1000*B%(N%) : REM TIDFÖRDRÖJNING
510 FOR P=1 TO T : NEXT P
520 GOTO 560
530 OUT 0%,A%(N%) : REM CYL-RÖRELSE
540 X%=INP(0%) : REM KVITTENSSIGNAL
550 IF X%=B%(N%) THEN 560 ELSE 540
560 N%=N%+1%
570 REM ÄR CYKELN SLUT?
580 IF A%(N%)=253% THEN 470 ELSE 490
```

16.3 Förbättringar av cylinderprogrammen

A. Förbättring av CYL1A och CYL1B

I programmen CYL1A och CYL1B följer körning av cylindrar direkt efter inmatning av sekvens. Varje gång vi vill köra cylindrarna måste först koderna matas in.

En första förbättring är att lagra koderna direkt i minnet med POKE-satsen under inmatning. Vid körning hämtas koderna i minnet med PEEK-satsen. Om körningen avbryts med CTRL-C skall den kunna fortsätta igen i samma fas som den slutade. Därför behövs en meny i början av programmet, t ex

1 programmering av sekvens

2 körning

Förslag på lösning lämnas inte här. Principen för lagring av data direkt i minnet finns angiven i uppgift E4 och E6 i punkt 4 och 5. Se även nedan.

B. ON-LINE-programmering

Vi vill bespara oss mödan att räkna ut koderna och låta datorn göra det i stället.

När provkörningen utförts för ett följddiagram vill vi dessutom spara undan koderna på en datafil på kassett.

I början av programmet göres ett körval, vidstående meny visar ett exempel.

```
; "VÄLJ MELLAN" : ; : ;
; "1  PROGRAMMERING AV SEKVENS" : ;
; "2  KÖRNING" : ;
; "3  ÖVERFÖRING AV DATA TILL KASSETT" : ;
; "4  ÖVERFÖRING AV DATA FRÅN KASSETT" : ;
```

Programmering av sekvens on-line kan tillgå enligt följande.
För en fas väljer vi först efter vidstående meny.

```
; "VÄLJ MELLAN" : ; : ;
; "1  CYLINDERRÖRELSE" : ;
; "2  TIDFÖRDRÖJNING" : ;
; "3  SLUT"
```

Väljes cylinderrörelse måste först anges vilken cylinder som skall ändra tillstånd.

```
; "VÄLJ MELLAN";TAB(20);"FAS NR";F% : ;
; "1  CYL 1" : ;
; "2  CYL 2" : ;
; "3  CYL 3" : ;
; "4  CYL 4" : ;
```

Sedan måste vi ange om cylindern i fråga skall gå fram eller back.

```
; "VÄLJ MELLAN" : ;
; "0  BACK" : ;
; "1  FRAM"
```

Från de två sista menyvalen beräknar datorn en sekvenskod. Denna skickas ut till UTO och den aktuella cylindern rör sig. Då cylinderrörelsen är klar anger vi detta till datorn. Datorn känner av kvittenskoden genom att läsa in INO. Koderna lagras i minnet.

Programmen
CYL2A1 för 1-4 cylindrar utan fjäderretur och
CYL2B1 för 1-4 cylindrar med fjäderretur
är gjorda enligt ovanstående beskrivning.
Programlängd: printerutskriften tar 2,5 sidor A4.
Programmen ingår i programkassett som finns att köpa.
Dokumentation medföljer.

C. Komplettering av föregående program

Föregående program kompletteras med

- val mellan encykelförlopp och flercykelförlopp
- tryckknappar för start och stopp,
- watch-dog.

För start och stopp utnyttjas tryckknapparna på kortet.

Watch-dog innebär att varje fas med cylinderrörelse tidövervakas. Om en fas inte är utförd inom en viss tid stoppas körningen och datorn meddelar vad som hänt. Man kan simulera kabelfel genom att dra ur en sladd så att datorn inte får rätt kvittenskod och därigenom utlösa watch-dog.

Programmen
CYL2A2 för 1-4 cylindrar utan fjäderretur och
CYL2B2 för 1-4 cylindrar med fjäderretur
har ovanstående faciliteter.
Programlängd: printerutskriften tar mindre än 3 sidor A4.
Programmen ingår i programkassett som finns att köpa.
Dokumentation medföljer.

16.4 Provning av koppling

Vid kontroll av uppkopplingen kan man utnyttja direktmoden i ABC80.
Med kommandot OUT 1,19 adresseras kortet.
Vi antar att cylindrar utan fjäderretur är anslutna.

Exempel:
Med kommandot OUT 0,8 skall C2 gå till läge FRAM.
Med kommandot OUT 0,1 skall C1 gå till läge BACK.
Med ; INP(0) visas kvittenskoden på skärmen.

Med OUT-kommandot kan cylindrarna ställas i utgångsläge före körning.

Man kan även mäta spänningar på klämlisterna med -VEXT som nolla.

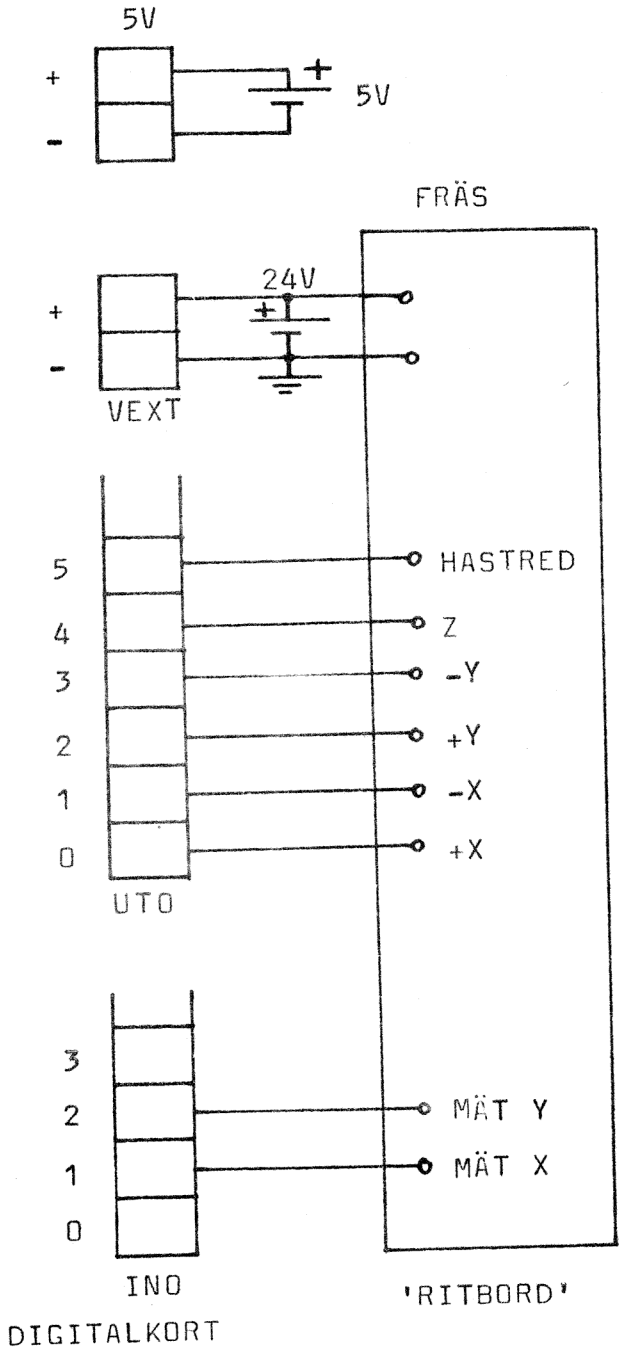
17. PROGRAM FÖR FRÄS (INTRONIC AB)

Under denna punkt visas ett program för styrning av fräs (INTRONIC AB).

Kort: Digitalkort-R-24V.

Avstörning: De två likströmsmotorerna i fräsen skall avstöras genom att en kondensator om 0,1 µF lödes in över vardera motorn, se sid 1.6. Man kommer åt motorerna om bottenplattan avlägsnas.

Uppkoppling:



Kommentarer

Yttre 5V-matning till kortet, se sid 1.3.

Minuspolen på VEXT skall jordas.

Minskar motorernas hastighet.

Penna ned.

Ger rörelse i respektive riktning.

Ger en puls för varje varv i Y- respektive X-led.

Programbeskrivning

Program RIT visas sist i denna punkt.

Då en motor går erhålles en puls per varv från mätdonets kamskiva. Detta motsvarar 2 mm på ritbordet.

I början av programmet nollställes utgångarna (rad 50). Kortet är enbart adresserat då datorn skall kommunicera med den yttre enheten.
För inmatning av data från tangentbordet är felhantering inlagd i programmet.
I början av programmet göres ett körval mellan programmering av sekvens, körning eller slut. (Rad 70-150).
En hel körning kallas här för en sekvens. Denna är uppdelad i ett antal rörelser i X- eller Y-led. En rörelse omfattar ett visst antal steg om 2 mm.

Programmering av sekvens, rad 170-370.

Data för en sekvens lägges in i en tabell i minnet.
Varje rörelse bestäms av två bytes. Den första innehåller koden för rörelsen. Den andra innehåller antalet steg a 2 mm.
Hur tabellen nås framgår av uppgift E4 och E6 vilka är lämpliga förövningar. (sid 4.4).
Om en sekvens avbryts kan man i detta fall inte fortsätta körningen där den avslutades.
På rad 220-260 finns en mall för att beräkna koden, jämför figuren på sid 17.1. Koden 128 anger tabellslut och medför återhopp till menyvalet. 128 är en påhittad kod för tabellslut därför att den inte används för någon rörelse.

Körning, rad 390-810.

Under körningen printas data på skärmen. Rubrik ges på rad 400 och data printas på rad 540-550 och 660.
Startvärden för tabelladress och rörelsenummer ges på rad 410.
Rad 430-480 är förberedelser för en rörelse. Kod och antal steg a 2 mm tas fram ur tabellen. Dessutom bestäms en mask V% som anger vilket mätdon som skall kännas av. För X-rörelse blir V%=2 och för Y-rörelse 4, jämför figur sid 17.1.

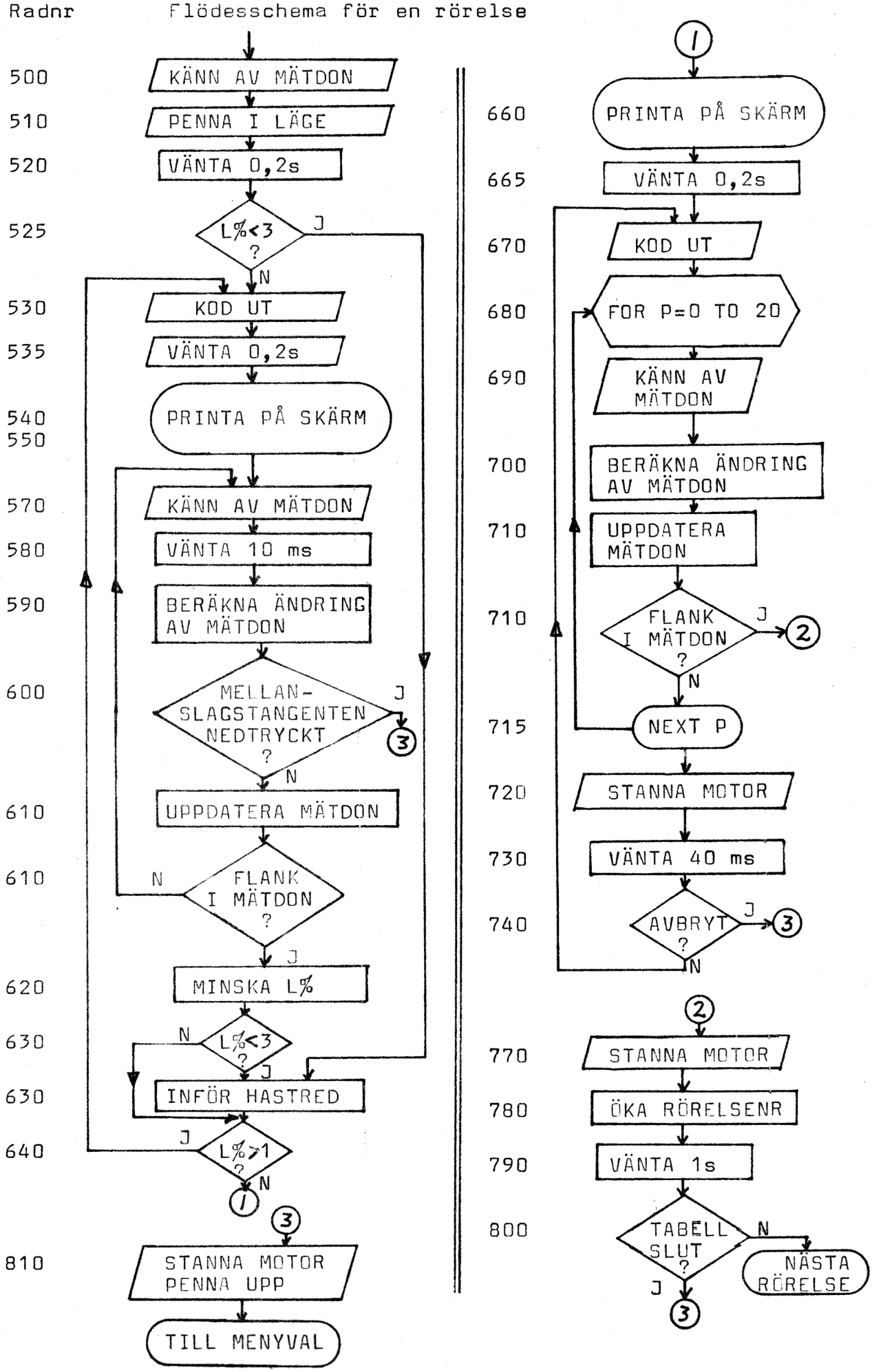
En rörelse utförs i princip så att koden läggs ut och antalet körda steg a 2 mm räknas in via mätdonet. Då angivet antal steg körts är rörelsen klar.

Antalet steg räknas in genom att mätdonets positiva flank känns av. Principen för detta visas i uppgift E2 på sid 4.4 och 5.10.

Körningen kan avbrytas med mellanslagstangenten, rad 600 och 740.
För att motorerna inte skall gå för långt när en rörelse är klar kopplas hastighetsreduktionen in när 2 steg återstår, rad 630. Dessutom pulsas motorn ifråga under sista steget för att ytterligare minska hastigheten.
Detta sker på rad 660-750.
När en rörelse påbörjas går först pennan i läge, rad 510-520.
Då en rörelse är klar lämnas pennan kvar i sitt läge.
Om pennan skall vara nere under flera rörelser i rad kommer den inte att lyftas mellan rörelserna.
Flödesschema för en rörelse visas på nästa sida.

Om en motor går för långt innan den stannar kan man minska siffran 20 något på rad 680.
En tabell kan användas flera gånger för körning.

Radnr Flödesschema för en rörelse



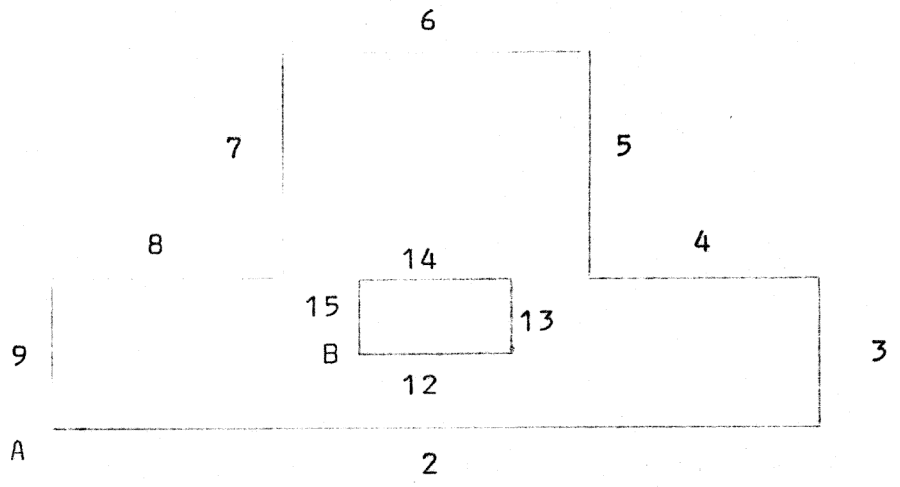
Körexempel

Innan en körning påbörjas ställs bordet i lämpligt läge, eventuellt med manuell körning.

Följande tabell programmerades in för den figur som visas nedan.

Rörelsenr	Kod	Antal steg	Kommentar
0	33	1	Pennan uppe.
1	36	1	Pennan uppe.
2	17	50	10 cm i X-led. Börjar i A.
3	20	10	2 cm i Y-led.
4	18	15	3 cm i -X-led.
5	20	15	3 cm i Y-led.
6	18	20	4 cm i -X-led.
7	24	15	3 cm i -Yled.
8	18	15	3 cm i -X-led.
9	24	10	2 cm i -Y-led. Slutar i A.
10	1	20	Transport till B
11	4	5	med pennan uppe.
12	17	10	
13	20	5	
14	18	10	
15	24	5	Slutar i B.
16	8	5	Transport till A
17	2	20	med pennan uppe.
18	128		Tabellen slut.

Varje tabell bör börja som ovan med att mätdonen körs till väldefinierat läge så att ett helt antal varv körs.
Nedan visas resultatet av en körning med rörelsenr ditskrivna.



```
10 REM RIT 810423
20 REM PROGRAM FöR MODELLFRäS INTRONIC
30 REM DIGITALKORT-R-24V
40 REM TAGE OLSSON MIKRONIK SUNDSVALL
50 OUT 1,19,0,0,2,0,1,0
60 REM ++++++++++++++++++++++++++++++++
70 REM KöRVAL **************************
80 ; CHR$(12) : ONERRORGOTO 80
90 ; "VäLJ MELLAN " : ; : ; : ;
100 ; " 1   PROGRAMMERING AV SEKVENS" : ;
110 ; " 2   KöRNING " : ;
120 ; " 3   SLUT" : ; : ; : ;
130 INPUT C$ : C=VAL(C$)
140 IF C<1 OR C>3 THEN 80
150 ON C GOTO 170,390,820
160 REM ++++++++++++++++++++++++++++++++
170 REM PROGRAMMERING AV SEKVENS ******
180 A%=65408% : REM TABELLBöRJAN
190 N=0 : REM RöRELSENUMMER
200 ; CHR$(12)
210 ONERRORGOTO 290
220 ; "NäSTA RöRELSE"; : ; "     NR";N ; ; : ;
230 ; "          32   16    8    4    2    1"
240 ; "D7   D6   D5   D4   D3   D2   D1   D0"
250 ; "        HRED    Z   -Y   +Y   -X   +X"
260 ; : ; : ; "128 ANGER SLUT" : ;
270 ; "GE KOD "; : INPUT K$
280 K%=VAL(K$) : GOTO 300
290 ; CHR$(12) : ; CHR$(7) : ; "FELAKTIG KOD, OMIGEN" : ; : ; : GOTO 210
300 IF K%<>128% THEN 320
310 POKE A%,K% : GOTO 80
320 ONERRORGOTO 350
330 ; : ; : ; : ; "GE ANTAL STEG A 2 MM" : ;
340 INPUT L$ : L%=VAL(L$) : GOTO 360
350 ; CHR$(7) : ; "FELAKTIG KOD, OMIGEN" : GOTO 320
360 POKE A%,K%,L%
370 N=N+1 : A%=A%+2% : GOTO 200
380 REM ++++++++++++++++++++++++++++++++
390 REM KöRNING ************************
400 ; CHR$(12);"RöRELSENR","KOD","STEG KVAR"
410 A%=65408% : N=0
420 REM S%=GAMLA, T%=NYA MäTDON
430 K%=PEEK(A%) : A%=A%+1%
440 L%=PEEK(A%) : A%=A%+1%
450 IF (K% AND 3%)=0% THEN 480
460 V%=2% : GOTO 490
470 REM V% AR MASK FöR MäTDON X ELLER Y
480 V%=4%
490 REM EN RöRELSE **********************
500 OUT 1,19 : S%=(INP(0) AND V%) : OUT 1,0
510 OUT 1,19,0,K% AND 16%
520 FOR P=0 TO 200 : NEXT P
525 IF L%<3% THEN 630
530 OUT 1,19,0,K%,2,0,1,0
535 FOR P=0 TO 200 : NEXT P
540 ; CUR(3,0)"    ","    ","     "
550 ; CUR(3,0)N,K%,L%
570 OUT 1,19 : T%=(INP(0%) AND V%) : OUT 1,0
580 FOR P=0 TO 10 : NEXT P : REM ANTISTUDS
590 U%=(T% XOR S%) AND T%
600 IF INP(56)=160 THEN 810
610 S%=T% : IF U%=0% THEN 570
620 L%=L%-1%
630 IF L%<3% THEN K%=K% OR 32%
640 IF L%>1% THEN 530
650 REM ETT STEG KVAR ***************
660 ; CUR(3,0)N,K%,L%
665 FOR P=0 TO 200 : NEXT P
670 OUT 1,19,0,K%,2,0,1,0
680 FOR P=0 TO 20
690 OUT 1,19 : T%=(INP(0%) AND V%) : OUT 1,0
700 U%=(T% XOR S%) AND T%
710 S%=T% : IF U%=V% THEN 770
715 NEXT P
720 OUT 1,19,0,K% AND 16%,2,0,1,0
730 FOR P=0 TO 40 : NEXT P
740 IF INP(56)=160 THEN 810
750 GOTO 670
760 REM ********************************
770 OUT 1,19,0,K% AND 16%,2,0,1,0
780 N=N+1
790 FOR P=0 TO 1000 : NEXT P
800 IF PEEK(A%)=128% THEN 810 ELSE 430
810 OUT 1,19,0,0,1,0 : GOTO 80
820 END
```

18. PROGRAM FÖR STEGMOTORER

Under denna punkt visas ett program för styrning av stegmotorer.

Kort: Digitalkort-T-24V eller Digitalkort-T-PUR-24V.

18.1 Stegmotorer

I stegmotorn finns en permanentmagnetisk rotor med fast inpräglade poler. Runt statorn finns ett antal statorok med lindningar så att N-S-poler kan bildas diagonalt, en N-S-pol åt gången. Genom att växla strömmen genom okens lindningshalvor kan N-S-polen i statorn vridas stegvis. Rotorn med sin fasta N-S-pol vrids med.
Stegmotorns funktion behandlas i artikeln "Motorer för elektronik och dataåldern" av Wilhelm Liander i tidskriften Elteknik med aktuell elektronik, nr 5-6, maj/1974. Av samma författare finns boken "Små elektriska motorer".
Exempel på experiment med stegmotorer finns i artikeln "Styra synkronmotorer, inhämta mätdata" av Lars Erik Jansson i tidskriften Radio&Television-nr8-1978.

Här behandlas stegmotorn SP4-415-280 från Copal electronics (finns hos Elproman AB och Multikomponent). På sid 18.2 och 18.3 finns delar av databladet.

Av databladet framgår:
Spänning: 24V DC.
Ström per lindning: 86 mA.
Lindningsresistans per fas: 280 ohm.
Antal faser: 4.
Antal poler: 12.
Stegvinkel: 15^{0}.
Antal steg per varv: 24.

Motorn är hopbyggd med växel. Växel från 1/10 till 1/600 kan väljas. Vi väljer växel 1/25.

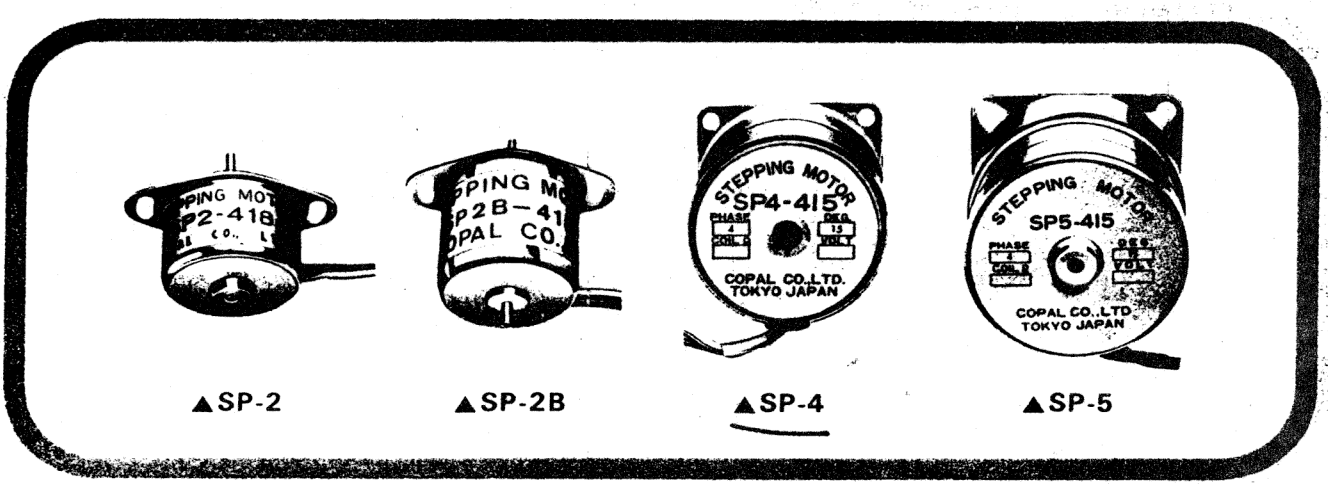
Vi tittar på ett fall då två stegmotorer har monterats så att de bildar en enkel robot. Den ena stegmotorn ger horisontell rörelse och den andra vertikal rörelse av en arm, se sid 18.3. Längst ut på armen sitter en elektromagnet bestående av spole med järnkärna. Genom att skicka ström genom spolen kan små magnetiska föremål, t ex gem, lyftas.

Målsättning för programmet

Med programmet bör ett rörelseschema kunna läggas in. Vid körning bör man kunna stanna roboten och sedan köra den på nytt utan att behöva programmera in rörelseschemat på nytt. Programmet bör vara så utformat att rörelsesekvensen upprepas ända tills en tangent nedtryckes. Roboten bör därvid fullfölja pågående sekvens och stanna vid utgångsläget.
Som exempel på rörelseschema kan vi ta det fall då föremål flyttas från en plats till en annan.
Nedan visas en lösning.
Först måste vi dock titta på hur vi kan få stegmotorerna att rotera.

COPAL SP SERIES STEPPING MOTORS

COPAL ELECTRONICS



▲SP-2 ▲SP-2B ▲SP-4 ▲SP-5

Features

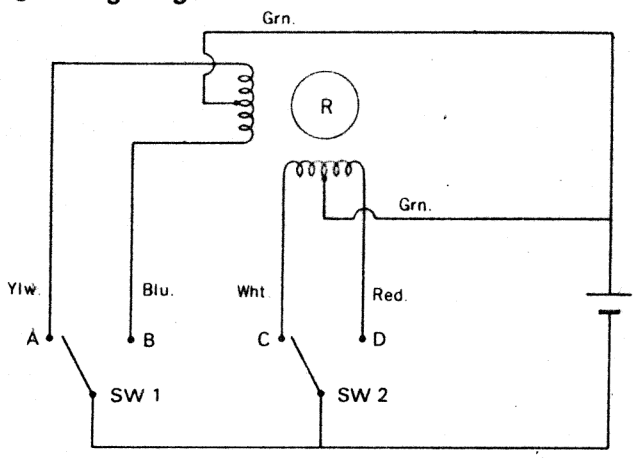
- Ferrite permanent magnet rotor provides high detent torque and excellent start-stop response.
- Wide slew range and bi-directional mode of operation.
- Sealed bearings need no lubrication.
- Motors are compact and light weight.

ELPROMAN AB
Källängsvägen 2
141 71 HUDDINGE
Tel. 08/88 02 50 · Telex 11777

Specifications	SP2-418-60	SP2B-418-60	SP4-415-70	SP4-415-280	SP5-415-40	SP5-415-10		
Winding resistance per phase (ohm)	63	63	70	280	40	10	10	10
External series resistors(ohm)	0	0	0	0	0	0	10	30
DC input voltage(V)	6	6	12	24	12	6	12	24
NO. of phases	4	4	4	4	4	4	4	4
NO. of poles	10	10	12	12	12	12	12	12
Step angle(°)	18	18	15	15	15	15	15	15
NO. of steps per revolution	20	20	24	24	24	24	24	24
Excitation(phases)	2	2	2	2	2	2	2	2
Pull-out torque, min (g·cm) @ 150 PPS	6.5	16	70	60	200	200	300	300
Pull-in pulse rate(PPS), min., @ no load	400	420	250	250	180	180	230	230
Pull-out pulse rate(PPS), min. @ no load	1150	500	320	320	200	200	400	600
Current per winding(mA) @ 0 PPS	95	100	170	86	300	600	600	600
Power input per winding(W) @ 0 PPS	0.6	0.6	2	2.1	3.6	3.6	3.6	3.6
Net weight(g)(motor only)	25	30	100	100	260	260	260	260
Insulation resistance @ DC 500V(Mohm)	100	100	100	100	100	100	100	100
Dielectric strength (Vrms, 1 minute)	500	500	500	500	500	500	500	500
Operating ambient temp. range (C)	−10~+50							
Temp. rise(°C)	40	40	50	50	45	50	50	50

Stepping motor schematic

●Wiring diagram

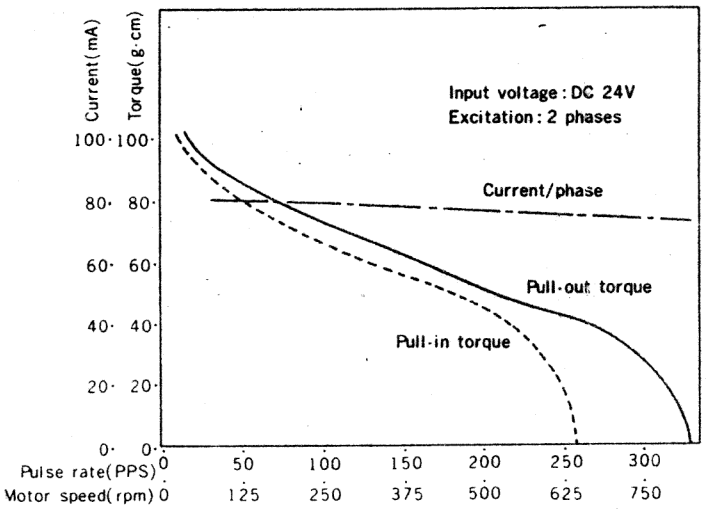


●Excitation

	SW 1		SW 2	
	A	B	C	D
1	ON		ON	
2		ON	ON	
3		ON		ON
4	ON			ON
1	ON		ON	

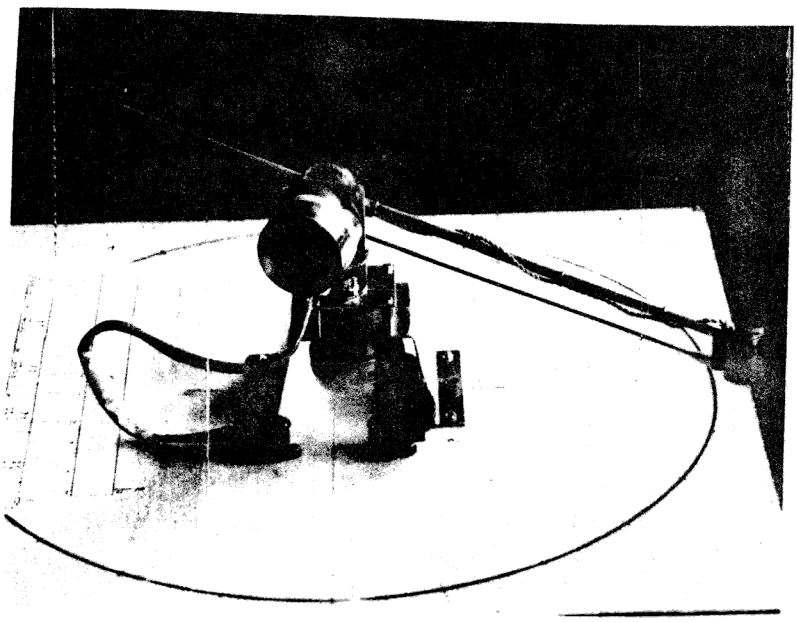
★Switching sequence for CW rotation

●SP4-415-280



Figuren ovan till vänster visar motorns anslutningar.
Figuren ovan till höger visar hur motorn skall pulsas för rotation medurs.

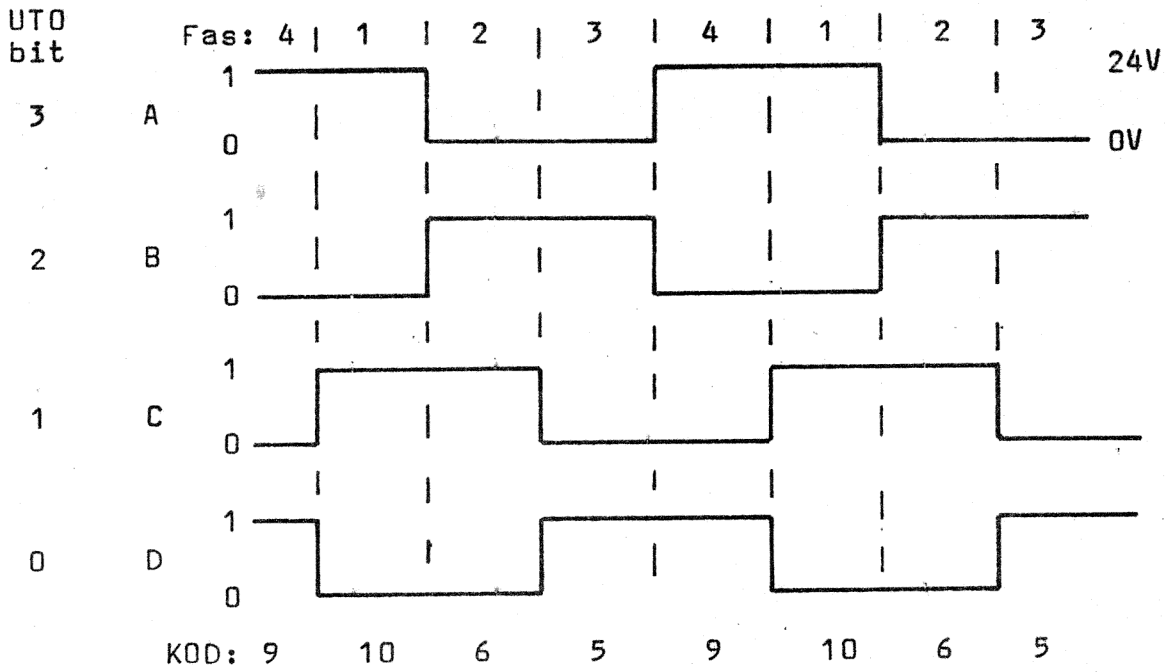
Figuren till vänster visar momentet som funktion av pulsfrekvensen.



Figuren till vänster visar exempel på hur två motorer kan monteras så att de bildar en enkel robot.
I ena änden av armen sitter en elektromagnet och i den andra en motvikt.
Armen är ledad så att magneten alltid är vertikal.

18.2 Styrning av stegmotorn

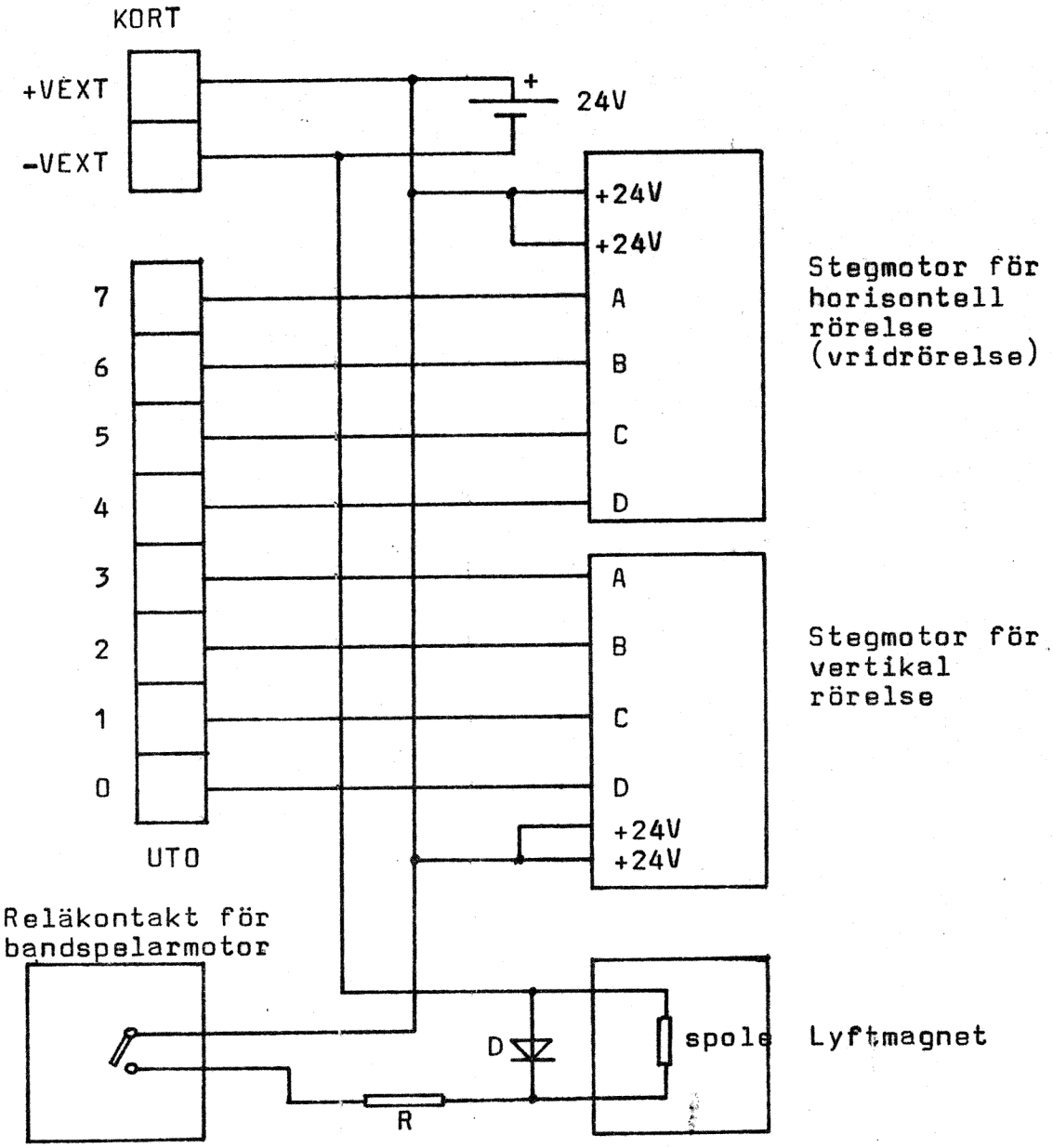
Hur får man motorn att rotera?
Motorn roterar om man lägger spänning över lindningarna efter ett visst mönster som framgår av databladet, sid 18.3.
Programmet skall alltså generera lämpliga pulståg.
Switcharna SW1 och SW2 på sid 18.3 ersättes med 4 transistorer på kortet. Programmet skall generera nedanstående pulståg.



I figuren ovan antages att lindningarna A, B, C och D är anslutna till UTO bit 3-0. Pulstågen erhålles från tabell i databladet. Under figuren finns de koder som ger spänningsnivåerna.
Motorn roterar medurs (CW rotation) om koderna 10, 6, 5, 9, 10, 6 ..... skickas ut till UTO med visst tidsmellanrum.
Vill man att motorn skall rotera åt andra hållet vänder man på fasföljden. Skicka ut koder 9, 5, 6, 10, 9, 5, 6 .....
Rotationshastigheten bestäms av hur lång tid varje kod finns på utgången.
För 4 faser vrids motoraxeln 60^{o}. Med växel 1/25 vrids utaxeln $2,4^{o}$. För att veta var motorn står är det säkrast att alltid köra motorn ett antal multiplar av 4 faser, således $2,4^{o}$. 4 faser, $2,4^{o}$, kallas här för ett steg.

18.3 Anslutningar

På nästa sida visas hur stegmotorerna ansluts till kortet och lyftmagneten till reläkontakten för bandspelarmotorn (på baksidan av tangentbordet).
Reläkontakten klarar av att bryta 0,1A. Resistorn R väljes så att denna gräns icke överskrides. Strömmen genom spolen i lyftmagneten klingar av till 0 genom dioden D vid frånslag.
Reläkontakten sluts med OUT 58,32 och öppnas med OUT 58,0.
Detta kallas för GRIP respektive SLÄPP i programmet.



För att köra en stegmotor används en av de 4 vidstående tabellerna.
I adress 65416-65419 finns de koder som finns under figuren på sid 18.4. De ger rörelse UPP.
Rörelse NED fås med tabellen i adress 65420-65423.
Stegmotorn för horisontell rörelse är ansluten till de 4 mest signifikanta bitarna i UTO. Koderna för MEDURS fås genom att ta koderna för UPP och multiplicera med 16.
Vid programmering av en sekvens måste vi ange både vilken av de 4 tabellerna som skall användas och antal steg (hur många varv tabellen skall genomlöpas).

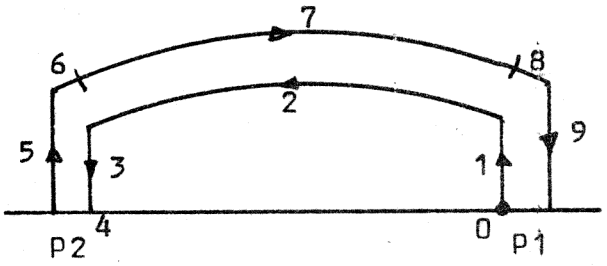
	ADRESS	KOD
MEDURS	65408	160
	409	96
	410	80
	411	144
MOTURS	65412	144
	413	80
	414	96
	415	160
UPP	65416	10
	417	6
	418	5
	419	9
NED	65420	9
	421	5
	422	6
	423	10

18.4 Beskrivning av program

Först beskrivs program ROBOTT för Digitalkort-T. Robotens rörelseschema bestäms av ett antal delsekvenser. Varje delsekvens bestäms av två bytes, den första innehåller sekvenskod och den andra antal steg. Koderna lagras i en sekvenstabell. Sekvenskodens uppbyggnad framgår av rad 210-330 i programmet. Programmeraren bestämmer sig för att vissa bitar ges en viss betydelse. I detta fall har bitarna D7 och D6 reserverats för GRIP och SLÄPP. Med D3 bestäms motorernas hastighet. D1D0 ger ett tal 0, 1, 2 eller 3. Med detta tal får man begynnelseadressen till en av de 4 tabellerna på sid 18.5. Sekvenstabellen illustreras med ett exempel.

Antag vi vill flytta ett föremål från plats P1 till P2.

Rörelseschema med nummer på varje delsekvens:



Som förebild för hur tabellen lagras kan man ha uppgift E4 och E6 i punkt 4 och 5. Efter koderna för GRIP och SLÄPP lägger vi in 0 i stället för antal. Av datablad sid 18.3 framgår hur snabbt motorerna kan pulsas. I exemplet är alla rörelser långsamma utom nr 7. Vill man köra snabbt måste man först accelerera, nr 6, och på slutet retardera, nr 8.

Sekvenstabell:

delsekvens nummer	KOD	kommentar
0	192	GRIP
	00	
1	2	UPP
	10	10 steg
2	1	MOTURS
	30	30 steg
3	3	NED
	10	10 steg
4	128	SLÄPP
	0	
5	2	UPP
	10	10 steg
6	0	MEDURS
	2	2 steg
7	8	MEDURS snabb
	26	26 steg
8	0	MEDURS
	2	2 steg
9	3	NED
	10	10 steg
10	255	SLUTKOD

Ett program som medger att vi kan programmera in en sekvenstabell och sedan köra visas på sid 18.7.

Radnr	Kommentar
40	Inläggning av tabellerna från sid 18.5.
50	Kortadressering. Nollställ utgångar.
70-140	Körval. Felhantering ingår.
170-180	Startadress för sekvenstabell. Delsekvensnummer=0.
200-390	Sekvenskod matas in. Felhantering ingår.
390	Slutkod lagras. Återhopp till körval.
410-510	Antal steg matas in. Felhantering ingår.
420-430	Kod för GRIP/SLÄPP medför att antal steg sätts=0.
480	Sekvenskod och antal steg lagras i tabellen.
540-780	Roboten körs.
560	Startadress för sekvenstabell.
570-580	Sekvenskod och antal steg hämtas från tabellen.
590	Om GRIP/SLÄPP hoppa till 720.
610	Bestäm snabb eller långsam rörelse.
620	R%=A1A0
630-690	En motor körs angivet antal steg (L%·2,4°).
640	Startadress för en av de 4 tabellerna på sid 18.5 ber..
650-680	Motorn körs ett steg (4 faser på sid 18.4).
670	H%=40% ger frekvensen 125 Hz, H%=80% ger drygt 200 Hz. Detta är frekvensen på pulstågen sid 18.4. Frekvensen kan mätas med oscilloskop. (Vid den lägre frekvensen tar 1 steg 80 ms, således ett motorvarv 480 ms. Motorns varvtal = 125 varv/min).

Radnr	Kommentar
700	Hopp till delsekvensslut.
720	GRIP/SLÄPP
740-750	Om nästa kod anger sekvens slut känner datorn av bit 7 i minnesadress 65013. Om denna bit är 1 har en tangent nedtryckts. Se ref 5.
740-780	Om en tangent nedtrycks stannas roboten sedan sekvensen är slut. I annat fall upprepas sekvensen. Fördelen med detta är att roboten kan stannas där rörelsen påbörjats.

Anmärkning. Om det vid provkörning visar sig att en motor går åt fel håll är det enklast att i menyn låta medurs och moturs byta plats (rad 290-300). Detsamma gäller upp-ned (rad 310-320).

```
10 REM ROBOTT  810202
20 REM MIKRONIK SUNDSVALL
30 REM DIGITALKORT-T
40 POKE 65408,160,96,80,144,144,80,96,160,10,6,5,9,9,5,6,10
50 OUT 1,19,0,0,2,0 : REM KORTADR
60 REM ++++++++++++++++++++++++++++++++
70 ; CHR$(12) : ONERRORGOTO 70
80 ; "KÖRVAL,  VÄLJ MELLAN" : ; : ; : ; : ;
90 ; "1   PROGRAMMERING AV RÖRELSE" : ;
100 ; "2   KÖRNING" : ;
110 ; "3   SLUT" : ; : ;
120 INPUT K$ : K=VAL(K$)
130 IF K<1 OR K>3 THEN 70
140 ON K GOTO 150,520,790
150 REM ++++++++++++++++++++++++++++++++
160 REM INMATNING AV SEKVENS **********
170 A%=65440% : REM START TABELLADRESS
180 N%=0% : REM DELSEKVENSNUMMER
190 REM SEKVENSKOD **********************
200 ; CHR$(12) : ONERRORGOTO 360
210 ; "NÄSTA DELSEKVENS   NR";N% : ;
220 ; "128  64  32  16   8   4   2   1"
230 ; " D7  D6  D5  D4  D3  D2  D1  D0"
240 ; "  S   I           H      A1  A0" : ;
250 ; "S=1  GRIP/SLÄPP,   S=0  RÖRELSE" : ;
260 ; "I=1  GRIP,    I=0  SLÄPP" : ;
270 ; "H=1  SNABB RÖRELSE"
280 ; "H=0  LÅNGSAM RÖRELSE" : ;
290 ; "A1A0=00 RÖRELSE MEDURS"
300 ; "     =01         MOTURS"
310 ; "     =10  RÖRELSE UPP"
320 ; "     =11 RÖRELSE NED" : ;
330 ; "KOD 255 ANGER SLUT PÅ DELSEKVENSER" : ;
340 ; "GE KOD " : INPUT K$
350 K%=VAL(K$) : GOTO 380
360 ; CHR$(12) : ; CHR$(7)
370 ; "FELAKTIG KOD, OMIGEN" : GOTO 200
380 IF K%<>255% THEN 400
390 POKE A%,K% : GOTO 70
400 REM ANTAL STEG **********************
410 ; CHR$(12)
420 IF (K% AND 128%)=0% THEN 440
430 L%=0% : GOTO 480
440 ONERRORGOTO 500
450 ; "FÖR RÖRELSE GE ANTAL STEG" : ; : ;
460 INPUT L$
470 L%=VAL(L$)
480 POKE A%,K%,L% : A%=A%+2%
490 N%=N%+1% : GOTO 200
500 ; CHR$(7) : ; "FELAKTIG KOD, OMIGEN"
510 GOTO 440
520 REM ++++++++++++++++++++++++++++++++
530 REM KÖRNING **********************
540 ; CHR$(12) : ; "NU KÖR VI" : ; : ; : ;
550 ; "AVBRYT MED ATT NEDTRYCKA TANGENT"
560 A%=65440%
570 K%=PEEK(A%) : A%=A%+1%
580 L%=PEEK(A%) : A%=A%+1%
590 IF (K% AND 128%)=128% THEN 720
600 REM RÖRELSE ***********************
610 IF (8% AND K%)=8% THEN H%=40% ELSE H%=80%
620 R%=3% AND K%
630 FOR M%=1% TO L%
640 A1%=65408%+4%*R%
650 FOR N%=0% TO 3%
660 OUT 0%,PEEK(A1%+N%)
670 FOR T%=1% TO H% : NEXT T%
680 NEXT N%
690 NEXT M%
700 GOTO 740
710 REM GRIP/SLÄPP **********************
720 OUT 58%,(K% AND 64%)/2%
730 REM DELSEKVENSSLUT ***************
740 IF PEEK(A%)<255% THEN 570
750 IF (PEEK(65013%) AND 128%)=0% THEN 780
760 OUT 0,0 : REM NOLLSTÄLL UTGÅNGAR
770 GET A$ : GOTO 70
780 GOTO 560
790 END
```

För Digitalkort-T-PUR ändras nedanstående rader.
Se även sid 1.5.

```
10 REM ROBOTPUR  810202
20 REM MIKRONIK SUNDSVALL
30 REM DIGITALKORT-T-PUR
50 OUT 1,19,0, NOT 0,2,0 : REM KORTADR
660 OUT 0%, NOT PEEK(A1%+N%)
760 OUT 0, NOT 0 : REM NOLLSTÄLL UTGÅNGAR
```

19. PROGRAM FÖR TIDMÄTNING

I ABC80 finns en realtidsklocka som ändras ett steg var 20:e ms. För tider kortare än 1 sek kan den därför inte användas för tidmätning.
Med de program som presenteras nedan kan tiden mellan två händelser mätas. Med händelse menas att en ingång ändrar tillstånd.
Tiden mätes genom att programmen beräknar det antal states (periodtider för klocksignalen i ABC80) som förflyter mellan händelserna. Programdelen för tidmätning är skriven i assembler. Av ref 8 framgår det antal states varje assemblerinstruktion tar i Z80 CPU.

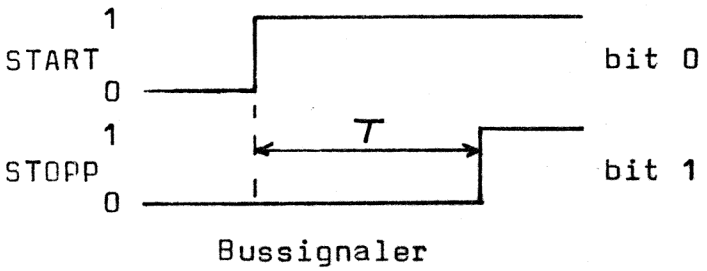
Kort: Tidmätningen startas med en START-signal och stoppas med en STOPP-signal. Det beror på givarna för dessa signaler om TTL-ingång eller optoisolerad ingång är lämplig för ett visst experiment. Av beskrivningarna för de olika korten framgår vilka ingångar respektive kort har.

Innehåll

19.1 Principen för tidmätning
19.2 Program TID1A för tider 2ms - 0,5s
19.3 Program TID1B för tider 2ms - 0,5s
19.4 Program TID2 för tider 2ms - 0,5s med upprepade mätningar
19.5 Program LÅNGTID för tider upp till 230s
19.6 Exempel på givare
19.7 Felberäkning

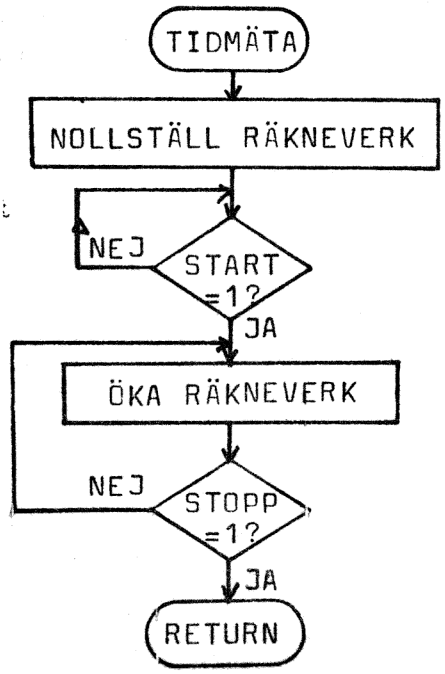
19.1 Principen för tidmätning

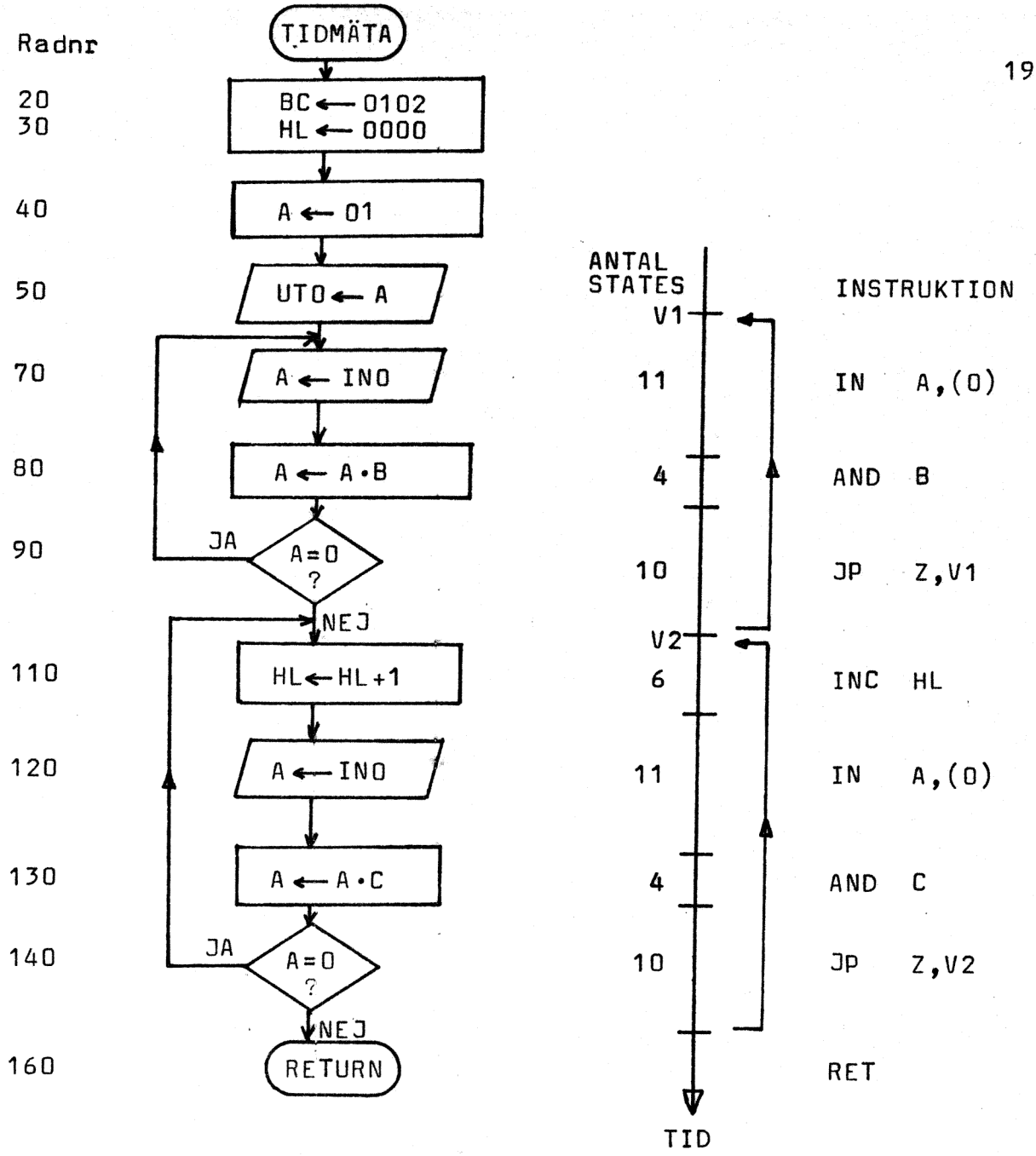
Principen visas för subrutin TIDMÄTA som ingår i TID1A.



Bussignaler

I detta fall startas tidmätningen då bit 0 i IN0 ger logisk ETTA till datorn på bussen. Tidmätningen stoppas då bit 1 i IN0 ger logisk ETTA på bussen till datorn.
Vidstående flödesschema visar principen i stort. Registerparet HL används som räkneverk.
På nästa sida visas detaljerna.





Kommentar till subrutin TIDMÄTA. Radnr efter källkod.

Radnr	Kommentar
10	Subrutinen börjar på adress 65408.
20	B laddas med 01 och C med 02. B utgör mask för START och C för STOPP. AND B tar 4 states men AND 01 7 states.
30	Räkneverket HL nollställs.
40	Bit 0 i A 1-ställs, övriga bitar nollställs.
50	A till UTO. Bit 0 i UTO kan användas för att starta ett förlopp.
70	Läs in INO till A.
80	Nollställ alla bitar utom nr 0 (START).
90	Hoppa tillbaka till 70 om bit 0 = 0 (ingen START-puls).
110	Öka räkneverk med 1.
120	Läs in INO till A.
130	Nollställ alla bitar utom nr 1 (STOPP).
140	Hoppa tillbaka till 110 om bit 1 = 0 (ingen STOPP-puls).
160	Åter till BASIC.

Under väntan på START går programmet runt i den övre loopen som är på 25 states. Efter START går programmet runt i den nedre loopen ända tills STOPP inträffar.
Ett varv tar 31 states.
Vid återhopp överförs innehållet i HL till Basic.
I medeltal har halva övre loopen avverkats då START-puls kommer. Likaså har i medeltal halva nedre loopen avverkats då STOPP-puls kommer.
Vid beräkning av antalet states börjar vi med (25+31)/2.
Om vi antar att STOPP-pulsen kommer under andra varvet i nedre loopen får HL innehållet 2 vid återhopp till Basic. Därför minskas antalet med 1 i Basic.

I Basic beräknas X= innehållet i HL minus 1.

Antalet states blir (25+31)/2 + X•31 .

Om STOPP-pulsen kommer under andra varvet enligt ovan får HL innehållet 2 och X beräknas till 1. Antalet states beräknas för ett och ett halvt varv i nedre loopen enligt formeln ovan.
Klockfrekvensen i ABC80 är 2,9952 MHz och styrs av en kristall med fyrdubbla frekvensen.
Tiden i ms fås genom att dividera antalet states med frekvensen i kHz.

T=((25+31)/2+X•31)/2995.2 ger tiden i ms (rad 220 i TID1A och TID1B).

Ovan beskrivna princip för tidmätning ger en osäkerhet som är ±(25+31)/2/2.9952 µs = ±10 µs. Se vidare 19.7 nedan.

Maximala tiden som kan mätas erhålles med X=65534 som ger T=678ms. (Av säkerhetsskäl anges 0,5s).

19.2 Program TID1A för tider 2ms - 0,5s

I programmet ingår subrutin TIDMÄTA. Programmen finns på sid 19.4. Eftersom vissa ingångskretsar ger invertering anges i programmet de olika triggnivåerna.

Radnr	Kommentar
10-20	Subrutin TIDMÄTA.
60-115	Beskriver vilka nivåer på olika kortingångar som ger triggning.
146	Kortadressering. Utgångar nollställs.
148-165	Kontroll av att insignalerna står i rätt utgångsläge.
200	Subrutinen TIDMÄTA anropas.
205	Utgång nollställs.
210	Justering från tvåkomplement.
220	Tiden beräknas, se ovan.
230-240	Var 20:e ms stjäl realtidsklockan 6 µs. Tillägg för detta.
250	Utskrift av resultat.
260-280	Beslut om fortsättningen.

Är man osäker på om ett förlopp tar längre tid än 0,5s kan man först använda programmet LÅNGTID, se punkt 19.5.

Källkod för subrutin TIDMÄTA.

```
10        ORG   65408    ;TIDMÄTA
20 INIT:  LD    BC,0102H ;MASKER
30        LD    HL,00000H ;RÄKNEVERK
40        LD    A,01     ;GE
50        OUT   (0),A    ;STARTPULS
60                       ;
70 V1:    IN    A,(0)    ;LÄS IN START
80        AND   B        ;OCH VÄNTA PÅ
90        JP    Z,V1     ;ACTION
100                      ;
110 V2:   INC   HL       ;ÖKA RÄKNEVERK
120       IN    A,(0)    ;LÄS IN STOPP
130       AND   C        ;OCH VÄNTA PÅ
140       JP    Z,V2     ;ACTION IGEN
150                      ;
160       RET            ;NU ÄR DET SLUT
```

Program TID1A.

```
1 REM TID1A 810206
2 REM MIKRONIK SUNDSVALL
6 ; CHR$(12)
7 ; "ENSTAKA TIDER UNDER 0.5 S MÄTS" : ; : ;
8 REM ++++++++++++++++++++++++++++++++
9 REM SUBRUTIN TIDMÄTA **********************
10 POKE -128%,1%,2%,1%,33%,0%,0%,62%,1%,211%,0%,219%,0%,160%
20 POKE -115%,202%,138%,255%,35%,219%,0%,161%,202%,144%,255%,201%
30 REM
40 REM
50 REM ++++++++++++++++++++++++++++++++
60 ; "TRIGGNING DÅ" : ; : ;
70 ; "-OPTOISOLERADE INGÅNGAR BLIR HÖGA" : ; : ;
80 ; "-TTL-INGÅNGAR PÅ" : ;
90 ; "     DKORT-0-16TTL,"
95 ; "     DKORT-32TTL BLIR HÖGA" : ; : ;
100 ; "-TTL-INGÅNGAR PÅ" : ;
110 ; "     DKORT-T, DKORT-T-PUR,"
115 ; "     DKORT-R BLIR LÅGA" : ; : ;
120 ; "TRYCK PÅ TANGENT FÖR FORTS"
130 GET A$
140 REM ++++++++++++++++++++++++++++++++
145 REM KONTROLL AV INSIGNALER ********
146 OUT 1,19,2,0,0,0 : REM KORTADR, NOLLSTÄLL UTGÅNGAR
148 ; CHR$(12)
150 IF (INP(0) AND 1)=0 THEN 160
155 ; CUR(0,0);"STARTSIGNAL STÅR FEL" : GOTO 150
160 IF (INP(0) AND 2)=0 THEN 190
165 ; CUR(4,0);"STOPPSIGNAL STÅR FEL" : GOTO 160
170 REM ++++++++++++++++++++++++++++++++
180 REM KÖRNING **********************
190 ; CHR$(12) : ; "NU KÖR VI" : ; : ; : ;
200 X%=CALL(65408) : REM MÄTNING
205 OUT 0,0 : REM NOLLSTÄLL UTGÅNG
210 IF X%<0% THEN X=65536+X%-1% ELSE X=X%-1%
220 T=((25+31)/2+X*31)/2995.2
230 P=INT(T/20) : REM KORRIGERA FÖR
240 T=T+P*.006 : REM REALTIDSKLOCKAN
250 ; "TIDEN ÄR =";INT(100*T+.5)/100;" MS" : ; : ; : ;
260 ; "VILL DU KÖRA IGEN (J/N) ?" : ; : ;
270 GET A$
280 IF A$="J" OR A$="J" THEN 140
290 END
```

Källkod för subrutin TIDMÄTB.

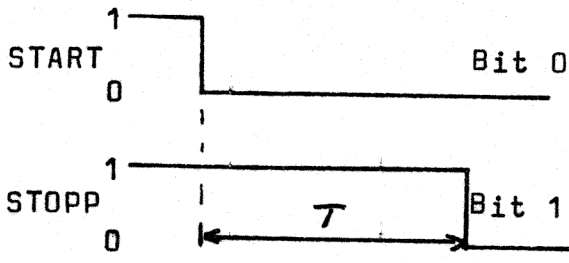
```
10        ORG   65408     ;TIDMÄTB
20 INIT:  LD    BC,0102H  ;MASKER
30        LD    HL,00000H ;RÄKNEVERK
40        LD    A,01      ;GE
50        OUT   (0),A     ;STARTPULS
60                        ;
70 V1:    IN    A,(0)     ;LÄS IN START
80        AND   B         ;OCH VÄNTA PÅ
90        JP    NZ,V1     ;ACTION
100                       ;
110 V2:   INC   HL        ;ÖKA RÄKNEVERK
120       IN    A,(0)     ;LÄS IN STOPP
130       AND   C         ;OCH VÄNTA PÅ
140       JP    NZ,V2     ;ACTION IGEN
150                       ;
160       RET             ;NU ÄR DET SLUT
```

Program TID1B.

```
1 REM TID1B 810206
2 REM MIKRONIK SUNDSVALL
6 ; CHR$(12)
7 ; "ENSTAKA TIDER UNDER 0.5 S MÄTS" : ; : ;
8 REM ++++++++++++++++++++++++++++++++
9 REM SUBRUTIN TIDMÄTB **********
10 POKE -128%,1%,2%,1%,33%,0%,0%,62%,1%,211%,0%,219%,0%,160%
20 POKE -115%,194%,138%,255%,35%,219%,0%,161%,194%,144%,255%,201%
30 REM
40 REM
50 REM ++++++++++++++++++++++++++++++++
60 ; "TRIGGNING DÅ" : ; : ;
70 ; "-OPTOISOLERADE INGÅNGAR BLIR LÅGA" : ; : ;
80 ; "-TTL-INGÅNGAR PÅ" : ;
90 ; "     DKORT-0-16TTL,"
95 ; "     DKORT-32TTL BLIR LÅGA" : ; : ;
100 ; "-TTL-INGÅNGAR PÅ" : ;
110 ; "     DKORT-T, DKORT-T-PUR,"
115 ; "     DKORT-R BLIR HÖGA" : ; : ;
120 ; "TRYCK PÅ TANGENT FÖR FORTS"
130 GET A$
140 REM ++++++++++++++++++++++++++++++++
145 REM KONTROLL AV INSIGNALER ********
146 OUT 1,19,2,0,0,0 : REM KORTADR, NOLLSTÄLL UTGÅNGAR
148 ; CHR$(12)
150 IF (INP(0) AND 1)=1 THEN 160
155 ; CUR(0,0);"STARTSIGNAL STÅR FEL" : GOTO 150
160 IF (INP(0) AND 2)=2 THEN 190
165 ; CUR(4,0);"STOPPSIGNAL STÅR FEL" : GOTO 160
170 REM ++++++++++++++++++++++++++++++++
180 REM KÖRNING **********************
190 ; CHR$(12) : ; "NU KÖR VI" : ; : ; : ;
200 X%=CALL(65408) : REM MÄTNING
205 OUT 0,0 : REM NOLLSTÄLL UTGÅNG
210 IF X%<0% THEN X=65536+X%-1 ELSE X=X%-1
220 T=((25+31)/2+X*31)/2995.2
230 P=INT(T/20) : REM KORRIGERA FÖR
240 T=T+P*.006 : REM REALTIDSKLOCKAN
250 ; "TIDEN ÄR =";INT(100*T+.5)/100;" MS" : ; : ; : ;
260 ; "VILL DU KÖRA IGEN (J/N) ?" : ; : ;
270 GET A$
280 IF A$="J" OR A$="J" THEN 140
290 END
```

19.3 Program TID1B för tider 2ms - 0,5s

Program TID1B avviker från TID1A enbart i fråga om avkänning av insignalerna, se vidstående figur.
Till TID1B hör subrutin TIDMÄTB. Av källkoden för TIDMÄTB ser vi att det är hoppen på rad 90 och 110 som avviker från TIDMÄTA.
Programmen finns på sid 19.5.
De ingångssignaler som ger triggning anges på rad 60-115 i program TID1B.



Bussignaler

19.4 Program TID2 för tider 2ms - 0,5s med upprepade mätningar

Detta program innehåller subrutin TIDMÄTA. De insignaler som ger triggning framgår av program TID2.
I början av programmet ger användaren en del uppgifter om försöket. Då försöket utförts erhålles en del statistiska uppgifter såsom medelvärde, standardavvikelse, undre gräns och övre gräns. Programmet finns på sid 19.7.

Radnr	Kommentar
10-20	Subrutin TIDMÄTA.
120	Kortval. Nollställning av utgångar.
130-190	Inmatning av antal försök, tid mellan försök samt troligt medelvärde. Tiden mellan försök bör vara så lång att försöksobjektet hinner återhämta sig. Om man ger ett troligt medelvärde förbättras noggrannheten av beräkning av standardavvikelse. Troligt medelvärde kan givetvis sättas till 0.
220-250	Begynnelsevärden för statistikstorheter.
260	Försöksnummer=1.
330-350	Kontroll av att allt är klart.
370	Ett försök utföres.
375	Utgång nollställs.
380	Justering från tvåkomplement.
390	Tiden beräknas.
400-410	Justering med hänsyn till realtidsklockan.
420	Avrundning.
430	Utskrift av försöksresultat på skärm.
510-570	Statistikuppföljning. K är en summa av kvadrater som sedan används för beräkning av standardavvikelse.
610	Försöksnummer ökas med 1.
630	Väntetid mellan försök.
640	Fler försök?
700-800	Statistikuppföljning.
720	Medelvärde beräknas.
730	Standardavvikelsen Z beräknas. Allmänt gäller att

$$Z^2 = \frac{\sum^{n}(X-M)^2}{n}$$ där X=värdet från ett försök, M=medelvärde, n=antal försök

Vidare gäller $M = \frac{\sum^{n} X}{n}$. Utveckling av Z^2 ger

$$Z^2 = \frac{\sum^{n} X^2 + \sum^{n} M^2 - 2M\sum^{n} X}{n} \; ; \; \sum^{n} M^2 = nM^2 = (\sum^{n} X)^2/n.$$

Förenkling ger $Z^2 = \sum^{n} X^2/n - (\sum^{n} X)^2/n/n$ som används.

$K = \sum^{n} X^2$, $S = \sum^{n} X$, N1=n i programmet.

```
1 REM TID2 810206
2 REM MIKRONIK SUNDSVALL
8 REM ++++++++++++++++++++++++++++++++
9 REM SUBRUTIN TIDMÄTA ****************
10 POKE -128%,1%,2%,1%,33%,0%,0%,62%,1%,211%,0%,219%,0%,160%
20 POKE -115%,202%,138%,255%,35%,219%,0%,161%,202%,144%,255%,201%
30 REM
40 REM
50 REM ++++++++++++++++++++++++++++++++
60 ; CHR$(12) : ; "TRIGGNING DÅ" : ; : ;
70 ; "-OPTOISOLERADE INGÅNGAR BLIR HÖGA" : ; : ;
75 ; "-TTL-INGÅNGAR PÅ" : ;
80 ; "     DKORT-0-16TTL,"
85 ; "     DKORT-32TTL BLIR HÖGA" : ; : ;
90 ; "-TTL-INGÅNGAR PÅ" : ;
92 ; "     DKORT-T, DKORT-T-PUR,"
94 ; "     DKORT-R BLIR LÅGA" : ; : ; :
96 ; "TRYCK PÅ TANGENT FÖR FORTS."
98 GET A$
100 REM ++++++++++++++++++++++++++++++++
110 REM KÖRVAL ************************
120 OUT 1,19,0,0,2,0 : REM KORTADR, NOLLSTÄLL UTGÅNGAR
130 ; CHR$(12) : ONERRORGOTO 130 : ; CHR$(7)
140 ; "MED DETTA PROGRAM GÖRES ETT" : ;
150 ; "ANTAL FÖRSÖK MED TIDMÄTNING" : ;
155 ; "FÖR TIDER UNDER 0.5 S" : ; : ; : ;
160 ; "GE ANTAL FÖRSÖK  "; : INPUT N1 : ; : ;
170 ; "GE TID MELLAN FÖRSÖK I SEK  ";
180 INPUT T1 : T2=1000*T1 : ; : ; : ;
190 ; "GE TROLIGT MEDELVÄRDE  " : INPUT M0 : ; : ; : ;
200 REM ++++++++++++++++++++++++++++++++
210 REM STATISTIK BEGYNNELSEVÄRDEN ****
220 S=0 : REM SUMMA
230 K=0 : REM KVADRATSUMMA
240 U=1E+7 : REM UNDRE GRÄNS
250 ö=0 : REM ÖVRE GRÄNS
260 N=1 : REM FÖRSÖK NUMMER
300 REM ++++++++++++++++++++++++++++++++
310 REM KÖRNING ************************
320 ; "ÄR DET KLART FÖR KÖRNING?"
340 GET A$ : ;
350 IF A$="J" THEN 360 ELSE 320
360 ; CHR$(12)
370 X%=CALL(65408) : REM MÄTNING
375 OUT 0,0 : REM NOLLSTÄLL UTGÅNGAR
380 IF X%<0% THEN X=65536+X%-1 ELSE X=X%-1
390 T=((25+31)/2+X*31)/2995.2
400 P=INT(T/20) : REM KORRIGERA FÖR
410 T=T+P*.006 : REM REALTIDSKLOCKAN
420 T=INT(100*T+.5)/100
430 ; "FÖRSÖK NR ";N;"   TID=";T;" MS" : ;
500 REM STATISTIKUPPFÖLJNING **********
510 T0=T-M0
520 S=S+T0
530 K=K+T0*T0
540 IF T>U THEN 560
550 U=T
560 IF T<ö THEN 600
570 ö=T
600 REM ÄR DET SLUT ********************
610 N=N+1
630 FOR P=1 TO T2 : NEXT P
640 IF N>N1 THEN 700 ELSE 370
700 REM ++++++++++++++++++++++++++++++++
710 REM STATISTIK UTSKRIFT *************
715 ; : ; : ; "TRYCK PÅ TANGENT FÖR STATISTIK" : GET A
720 M=S/N1+M0
730 Z=SQR(K/N1-S*S/N1/N1)
740 ; CHR$(12)
750 ; "RESULTAT" : ; : ; : ;
760 ; "MEDELVÄRDE= ";M;" MS" : ; : ;
770 ; "STANDARDAVVIKELSE= ";Z;" MS" : ; : ;
780 ; "UNDRE GRÄNS= ";U;" MS" : ; : ;
790 ; "ÖVRE GRÄNS = ";ö;" MS" : ; : ;
800 ; "ANTAL FÖRSÖK= ";N1 : ; : ;
810 END
```

19.5 Program LÅNGTID för tider upp till 230s.

Programmet LÅNGTID med subrutinen TIDLÅNG finns på sid 19.9.
Här ges enbart en kortfattad beskrivning.
Tidmätningen startas och stoppas då insignalerna ändrar tillstånd. Detta åstadkommes med operationen XOR.

Subrutin TIDLÅNG

För tider över 0,5s behövs 3 register för räkneverket.
Loopen för START tar 29 states.
Efter denna känns STOPP-signalen av och detta tar 22 states.
Som räkneverk används registren D och HL.
I loopen för STOPP räknas D ned ett steg för varje varv.
Varje gång D=0 ökas HL med 1. Således räknas D ned under 255 varv och dessa tar 43 states var. Under det 256:e varvet då D=0 och HL ökas med 1 tar varvet 49 states.
HL överförs till BASIC på vanligt sätt. D överförs via minnescell 65535.

Program LÅNGTID

I BASIC-programmet sätts X=innehållet i HL och Y=innehållet i D.
Antalet states beräknas som

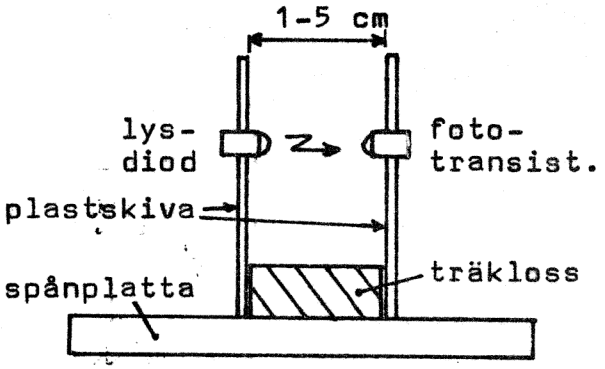
$$((29+43)/2 + 22 + X \cdot 255 \cdot 43 + X \cdot 49 + (256-Y) \cdot 43 - 43$$

- $((29+43)/2$: Antalet states för ett halft varv i START- och STOPP-loppen.
- 22: Avkänning av STOPP-signal.
- $X \cdot 255 \cdot 43$: 255 varv då HL icke ökats.
- $X \cdot 49$: 256:e varvet då HL ökas.
- $(256-Y) \cdot 43 - 43$: Antal varv sedan HL ökades sist.

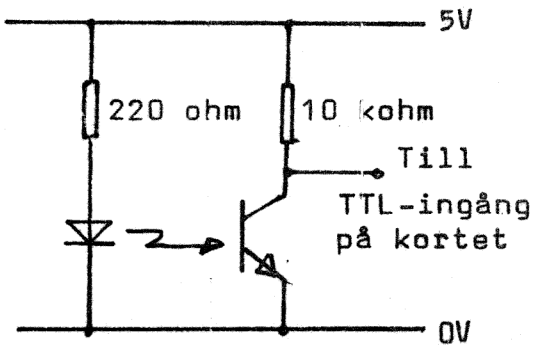
19.6 Exempel på givare

Optoelektriska givare

En lysdiod och en fototransistor monteras enligt vidstående figur.
Då ett föremål bryter ljusstrålen fås en puls.
Med två uppsättningar kan man ta tiden mellan brytning av ena ljusstrålen till brytning av den andra.



Den elektriska kopplingen visas i nedre figuren.
Komponentförslag:
Lysdiod: CQY11C (Philips)
Fototransistor: BPX25 (Philips).
Våglängd: nästan infrarött ljus.
Båda är försedda med linser och har smala lober.



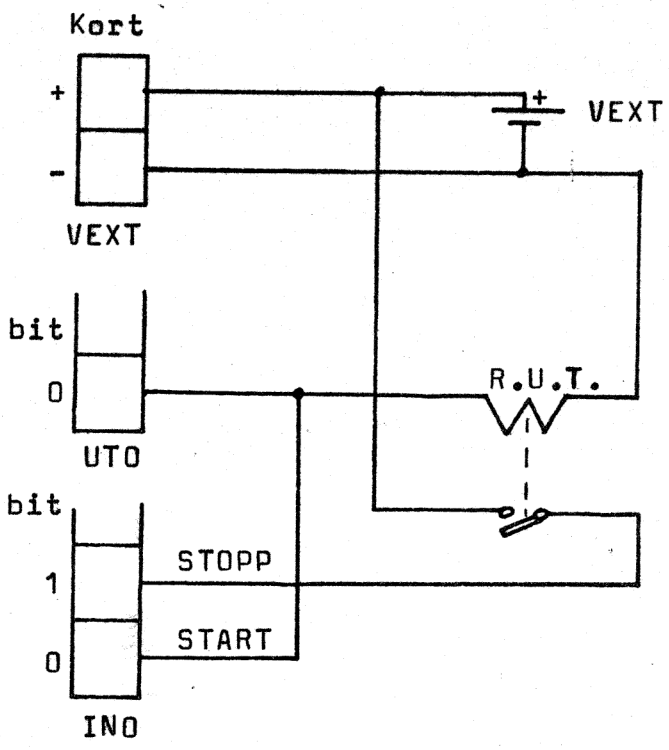
```
1 REM LANGTID 810206
2 REM MIKRONIK SUNDSVALL
5 REM ++++++++++++++++++++++++++++++++
6 REM SUBRUTIN TIDLANG ***************
10 POKE -128%,1%,2%,1%,33%,0%,0%,22%,0%,219%,0%,230%,1%,95%
20 POKE -115%,62%,1%,211%,0%,219%,0%,160%,171%,202%,145%,255%,219%,0%
30 POKE -102%,230%,2%,95%,21%,194%,162%,255%,35%,219%,0%,161%,171%,202%,157%,255%
40 POKE -87%,122%,50%,255%,255%,201%
50 REM
60 REM ++++++++++++++++++++++++++++++++
65 ; CHR$(12) : ; "MÄTNING AV TIDER" : ; : ;
70 ; "FRÅN 0.1 SEK TILL 230 SEK" : ; : ; : ; : ;
75 ; "TRIGGNING DÅ YTTRE" : ;
80 ; "INGÅNG ÄNDRAR TILLSTÅND" : ; : ; : ; : ;
90 ; "TRYCK PÅ TANGENT DÅ DET ÄR" : ;
95 ; "KLART FÖR START" : GET A$
100 OUT 1,19,0,0,2,0 : REM KORTADR, NOLLSTÄLL UTGÅNGAR
105 REM ++++++++++++++++++++++++++++++++
110 X%=CALL(65408) : REM MÄTNING
112 OUT 0,0 : REM NOLLSTÄLL UTGÅNG
115 Y=PEEK(65535) : REM HÄMTA MINST SIGNIFIKANTA BYTE
120 IF X%<0% THEN X=65536+X% ELSE X=X%
130 T=((29+43)/2+22+X*255*43+X*49+(256-Y)*43-43)/2.9952E+6
140 P=INT(T/.02) : REM KORRIGERING FÖR
150 T=T+P*.000006 : REM REALTIDSKLOCKAN
160 ; : ; : ; : ;
165 ; CHR$(12)
170 IF T<1 THEN 200
180 ; "TIDEN ÄR =";INT(10000*T+.5)/10000;" S"
190 GOTO 210
200 ; "TIDEN ÄR =";INT(1E+5*T+.5)/100;" MS"
210 ; : ; : ; : ; "VILL DU KÖRA OMIGEN ? (J/N) "
220 GET A$
230 IF A$="J" OR A$="J" THEN 65
240 END
```

Källkod för subrutin TIDLÅNG.

```
10          ORG   65408     ;TIDLÅNG
20 INIT:    LD    BC,0102H  ;MASKER
30          LD    HL,00000H ;RÄKNE-
40          LD    D,0       ;VERK
50          IN    A,(0)     ;KÄNN
60          AND   01H       ;AV
70          LD    E,A       ;STARTSIGNAL
80                            ;
90          LD    A,01      ;GE
100         OUT   (0),A     ;STARTPULS
110                           ;
120 V1:     IN    A,(0)     ;LÄS IN START
130         AND   B         ;OCH VÄNTA PÅ
140         XOR   E
150         JP    Z,V1      ;ACTION
160                           ;
170         IN    A,(0)     ;KÄNN
180         AND   02H       ;AV
190         LD    E,A       ;STOPPSIGNAL
200                           ;
210 V2:     DEC   D
220         JP    NZ,HOPP
230         INC   HL
240 HOPP:   IN    A,(0)     ;LÄS IN STOPP
250         AND   C         ;OCH VÄNTA PÅ
260         XOR   E
270         JP    Z,V2      ;ACTION IGEN
280                           ;
290         LD    A,D
300         LD    (0FFFFH),A
310         RET
```

Mätning av tillslagstid för relä

Tillslagstiden för ett relä R.U.T. (Relay Under Test) kan mätas med vidstående koppling.
I detta fall användes Digitalkort-R med optoisolerade ingångar.
Ett antal mätningar kan göras med program TID2.
I början av subrutinen får R.U.T. spänning.
Denna ger även START-signal.
STOPP-signal erhålles då kontakten i R.U.T. har slutits.
Efter återhoppet till Basic blir reläet strömlöst igen.



19.7 Felberäkning

Följande felkällor kan förekomma.

A. Osäkerhet i kristallens frekvens: ±50 ppm är ett vanligt värde vid 25°C (miljondelar).

B. Metodfel. Beskrivs på sid 19.3.
För subrutin TIDMÄTA och TIDMÄTB är det max ±10 µs.
För subrutin TIDLÅNG är det 12 µs.

C. Beräkningen av antalet nedräkningar av realtids-klockan kan slå fel på en gång. Osäkerhet 6 µs.

D. Fördröjning i ingångskretsarna.
För TTL-ingångar är den försumbart liten.
Med optoisolerade ingångar fördröjs signalen:
Då den yttre insignalen ändras LÅG-HÖG är fördröjn. 15 µs, då den yttre insignalen ändras HÖG-LÅG är fördröjn. 80 µs.
Om START- och STOPP-signalen fördröjs lika mycket blir det inget fel. Antag maximala skillnaden mellan olika kanaler är 5 µs respektive 25 µs (konservativ gissning).

E. Fördröjning i eventuella yttre givare.
(För optoelektriska givaren i punkt 19.6 gäller 5 µs).

Exempel : Antag tillslagstiden för reläet ovan är c:a 20 ms.

Felkälla A: 1 µs
B: 10 µs
C: 6 µs
D: 5 µs

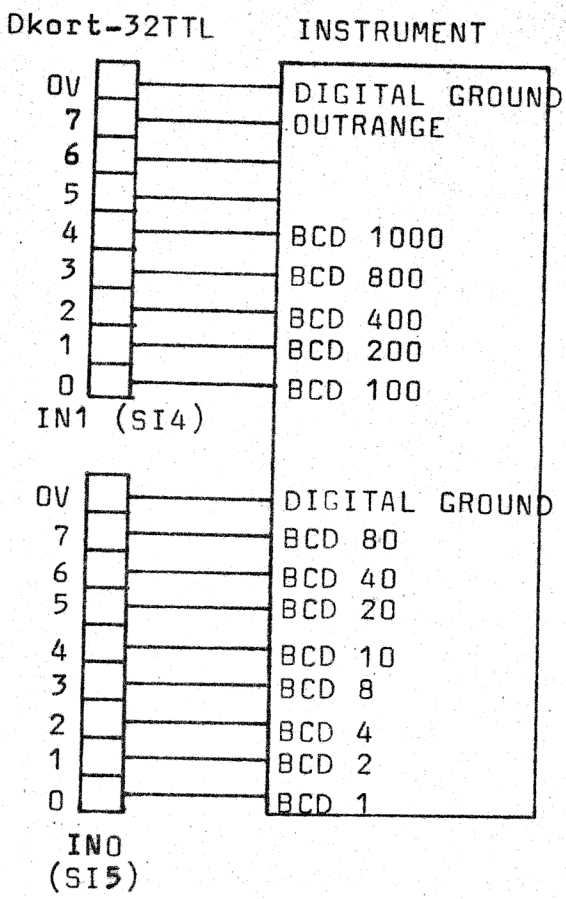
Avrundat uppåt är totala felet mindre än ±25 µs.

20. PROGRAM FÖR INSTRUMENT MED BCD-UTGÅNGAR

Kort: Digitalkort-32TTL.

Exempel på anslutning av digitalt panelinstrument med $3\frac{1}{2}$siffra (Weston 2462).

BCD-koden finns angiven på sid 9.10 i "Bruksanvisning och handledning, Mikronik Analogkort". Varje decimal siffra representeras med 4 bitar. Ett instrument med $3\frac{1}{2}$ siffra har 13 bitar för data ut. Av 16 ingångar på kortet finns 3 kvar för andra ändamål. I detta fall används en för OUTRANGE (överskridet mätområde). De två övriga kan användas för tecken eller område. Instrumentet är i detta fall kopplat för upprepade mätningar, 2-3 per sekund.



Instrumentet är kopplat för mätområdet 00,00V till 19,99V. Ett mycket enkelt program visas nedan. Eftersom två bytes läses in sker kontroll av att den första inte har ändrats då den andra lästs in (rad 130). Mätvärdet visas en gång per sekund på skärmen. Omräkningen sker på rad 150-170.

Exempel på förbättringar. Tidtagningen bör ske med realtidsklockan, se ref 1 och 5. Tid och mätvärde skrivs på skärm eller printer.

```
10 REM BCD  810213
20 REM MIKRONIK SUNDSVALL
30 REM SPÄNNINGAR 00.00 - 19.99 V MÄTS
50 OUT 1,19,0,0,2,0
100 REM ++++++++++++++++++++++++++++++++
110 X=INP(0) : REM IN0 TILL X
120 Y=INP(1) : REM IN1 TILL Y
130 IF X=INP(0) THEN 140 ELSE 110
140 IF (Y AND 128)=128 THEN 300
150 V1=(X AND 15)+(X AND 240)*10/16%
160 V2=(Y AND 15)*100+(Y AND 16)*1000/16%
170 U=(V1+V2)/100%
180 ; U
190 GOSUB 400
200 GOTO 110
290 REM ++++++++++++++++++++++++++++++++
300 ; "ÖVERSKRIDET MÄTOMRÅDE"
310 GOSUB 400
320 GOTO 110
390 REM VÄNTA 1 SEK ++++++++++++++++++++
400 FOR N=0 TO 1000 : NEXT N
410 RETURN
```

21. PROGRAM HEXDEC

Program HEXDEC är avsett för handassemblerade assemblerprogram. Med program HEXDEC kan man

1. skriva in hexadecimala objektkoder i minnet,
2. läsa i minnet (listning), varvid både adresser och data skrivs på skärmen i decimal och hexadecimal form,
3. köra ett assemblerprogram.

Kortadressen finns på rad 605. Felhantering ingår.

```
 10 REM HEXDEC  810212
 15 REM MIKRONIK SUNDSVALL
 20 ; CHR$(12) : ONERRORGOTO 110
 30 ; "MED DETTA PROGRAM KAN VI LÄSA IN"
 40 ; "HANDASSEMBLERADE SMÅPROGRAM FÖR"
 50 ; "Z80, 8080/8085 I HEXADECIMAL KOD"
 60 ; : ; : ; "VÄLJ MELLAN" : ;
 70 ; "1  INMATNING I HEXADECIMAL KOD" : ;
 80 ; "2  LISTNING" : ;
 90 ; "3  KÖRNING AV ASSEMBLERPROGRAM" : ;
 95 ; "4  SLUT" : ; : ;
100 INPUT I$ : I=VAL(I$) : GOTO 113
110 ; CHR$(12) : ; CHR$(7) : GOTO 20
113 IF I<1 OR I>4 THEN 20 : IF I=4 THEN 700
115 ONERRORGOTO 115
120 ; "GE STARTADRESS I DECKOD "; : INPUT A$ : A=VAL(A$)
130 S$="0123456789ABCDEF"
140 ON I GOTO 150,380,600,700
145 REM ++++++++++++++++++++++++++++++++
150 REM **********INMATNING**********
155 ; CHR$(12)
158 ; "INMATNING" : ; : ; : ; : ; : ;
160 ; "OM FÖREGÅENDE KOD BLEV FEL,"
170 ; "SKRIV 'FEL' " : ;
180 ; " NÄR PROGRAMMET ÄR SLUT,"
190 ; "SKRIV 'SLUT' " : ;
200 GOSUB 500
230 ; CUR(23%,0%);"HEXADRESS=";Y$+X$;"    GE HEXKOD  ";
240 INPUT H$
250 IF H$="FEL" THEN A=A-1 : GOTO 200
260 IF H$="SLUT" THEN 20
270 IF LEN(H$)<>2% THEN 280 ELSE 290
280 ; " FEL KOD" : GOTO 200
290 D1%=INSTR(1%,S$,LEFT$(H$,1%))
300 IF D1%=0% THEN 280
310 D0%=INSTR(1%,S$,RIGHT$(H$,2%))
320 IF D0%=0% THEN 280
330 D%=(D1%-1%)*16%+D0%-1%
340 POKE A,D%
350 A=A+1% : A1=A1+1%
360 GOTO 200
365 REM ++++++++++++++++++++++++++++++++
```

Fortsättning nästa sida.

```
370 REM *********LISTNING*************
380 ; CHR$(12)
382 ; "LISTNING" : ; : ; : ; : ; : ; : ; : ;
384 ; "TRYCK PÅ TANGENT FÖR NÄSTA ADRESS"
386 ; "TRYCK PÅ 'Å' FÖR ÅTER " : ; : ; : ;
390 ; "DECIMAL";TAB(15%);"HEXADECIMAL"
400 ; "ADRESS DATA    ADRESS DATA"
410 GOSUB 450
420 GET W$ : IF W$="Å" THEN 20 ELSE 430
430 A=A+1%
440 GOTO 410
445 REM ********************************
450 D%=PEEK(A)
460 GOSUB 500 : A$=Y$+X$
470 H%=PEEK(A) : GOSUB 560
480 ; A;TAB(8%);H%;TAB(15%);A$;TAB(23%);X$
490 RETURN
495 REM ADRESS TILL HEXADECIMAL KOD****
500 A1%=A/256% : REM HI BYTE
510 H%=A1% : GOSUB 560 : REM H% TILL HEXADECIMAL KOD
520 Y$=X$
530 A2%=A-A1%*256 : REM LO BYTE
540 H%=A2% : GOSUB 560
550 RETURN

554 REM ********************************
555 REM OMVANDLING TILL HEXADECIMAL KOD
560 X1%=H%/16%
570 X2%=H%-X1%*16%
580 X$=MID$(S$,X1%+1%,1%)+MID$(S$,X2%+1%,1%)
590 RETURN
595 REM ++++++++++++++++++++++++++++++++
600 REM **********KÖRNING**********
605 OUT 1,19,0,0,2,0
610 X=CALL(A)
620 GOTO 610
700 END
```

MIKRONIK DIGITALKORT

APPENDIX A

LITTERATURREFERENSER

1. Bruksanvisning ABC80. Luxor-Scandia Metric.
2. Bruksanvisning ABC80 assembler. Luxor.
3. Mikrodatorns ABC av Markesjö. Esselte Studium.
4. ABC om BASIC av Andersson, Kullbjer, Lundgren, Thornell. Didact.
5. Avancerad programmering på ABC80 av Isaksson, Kärrsgård. Scandia Metric-Studentlitteratur.
6. Z80-CPU Technical Manual. Scandia Metric.
7. Z80-PIO Technical Manual. Scandia Metric.
8. Z80-Assembly Language Programming Manual. Luxor-Scandia Metric.
9. Grundläggande Mikrodatorteknik av Hemert. Studentlitteratur.
10. Grundläggande Mikrodatorteknik, Arbetsbok av Hemert. Studentlitteratur.
11. Mikrodatorn av Bergström. Computer press.
12. Handbok för mikrodatorn av Bergström. Computer press.
13. ABC om mätdatorsystem av Eriksson, Magnusson, Rasmusson, Björklöf. Scandia Metric.
14. Z80 Instruction Handbook av Wadsworth. Scelbi Publications.
15. Z80 programming for logic design av Osborne m. fl.. Osborne & Associates.
16. Programming the Z80 av Zaks. Sybex.

Ref 9-12 behandlar Intel 8080.