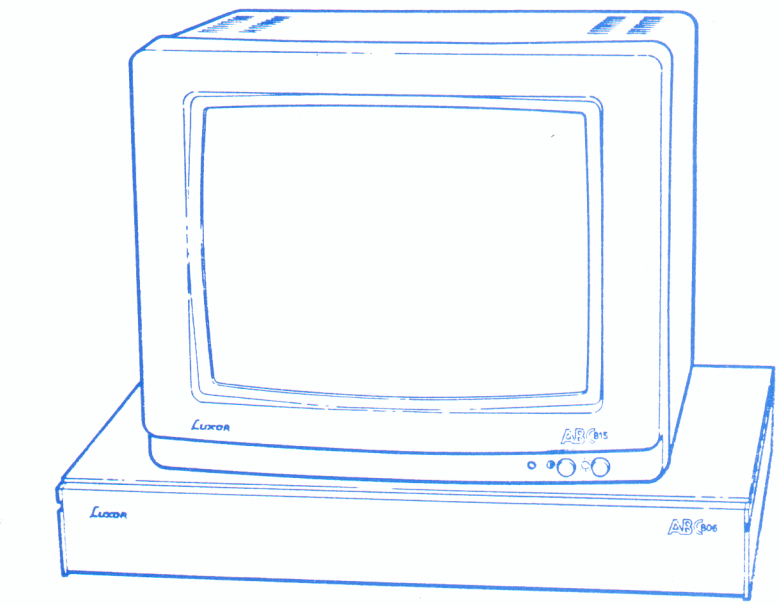
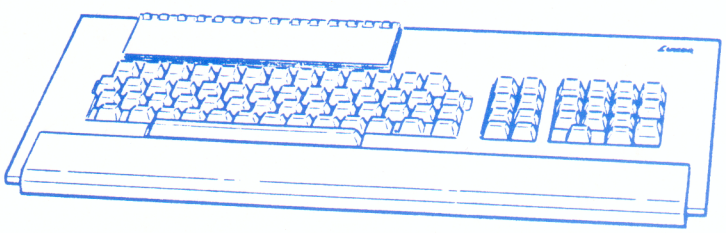
LUXOR
Datorer

Dator ABC 806 230 8089-15

Servicemanual





INNEHÅLL

1 INLEDNING

ABC 806 är en vidareutveckling av datorn ABC 800. Det är framförallt möjligheterna att återge och lagra högupplösningsbilder som utökats genom komplettering med ett grafiskt minne på 128 kbyte. 128 kbyte grafikminnet kan också användas för lagring av data.

1.1 Ingående enheter

ABC 806 kan byggas upp på flera olika sätt. Dels kan man använda en färgbildskärm ABC 812, dels en monokrom bildskärm ABC 815. Med monokrom menas en färg - i det här fallet amber (orange) mot mörk bakgrund.

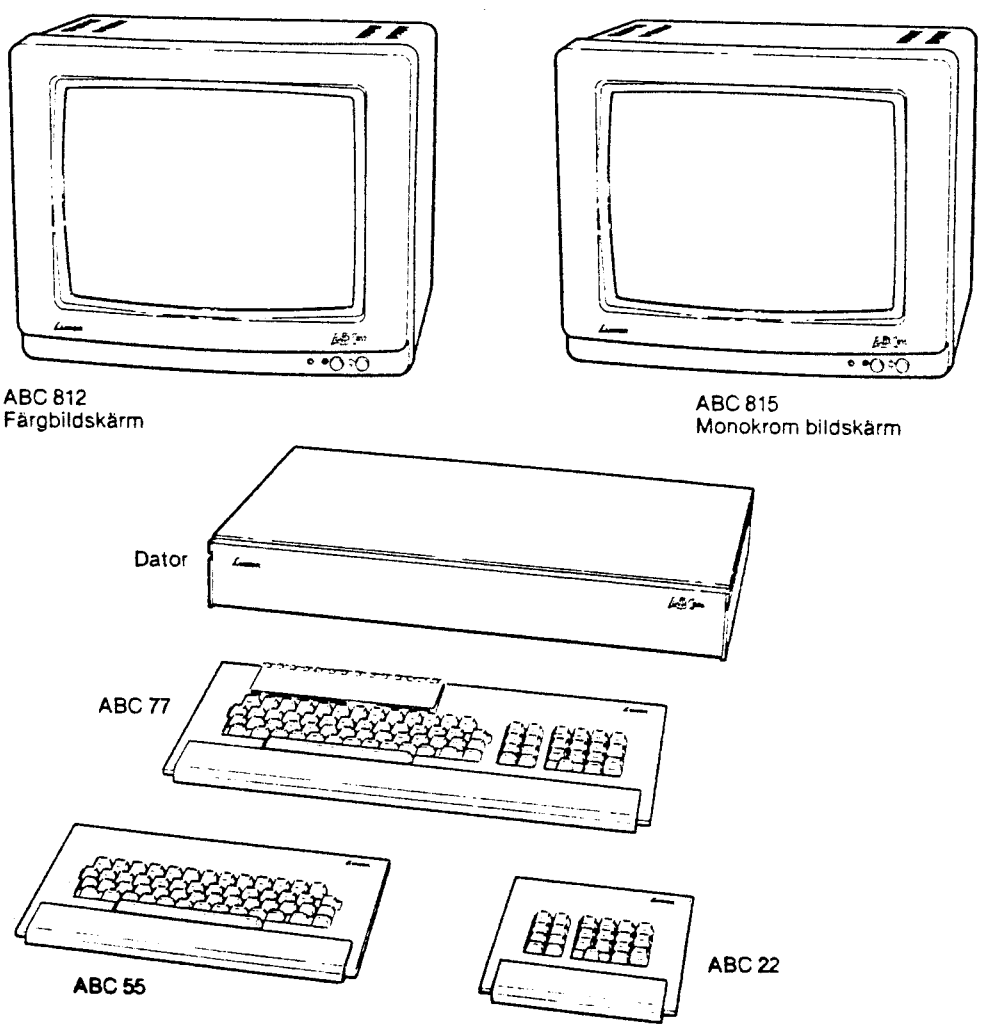


fig 1.

Till ABC 806 kan man använda två typer av tangentbord ABC 77 eller ABC 55. Det som skiljer de båda tangentborden åt är att ABC 77 har en funktions och numerisk tangentuppsätning.

Vid behov kan dock ABC 55 kompletteras med ABC 22 som är ett separat tangentbord med enbart funktions och numeriska tangenter. Tangentborden är uppbyggda enligt svensk standard.

Datorn innehåller systemprogram för högnivåspråket BASIC. Detta innebär att ABC 806 är klar att användas för programmering direkt efter spänningstillslag.

2 TEKNISKA DATA

Dimensioner	500 × 350 × 70 (b × d × h)
Vikt	5 kg
Reglage	RESET knapp
Effektanslutning	Kraftförsörjning över DC/DC-omvandlare via ansluten bildskärm. In: 17 V, 2.2 A till 24 V, 1.5 A. Ut: 5 V, 4 A + 12 V, 0.8 A —12 V, 0.3 A
Datoruppbyggnad	CPU CTC DART SIO/2 PROM (systemprogramminne) RAM (användarminne) Teckengenerator Teckenminne Attributminne Attributhanterare Grafikminne Kommunikationsanslutning, 2 st ABC-bussanslutning Bildskärmsanslutning Tangentbordsanslutning
Minne	PROM: 24 kbyte BASIC-interpretator 4 kbyte DOS (diskoperativsystem) 4 kbyte printer- och terminalrutin, samt rutin för HR-grafik RAM: 32 kbyte användarminne 128 kbyte grafik- och datalagringsminne (RAM-floppy), se grafikminne 2 kbyte teckenminne 2 kbyte attributminne Kan ställas om till RAM-laddad dator för t ex andra operativsystem som CP/M

Centralenhet	Mikroprocessor Z80A Ordlängd 8 bitar Klockfrekvens 3 MHz
Programspråk	BASIC II, se separat bruksanvisning
In- och utmatning	
Kommunikations-anslutning CH.A	Asynkron överföring. Programmerbar hastighet 75—19200 baud. Avsedd för skrivare
Kommunikations-anslutning CH.B	Synkron/asynkron sändning och/eller mottagning. Programmerbar hastighet 75—19200 baud. Möjlighet till olika sändnings- och mottagningshastighet. Avsedd för kommunikation. Olika typer av datorkommunikation väljs med en omkopplare. NRZI-modul som option.
ABC-buss	För anslutning av flexskiveenhet, expansionskort, expansionsenhet m fl
ABC NET	Maskinvaran för ABC NET ingår som standard
Kontaktdon	Bildskärm, 15-polig D-sub Tangentbord, 7-polig DIN Kommunikationsanslutning CH.A och CH.B, 9-polig D-sub ABC-buss, 64-polig Europakontakt
Driftsäkerhet	MTBF >10 000 timmar
Teckengenerator	
Bildformat	24 rader med 80 alternativt 40 tecken 1 systemrad med 80 alt 40 tecken
Teckenformat	Teckencell 10 × 6 punkter Teckenmatris, 9 × 5 punkter
Tecken	96 stycken enligt SS 636127
Teckengrafik	64 tecken enligt TELEDATA-modell
Markör	Blinkande "underline", fast vid skrivning eller efter ca fem sekunder om markören inte flyttas
Scrolling	Automatisk rullning uppåt av sidan. Möjlighet till soft-scroll
Teckenminne	2 kbyte
Attributminne	2 kbyte

Grafikminne	128 kbyte som kan användas enligt följande alternativ: 4 olika färgbilder 2 färgbilder och 66 kbyte data 1 färgbild och 98 kbyte data 128 kbyte data När antalet bilder understiger fyra kan datorn använda minnet för datalagring — RAM-floppy. Minnet kan också användas för laddning av andra operativsystem
Punktgrafik	240 x 240 punkter i fyra färger, kompatibelt med ABC 800 512 x 240 punkter i fyra individuellt valda färger 256 x 240 punkter i åtta individuellt valda färger Följande färger används: rött, grönt, blått, gult, cyan, magenta, vitt och svart
Kalender	Fast inbyggd CMOS kalender med klocka. Kalendern drivs med ett batteri med minst fem års gångtid
Miljökrav	
Drifttemperatur	+5 till +35°C
Lagringstemperatur	—40 till +55°C
Luftfuktighet	10 till 90% RH, icke kondenserande
Säkerhet	IEC, SEMKO, DEMKO
Störsäkerhet	VDE 0871 B

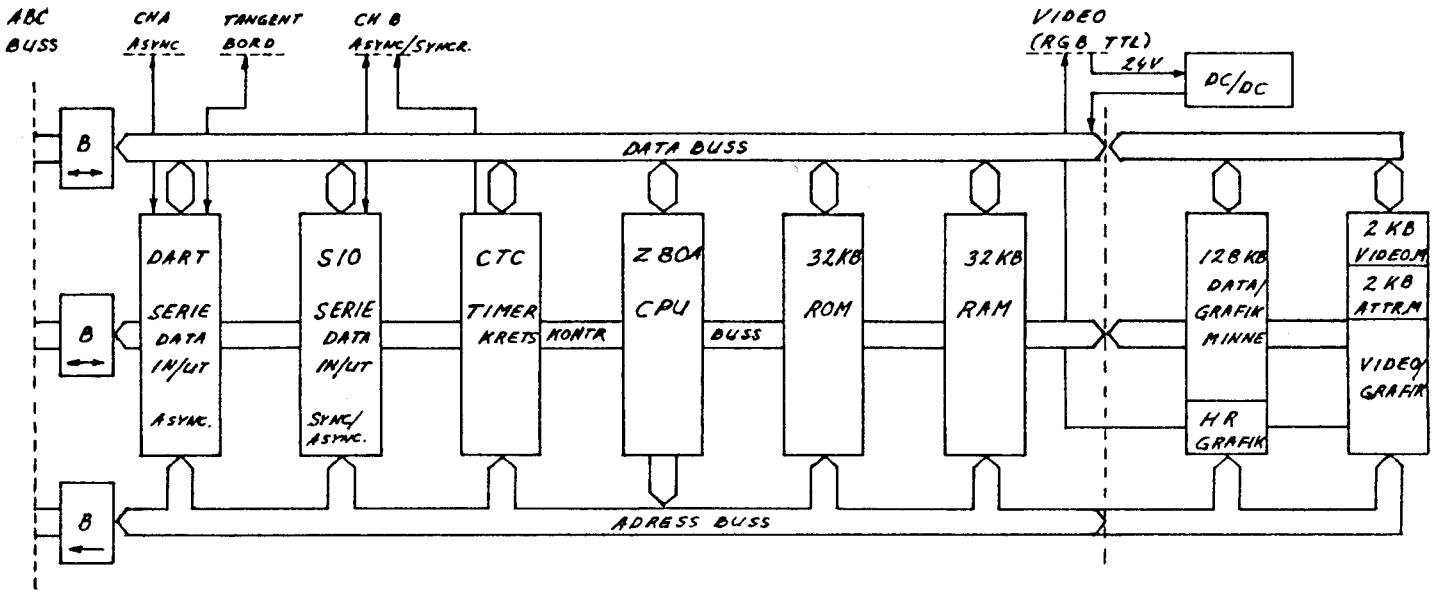
3 TEKNISK BESKRIVNING, ALLMÄN DEL (DATOR) OCH BLOCKSCHEMA

Datorn som är en separat enhet innehåller ett antal olika kretskort. Efter det att man tagit bort fem skruvar på undersidan av lådans framkant kan överdelen lyftas bort så att korten blir tillgängliga.

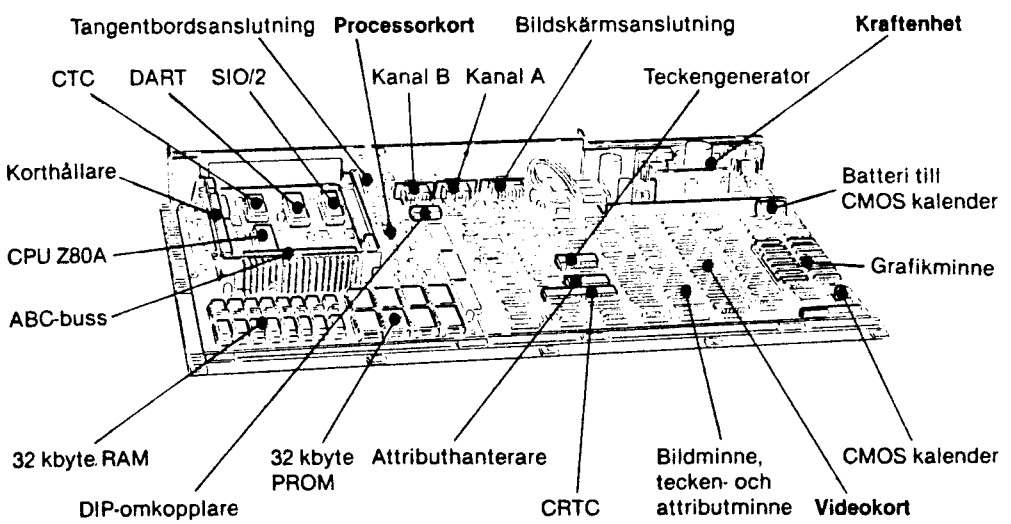
3.1 Uppbyggnad

Datorn innehåller följande enheter:

* PU-kort (Processor Unit), som består av
  - Mikroprocessorn Z80A
  - 32 kbyte PROM med systemprogram
  - 32 kbyte RAM som primärminne och arbetsminne
  - Seriekommunikationskretsar SIO, DART och CTC
  - Omkopplare för olika typer av kommunikation (serie)
  - Anslutningar för seriell kommunikation (3 st)
  - Busskontakt 64 polig - för anslutning av I/O och minneskort
* VU-kort (Video Unit), som består av
  - CRTC (Catode Ray Tube Controller) 6845 för generering av videosignal
  - Teckenminne 2 kbyte RAM
  - Attributminne 2 kbyte RAM. Detta minne innehåller styrkoder för respektive tecken i teckenminnet
  - Grafikminne 128 kbyte RAM för HR-grafik (High Resolution). Kan också användas för datalagring (RAM-floppy)
  - CMOS kalender, batteridriven klockkrets som innehåller data för ÅR,MÅN,VECKA,DAG,TIM,MIN,och SEK.
* Kraftenhet
  - En DC/DC omvandlare som förser datorn med erforderliga spänningar. En ostabiliserad 24 V likspänning tas från bildskärmen och omvandlas till de stabiliserade spänningarna +5 V, +12 V samt -12 V. Dessa spänningar är också tillgängliga i den 64 poliga busskontakten



Blockschema
fig 2.



Dator
fig 3.

3.2 Minnesdisposition

Mikroprocessorn Z80A kan direktadressera 64 kbyte minne. Hur minnesdispositionen ser ut visas i figur 4.

32 kbyte av primärminnet utgörs av användarminne (RAM) som används för lagring av applikationsprogram och data. En del av detta minne reserveras av systemprogrammet för lagring av systemvariabler,samt ytterligare en del för mellanlagring av data vid kommunikation med perferienheter.

Resten av primärminnet utgörs av 32 kbyte systemprogramm som är lagrat i åtta stycken PROM - kretsar. Detta program består av:

* 24 kbyte BASIC - interpretator, d v s. tolkar BASIC instruktioner

* 4 kbyte DOS (Disk Operating System) för styrning av flexskiveen-heten

* 4 kbyte Option Prom med drivrutin för de seriella portarna samt en del rutiner för hantering av högupplösningsgrafik

* 2 kbyte bildminne (RAM) som ligger parallelt med 2 kbyte av options prommet.

* 2 kbyte attribute minne som ligger parallelt med bildminnet.

* 2*64 kbyte data eller grafikminne som kan kopplas in på olika sätt över primärminnet.

Systemprogrammet som är lagrat i PROM kretsar sitter monterade i IC-hållare på PU-kortet enligt fig 5

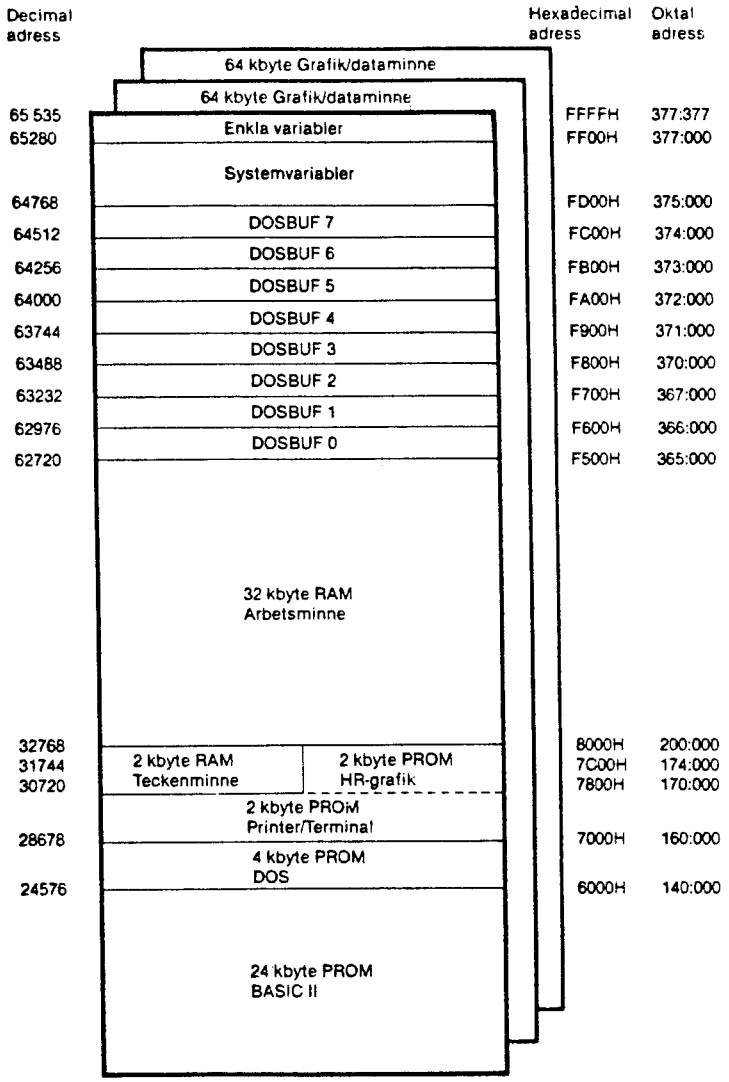
De olika minneskretsarna innehåller följande delar av systemprogrammet (X betecknar variant)

ABC06-X till ACB56-X innehåller BASIC II interpretator, drivrutiner samt initieringsrutiner för datorn 24 kbyte

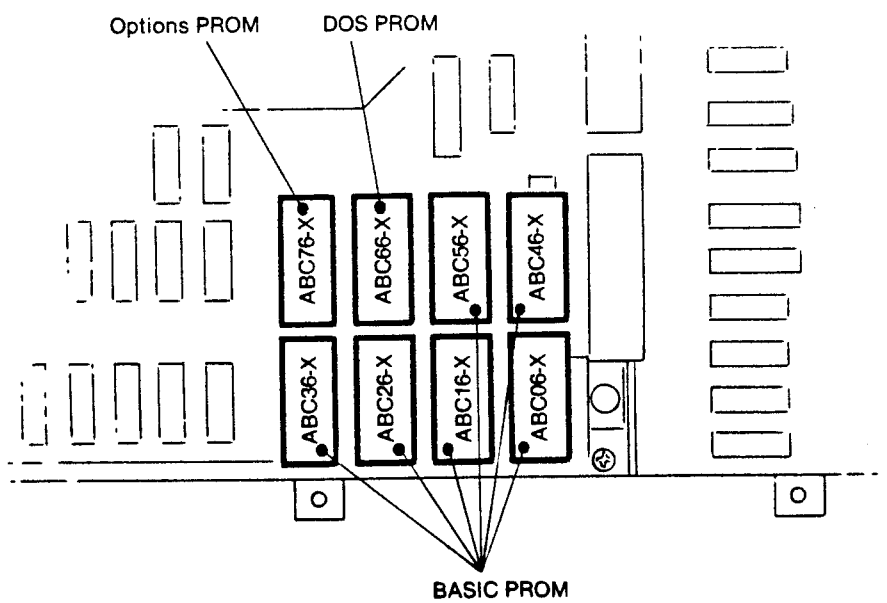
ABC66-X
innehåller DOS, för hantering av flexskiveenheter, 4 kbyte.

ABC76-X
är det sk. options - PROM:et som innehåller rutiner för seriell in/ut matning av data samt hantering av HR-grafik

Minneskarta med flexskiveenhet ansluten



Minneskarta
fig 4.



PROM placering
fig 5.

4 ANSLUTNINGSDON

Alla anslutningsdon finns samlade på baksidan av datorn.

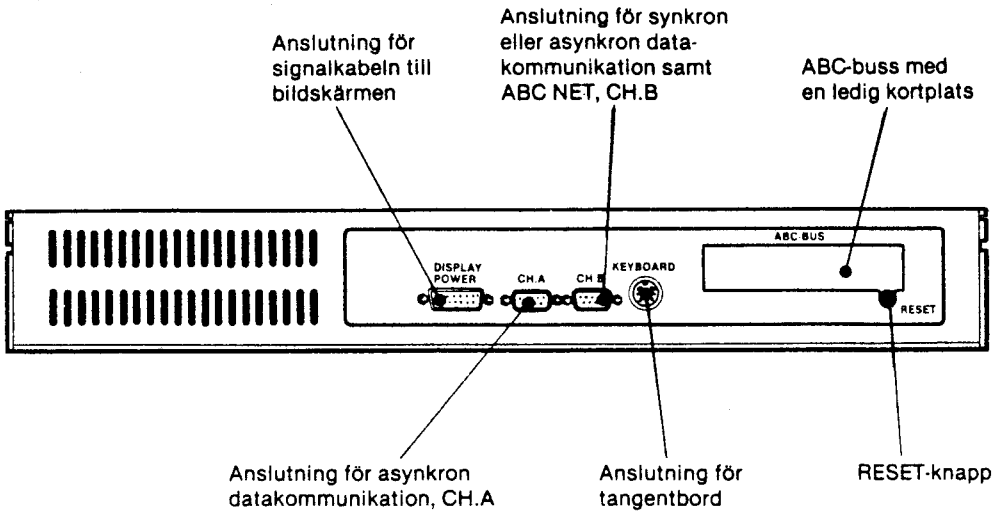
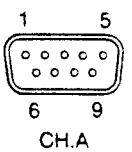


fig 6.

CH. A (Kanal A) är en seriell asynkron kommunikation-kanal. Kan hantera både in/ut data. Hastigheten på överföringen av data är samma både för sändning och mottagning. Vanligtvis används den för skrivar anslutning.

- Signaler till och från CH.A

1	DTR	(Data Terminal Ready)
2	TxD	(Transmit Data)
3	RxD	(Receive Data)
4	RTS	(Request to Send)
5	CTS	(Clear to Send)
6	DSR	(+ 12 V)
7	GND	(Ground)
8	DCD	(Data Carrier Detect)
9	—12 V	



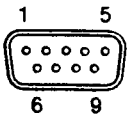
Signaler till och från CH. A
fig 7.

CH. B (Kanal B) är en seriell asynkron/syncron kommunikationskanal. Kanalen används även för anslutning till ABC NET fleranvändar system. Hur kanalen ska fungera ställs in med omkopplare som finns på PU - kortet.

Vilket läge omkopplaren ska stå i samt vad de olika brytarna har för funktion visas i följande tabellern och figurer.
Single speed (sändning och mottagning med samma hastighet)
Split speed (-"- -"- med olika hastighet)

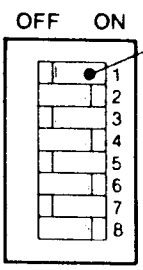
- Signaler till och från CH.B

1 DTR (Data Terminal Ready)
2 TxD (Transmit Data)
3 RxD (Receive Data)
4 RTS (Request to Send)
5 CTS (Clear to Send)
6 • ABC NET TxD • +12 V • TxC
7 GND (Ground)
8 DCD (Data Carrier Detect)
9 • —12 V • RxC

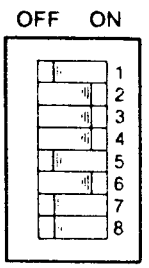


CH.B

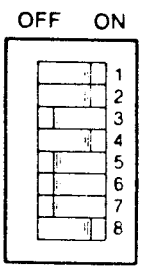
Signaler till och från CH.B
fig 8.



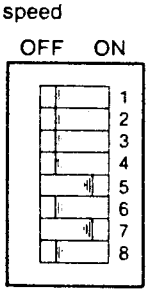
Omkopplarens inställning vid leverans.
Asynkron kommunikation, single speed



Asynkron kommunikation, split speed



ABC NET



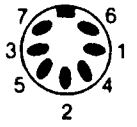
Synkron kommunikation

Omkpplarens inställning
fig 9.

Knapp	Funktion i läge ON	Stift i anslutning
1	ABC NET TxD	6, option 1
2	—12 V	9, option 2
3	CTC 0 till TxC	
4	CTC 1 till RxC	
5	RxC	9, option 2
6	+12 V	6, option 1
7	TxC	6, option 1
8	CTC 1 till TxC	

Knappfunktioner i omkopplaren
fig 10.

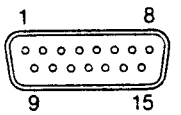
1	TxD	(Transmit Data)
2	GND	(Ground)
3	RxD	(Receive Data)
4	TRxC	(Clock)
5	Key down	
6	+ 12 V	
7	Reset	



KEYBOARD

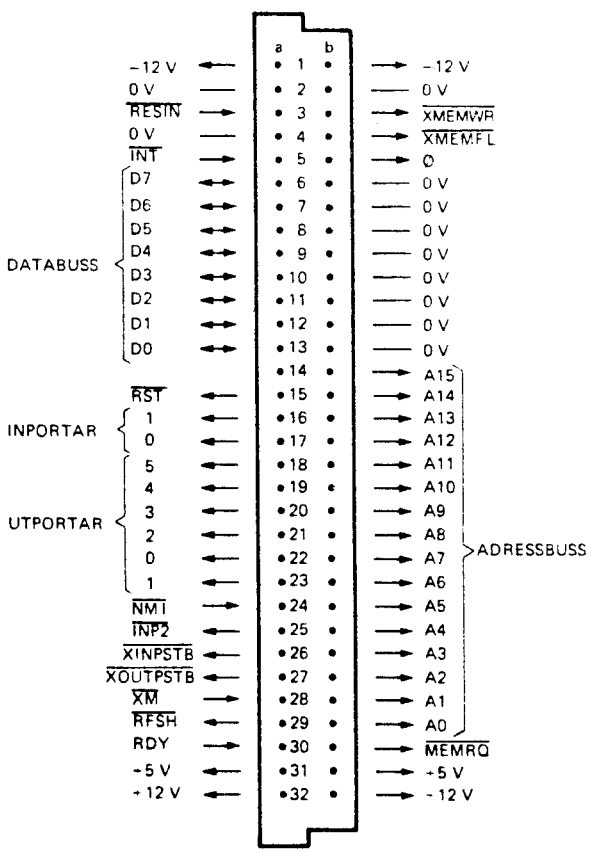
Signaler till och från tangentbordsanslutning
fig 11.

1	Matningsspänning (+ 17— + 24 V)
2	Kraftjord
3	
4	
5	Video
6	
7	Signaljord
8	SYNK (H + V)
9	B-signal (blå)
10	G-signal (grön)
11	R-signal (röd)
12	
13	
14	LF
15	



DISPLAY
POWER

Signaler till/från Display anslutningen
fig 12.



64 polig kontakt (ABC bussen)
fig 13.

5 RESERVDELSLISTA OCH KOMPONENTFÖRTECKNING

Reservdelslista Dator ABC 806, 230 8089-15

Kåpa kpl	40 08089-02	
Fot 11x5,5	40 30029-01	
*(Bildskärmskabel, kort	43 71694-01)	
Bildskärmskabel, lång	43 71691-01	Rekommenderas vid efterbeställning
(Signalkabel bildskärm-dator	43 71695-01)	
*Signalkabel bildskärm-dator	43 71696-01	Rekommenderas vid efterbeställning
Jordkabel	43 71699-01	
Kontaktplåt	44 20767-01	
Fästvinkel	44 20776-01	
*DC/DC modul kpl	44 30135-04	
Botten	44 70029-02	
Fäste för kretskort	53 30303-01	
*Processor-kort	55 20942-01	
*Videokort	55 20943-01	
ABC T06-1 Sv/Fi	64 90243-01	

*Komponentlista på följande sidor

Bildskärmskabel (kort), 43 71694-01

Kabel ämne	90 20024-04
Kontaktdon 14P - D-sub	43 60365-01
Kåpa 14P	43 90315-01

Signalkabel bildskärm-dator, 43 71695-01

Kabel 15-pol skärm	90 20042-01
DIN-kontakt 7-pol	43 60471-01

DC/DC modul kpl, 44 30135-04

Lådprofil	44 20529-01
DC/DC modul	55 20807-03

DC/DC modul, 55 20807-03

R01	100R 5% 0.3W	61 29250-01
R02	270R 5% 0.3W	61 29253-01
R03	2.2R 5% 0.3W	61 00071-01
R04	1.2k 5% 0.3W	61 29290-01
R05	4.7k 5% 0.3W	61 29263-01
R06	100R 5% 0.3W	61 29250-01
R07	680R 5% 0.5W	61 29489-01
R08	270R 5% 0.3W	61 29253-01
R09	2.2R 5% 0.3W	61 00071-01
R10	4.7R 5% 0.3W	61 00101-01
P1	Trimpot 500R	61 17948-01
C01	1000uF 35V	62 50132-01
C02	1000uF	62 50086-01
C03	2.2uF 63V	62 50108-01
C05	470uF 25V	62 50084-01
C08	1000uF 35V	62 50132-01
C09	470uF 25V	62 50084-01
C10	2.2uF 63V	62 50108-01
C12	560pF 10% 63V	62 00099-01
C13	47pF 2%	62 21245-01
C15	150nF 10% 100V	62 21347-01
C17	2200uF 40V	63 50128-01
D1	RGP 10 D	63 40058-01
D2	RGP 30 D	63 40074-01
D3	RGP 30 D	63 40074-01
D5	RGP 30 D	63 40074-01
D6	1N4148	63 08824-01
D4	BZX61C 12V	63 08748-01

T1	BU 406	63	20025-03
T2	BU 406	63	20025-03
T3	BC 636	63	10032-01
T4	BC 636	63	10032-01
I1	TL497AIN	64	90086-02
I2	STAB -12V 79M12	64	50022-01
I3	STAB 12V 7812	64	50019-01
L1	Drossel 230uH	59	00391-01
L2	Drossel 230uH	59	00391-01
L3	Trafo	58	10088-01
L4	Trafo	58	10088-01
K1	Kontakt 2-pol	43	60371-01
K2	Kontakt 4-pol	43	60361-01
	Kylplåt	44	10127-01

Processor-kort, 55 20942-01

R01	100R 5% 0.3W	61	29250-01
R02	1k 5% 0.3W	61	29258-01
R03	1k 5% 0.3W	61	29258-01
R04	Motst.nät 8x1k	61	90051-01
R05	Motst. nät 8x10k	61	90021-01
R08	1k 5% 0.3W	61	29258-01
R09	1k 5% 0.3W	61	29258-01
R10	1k 5% 0.3W	61	29258-01
R11	4.7k 5% 0.3W	61	29263-01
R12	4.7k 5% 0.3W	61	29263-01
R13	2k 1% 0.3W	61	40067-01
R14	1.5k 5% 0.3W	61	29259-01
R15	Motst.nät 7x1k	61	90036-01
R16	5.6k 5% 0.3W	61	29291-01
R17	22k 5% 0.3W	61	29268-01
R18	680R 5% 0.3W	61	29357-01
R19	22k 5% 0.3W	61	29268-01
R20	1k 5% 0.3W	61	29258-01
R21	220R 5% 0.3W	61	29252-01
R22	Motst.nät 7x1k	61	90036-01
R26	330R 5% 0.3W	61	29254-01
R27	1k 5% 0.3W	61	29258-01
R28	1k 5% 0.3W	61	29258-01
R29	10k 5% 0.3W	61	29265-01
R30	150R 5% 0.3W	61	29251-01
R31	Motst.nät 7x4.7k	61	90032-01
R32	220R 5% 0.3W	61	29252-01
R33	100R 5% 0.3W	61	29250-01
R35	1k 5% 0.3W	61	29258-01
R36	4.7k 5% 0.3W	61	29263-01
R37	10k 5% 0.3W	61	29265-01
R38	12R 5% 0.3W	61	29442-01
R39	470R 5% 0.3W	61	29256-01
R40	470R 5% 0.3W	61	29256-01
R41	4.7k 5% 0.3W	61	29263-01
R42	1k 5% 0.3W	61	29258-01

R43	1k 5% 0.3W	61 29258-01
R44	47R 5% 0.3W	61 29249-01
R45	1k 5% 0.3W	61 29258-01
R46	1k 5% 0.3W	61 29258-01
R47	47R 5% 0.2W	61 00129-01
R48	47R 5% 0.2W	61 00129-01
R49	47R 5% 0.2W	61 00129-01
R50	10k 5% 0.3W	61 29265-01
R51	47R 5% 0.2W	61 00129-01
R53	82R 5% 0.2W	61 00058-01
R54	82R 5% 0.2W	61 00058-01
R55	82R 5% 0.2W	61 00058-01
R56	82R 5% 0.2W	61 00058-01
R57	82R 5% 0.2W	61 00058-01
R63	270R 5% 0.3W	61 29253-01
R64	1k 5% 0.3W	61 29258-01
C01	10nF ker	62 21342-01
C02	100nF ker	62 00039-01
C03	100nF ker	62 00039-01
C04	100nF ker	62 00039-01
C05	100nF ker	62 00039-01
C06	100nF ker	62 00039-01
C07	100nF ker	62 00039-01
C08	100nF ker	62 00039-01
C09	100nF ker	62 00039-01
C10	100nF ker	62 00039-01
C11	100nF ker	62 00039-01
C12	100nF ker	62 00039-01
C13	100nF ker	62 00039-01
C14	100nF ker	62 00039-01
C15	100nF ker	62 00039-01
C16	100nF ker	62 00039-01
C17	100nF ker	62 00039-01
C18	10nF ker	62 21342-01
C19	10nF ker	62 21342-01
C20	10nF ker	62 21342-01
C21	47uF 16V tant	62 50151-01
C22	47uF 16V tant	62 50151-01
C23	220pF ker	62 00123-01
C24	10uF 25V tant	62 50021-01
C25	150uF 6.3V tant	62 50152-01
C26	100uF 16V tant	62 50150-01
C27	1uF 35V tant	62 50130-01
C28	1uF 35V tant	62 50130-01
C29	10nF ker	62 21342-01
C30	10uF 25V tant	62 50021-01
C31	10nF ker	62 21342-01
C32	10nF ker	62 21342-01
C36	10nF ker	62 21342-01
C37	10nF ker	62 21342-01
C39	10nF ker	62 21342-01
C40	10nF ker	62 21342-01
C41	10nF ker	62 21342-01
C42	10nF ker	62 21342-01
C43	10nF ker	62 21342-01
C44	470pF 20% ker	62 00055-01
C45	10nF ker	62 21342-01

C46	10nF ker	62 21342-01
C48	47uF 16V tant	62 50151-01
C49	47uF 16V tant	62 50151-01
C50	10nF ker	62 21342-01
C54	10nF ker	62 21342-01
C55	47uF 16V tant	62 50151-01
C56	47uF 16V tant	62 50151-01
C57	47uF 16V tant	62 50151-01
C58	10uF 35V E-lyt	62 50127-01
C59	10nF ker	62 21342-01
C60	10nF ker	62 21342-01
C61	10nF ker	62 21342-01
C62	470pF 20% ker	62 00055-01
C63	10nF ker	62 21342-01
C64	10nF ker	62 21342-01
C65	470pF 20% ker	62 00055-01
C66	47uF 16V tant	62 50151-01
C67	10uF 25V tant	62 50021-01
C68	10uF 25V tant	62 50021-01
C69	2.2uF 63V E-lyt	62 50108-01
C70	10nF 100V	62 20182-01
C71	1nF ker	62 21339-01
C78	220pF ker	62 00123-01
C79	220pF ker	62 00123-01
C80	220pF ker	62 00123-01
C81	220pF ker	62 00123-01
C82	1nF ker	62 21339-01
C83	1nF ker	62 21339-01
C85	1nF ker	62 21339-01
C86	10nF ker	62 21342-01
C87	470pF 20% ker	62 00055-01
C88	470pF 20% ker	62 00055-01
L01	Drossel 47uH	59 00229-01
L02	Ferritpärla	59 56759-01
L04	Ferritpärla	59 56759-01
L05	Ferritpärla	59 56759-01
L06	Ferritpärla	59 56759-01
L07	Ferritpärla	59 56759-01
L08	Ferritpärla	59 56759-01
L14	Ferritpärla	59 56759-01
L15	Ferritpärla	59 56759-01
L16	Ferritpärla	59 56759-01
L17	Ferritpärla	59 56759-01
D01	1N4148	63 08824-01
D02	1N4148	63 08824-01
D07	1N4148	63 08824-01
D08	1N4148	63 08824-01
D03	BZX83C 3.3V	63 08746-01
D05	BZX83C 3V	63 40139-01
D06	BZX83C 3V	63 40139-01
D09	BZX83C 8.2V	63 08732-01

T01	BC547B	63 10011-01
T02	BC547B	63 10011-01
T03	BC557B	63 10057-01
T04	BC557B	63 10057-01
1A	TMS4116-20NL	64 40028-01
1B	TMS4116-20NL	64 40028-01
1C	TMS4116-20NL	64 40028-01
1D	TMS4116-20NL	64 40028-01
1E	TMS4116-20NL	64 40028-01
1F	TMS4116-20NL	64 40028-01
1G	TMS4116-20NL	64 40028-01
1H	TMS4116-20NL	64 40028-01
2A	TMS4116-20NL	64 40028-01
2B	TMS4116-20NL	64 40028-01
2C	TMS4116-20NL	64 40028-01
2D	TMS4116-20NL	64 40028-01
2E	TMS4116-20NL	64 40028-01
2F	TMS4116-20NL	64 40028-01
2G	TMS4116-20NL	64 40028-01
2H	TMS4116-20NL	64 40028-01
3A	SN74LS258AN	64 40103-01
3C	SN7400	64 09307-01
3F	STAB -5V 79L05	64 50012-01
3G	SN74LS175	64 40043-01
3H	SN74LS258AN	64 40103-01
3L	SN74LS95	64 40113-01
3M	SN74LS158	64 40101-01
4C	SN74LS241	64 40044-01
4E	SN74LS241	64 40044-01
4G	SN74LS241	64 40044-01
4H	SN74LS245	64 40045-01
4M	SN74LS20	64 40097-01
4L	SN74LS138	64 40100-01
5A	SN74LS32	64 40032-01
5B	SN74ALS138	64 40052-01
5C	SN74ALS138	64 40052-01
5E	Z80A-CPU	64 90041-01
5H	LM339	64 90026-01
5K	SN7406	64 40000-01
5M	SN74ALS112	64 40099-01
6A	SN74LS107	64 40037-01
6B	SN74LS375	64 40050-01
6C	SN74LS139	64 40039-01
6G	SN74LS132	64 40038-01
6H	SN74LS32	64 40034-01
6J	SN74LS13	64 40096-01
6K	SN74LS51	64 40098-01
6L	SN74LS158	64 40101-01
6M	SN74LS241	64 40044-01
7A	SN74LS74	64 40035-01
7B	SN74LS191	64 40102-01
7C	SN74LS390	64 40105-01
7E	Z80A-CTC	64 90104-01
7J	Z80A-SIO/2	64 90102-01
7G	Z80A-SIO/0	64 90103-01
7K	SN74LS04	64 40031-01
7P	SN75189	64 90044-01

7R	SN75188	64 90043-01
7S	SN75189	64 90044-01
7T	SN75188	64 90043-01
1J	BASIC PROM ABC 36-11	64 90234-02
1K	BASIC PROM ABC 26-11	64 90233-02
1L	BASIC PROM ABC 16-11	64 90232-02
1M	BASIC PROM ABC 06-11	64 90231-02
2J	Options-PROM ABC 76-11	64 90238-02
2K	DOS PROM ABC 66-21	64 90314-01
2M	BASIC-PROM ABC 46-11	64 90235-02
2L	BASIC PROM ABC 56-11	64 90236-02
P01	DIN-kontakt 7-pol	43 60397-02
P08	Hylsdon 20-pol	43 60399-01
P09	Hylsdon 20-pol	43 60399-01
P10	Hylsdon 20-pol	43 60399-01
P11	Kontaktdon 64-pol, vinklad	43 60391-01
P12	Stiftdon 3-pol	43 60162-01
	Stiftlist 40-pol	43 60388-01
	D-sub 9-pol	43 60472-02
	D-sub 15-pol	43 60700-01
	Kontakthus 4P	43 60513-01
	Kontaktfjäder	43 90247-01
	Kortslutnings-bygel 2-pol	43 60389-01
	Kraftkabel 4-pol	43 71313-02
	Kraftkabel 2-pol, kpl	43 71418-01
S8	DIP-omk 8-pol	56 30115-01
	Tryckomkopplare	56 40200-01
	Styrskena	53 30483-01
5L	Balkong L5	55 21055-01

Balkong L5, 55 21055-01

R1	1k 5% 0.3W	61 29258-01
C1	10nF ker	62 21342-01
IC1	SN74LS375	64 40050-01
IC2	SN74ALS02	64 40155-01
	Stiftlist 40-pol	43 60388-01

Videokort, 55 20943-01

R01	470R 5% 0.3W	61 29256-01
R02	470R 5% 0.3W	61 29256-01
R03	1k 5% 0.3W	61 29258-01
R04	1k 5% 0.3W	61 29258-01
R05	10k 5% 0.3W	61 29265-01
R06	10k 5% 0.3W	61 29265-01
R07	10k 5% 0.3W	61 29265-01
R08	1k 5% 0.3W	61 29258-01
R09	1k 5% 0.3W	61 29258-01
R10	330R 5% 0.3W	61 29254-01
R11	330R 5% 0.3W	61 29254-01
R12	Motst-nät 9x1k	61 90040-01
R14	1k 5% 0.3W	61 29258-01
R15	1k 5% 0.3W	61 29258-01
R16	Motst-nät 9x1k	61 90040-01
R17	22k 5% 0.3W	61 29268-01
R18	1k 5% 0.3W	61 29258-01
R20	Motst-nät 9x10k	61 90039-01
R23	33R 5% 0.3W	61 29248-01
R24	33R 5% 0.3W	61 29248-01
R25	47R 5% 0.3W	61 29249-01
R26	Motst-nät 9x1k	61 90040-01
R27	47R 5% 0.3W	61 29249-01
R28	47R 5% 0.3W	61 29249-01
R29	Motst-nät 9x4.7k	61 90041-01
4J	DIL-motst 8x47R	61 90044-01
	4.7k 5% 0.3W	61 29263-01
	4.7k 5% 0.3W	61 29263-01
	4.7k 5% 0.3W	61 29263-01

C01	10nF ker	62 21342-01
C03	100nF ker	62 00039-01
C04	47pF	62 00036-01
C05	10nF ker	62 21342-01
C06	10nF ker	62 21342-01
C07	47uF 16V tant	62 50151-01
C08	10nF ker	62 21342-01
C09	47uF 16V tant	62 50151-01
C10	10nF ker	62 21342-01
C11	10nF ker	62 21342-01
C12	10nF ker	62 21342-01
C13	10nF ker	62 21342-01
C14	10nF ker	62 21342-01
C15	10nF ker	62 21342-01
C16	10nF ker	62 21342-01
C17	10nF ker	62 21342-01
C18	10nF ker	62 21342-01
C19	47uF 16V tant	62 50151-01
C20	10nF ker	62 21342-01
C21	10nF ker	62 21342-01
C22	10nF ker	62 21342-01
C23	10nF ker	62 21342-01
C24	10nF ker	62 21342-01
C25	10nF ker	62 21342-01
C26	10nF ker	62 21342-01
C27	10nF ker	62 21342-01

C28	10nF ker	62	21342-01
C29	10nF ker	62	21342-01
C30	10nF ker	62	21342-01
C34	10nF ker	62	21342-01
C35	10nF ker	62	21342-01
C36	10nF ker	62	21342-01
C37	10nF ker	62	21342-01
C38	10nF ker	62	21342-01
C39	10nF ker	62	21342-01
C40	10nF ker	62	21342-01
C41	10nF ker	62	21342-01
C42	10nF ker	62	21342-01
C44	10nF ker	62	21342-01
C45	10nF ker	62	21342-01
C46	10nF ker	62	21342-01
C47	10nF ker	62	21342-01
C48	10nF ker	62	21342-01
C49	10nF ker	62	21342-01
C50	10nF ker	62	21342-01
C51	10nF ker	62	21342-01
C55	10nF ker	62	21342-01
C56	10nF ker	62	21342-01
C57	100nF ker	62	00039-01
C58	100nF ker	62	00039-01
C59	100nF ker	62	00039-01
C60	100nF ker	62	00039-01
C61	100nF ker	62	00039-01
C62	100nF ker	62	00039-01
C63	100nF ker	62	00039-01
C64	100nF ker	62	00039-01
C65	10nF ker	62	21342-01
C67	10nF ker	62	21342-01
C68	Trimkond 2.5-27pF	62	80016-01
C69	47uF 16V tant	62	50151-01
C70	10nF ker	62	21342-01
C71	47uF 16V tant	62	50151-01
C72	47uF 16V tant	62	50151-01
D01	1N4148	63	08824-01
X01	Kristall 12MHz	63	90034-01
X02	Kristall 32.768	63	90053-01
1A	SN74LS240	64	40086-01
1C	SN74ALS158	64	40188-01
1D	SN74LS377	64	40122-01
1E	F 74 F 189	64	40200-01
1F	F 74 F 189	64	40200-01
1G	SN74ALS157	64	40187-01
2B	SN74LS378	64	40125-01
2C	SN74LS164	64	40121-01
2E	F 74 F 189	64	40200-01
2F	F 74 F 189	64	40200-01
2G	SN74ALS157	64	40187-01
2H	F 74 F 253	64	40201-01
2J	F 74 F 253	64	40201-01
3A	4020	64	40123-01
3B	SN74ALS163	64	40189-01

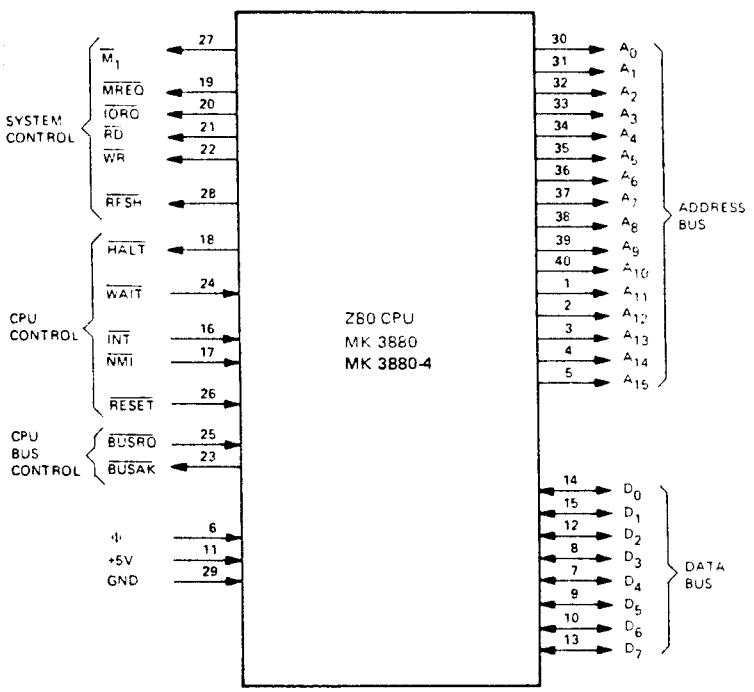
3C	SN74LS164	64	40121-01
3D	SN74ALS74	64	40157-01
3E	SN74ALS157	64	40187-01
3F	SN74ALS273	64	40191-01
3H	F 74 F 253	64	40201-01
3J	F 74 F 253	64	40201-01
4A	SN74ALS04	64	40184-01
4B	SN74LS107A	64	40037-01
4C	SN74LS164	64	40121-01
4D	SN74ALS109	64	40186-01
4F	SN74ALS1240	64	40205-01
4E	SN74LS125A	64	40083-01
4G	SN74LS393	64	40051-01
4H	F 74 F 253	64	40201-01
5A	SN74ALS175	64	40159-01
5B	SN74ALS00	64	40183-01
5C	SN74LS374	64	40117-01
5D	SN74ALS74	64	40157-01
5E	SN74LS107A	64	40037-01
5F	SN74ALS00	64	40183-01
5G	SN74LS393	64	40051-01
5H	HM 4864-3	64	90184-01
5J	HM 4864-3	64	90184-01
6A	SN74ALS74	64	40157-01
6B	4020	64	40123-01
6C	SN74LS377	64	40122-01
6D	SN74ALS163	64	40189-01
6F	SN74LS374	64	40117-01
6G	SN74LS166A	64	40042-01
6H	HM 4864-3	64	90184-01
6J	HM 4864-3	64	90184-01
7A	SN74ALS02	64	40155-01
7B	SN74LS155	64	40195-01
7D	F 74 F 273	64	40202-01
7F	SN74LS377	64	40122-01
7G	SN74LS166A	64	40042-01
7H	HM 4864-3	64	90184-01
7J	HM 4864-3	64	90184-01
8A	SN74ALS1020	64	40192-01
8B	SN74LS166A	64	40042-01
8D	F 74 F 273	64	40202-01
8E	4040	64	40131-01
8F	SN74LS373	64	40085-01
8G	SN74ALS00	64	40183-01
8H	HM 4864-3	64	90184-01
8J	HM 4864-3	64	90184-01
9A	SN74LS86	64	40036-01
9C	F 74 F 273	64	40202-01
9D	TC40H373	64	40152-01
9E	TC 5514 AP-2	64	90173-01
9F	SN74LS373	64	40085-01
9G	SN74LS166A	64	40042-01
9H	HM 4864-3	64	90184-01
9J	HM 4864-3	64	90184-01
10A	4557	64	40204-01
10B	SN 74LS379	64	40104-01
10D	TC40H374	64	40153-01
10E	TC 5514 AP-2	64	90173-01

10F	SN74LS377	64	40122-01	
10G	SN74LS166A	64	40042-01	
10H	HM 4864-3	64	90184-01	
10J	HM 4864-3	64	90184-01	
11A	SN74ALS1241	64	40193-01	
11B	F 74 F 02	64	40199-01	
11C	VIDEO ATTRIBUTE	64	90225-01	
11D	TC40H374	64	40153-01	
11E	TC 5514 AP-2	64	90173-01	
11F	TC40H245	64	40151-01	
11G	SN74LS125A	64	40083-01	
11H	HM 4864-3	64	90184-01	
11J	HM 4864-3	64	90184-01	
12A	TC40H373	64	40152-01	
12B	MC6845	64	90126-01	
12E	TC 5514 AP-2	64	90173-01	
12F	TC40H245	64	40151-01	
12H	HM 4864-3	64	90184-01	
12J	HM 4864-3	64	90184-01	
13A	SN74ALS00	64	40183-01	
13D	TC40H373	64	40152-01	
13E	TC 5514 AP-2	64	90173-01	
13F	SN74ALS1244	64	40194-01	
13G	SN74LS399	64	40197-01	
13J	E05-16	64	40198-01	
14A	SN74ALS175	64	40159-01	
14B	SN74ALS273	64	40191-01	
14C	SN74ALS273	64	40191-01	
14D	TC40H374	64	40153-01	
14E	TC 5514 AP-2	64	90173-01	
14F	DM81LS95	64	40113-01	
14G	SN74LS399	64	40197-01	
14H	SN74ALS04	64	40184-01	
15A	SN74ALS74	64	40157-01	
15B	SN74LS258	64	40103-01	
15C	SN74LS258	64	40103-01	
15D	TC40H374	64	40153-01	
15E	TC 5514 AP-2	64	90173-01	
15F	SN74LS245	64	40045-01	
15G	SN74LS259	64	40206-01	
15H	SN74ALS02	64	40155-01	
16A	SN74ALS27	64	40185-01	
16B	SN74LS258	64	40103-01	
16C	SN74LS258	64	40103-01	
16D	TC40H373	64	40152-01	
16E	TC 5514 AP-2	64	90173-01	
16F	SN74LS245	64	40045-01	
1B	ABC P3-1	64	90239-01	
2D	ABC P4-1	64	90240-01	
6E	HRU I	64	90128-01	
12G	HRU II	64	90127-01	
9B	RAD	64	90241-01	
7E	V50	64	90242-01	
	V60	64	90319-01	tillbehör kan ersätta V50

BAT1	Batteri BR-2/3A Litium	65 50561-01
	Stiftlist 40-pol	43 60388-01
	Stiftdon	43 60398-01

6 TEKNISK BESKRIVNING AV KRETSAR OCH FUNKTIONER PÅ PU-KORT

6.1 Microprocessor Z80A



Z80A pinkonfiguration
fig 14.

A0-A15 (Address Bus) Tri-state utgångar. A0-A15 ger en 16-bitars adress (64 kbytes). Adresserar minnen eller in/ut enheter. Vid I/O adressering används dom 8 lägst signifikanta bitarna (A0-A7) för att adressera upp till 256 in/ut portar. Under refresh perioden innhåller A0-A6 en refresh adress.

D0 - D7 (Data Bus) Tri-state in/utgångar. D0-D7 bildar en 8-bitars dubbelriktad databuss för datautbyte med minnen och in/ut enheter.

M1 (Machine cykel one) Utgång, aktiv låg. M1 indikerar att pågående maskincykel är en opkodshämtning, i en instruktions cykel. Under utförande av 2-bytes instruktioner, genereras M1 för varje opkodshämtning. Dessa opkoder börjar alltid med CBH, DDH, EDH eller FDH.

M1 är också aktiv tillsammans med IORQ för att indikera ett godkännande av en interruptbegäran.

MREQ — (Memory Request) Tri-state utgång, aktiv låg.
MREQ signalen indikerar att adressbussen innehåller en giltig adress för läsning eller skrivning i minnet.

IORQ — (Input/Output Request) Tri-state utgång, aktiv låg. IORQ signalen indikerar att A0-A7 innehåller en giltig adress för en läsning eller skrivning till en in/ut enhet.
En IORQ genereras också tillsammans med en M1 signal när en interrupt blir godkänd så att en interruptvector kan placeras på databussen.
En opkodshämtning kan aldrig ske från en in/ut enhet.

RD — (Read) Tri-state utgång, aktiv låg.
RD indikerar att CPU:n vill läsa data från en minnespositon eller från en in/ut enhet.
Är endast aktiv tillsammans med MREQ eller IORQ. Det adresserade minnet eller in/ut enheten skall använda den här signalen för att grinda data ut på databussen

WR — (Write)Tri-state utgång, aktiv låg. WR indikerar att CPU:ns databuss innehåller giltiga data för lagring i adresserad minne eller in/ut enhet.
Är endast aktiv tillsammans med MREQ eller IORQ.
Signalen används för skriva in data till adresserad ut-enhet eller minnesposition.

RFSH — (Refresh) utgång, aktiv låg.
RFSH indikerar att A0-A6 innehåller en refresh adress (från R-registret) för dynamiska minnen och att MREQ signalen ska användas för att göra en refresh läsning av alla dynamiska minnen i systemet.
RFSH är alltid aktiv tillsammans med MREQ.
A7 är alltid låg och A8-A15 i adressbussen kommer att innehålla värdet i I-registret.

HALT — (Halt state) Utgång, aktiv låg.
HALT indikerar att CPU:n har läst in programinstruktionen HALT vilket innebär att den stoppar programkörningen och väntar tills en NMI eller INT inträffar, innan den fortsätter programexekveringen.

När CPU:n stoppats av HALT instruktionen kommer den att utföra sk. NOP´s (No operation) för refresh adresseringen ska fortgå.

WAIT — (Wait) Ingång, aktiv låg.
WAIT informerar CPU:n om att det adresserade minnet eller in/ut enheten inte är klar för att göra en dataöverföring.
CPU:n kommer att lägga till sk. wait states tills WAIT signalen blir hög.
Används vid kommunikation med långsamma minnen eller in/ut enheter.
OBS refresh utförs ej när WAIT signalen är aktiv.

INT — (Interrupt Request) Ingång, aktiv låg.
Avbrott i pågående programexekvering.
INT genereras av in/ut enheter, en avbrottsbegäran besvaras i slutet av pågående instruktionscykel, om den av mjukvaran kontrollerade enable flaggan är satt och om signalen BUSRQ inte är aktiv.
Om CPU:n godkänner avbrottsbegäran sänder den ut en godkännande signal (IORQ och M1 aktiva) i början under nästa instruktionscykel.

NMI — (Non Maskable Interrupt) Ingång, aktiv låg (triggas på den negativa flanken).
NMI ingången har en högre prioritet än INT ingången och besvaras alltid i slutet av pågående instruktionscykel oberoende av om enable flaggan är satt eller inte.
NMI signalen gör att CPU:n kommer att hämta sin nästa instruktion på adress 0066H.
Programräknarens innehåll sparas automatiskt på stacken så programkörning kan fortsätta där avbrottet inträffade.

RESET — Ingång, aktiv låg.
RESET signalen nollställer programräknaren och initierar CPU:n. CPU initieringen består av följande:

1) Disable interrupt enable flaggan
2) Nollställer I-registret
3) Nollställer R-registret
4) Sätter interrupt mode 0

När RESET är aktiv kommer adressbussen och data bussen läggas på en Hög Impediv nivå samt alla kontrollsignaler att läggas inaktiva. Ingen refresh utförs.

BUSRQ	(Bus Request) Ingång, aktiv låg. När BUSRQ blir aktiv kommer CPU:n att lägga adress och data buss till en hög-impediv nivå så snart pågående instruktionscykel är avslutad. Detta används när någon annan enhet ska kontrollera adress och databuss t.ex. en DMA-krets.
BUSAK	(Bus Acknowledge) Utgång, aktiv låg. Används för att tala om för den enhet som aktiverat BUSRQ att CPU:n har lagt adress och data buss i tristate läge (hög impedivt).
Ø	Klockingång till CPU. Styr tidsmässigt alla signaler ut från CPU:n.

7 CPU OPERATIONER

Z80 - CPU exekverar instruktioner (hårdvarumässigt sett) genom att utföra några få mycket fundamentala operationer. Dessa består av:

Läs eller skriv till en minnecell

Läs eller skriv till in/ut enhet

Godkänn en avbrottsbegäran

Alla instruktioner är egentligen mer eller mindre en serie av dessa operationer.
Var och en av dessa operationer kan tidsmässigt sträcka sig tre till sex klockperioder för att slutföras, eller de kan bli förlängda (av WAIT-signalen) för att synkronisera CPU:n med långsammare enheter.
Klockperioderna kommer att refereras till som T lägen (state) och dom angivna operationerna som M (Maskin) cykler.

I fig 15 illustreras hur en typisk instruktions cykel består av ett visst antal M och T lägen, den här instruktionen består av tre maskincykler (M1,M2 och M3). Den första maskincykeln i varje instruktion är en op--kodshämtning (operations kod) som är fyra,fem eller sex T lägen lång (om den inte blir förlängd med hjälp av WAIT).
Op-kodscykeln (M1) används för att hämta op-koden för nästa instruktion som ska utföras, de följande maskincyklerna förflyttar data mellan CPU:n och minnes eller I/O enheter och de kan vara tre till fem T lägen långa (om de inte förlängs med WAIT).

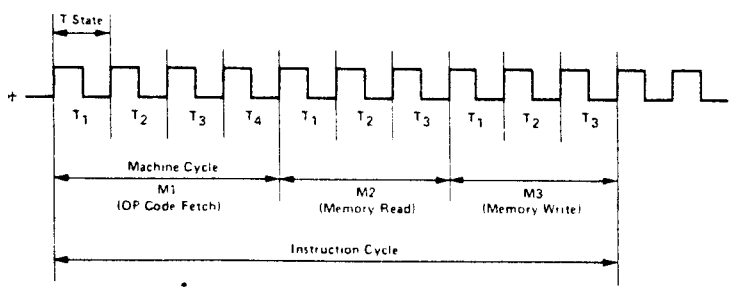


fig 15.

Alla CPU:ns operatiner kan visas med några få enkla tidsdiagram som visas i figurerna 16 till 21. Figurerna visar följande operationer med och utan WAIT lägen.

fig 16 OP-kods läsning (M1 cykel)
fig 18 Läsning och skrivning i minnet
fig 20 Läsning och skrivning till In/Ut enhet
fig 22 Interrupt Request/acknowledge cykel

7.1 OP-kods hämtning

I fig.16 visas tidsdiagrammet för en M1 cykel (op-kod-shämtning).
Notera att innehållet i PC (program counter) placeras på adressbussen i början av M1 cykeln.
En halv klockperiod senare blir MREQ aktiv. Adressen har nu hunnit att stabilisera sig på bussen så att den negativa flanken på MREQ kan användas för att göra Chip Enable på minnesenheten. RD signalen blir också aktiv för att indikera till minnesenheten att data i adresserad minnesposition ska läggas ut på databussen. CPU:n kommer att läsa av databussen på den positiva flanken under läge T3 och med samma flank gör CPU:n RD och MREQ inaktiva. CPU:n har läst av data på bussen innan RD blir inaktiv.
Under lägena T3 och T4 av en opkodshämtning görs en refresh av dynamiska minnen. (CPU:n använder den här tiden till att avkoda och exekvera den hämtade instruktionen, eftersom CPU:n inte gör någon yttre operation under den här tiden kan den användas för att utföra en refreshoperation).
Under T3 och T4 innehåller adressbussen (A0-A6) en refresh adress (från R-registret) och RFSH signalen blir aktiv för att indikera att en refresh av dynamiska minne ska utföras.
Läg märke till att RD signalen inte är aktiv under RFSH tiden, detta för att förhindra att data från minnet läggs ut på databussen.
MREQ signalen under RFSH tiden ska användas för att göra en refresh läsning av minnet. RFSH signalen kan inte ensam användas på grund av att refresh adressen bara garanteras vara stabil under tiden när MREQ är aktiv. Fig 16 visar ett tidsdiagram där WAIT signalen aktiverats. Under T2 känner CPU:n av WAIT signalen och så länge den är låg lägger CPU:N till Tw.

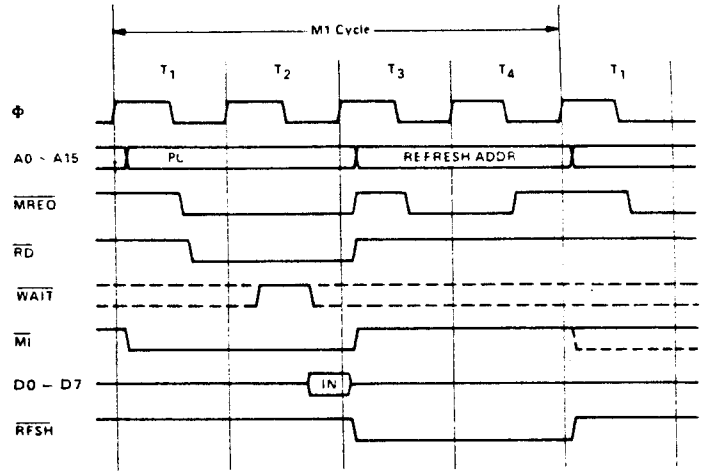

M1 Cycle
T1
T2
T3
T4
T1
Φ
A0 ~ A15
PC
REFRESH ADDR
MREQ
RD
WAIT
MI
D0 – D7
IN
RFSH

fig 16.

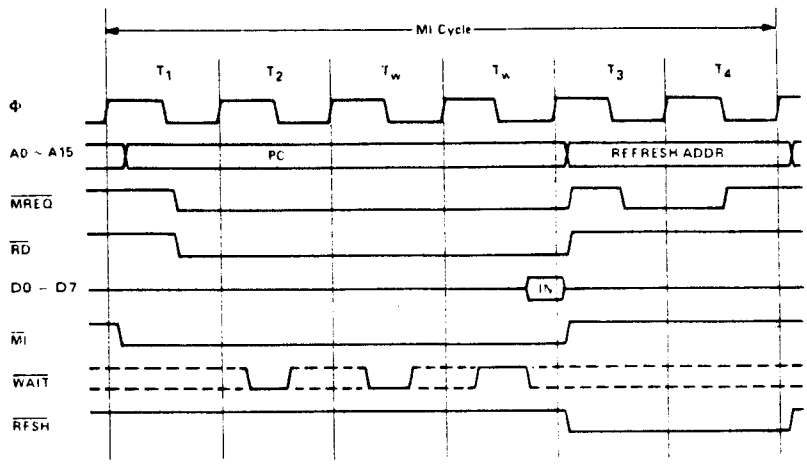

M1 Cycle
T1
T2
Tw
Tw
T3
T4
Φ
A0 ~ A15
PC
REFRESH ADDR
MREQ
RD
D0 – D7
IN
MI
WAIT
RFSH

fig 17.

7.2 Läsning och skrivning i minnet

Fig. 18 visar tidsdiagrammet för en read eller write cykel som inte är op-kodshämtning. Dessa cykler är vanligtvis tre klockcykler långa om inte wait lägen har tillförts via WAIT signalen. MREQ och RD fungerar på samma sätt som under en M1 cykel.
Under en write cykel, indikerar MREQ att adressbussen är stabil. WR signalen blir aktiv först när data på databussen är stabila och kan användas direkt som R/W signal till minnet. WR signalen går inaktiv en halv klockperiod innan adress och databuss ändras så att den kan användas för att klocka in data till minnet.

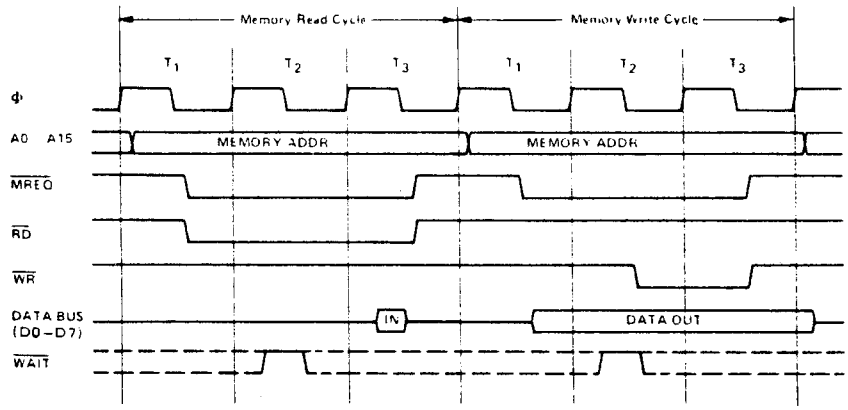


fig 18.

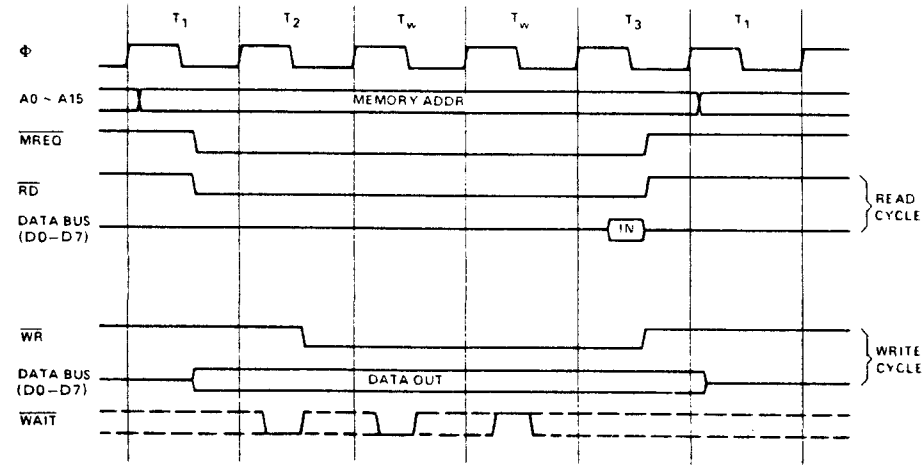


fig 19.

Läsning och skrivning i minnet med WAIT signalen aktiverad.

7.3 Läsning och skrivning till In/Ut enhet

Fig 20 illustrerar en I/O read eller en I/O write operation.
Lägg märke till att under en I/O operation lägger CPU:n automatiskt till en wait period. Anledningen till det är att tiden från det att IORQ går låg tills CPU:n måste känna av WAIT signalen är mycket kort och utan den här wait perioden skulle inte I/O enheten som adresseras hinna avkoda adressen och generera en WAIT signal. Under wait perioden känner CPU:n av värdet på WAIT ingången.

Vid en I/O läsning används RD för att ge enable till den adresserade I/O porten så att den lägger ut data på bussen och CPU:n läser sedan av data på den negativa flanken under T3.

För en I/O skrivning aktiveras WR tillsammans med IORQ och WR signalen går också här inaktiv en halv klockperiod före adress och databuss ändrar sig, så att den direkt kan användas för att skriva in data till adresserad I/O enhet.

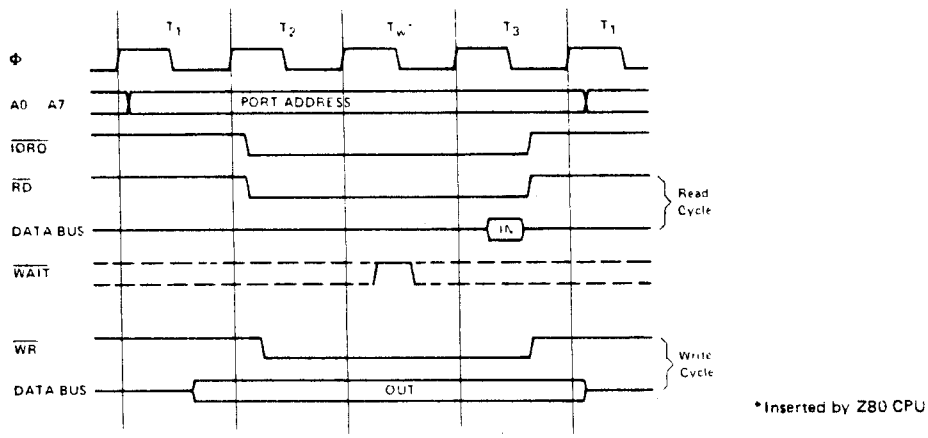


fig 20.

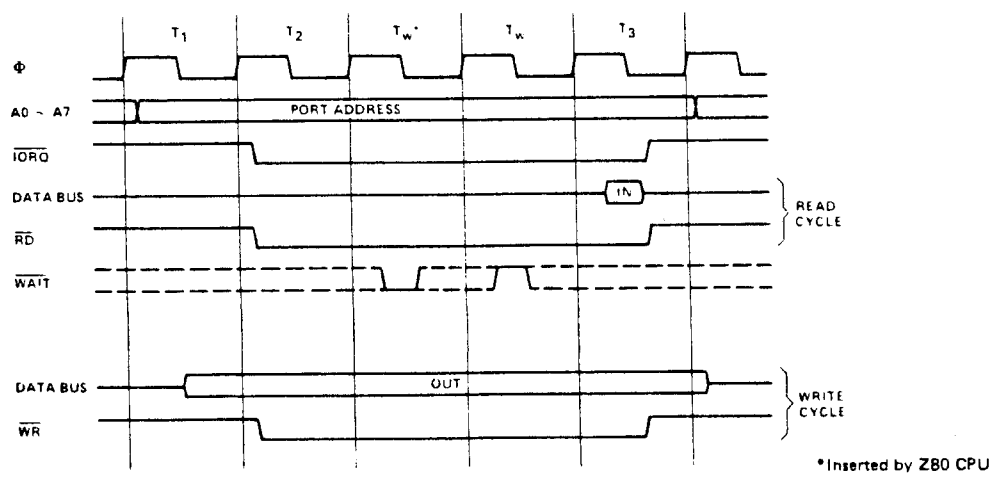


fig 21.

Läsning eller skrivning till I/O enhet med WAIT aktiverad.

7.4 Interrupt begäran

Figure 22 visar tidsdiagrammet för en interrupt cykel.
CPU:n känner av Interrupt signalen på den positiva flanken i sista
T läget i varje instruktion. Interrupten accepteras inte av CPU:n ifall att den interna enable flaggan inte är satt (av instruktionen DI) eller om BUSRQ är aktiv.
När signalen accepteras kommer en speciell M1 cykel att genereras.
Under den här M1 cykeln kommer IORQ att genereras (istället för
MREQ) för att tala om för den I/O enhet som avgett interrupt signalen att den kan placera en 8 bitars vector adress på databussen.
Lägg märke till att två wait lägen är adderade till den här cykeln.
De två wait lägena är till för att ge I/O enheten tid på sig att lägga ut vectorn på databussen.

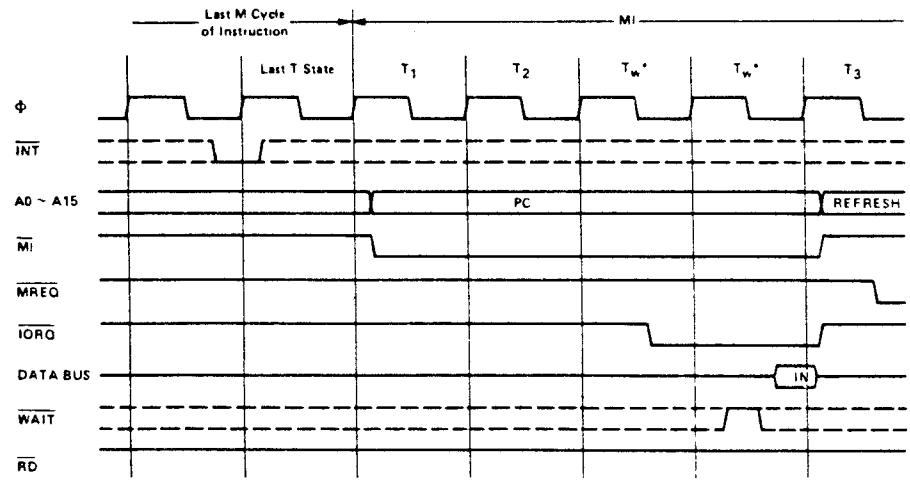


fig 22.

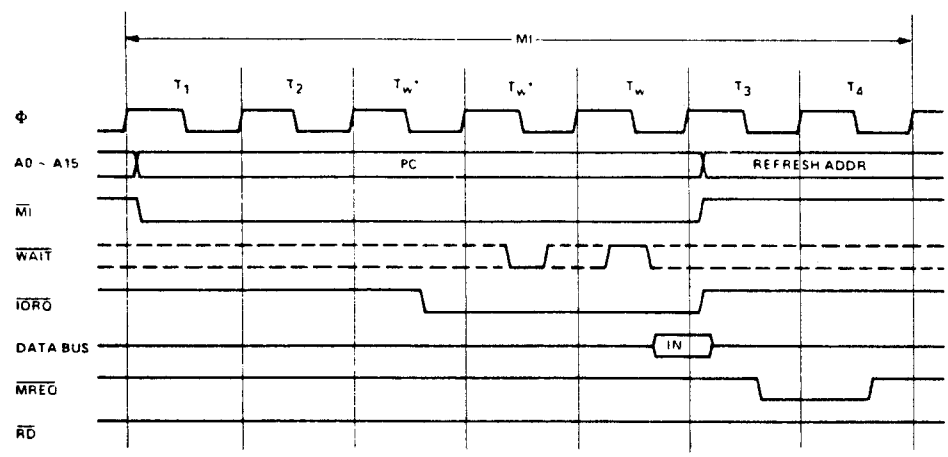


fig 23.

Inerrupt Request med WAIT lägen.

8 SYSTEM RESET

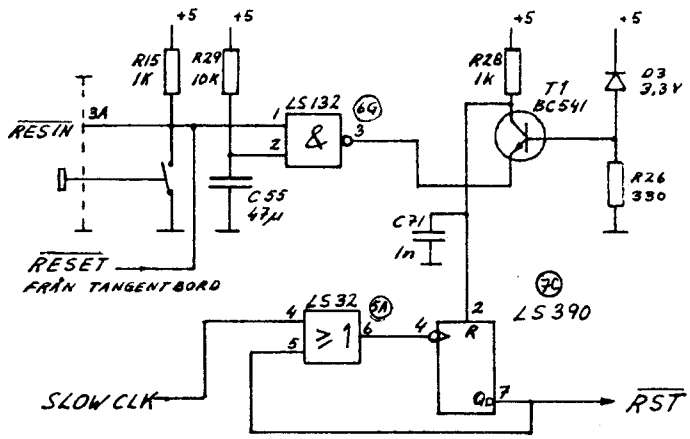
Reset av system ABC 806 genereras på fyra sätt:

1. Vid nätpåslag.
2. Genom att aktivera signalen RES IN pin 3A i den 64poliga busskontakten. 3. Genom att trycka in knappen på baksidan av datorn.
4. Genom att trycka in knappen på baksidan av tangentbordet

När en RST signal har genererats på något av dess sätt kommer den att sändas till CPU,SIO2,CTC,och DART samt VU kontakten och den 64poliga busskontakten.

RST signalen som går ut till busskontakten används för att ge reset till I/O kort som sitter anslutna på bussen. RST signal till I/O kort kan även genereras separat (via programstyrning) med instruktionen INP (7).

Reset signalen är framkopplad till VU-kontakten P9 pin 2 och ger reset till de kretsar som ska nollställas på VU-kortet.

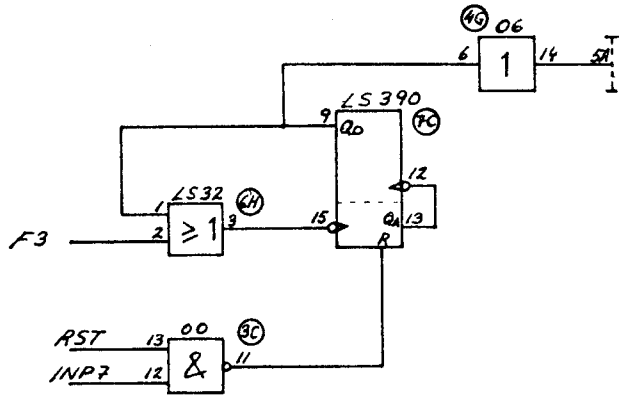


Intern Reset generering.
fig 24.

För att ge en säker RESET i alla lägen låter man NAND grinden i pos 6G styra transistor T1. Grinden utgång kommer att gå hög när någon av dess ingångar går låg. Ingången pin 2 aktiveras vid nätpåslag, då C55 laddas upp och spänningsfall uppstår över R29 som sänker ingången.
Ingången pin 1 kan aktiveras av signalen RES IN, resetknappen på datorn eller med resetsignalen från tangentbordet.

När T1:s emmitter höjs över en viss nivå, som bestäms av spänningsdelaren på basen, stryps T1 och ger reset till räknaren i pos 7C.
T1 kommer att liga strypt en viss minimitid som bestäms av RC-nätet R28 och C71, vilket garanterar att räknaren ges reset även vid mycket kort reset signal från T1.

När R-ingången på räknaren går hög, nollställs Q utgången och RST signalen blir aktiv och ger systemreset, samtidigt öppnas OR grinden och släpper fram klocksignaler till räknaren.
När sedan R-ingången till räknaren blir låg kommer Q utgången att bli hög (efter fyra klockpulser) RST signalen blir inaktiv och stänger samtidigt OR-grinden.
Den interna RST signalen kan därmed inte bli kortare än än tiden mellan fyra pulser på SLOW CLOCK-ingången.
Pulserna SLOW CLOCK kommer från VU-kontakten P9 pin 1 och har periodtiden 16 ms.



Extern Reset generering.
fig 25.

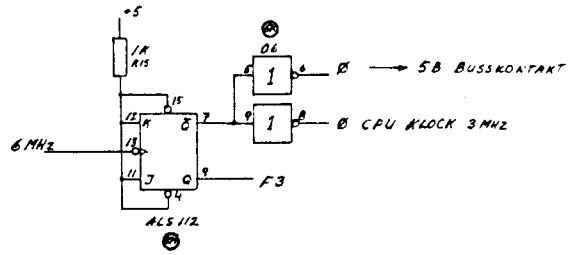
Extern RST signal aktiveras av den interna RST signalen via NANDgrinden. Grindens utgång ger reset till räknaren vilket gör att dess Q-utgång går låg och via buffertkretsen ger RST signal på pin 5A i busskontakten.
Då Q utgången på räknaren blir låg öppnas OR grinden ock släpper fram klockpulser till räknaren. När sedan den interna RST signalen blir inaktiv blir R-ingången låg och räknaren kommer att stegas upp. Efter 8 klockpulser blir Q-utgången hög och den externa RST signalen inaktiv.
Extern RST kan också avges med en programinstruktion INP(7), signalen INP7 kommer då att bli aktiv och extern RST enligt ovan.
Räknarens uppgift är att ge RST signalen tillräcklig längd vid reset med INP7 signalen, tiden blir INP7 stroben+8 klockpulser, ca 3,3 uS.

Reset av systemet innebär att CPU:n kommer att hämta sin första instruktion på adress 0 (PROM i pos 1H) och därifrån sedan fortsätter med att en initiering av system ABC 806. En del av de initieringar som görs är följande:

RAM minnet fylls med nollor, samtidigt testas hur mycket RAM som finns.
Programmera CTC kretsen kanal 3 så att klockfunktionen startar.
Nollställ bildminne (Tecken minne och Attribute).
Programmering av CTC 6845 för att generera en videosignal.
Programmering av DART kretsens kanal B för att kunna ta emot data från tangentbord.
Titta efter om något DOS finns inkopplat om det finns, initiering av flexskiveenhet.
Samt initiering av systemvariabler m.m.

9 SYSTEMKLOCKA, INT OCH NMI

CPU:n s klocksignal som är 3 MHz betecknas Ø den genereras av en 12 Mhz kristalloscillator som finns på VU kortet, där den först delas ned till 6 Mhz för att sedan kopplas in till PU-kortet via VU kontakten P8 pin 14.

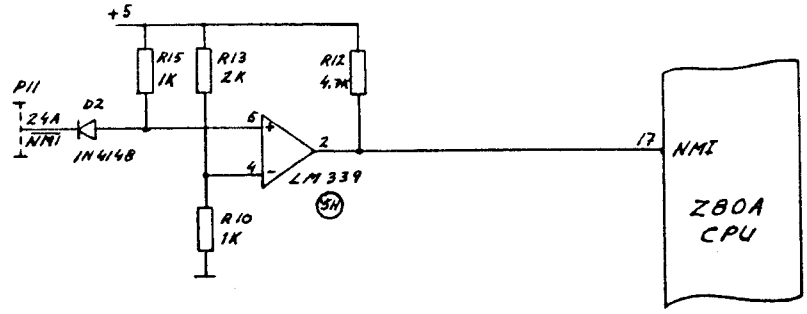


Systemklocka.
fig 26.

6 MHz klockan kopplas sedan till en JK vippa som är kopplad så att den kommer att ge 3 MHz på utgångarna. Q invers utgången är sedan kopplad till två stycken inverterare som även buffrar upp klockan. Från den ena inverteraren går klockan till pin 5B i busskontakten och från den andra får CPU:n sin klocka.
Q utgångens 3 MHz (F3) kommer att ligga i fas med CPU klockan och används för att synkronisera olika förlopp med CPU:n på kortet.

9.1 NMI till CPU

NMI (non maskable interrupt) kan bara genereras via busskontakten pin 24A.

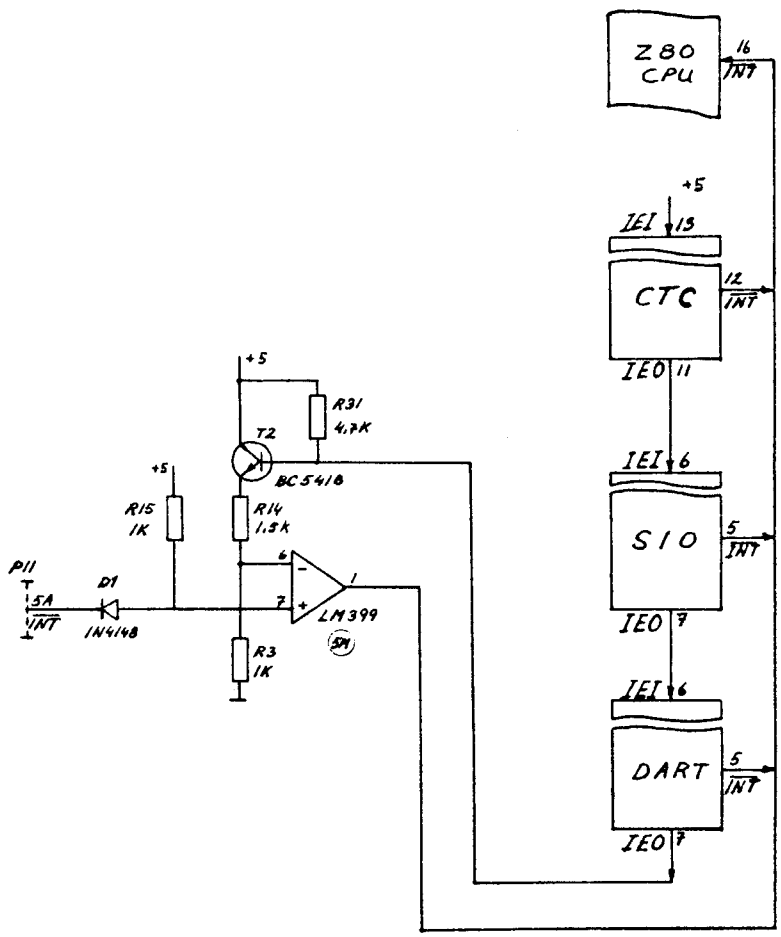


NMI signal.
fig 27.

Signalen läggs in till CPU via en komparatorkrets, komparatorns utgång kommer att gå låg när + ingången får en lägre nivå än - ingången. När CPU:ns NMI ingång blir aktiv kommer den avsluta den instruktion som var under utförande och sedan hämta sin nästa instruktion från adress 0066H.
Instruktionen som ligger i på den adressen är en JP FF8AH där nästa instruktion blir JP 0000H vilket leder till att en restart görs.
Om man vill utnyttja NMI funktionen får man på Adress FF8AH lägga in ett jump till sin egen NMI rutin.

9.2 INT till CPU

INT-funktionen hos CPU:n utnyttjas av CTC,SIO2 och DART kretsarna internt i systemet, men kan också användas från externa I/O enheter via busskontakten. Extern interrupt läggs in via en komparator på liknande sätt som vid NMI begäran.



INT begäran.
fig 28.

Interrupt begäran från de olika enheterna ges olika prioritet, där CTC kanal 0 har högsta och externa interrupt lägsta prioritet.
CTC,SIO2 och DART kretsen har en ingång (IEI) och en utgång (IEO) som används för ge dessa olika prioritet. IEI står för interrupt enable in och IEO för interrupt enable out. Om t.ex CTC har avgett en INT kommer dess IEO utgång att läggas låg och den signalen kopplas sedan till SIO2:s IEI som ger hög nivå på IEO osv. sist ser vi att den DART:ens IEO utgång kopplas till T2 som i sin tur förhindrar externa interrupt. När IEI till en krets är aktiv väntar den kretsen med att avge interrupt till dess IEI signal blir inaktiv.

10 MINNESAVKODNING FÖR ADRESSER 0—32k (ROM-AREAN)

ROM-arean är uppdelad i 8 block om vardera 4 kbyte, varje block ligger lagrat i en PROM krets 2732.

Adressarean är 0000H till 7FFFH. På adress 7800H till 7FFFH kopplas video-minnet tillfälligt in, när skrivning eller läsning av detta görs.

Med adressbitarna A0-A11 adresserar man sig inom varje block och med adressbitarna A12-A15 väljer man vilket block som ska adresseras.

	Adressbit 15	14	13	12	11	Hexadr	Benämning
	1	1	X	X	X	C000-FFFF	HIGH RAM
RAM	1	0	X	X	X	8000-BFFF	LOW RAM
ROM	0	1	1	1	1	7800-7FFF	ABC 76-X/VIDEORAM
	0	1	1	1	0	7000-77FF	ABC 76-X OPT-PROM
	0	1	1	0	X	6000-6FFF	ABC 66-X UFD-DOS
	0	1	0	1	X	5000-5FFF	ABC 56-X
	0	1	0	0	X	4000-4FFF	ABC 46-X
	0	0	1	1	X	3000-3FFF	ABC 36-X
	0	0	1	0	X	2000-2FFF	ABC 26-X
	0	0	0	1	X	1000-1FFF	ABC 16-X
	0	0	0	0	X	0000-0FFF	ABC 06-X

I 1 av 8 avkodaren i pos 4L avkodas adresserna A12-A15 där A12-14 ger vilken 4 kbytes block som ska kopplas in och A15 tillsammans med signalerna ROMDIS och XM ger enable till kretsen.
Om ROMDIS eller XM är aktiva när A15 går låg kommer inte någon avkodning att utföras vilket innebär att ROM arean 0-32k är bortkopplad, dessa signaler används när extraminne kopplas in.

Utgångarna från avkodaren ger CS till respektive krets medan signalen MRD ger OE så att adresserade data läggs ut.

Data ut från 7 av PROM-kretsarna läses via en buffert krets (pos 3L), medan options prommets datutgångar ligger anslutna direkt till databussen.
Vid läsning i ramminnet läggs data ut till bussen via samma buffertkrets.

Buffertkretsens två enable ingångar styrs av signalerna MRD och MOE.
MRD, som är en kombination av CPU signalerna MRQ och RD, är aktiv låg var gång läsning görs i minnet. MOE (Memoery Output Enable) signalen kommer från kombinations grinden LS 51 och är aktiv låg när minnen som använder sig av buffert kretsen adresseras (0000H-6FFFH och 8000H-FFFFH).

MOE signalen genereras från NAND grindarna i pos 4M via LS51, grindarna känner av utgångarna från adress avkodaren och så snart någon ingång går låg kommer en av grindarnas att ge hög nivå och generera MOE via LS 51. Den ena NAND grinden i pos 4M avkodar på pin 1 en signal som indikerar att ramminnet adresseras och ger MOE vid läsning i adressområde 8000H-FFFFH.

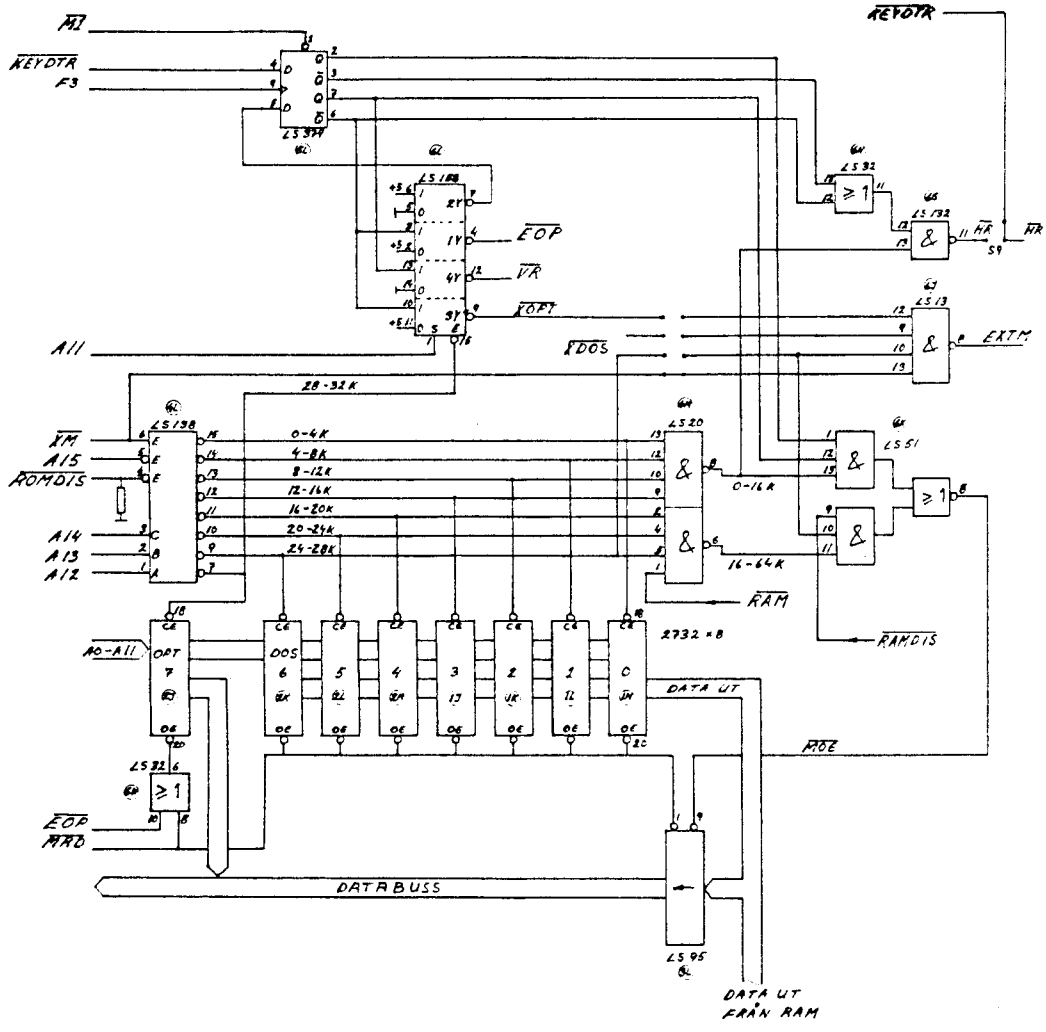


fig 29.

De andra ingångarna till LS 51 används för att koppla bort delar av den interna minnesarean. Genom att lägga någon av ingångarna låg kommer respektive utgång att ligga låg, och signalen MOE kan inte aktiveras.
Ingången på stift 9 kan fås att gå låg genom att bygla fram signalen XLRAM (External Low Ram).
XLRAM signalen är aktiv för adresserna 8000H-BFFFH vilket gör att när ingången pin 11 går hög för att generera MOE går samtidigt pin 9 låg och gör att MOE inte aktiveras.

NAND-grinden i pos 6J avger samtidigt signalen EXTM vilken i sin tur aktiverar signalen XMEMFL på pin 4B i busskontakten för att indikera att läsning ska göras i ett externt minne.

Om man kopplar fram signalen XDOS kommer det interna DOS prom:et att kopplas bort på samma sätt.

Ska ett externt Options prom användas kopplas signalen XOPT fram genom bygling, men i det fallet måste det interna optionsprommet monteras bort.

De två andra ingångarna LS 51 pin 1,12 används inte på ABC 806 utan är kvar från ABC 800 konstruktionen. Samma sak gäller OR-grinden och NAND-grinden som genererar signalen HR.

Multiplexern i pos 6C och D-vipporna i pos 5L används för minnesdelningen mellan optionsprommet och videominnet.
I multiplexern avkodas A11 och CE till options prommet, CE signalen gör enable på multiplexern medan A11 väljer vilken av ingångarna som sk generera respektive utsignal. Utsignalerna är följande:

VR som gör att CPU kan adressera videominnet.

EOP som tillsammans med MRD gör output enable på optionsprommet.

XOPT som är samma signal som EOP men kan användas för externt optionsprom.

Utgången pin 7 används för att välja om EOP eller VR signal ska genereras, detta görs via D-vippan i pos 5L, i ABC 806 används bara den undre vippan.
Vippan enablas av M1 och klockas av F3 (samma som CPU klockan), om en adress till optionsprommet läggs ut av CPU:n (ger CE opt.prom) och adressbit A11 är hög (7800-7FFF) kommer D-ingången på vippan att bli låg, och om M1 också är låg (CPU:n gör en instruktionshämtning) kommer Q utgången att bli låg vilket ger hög VR, samtidigt som Q ger hög nivå och aktiverar EOP som via OR grinden i pos 6H kopplar in optionsprommet.
Om det inte hade varit en instruktionshämtning (M1 hög) hade Q utgången legat kvar på hög nivå och gett VR låg vilket kopplat in videominnet istället.

Om adressbit A11 är låg kommer multiplexerns 0 ingångar att kopplas till utgångarna vilket ger att VR blir hög, EOP blir låg och kopplar in optionsprommet.

Det här innebär att om CPU:n gör en instrukrtionshämtning på adresserna 7800H-7FFFH så kopplas optionsprommet in och ligger inkopplat tills en instruktionshämtning görs på en adress utanför detta område. När en data skrivning eller läsning utförs i det här adressområdet kommer läsning och skrivning att ske i videominnet.

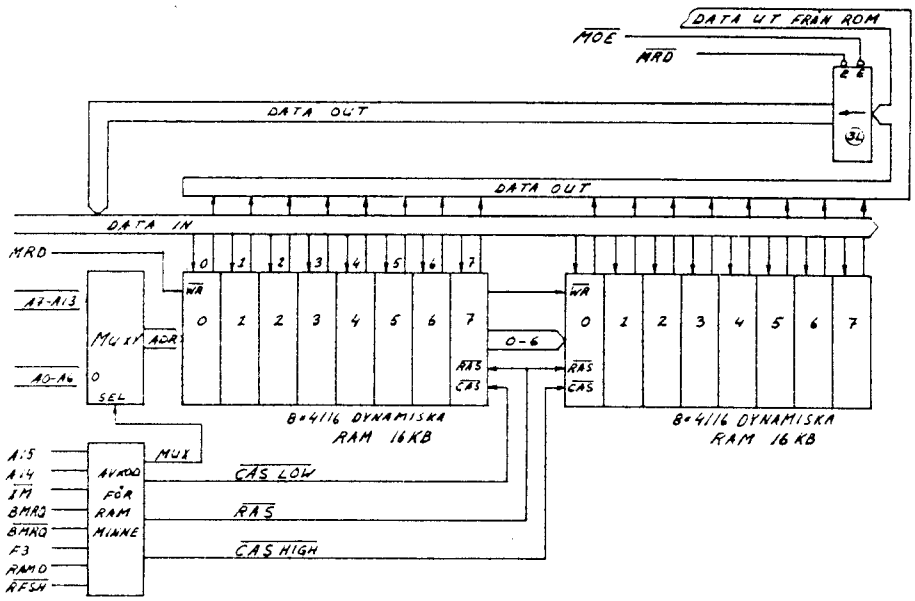
Signalen XM aktiveras när externt extra minne ska kopplas in och kommer då att koppla bort allt internt minne på PU-kortet.

ROMDIS aktiveras från VU-kortet när extraminne ska kopplas in över adressområde 0-32k, ROMDIS aktiv gör att inga CE enable signaler till prom kretsarna aktiveras.

11 ADRESSERING AV RAM 32 KBYTE

Det dynamiska RAM-minnet är 32kbyte stort och är uppbyggt av 16 st 4116 med 16 kbit i varje krets. Kretsarna är kopplade så at de bildar två block om vardera 16 kbyte, vilka benäms Low RAM och High RAM. LRAM ligger på adressområde 8000H-BFFFH och HRAM C000H-FFFFH.

Data läses från RAM-minnet via buffertkretsen i pos 3L, som är gemensam med ROM-minnet. Bufferten kontrolleras med signalerna MRD och MOE , som beskrivits tidigare. Vid skrivning i minnet läggs data in direkt från bussen till datingångarna.



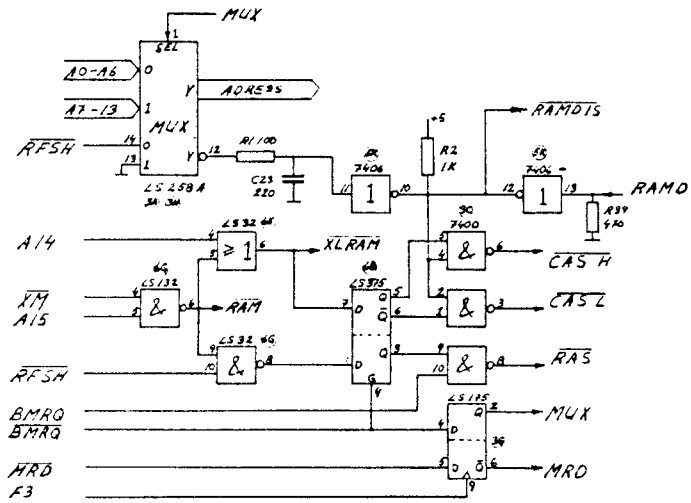
Adressering av RAM-minne
fig 30.

RAM-kretsarna har bara 7 adress ingångar ,vilket gör att adresseringen sker i två steg. Först läggs adressbitarna A7-A13 fram till adressingångarna (via multiplexerkretsen), från avkodningsblocket aktiveras sedan RAS signalen som är kopplade till alla kretsarna, och radadressen läses in.
Sedan aktiveras signalen MUX som styr selektingången på multiplexern och adressbitarna A0-A6 kopplas fram till adressingångarna, när A0-A6 stabilliserat sig aktiveras signalen CASL eller CASH och läser in kolumnadressen till minnet.

En bit i respektive ramkapsel (8*1bit) har nu adresserats och om data ska skrivas eller läsas styrs av signalen MRD som är kopplad till kretsarnas write ingång. CAS fungerarar också som Output Enable för databufferten i kretsen.
Vilken av signalerna CASL och CASH som aktiveras bestäms av vilket adressområde man hamnar i.

11.1 RAM avkodning

När CPU:n gör en läsning eller skrivning på adressområde 8000H till FFFFh kommer RAM-minnet att adresseras.
I ramavkodningen avkodas A15 tillsammans med signalen XM/ i en nand-grind, om XM/ är inaktiv kommer utgången att på grinden att bli låg när A15 går hög och signalen RAM aktiveras.
RAM signalen inverteras i den efterföljande nandgrinden så att D-ingången på låskretsen blir hög, Q-utgången på låskretsen blir också hög och låses till detta värde när BMRQ/ signalen på Gate ingången blir aktiv.
När BMRQ blir aktiv (inversen av BMRQ/) kommer den tillsammans med signalen från Q-utgången via nand-grinden att aktivera RAS/ signalen till RAM-minnet och A7-A13 som då finns på adressingångarna kommer att läsas in som en radadress.



Ram-avkodning.
fig 31.

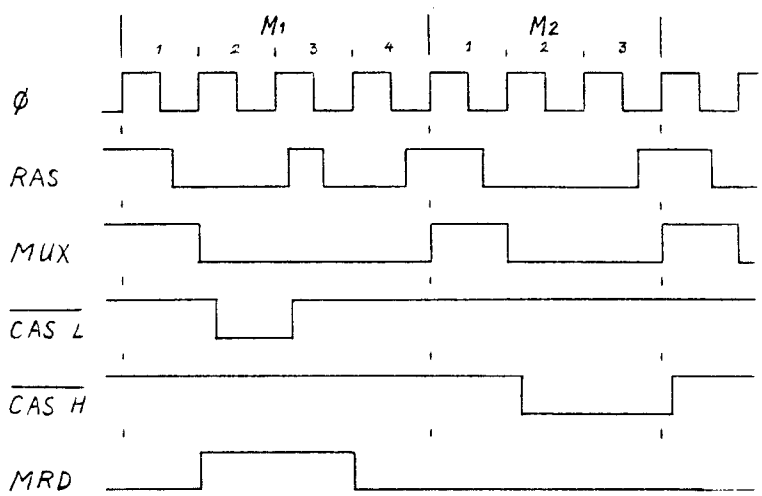
Signalen BMRQ är också ansluten till en D-vippa i kretsen i pos 3G, som klockas av F3 (CPU klocka 3MHz), BMRQ/ aktiveras från CPU:n på den negativa flanken av klockan, medan vippan klockas på den positiva , signalen MUX som aktiveraras av BMRQ/ på vippans Q-utgång kommer därför att bli fördröjd ca 80 nS vilket ger tillräckligt med tid för att läsa in radadressen.

Signalen MUX styr multiplexerns selektingång och kommer att koppla fram adressbitarna A0-A6 när den blir låg.
I multiplexern kopplas RFSH/ signalen till utgången pin 12 och ger en låg nivå som fördröjs i ett RC nät, inverteras och ger hög nivå till en av ingångarna på två nand grindar, de andra ingångarna på nand grindarna kontrolleras av A14 och A14/ via låskretsen som har A14 som insignal från or grinden.
Beroende på vilket värde har kommer CASL eller CASH att aktiveras och läsa in A0-A6 som kolumn adress. Fördröjningen av CAS signalen gör att A0-A6 hinner stabilisera sig innan CAS blir aktiv.

Under slutet av varje instruktionshämtning som CPU:n utför kommer signalen RFSH/ att bli aktiv för att indikera att en refresh adress finns tillgänglig på adressledningarna A0-A6 (från R-registret), RFSH/ och BMRQ signalerna aktiverar RAS/ via nand grinden och låskretsen på samma sätt som A15 gjorde vid en normal läsning eller skrivning, signalen MUX har låg nivå på grund på grund av tidigare BMRQ/ siganlen (se tidsdiagram) vilket gör att A0-A6 adresserar minnet som refresh adress.

RFSH/ signalen som tidigare genererade CAS/ signal är nu låg vilket gör att CAS/ signalen inte kommer att aktiveras.
RFSH görs internt i minneskretsarna när man gör en läsning, under RFSH adressering aktiveras inte CAS signalen vilket gör att databuffertarna i kretsen inte aktiveras.

RFSH läsningen görs på en hel rad i kretsarna vilket gör att det räcker med en refresh adress 0-127. En refreshläsning av varje bit i minnet måste göras med minst 2 mS mellanrum.

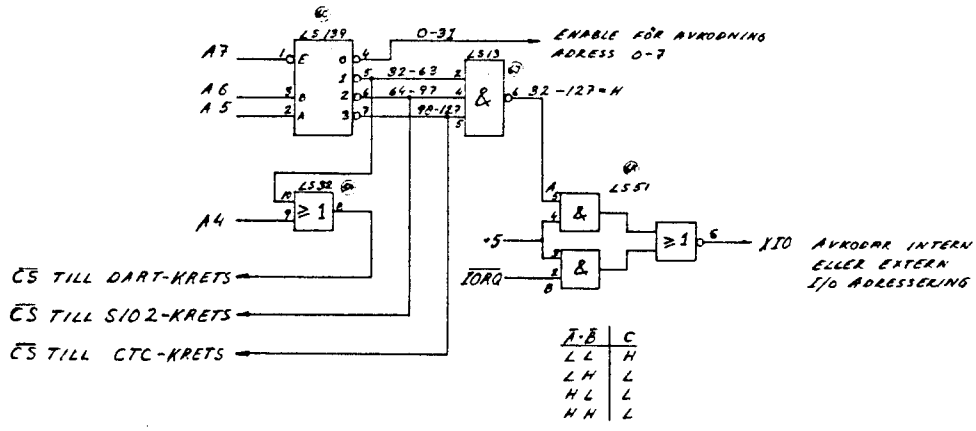


Tidsdiagram läsning skrivning i RAM.
fig 32.

12 AVKODNING I/O ADRESSER

När CPU:n utför en I/O read eller write använder den adressbitarna A0-A7 för att adressera I/O enheterna samtidigt som IORQ indikerar en I/O operation.
I/O adresserna tillsammans med IORQ avkodas i datorn och kommer för vissa adresser att generera kontrollsignaler för in och utläsning av data från I/O enheter.

Adresserna 0-31 och 128 till 255 genererar externa I/O strobar (till busskontakten) medan adress 32-127 används för internt bruk.



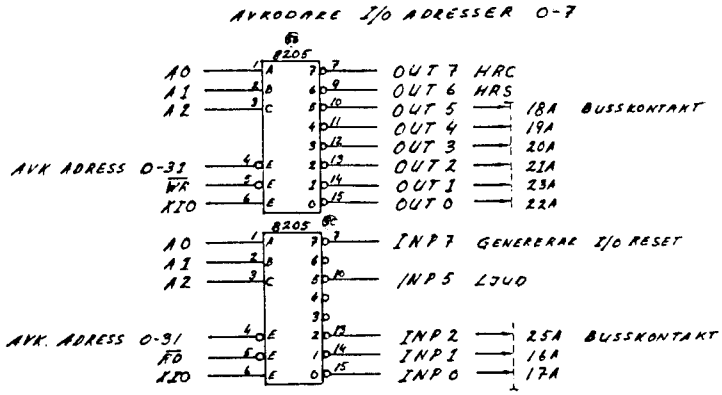
Avkodning intern, extern I/O adress.
fig 33.

I 1 av 4 avkodaren avkodas A7-A5 så att utgång 0 blir låg för adresserna 0-31, 2 för adress 32 till 63, 3 för adress 64 till 97 och 4 för adresserna 98 till 127.
Utgång 1 ger tillsammans med adressbit A4 chip enable till DART kretsen som sedan själv avkodar adressbit A0, A1, IORQ och RD/WR.
Utgång 2 ger enable till SIO2 kretsen som också sköter resterande avkodning själv och utgång 3 enablar CTC kretsen på samma sätt.

När någon av dessa tre utgångar går låg kommer NAND grindens utgång att gå hög vilket avkodas i kombinations grinden LS 51 (som också avkodar IORQ) så att signalen XIO förhindras att gå hög vilket i sin tur gör att inga externa I/O strobar avges.
Signalen XIO blir aktiv hög när IORQ är låg tillsammans med en låg nivå ut från NAND grinden vilket indikerar en extern I/O adress 0-31 eller 128-255.

Utgång 0 och XIO signalerna är kopplade till två stycken 1 av 8 avkodare som också avkodar A0-A2, RD och WR för att generera in och ut strobar.

Avkodaren i pos 5B kommer att generera OUT strobar för skrivning till externa I/O kort. För att kretsen ska avkoda adressbitarna A0-A2 måste de tre enablevillkoren vara uppfyllda, E1 kontrolleras av signalen från den tidigare nämda avkodarens utgång noll vilken är aktiv låg för en adress med värde 0-31.



Generering av Out och Inp strobar.
fig 34.

Enable ingång E3 styrs av signalen XIO som är aktiv hög för I/O adresserna 0-31 och 128-255 och enable ingång E2 kontrolleras av WR signalen från CPU:n. När CPU:n utför I/O write på adresserna 0-31 kommer avkodaren att känna av värdet på A0-A2 och aktivera någon av utgångarna.
En del av Out strobarna används för vissa förutbestämda funktioner,
vilka är följande:

OUT1 används för att selekta ett I/O kort, varje I/O kort anslutet till bussen ska ha ett eget kortnummer (ex styrkort flexskiveenhet ABC 830 har adress 45).

OUT0 Skriver data till selektat I/O kort.

OUT2-5 dessa har ingen förutbestämd funktion utan används för olika funktioner på I/O korten.

OUT6 Genererar inte någon extern strob utan används på VU-kortet för att skriva i en I/O port vid hantering av grafikminnet.

OUT7 Genererar inte heller någon extern strob utan används på VU-kortet vid grafikgenerering.

Den andra 1 av 8 avkodaren avkodarsamma signaler A2-A0, adress 0-31 och XIO men till enable ingång E2 är CPU:ns signal RD kopplad, kretsen kommer därför att generera signaler när CPU:n gör en I/O read på adresserna 0-31. Signalerna som genereras är följande:

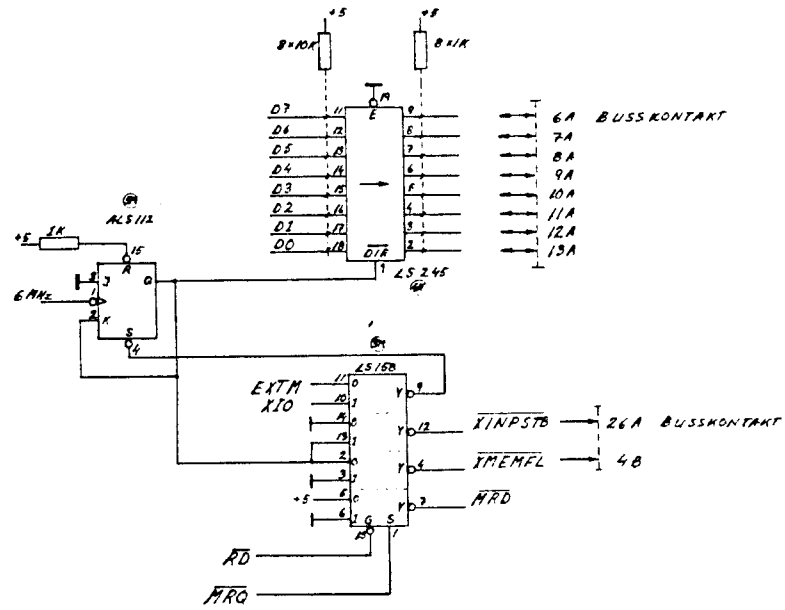
INP0 Läser data från ett selektat I/O kort.
INP1 Läser status från ett selektat I/O kort.

INP2 Ingen förutbestämd funktion kan användas för olika funktioner på I/O korten.

INP5 Används ej i ABC 806, i ABC 800 genererade man ljud med den här signalen.

INP7 Ger RESET till I/O kort anslutna till den 64-poliga busskontakten.

När CPU:n gör en I/O read på adresserna 0-31 och 128-255 ska data läsas in från en extern I/O enhet, D7-D0 i busskontakten är anslutna till CPU:ns databuss via en dubbelriktad buffertkrets.



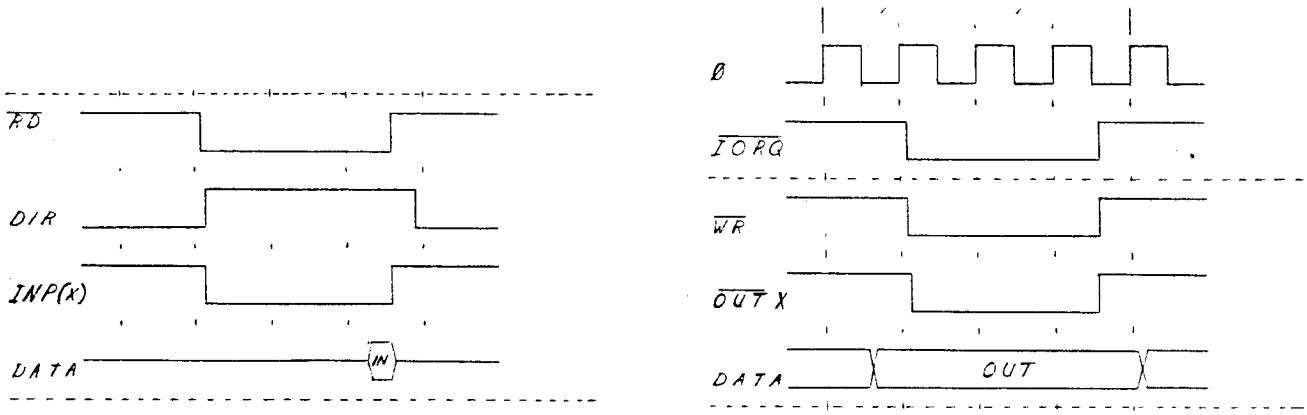
Läsning av data från busskontakten.
fig 35.

Databufferten ligger vanligt vis riktat ut mot busskontakten genom att dess DIR ingång är låg, när en extern I/O read utföres måste DIR ingången läggas hög så att databufferten riktas in mot CPU:n så den kan läsa av data från I/O enheten.
DIR signalen styrs via en multiplexerkrets som avkodar signalerna RD MRQ och XIO samt EXTM.
RD styr multiplexerns Gate ingång vilket gör att den är aktiv under både en minnesläsning och en I/O läsning, MRQ bestämmer sedan vilken av multiplexerns ingångar som ska kopplas till respektive utgång.

MRQ kommer att vara hög vid en I/O läsning vilket gör att ingångarna 1 ansluts till utgångarna.
Signalen XIO som är aktiv hög för adresserna 0-31 och 128-255 kommer då att kontrollera utgången 3Y som i sin styr Set ingången på en JK-vippa, JK vippans Q utgång kontrollerar i sin tur DIR-ingången på multiplexerkretsen.
När Y3 utgången går låg (XIO hög) kommer Q-utgången på JK-vippan att bli hög och bufferten att riktas in mot CPU:n, om XIO varit låg (I/O adress 32-127) hade 3Y utgången varit hög och DIR hade inte påverkats.

När RD signalen sedan blir inaktiv kommer DIR signalen att återställas till låg nivå av den negativa flanken på klockingången genom att JK vippans D-ingång är ansluten till låg nivå.
Återställningen av riktningen på buffertkretsen fördröjs lite genom klockningen med 6 Mhz för att I/O enhetens buffertkrets och datorns buffertkrets inte ska få utgångarna riktade mot varandra.

Q utgången från JK-vippan kommer också att generera signalen XINPST på pin 26A i busskontakten samtidigt med DIR signalen. XINPST kommer därför att vara aktiv vid en I/O läsning på adresserna 0-31 och 128-255, signalen kan därför sägas vara en generell I/O strob för dessa adresser.



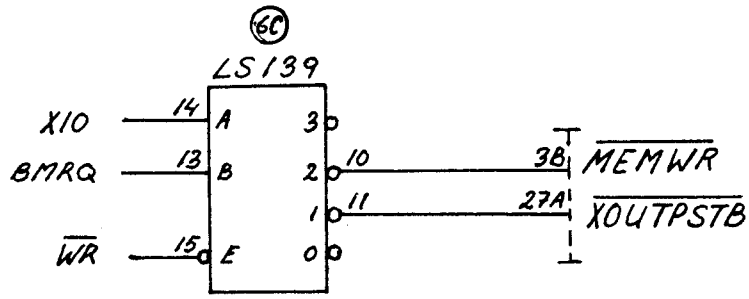
Tidsdiagram för externa I/O signaler.
fig 36.

Vid minnesläsningar när MRQ är låg och ingångarna 0 är kopplade till respektive utgång kommer signalen EXTM att kontrollera DIR signalen på sammma sätt som beskrivits för XIO.

EXTM kommer att bli aktiv när signalen XM i busskontakten aktiveras vilket görs när externa minnen är anslutna till bussen.
EXTM signalen kommer då att vända databussen in mot CPU:n och samtidigt aktivera utgången 1Y som är ansluten till busskontakten och kallas XMEMFL.
XMEMFL indikerarar att läsning ska göras i ett externt minne.

Motsvarande signal för intern minnesläsning är signalen MRD som också skapas i multiplexern på utgång 2Y, denna används för att göra läsningar i det interna minnet.

Motsvarande signaler för extern I/O skrivning och extern minnesskrivning genereras av 1 av 4 avkodaren i pos 6C.



Skrivningsignaler för externt minne eller I/O enhet.
fig 37.

Kretsen avkodar signalerna XIO,BMRQ och WR.
När XIO och WR är aktiva (XIO=hög,WR=låg,BMRQ=låg) kommer utgång 1 att gå låg och avge XOUTSTB i busskontakten, XOUTSTB kommer att vara aktiv när CPU:n gör en I/O skrivning på adresserna 0-31 och 128-255.

MEMWR signalen som används för skrivning i externa minnen kommer att vara aktiv när BMRQ och WR är aktiva (BMRQ=hög,WR=låg,XIO=låg) denna signal avges alltid när CPU:n gör en läsning i minnet.

13 Z80 CTC

Z80 CTC (Counter Timer Circuit) är en programmerbar krets med fyra av varandra oberoende kanaler, där varje kanal innehåller räknare och tidsfunktioner för mikrodatorsystem baserade på Z80-CPU. Några av funktioner i CTC är följande:

- Var och en av kanalerna kan väljas att arbeta i antingen countermode or timermode.
- I varje mode finns en nedräknare vilken räknas ned av klockpulser på CLK/TRG ingången eller av systemklockan på pin 15, räknaren kan läsas av CPU och visar hur långt den har kvar till noll.
- Systemklockan delas valbart med 16 eller 256.
- Ett tidskonstant register som automatiskt omladdar nedräknaren när den räknat till noll
- Valbart positiv eller negativ trigg på klockingången för att initiera tidsfunktioner i timer mode. Samma ingång används som klocka för nedräknaren i countermode.
- Tre av kanalerna har en utgång som indikerar med en puls att räknaren har kommit till noll i counter mode och timer mode.
- Varje kanal kan programmeras för att avge en interrupt när räknaren kommit till noll,varje kanal avger en egen interrupt vektor.

13.1 CTC pin konfiguration

D7-D0 Z80-CPU data buss. (dubbelriktat, tri-state.) Bussen används för att överföra alla data och kommando ord mellan CPU:n och CTC.

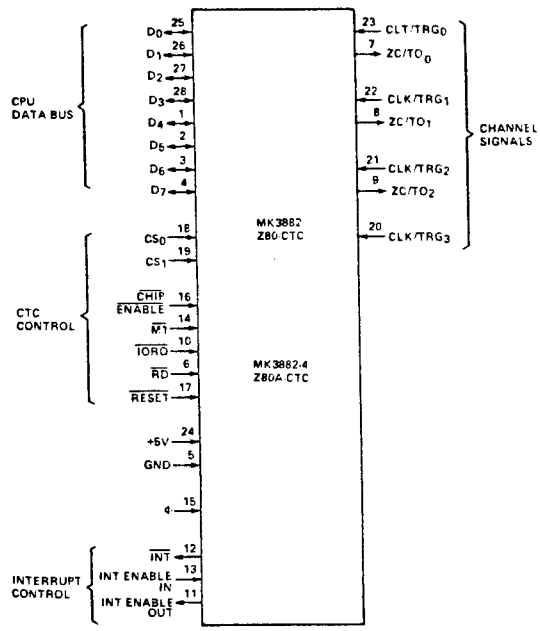
CS1-CS0 Channel Selekt. (ingång.) Är en två bitars adress som väljer ut en av kanalerna för en I/O write eller read.

	CS1	CS0
CH 0	0	0
CH 1	0	1
CH 2	1	0
CH 3	1	1

Är i ABC 806 anslutna till A0 och A1.

CE — Chip Enable (Ingång, aktiv låg.)
Gör att CTC:n accepterar kontrollord, interrupt vektorer eller tidskonstant data från data bussen under en I/O write cykel eller lägger ut innehållet i räknaren under en I/O read.
CE aktiveras i ABC 806 av avkodade I/O adressbitarna A7-A6 medan A0 och A1 väljer kanal.

O — System Clock (Ingång.)
En klocksignal som synkroniserar signaler i CTC med CPU:n samt styr andra kontrollsignaler i CTC.



Pinkonfiguration CTC-kretsen.
fig 38.

M1 — M1 signalen från CPU:n (ingång, aktiv låg.)
När M1 är aktiv tillsammans med MREQ hämtar CPU:n en instruktion från minnet, när M1 är aktiv tillsammans med IORQ indikerar det ett godkännande på en interruptbegäran. CTC kommer då att placera en interruptvektor på data bussen om någon av kanalerna har aktiverat INT signalen.

IORQ — Input/Output Request från CPU:n (Ingång, aktiv låg.) IORQ tillsammans med CE och RD signalerna överför data eller kontrollord mellan CPU:n och CTC:n.
Under en en CTC write cykel måste IORQ och CE vara aktiva och signalen RD inaktiv. CTC:n har ingen separat signal för write, utan genererar en egen writesignal internt om RD är hög.
Under en CTC read cykel, är IORQ,RD och CE aktiva och CTC kommer att placera ut innehållet i räknaren på databussen.

Om IORQ och M1 är aktiva, godkänner CPU:n en avbrottsbegäran och CTC:n lägger ut en interuppt vektor på bussen.

RD — Read från CPU:n (Input, aktiv låg.)
Aktiverar tillsammans med CE och IORQ en läsoperation eller om RD är hög en skrivoperation.

IEI — Interrupt Enable In. (Ingång, aktiv låg.)
Den här signalen används för att i ett system inordna olika enheter i en kedja med olika prioritet för att begära interrupt. En hög nivå talar om för kretsen att ingen annan enhet med högre prioritet har avgett interrupt till CPU:n, om den varit låg hade CTC:n inte aktiverat sin INT signal utan väntat tills IEI hade blivit hög.

IEO — Interrupt Enable Out. (Utgång, aktiv låg)
IEO signalen kommer att aktiveras om någon av kanalerna i CTC:n har avgett interrupt och fått ett godkännande på densamma eller om IEI ingången har blivit aktiv vilket indikerar att en enhet med högre prioritet ligger i en interruptrutin.

INT — Inerrupt Request. (Utgång, aktiv låg.)
Aktiveras när en kanal har programmerats så att den ska avge interrupt när dess räknare har räknat ner till noll.

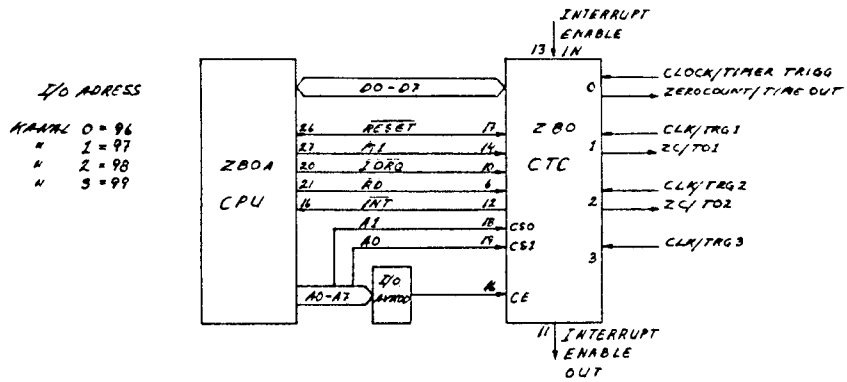
RESET — Reset. (Ingång, aktiv låg.)
Nolställer CTC kretsen register och sätter signalerna ZC/TO, INT till inaktivt läge och datautgångarna i hög impedivt läge.

CLK/TRG — External clock/Timer trigger. (Ingång.)
Det är fyra CLK/TRG ingångar en för respektive kanal.
I counter mode kommer varje aktiv flank att räkna ner räknaren med ett. I timer mode kommer en aktiv flank att starta nedräkningen i räknaren. Vilken flank som ska vara aktiv bestäms i programmeringen av kretsen.

ZC/TO — Zero count/Time Out.(Utgång, aktiv hög.)
Det finns tre ZC/TO utgångar en för varje kanal 0-2 medan kanal 3 inte har någon sådan utgång.
I både counter och timer mode kommer en aktiv hög puls att avges var gång räknaren räknat ner till noll. (i timer mode räknas räknaren ned av systemklockan delat med 256 eller 16.)

13.2 CTC adressering och programmering

CTC är inkopplad till CPU:n enligt fig, adressbitarna A7-A5 avkodas i I/O avkodeningen och som kommer att göra chip enable på ctc kretsen, A0 och A1 är kopplade till CS0 och CS1 för att välja vilken kanal som CPU:n vill kommunicera med.



CTC adressering.
fig 39.

I/O adresser som används för CTC-kretsen är följande:

Kanal 0 = Adress 96 (01100000B,60H)
Kanal 1 = Adress 97 (01100001B,61H)
Kanal 2 = Adress 98 (01100010B,62H)
Kanal 3 = Adress 99 (01100011B,63H)

CTC:ns CLK/TRG ingångar för kanal 0-2 är anslutna till en klocksignal på 1.5 Mhz denna används för att räkna ned ett register i CTC:n, även klockan på pin 15 som är 3Mhz kan användas för att räkna ned registret, vilken av de två klocksignalerna man använder beror på vilken frekvens man vill ha ut på ZC/TO utgångarna.

ZC/TO utgångarna på kanal 0-2 används för att ge sändnings och mottagningsklock till SIO2 och DART kretsen.

Kanal 3:s CLK/TRG ingång är kopplad till SIO2 kanal A:s klockingång som bestämmer sändningshastigheten,och kan användas för att eventuellt mäta klockfrekvensen eller titta efter om där finns någon klocksignal.

CTC:n programmeras vid initiering av system ABC806 så att kanal 3 kommer att generera interrupt till CPU:n med ett intervall av 10.6 mS, den här interruptrutinen används för att räkna tiden i basic funktionen TIME¤.

Rutinen räknar i ett antal minnespositioner upp sek,-min,tim,datum,mån och år. Dessa måste naturligtvis sättas efter varje gång som strömmen varit avslagen till systemet.
Minnespositioner som används för de här värdena är följande:

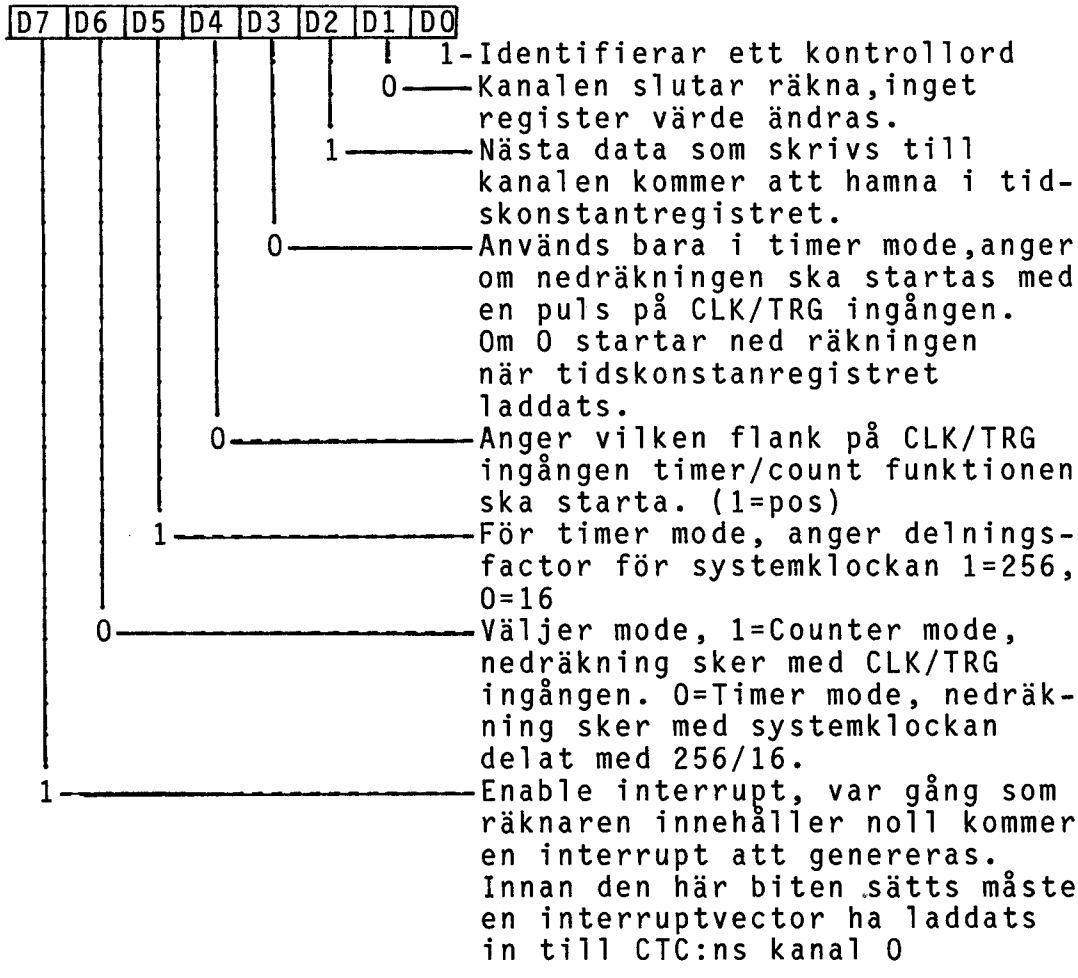
(FFF4H) 65524 = Sekunder
(FFF3H) 65523 = Minuter
(FFF2H) 65522 = Timmar
(FFF1H) 65521 = Datum
(FFF0H) 65520 = Månad
(FFEFH) 65519 = År

I ABC806 finns på VU-kortet en CMOS klocka med batteribackup som kan användas för att sätta rätt värden efter varje gång ABC806 varit avstängd.

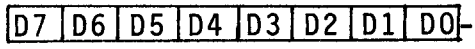
De övriga kanalerna i CTC:n programmeras inte förrän någon av de två kanalerna för seriell överföring programmeras.

Programmering av CTC-kretsen görs genom att skriva till kontrollregistren för respektive kanal. Nedan följer en kort beskrivning av registren. Värdena som är angivna är de värden som sätts vid reset av ABC806.

- Programmering av kontrollregistret.(Kanal 3)

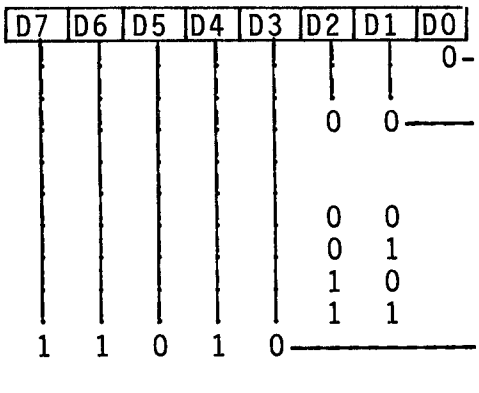


- Programmering av tidskonstantegister.



Innan kanalen kan börja fungera enligt kontrollregistret, måste det här registret laddas med ett värde, innhållet i det här registret laddas in i räknaren var gång den räknat till noll

- Programmering av interruptvektor.



0 – Identifierar att det är en interrupt vector som laddas.

0 0 – Dom här två bitarna räknaren CTC:n själv ut när den lägger ut vectorn.

0 0 Kanal 0
0 1 Kanal 1
1 0 Kanal 2
1 1 Kanal 3

1 1 0 1 0 – Den del som bestämmes av användaren.

Vid en interrupt till CPU:n kommer den att läsa in vektorn och och använda den tillsammans med innehållet i I-registret för att få fram en adress till en position i minnet där den ska hämta adressen till interruptrutinen för respektive kanal.

Adresserna i minnet för respektive kanal där adressen till interruptrutinen ligger är följande:

(FFD0H) 65488 LSB Kanal 0
(FFD1H) 65489 MSB
(FFD2H) 65490 LSB Kanal 1
(FFD3H) 65491 MSB
(FFD4H) 65492 LSB Kanal 2
(FFD5H) 65493 MSB
(FFD6H) 65494 LSB Kanal 3
(FFD7H) 65495 MSB

14 SERIEKOMMUNIKATIONSKRETSAR SI02, DART

Z80 SIO (Serial Input/Output) och DART (Dual Asyncron Reciever Transmitter) är kretsar med två kanalers, programmerbara för olika funktioner som är konstruerade för att klara ett brett område av seriell kommunikation i ett mikrodatorsystem.
Grundfunktionen är att omvandla data från parallel till serieform eller från serie till parallela data. Några av funktionerna i SIO2 och DART är följande:

- Två av varandra oberoende kanaler som kan sända och ta emot data i full duplex.

- Fyra bytes buffer vid mottagning,två bytes vid sändning.

- Vid asynkron överföring:
  - 5,6,7 eller 8 bit/tecken
  - 1,1 1/2 eller 2 stoppbitar
  - Jämn,udda eller ingen paritet
  - Paritetsfel,överskrivnings detektering
  - Klockfrekvensen kan vara bithastigheten x1,x16,x32 och x64

- Vid synkron överföring: (ej DART)
  - Intern eller extern tecken synkronisering
  - En eller två synktecken i separata register
  - Automatisk sändning av synktecken
  - CRC generering och kontroll av CRC
  - Kan sända och ta emot HDLC och IBM SDLC

- Separata modem kontrollsignaler för båda kanalerna

- Modem status kan övervakas.

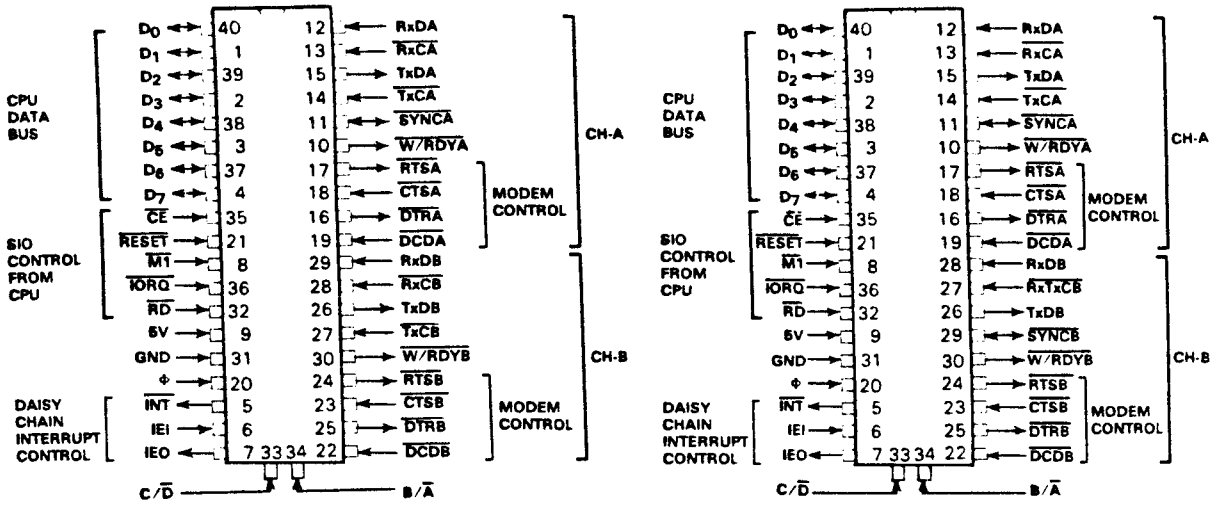
14.1 Pinkonfiguration SIO2 och DART

D0-D7 Data buss. (dubbelriktad, tri-state)
Bussen används för att överföra data och kommando ord mellan CPU och SIO2/DART

CE Chip Enable (ingång, aktiv låg)
Aktiverar SIO2/DART för att ta emot data eller kommando under en skrivoperation eller lägga ut data på bussen inder en läsoperation. Aktiveras i ABC806 av en avkodad I/O adress.

B/A Selekt kanal A eller B (ingång, hög selektar kanal B)
Med den här ingången bestämmer man vilken kanal man ska kommunicera med under en skriv eller läs operation. I ABC806 används A0 för att välja kanal.

C/D Control/Data select (ingång, hög=kontroll) Med den här ingången bestäms hur SIO2/DART:n ska tolka data som skrivs till den, en hög nivå gör att data tolkas som kommando för den kanal som är vald med signalen B/A, en låg nivå innebär att informationen som läses eller skrivs är data.



Pinfiguration SIO2,DART
fig 40.

Φ System clock. (Ingång)
Används för att synkronisera interna signaler samt synkronisera SIO2/DART med CPU:n.

M1 M1 från CPU. (ingång, aktiv låg)
När M1 är aktiv tillsammans med MRQ hämtar CPU:n en instruktion från minnet, när M1 är aktiv tillsammans med IORQ indikerar det ett godkännande på en interrupt begäran. SIO2/-DART:n kommer då att placera en interrupt vektor på databussen om någon av kanalerna har aktiverat INT signalen.

IORQ Input/Output Request från CPU:n. (Ingång, aktiv låg)
IORQ tillsammans med CE och RD signalerna överför data eller kontrollord mellan CPU och SIO2/DART:n.
När CE,IORQ och RD aktiva kommer data/kontrollord att läsas från adresserad kanal, När CE,IORQ är aktiva och RD inaktiv kommer kommer data/kontrollord att skrivas till SIO2/DART. Om IORQ och M1 är aktiva samtidigt innebär det ett godkännande på en avbrottsbegäran från CPU, och om kretsen har genererat en INT signal ska den lägga ut en interruptvector på bussen.

RD — Read från CPU:n. (ingång, aktiv låg)
Aktiverar tillsammans med CE,IORQ en skrivoperation eller om RD är hög en skrivoperation.

IEI — Interrupt Enable In. (Ingång, aktiv låg)
Den här signalen används för att i ett system inordna olika enheter i en kedja med olika prioritet för att begära inerrupt. En hög nivå talar om för kretsen att ingen annan enhet med högre prioritet har avgett interrupt till CPU:n, om den varit låg hade kretsen inte aktiverat sin INT signal utan väntat tills IEI signalen blivit hög.

IEO — Interrupt Enable Out. (Utgång, aktiv låg) IEO signalen kommer att aktiveras om någon av kanalerna i SIO2/DART har avgett interrupt och fått en gokännande på densamma eller om IEI ingången har blivit aktiv vilket indikerar att en enhet med högre prioritet ligger i en interruptrutin.

RESET — Reset. (Ingång, aktiv låg)
En aktiv RESET inaktiverar både sändning och mottagning, lägger TxD signalerna och modem signalerna på hög nivå. Registren i SIO2/DART:n måste sedan omprogrammeras.

INT — Interrupt Request. (Utgång, aktiv låg)
När SIO2/DART har programmerats för att avge interrupt, kommer en interruptbegäran till CPU:n att göras genom att SIO2/DART:n aktiverar INT signalen.

W/RDY (CH A,B) — Wait/Ready. (Utgång)
Kan programmeras för att fungera antingen som en WAIT signal för synkronisering med CPU:n eller en READY signal vid DMA överföring.

CTS (CH A,B) — Clear To Send. (Ingång , aktiv låg)
Kan programmeras för Auto enable för sändning av data vilket innebär att CTS signalen måste vara låg för att data ska sändas på TxD. Nivån på CTS kan läsas av via ett register. SIO2/-DART:n kan programmeras så att den avger interrupt när ett omslag sker på CTS signalen.

DCD (CH A,B) — Data Carrier Detect. (Ingång, aktiv låg)
Funktionen är samma som för CTS signalen med skillnaden att den kan användas för att göra mottagning av data möjlig.

RxD (CH A,B) — Receive Data. (Ingång)
Ingång för serie data.

TxD (CH A,B) — Transmit Data. (Utgång)
Utgång för serie data.

RxC — Receiver Clocks. (Ingång)

(CH A,B) I asynkron mode kan mottagarklockan kan vara 1,16,32 eller 64 gånger överföringshastigheten av data. RxD ingången avkänns under den positiva flanken på RxC signalen.

TxC (CH A,B) Transmitter clock. (Ingång)
I asynkron mode kan sändningsklockan vara 1,16,32 eller 64 gånger överföringshastigheten av data. Klockan delas ned internt i kretsen och multipeln måste vara samma för både RxC och TxC. TxD utgången växlar data på den negativa flanken på TxC.

RTS (CH A,B) Request To Send. (Utgång, aktiv låg)
I asynkron mode blir RTS hög när sändningsbufferten är tom, och låg när data har laddats i bufferten.
Kan kontrolleras av CPU:n via ett register.

DTR Data Terminal Ready. (Utgång, aktiv låg)
Kontrolleras av CPU:n via ett register

SYNC Synchronization (Ingång/Utgång, aktiv låg)
SIO2

RI Ring Indikator (Ingång, aktiv låg) DART
I asynkron mode fungerar den liknande som CTS och DCD och kan läsas av CPU:n via ett register.

Skillnaderna på SIO2 och DART kretsen är att DART kretsen inte kan sända eller ta emot synkrona data vilket SIO2 kretsen kan.
Det skiljer också på pin konfigurationen mellan kretsarna. På grund av att pinnarna inte räcker för att ha alla signalerna RxC,TxC,READY,SYNC och DTR i båda kanalerna samtidigt tar man bort en av dom i kanal B, på SIO2 saknas SYNC signalen och på DART:n har man en gemensam klockingång för RxC,TxC.

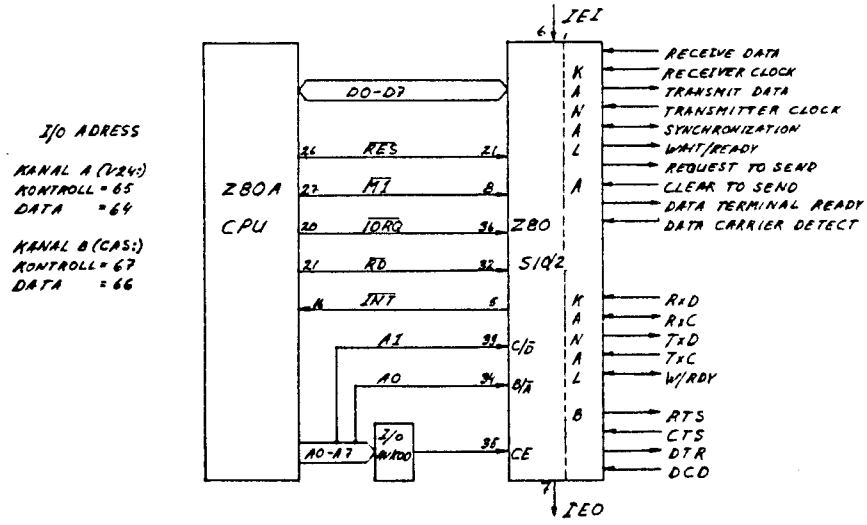
14.2 Adressering av SIO2

SIO2 kretsen är inkopplad till CPU:n enligt fig 41, adressbitarna A7-A5 avkodas i I/O avkodningen och aktiverar CE på SIO2 kretsen, A0 är kopplad till B/A och väljer kanal och A1 är kopplad til C/D för ange om dataöverföringen gäller data eller kontroll ord.

I/O adresser som används för SIO2 kretsen är följande:

Kanal A (V24:) Kontroll=Adress 65 (01000001B,33H)
Data =Adress 64 (01000000B,32H)

Kanal B (CAS:) Kontroll=Adress 67 (01000011B,35H)
Data =Adress 66 (01000010B,34H)



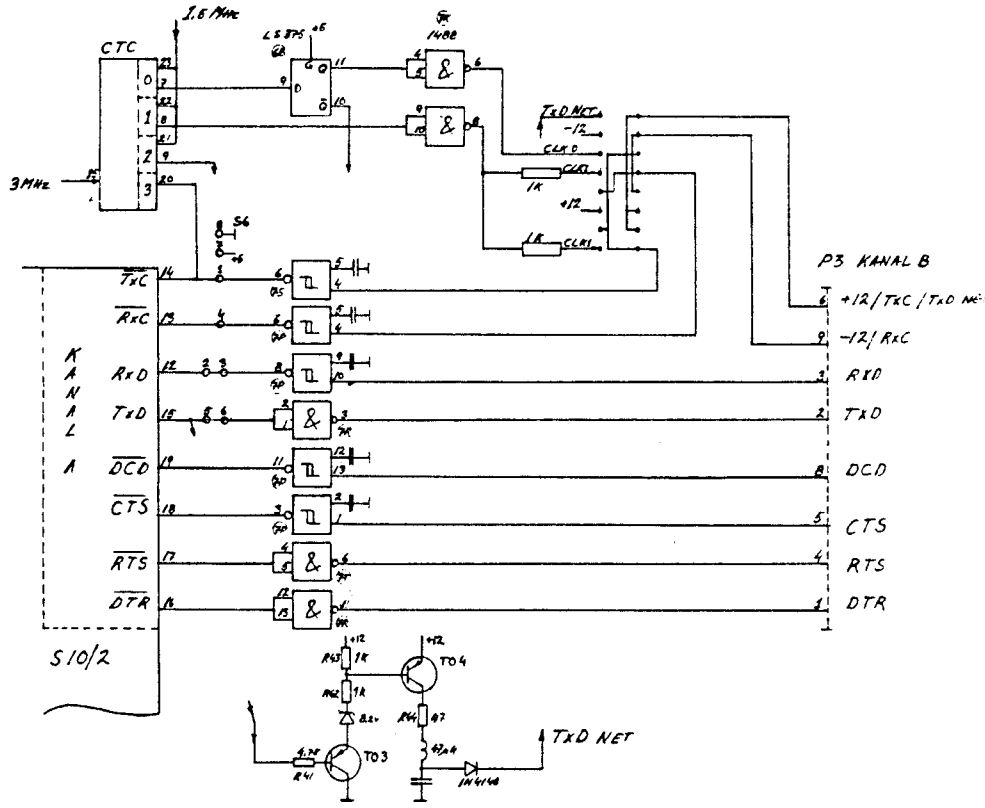
Adressering SIO2.

fig 41.

IEI (Interrupt Enable In) på SIO2 kretsen är kopplad till CTC kretsens IEO (Interrupt Enable Out) utgång vilket innbär att SIO2 kretsen får en lägre prioritet vid interrupt än CTC:n.
Inom kretsen har kanal A en högre prioritet än kanal B.

14.3 SIO2:s anslutning till RS232 kanal B.

Det är SIO2:s kanal B som sänder och tar emot data från ABC806 RS232 port kanal B. Hur anpassning är gjord visas i fig. 42.



RS232 anpassning.
fig 42.

Utsignalerna från SIO2 kretsen har TTL nivå och görs därför om till RS232 nivå i de så kallade "Line Drivers" som ger ca -7V för en hög nivå och +7V för en låg nivå.
Kretsarna är av typ MC1488 som spänningsmatas med ±9 volt och är kapabla att driva en utgångsström på 10 mA, utgångsimpedansen är 300 ohm.

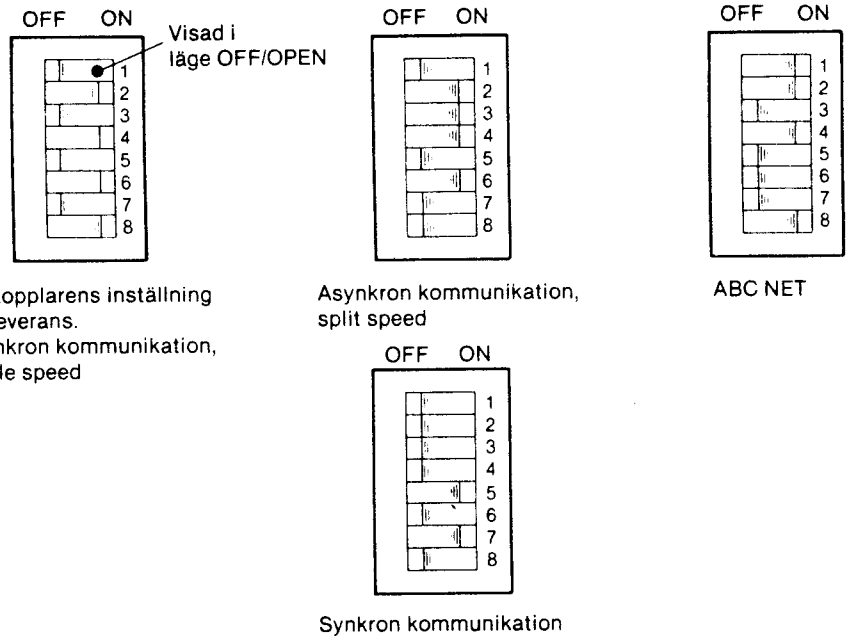
Signalerna in från RS232 kontakten matas till speciella mottagare kretsar som gör om från RS232 nivå till TTL nivå, kretsarna har också en separat ingång där man ansluter en kondensator för att filtrera bort brus och andra störningar på insignalen. Kretstypen är MC1489

Kanal B är den kanal som utformats för en generell användning här ansluter man t.ex modem vid terminalkörning med asyncron eller syncron överföring, om man använder datorn i ett ABC NET system ska anslutningen ske här, osv.

För att kunna klara av olika överföringstyper finns en omkopplare som man ställer om för olika funktioner.

De streckade linjerna visar inställningen av omkopplaren vid leverans, som ger asyncron överföring, samma hastighet på sändning och mottagning då TxC och RxC tas från CTC:ns utgång kanal 1.

Figuren visar hur omkopplarna ska ställas för några olika användnings områden.



DIP omkopplare.
fig 43.

Transistorerna T03 och T04 fungerar som drivsteg vid anslutning till ABC NET, TxD är kopplad till förstärkarsteget och kommer att kopplas till pin 6 när omkopplaren ställs in enligt bild ovan.
Effektförstärkningen är ca 3* nominella drivförmågan på RS232 signalen (ca 30 mA), i centralenheten för ABC NET sitter ett pull down motstånd på 470 ohm anslutet till -12 volt vilket gör att signalnivån kommer att bli ±7V.
Utgången är också försedd med ett avstörningsfilter för radiofrekvenser.
När anslutning till ABC NET görs måste också DOS prommet bytas ut mot ett prom som innehåller drivrutiner för ABC NET.
Kanal B på SIO2 används inte i ABC806 systemet, i ABC800 användes den för att sända och ta emot data från ett yttre kassetminne.

14.4 Allmänt om anslutning av periferiutrustning

Om man ska ansluta någon form av periferiutrustning till kanal B, ska den för det första ha en anslutning med RS232 sedan måste man göra en anslutningskabel, eftersom det vanligtvis är en 25 polig anslutningskontakt på enheten och ABC datorn har en 9 polig.

Funktionerna på signalerna är standardiserade men eftersom man kan tolka en standard på olika sätt varierar signalfunktionerna ibland.
I anslutningen till ABC datorn är det två signaler utöver TxD och RxD som är viktiga för att sändning och mottagning ska fungera.
Drivprogrammen för V24: kanalen kontrollerar när man gör "open ´V24:´ as file 1" att DCD signalen in på kontakten är hög, om så inte är fallet kan drivrutinen ligga kvar och vänta tills DCD blir hög. Det fenomenet kan man få bort genom att ange en parameter när man öppnar filen då kommer man att få en retur från drivrutinen med en felkod.
Den andra viktiga signalen är CTS, den signalen känner drivrutinen av varje gång som den skriver (PRINT #1) ett tecken till V24: porten och om den inte är hög väntar drivrutinen med att skicka data tills CTS blivit låg.

I anslutningskabeln kan man därför koppla på följande sätt, förutsatt att periferiutrustningen betraktas som DTE (Data Terminal Equipment):

ABC 9-polig		Periferiutrustning 25 pol
TxD	2	RxD 3
RxD	3	TxD 2
SG	7	SG 7
DCD	8	Någon signal på enheten som blir hög när spänningen slås på. Kan också byglas till pin 6 i den 9 poliga kontakten.
CTS	5	DTR 20 är det vanligaste stället den här signalen återfinns på. Det ska vara den signal som enheten använder för att signalera busy med (att den inte kan ta emot mer data).

Överföringshastighet och paritet kan sedan anges med parametrar när man öppnar filen ut mot V24: porten.
Om man ställer överföringshastigheten och pariteten på periferiutrustningen ska den ställas till 2400 Baud och ingen paritet, vilket är standardvärden som tas när man öppnar en fil första gången. För mer ingående beskrivning av parameterlistan hänvisas till bruksanvisning Options Prom.

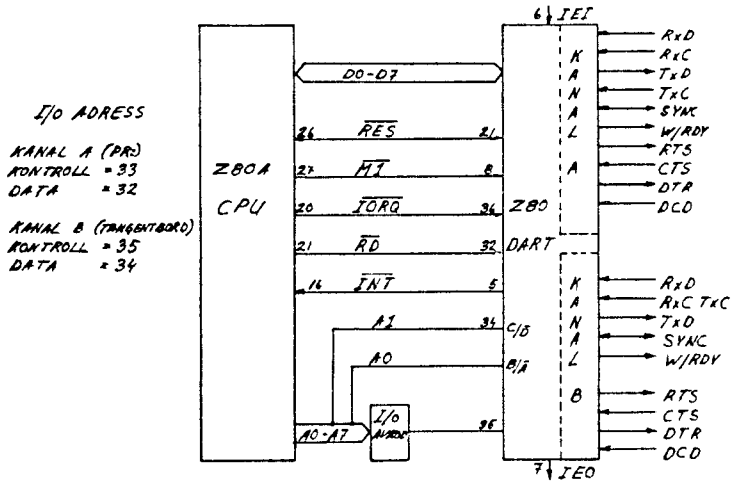
Det som nämts ovan angående anslutning till kanal B, gäller också för kanal A på ABC datorn.

14.5 Adressering av DART

DART kretsen är inkopplad till CPU:n enligt fig 44, adressbitarna A7-A5 och A4 avkodas i I/O avkodningen och aktiverar CE på SIO2 kretsen, A0 är kopplad till B/A och väljer kanal och A1 är kopplad til C/D för ange om dataöverföringen gäller data eller kontroll ord.

I/O adresser som används för DART kretsen är följande:

Kanal A (PR:)	Kontroll=Adress 33	(0010XX01B,35H)
	Data =Adress 32	(0010XX00B,34H)
Kanal B (CON:)	Kontroll=Adress 35	(0010XX11B,33H)
Tangentbord	Data =Adress 34	(0010XX10B,32H)



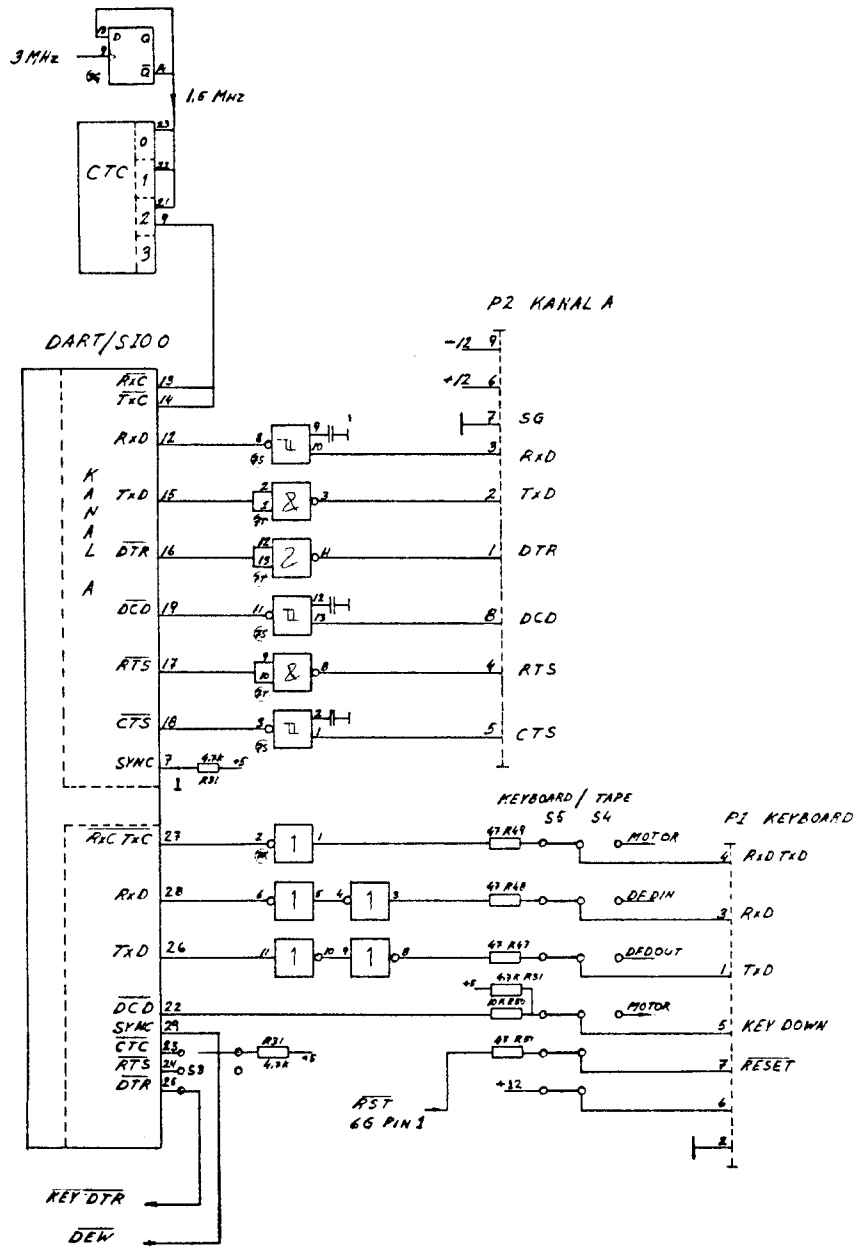
Adressering DART.
fig 44.

IEI (Interrupt Enable In) på DART kretsen är kopplad till SIO2 kretsens IEO (Interrupt Enable Out) utgång vilket innbär att DART kretsen får en lägre prioritet vid interrupt än SIO2 kretsen.
Inom kretsen har kanal A en högre prioritet än kanal B.

14.6 Anslutning till tangentbord och V24:ugången kanal A

Via DART kretsens kanal A som är ansluten till den 9 poliga V24 kontakten som också benäms kanal A kan man via drivrutinen PR: kommunicera seriellt med utrustning som har ett gränssnitt av V24 eller RS 232 typ. Dataöverföring kan bara göras i asynkron form och med samma hastighet på både sändning och mottagning.

Utsignalerna från DART kretsen har TTL nivå och görs därför om till V24/RS232 nivå i så kallade "Line Drivers" som ger ca -7V för en hög nivå och +7V för en låg nivå.
Kretsarna är av typ MC1488 som spänningsmatas med ±9 volt och är kapabla att driva en utgångsström på 10 mA, utgångsimpedansen är 300 ohm.
Klocksignal för RxC och TxC tas från CTC:ns kanal 2.
För anslutning av utrustning till kanal A gäller samma som beskrivits tidigare för kanal B med avseende på signalfunktioner.



Anpassning mot tangentbord och serieport kanal A
fig 45.

Till DART:ens kanal B är tangentbordet anslutet, tangentbordet kan både sända och ta emot data. Signalerna från tangentbordet har TTL nivå. Dataformat på överföringen mellan tangentbord och dator är 650 b/s, 8 databitar,ingen paritet och två stoppbitar, RxC,TxC klockan som kommer från tangentbordet har frekvensen 10 kHz.

RxD och TxD är kopplade till kontakten via två inverterare som buffrar signalen medan klockan inverteras och buffras.
Motstånden är till för att dämpa reflexer på ledningarna.
Signalen KEYDOWN från tangentbordet indikerar att en tangent är nedtryckt när den är aktiv, CPU:n kan via ett register läsa av värdet, BASIC funktionen Sys(?) använder sig av signalen.

RI signalen på pin 29 kan läsas av CPU via ett register i DART kretsen, till ingången är signalen DEW/ kopplad vilken indikerar start för en bild på bildskärmen (Vert-synk), den används bland annat av programvaran som gör att man får Soft Scroll på bildskärmen.

Signalen KEYDTR/ från DART kretsens pin 25 kontrollerar inkoppling av adressområde 0-32k från extraminnet. Om den här signalen aktiveras kopplar den bort ROMarean och bildminnet (0 till 32k) på PU-kortet och aktiverar ramminne i den arean.

Via bygel fältet som finns mellan DART:en och den 6 poliga DIN kontakten kan signalerna från SIO2:s kanal B kopplas fram till kontakten istället för signalerna från DART:ens kanal B.

15 PROGRAMMERBARA REGISTER I SI02 OCH DART

Nedan följer en kort beskrivning av hur man från Basic kan läsa av signalerna CTS,DCD eller RI/SYNC eller sätta DTR och RTS signalerna på SI02 och DART kretsen.

Kretsarna har åtta stycken register som man kan skriva till och tre stycken som kan hämta information från. För programmering av registren använder man sig av kontrolladressen för respektive krets när den är programmerad och man vill sända eller ta emot data använder man Data adressen.

Ex. Out 33,0 skriver till Write Register 0
    Inp (33) läser Read Register 0

Om man vill skriva eller läsa i något av registren WR1-WR8 eller läsa RR1-RR2 så skriver man först en registeradress i register 0 och sedan kan man läsa eller skriva till adresserat register.

Ex Out 33,5   adresserar WR 5
    Out 33,128 skriver värdet 128 i register 5

    Out 33,2 adresserar register 2
    Inp 33   läser RR2

```
10 !Läs värdet på DCD,RI/SYNC och CTS
15 !Vilken kanal och vilken krets som läses bestäms
    med adress
20 OUT 33,16 !Läser in aktuellt värde till reg 0
30 RR0%=INP(33)
40 IF RR0% AND 8 THEN DCD%=0 ELSE DCD%=1
50 IF RR0% AND 16THEN RI%=0 ELSE RI%=1
60 IF RR0% AND 32THEN CTS%=0 ELSE CTS%=1
40 PRINT "DCD" DCD%,"RI"RI%,"CTS"CTS%
50 GOTO 20
```

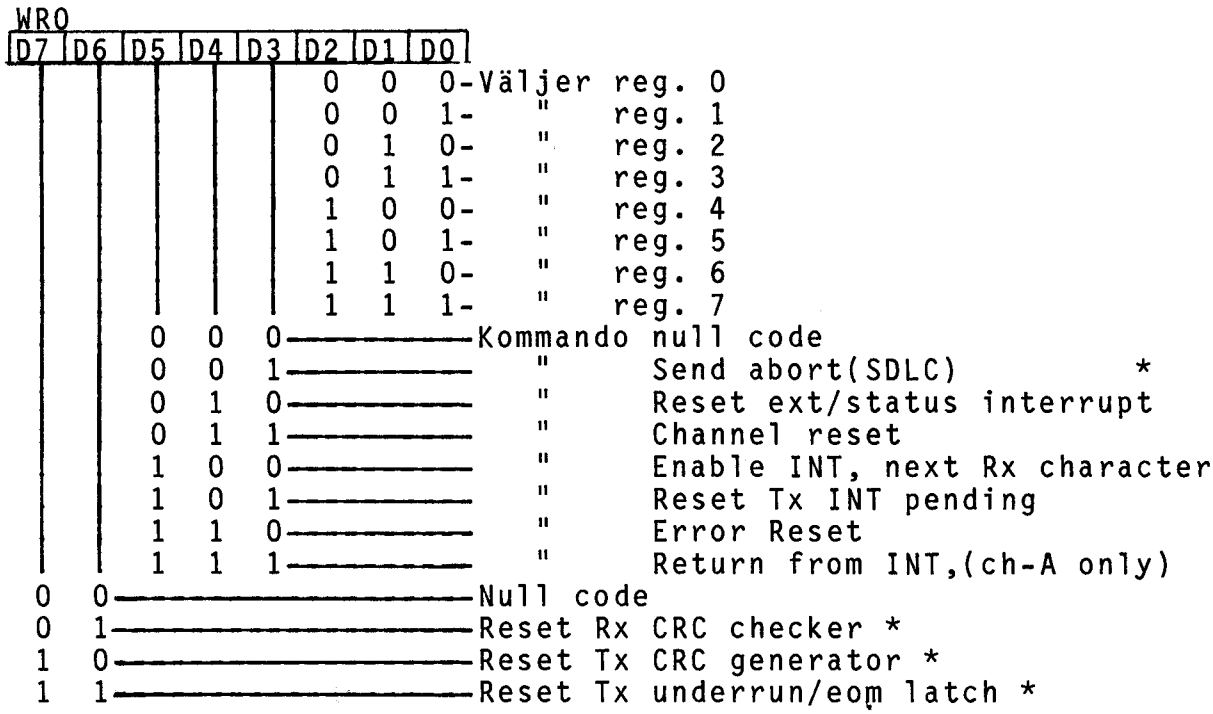
OBS. att biten i registret får inversen av signalen på respektive ingång.

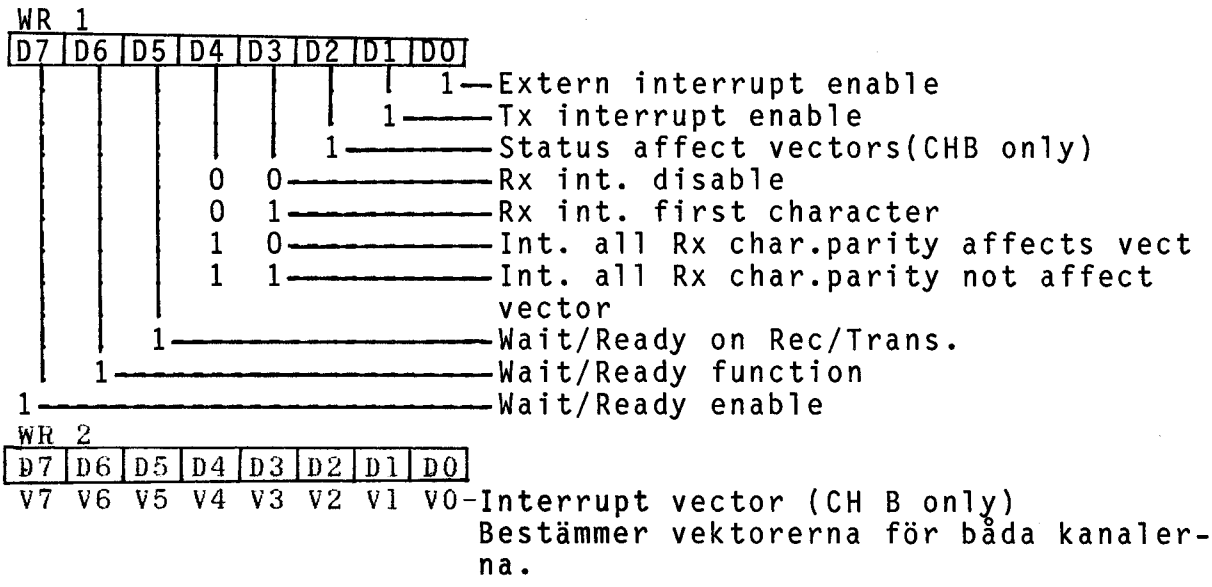
```
10 !Sätt DTR och RTS låg och hög växlande.
20 OUT 33,5 !Välj register 5
30 OUT 33,128+2 !Låg nivå till DTR och RTS (på kretsutg.)
40 OUT 33,0 ! Hög nivå till DTR och RTS (på kretsutg.)
50 GOTO 20
```

OBS. Gör man det här på fel kanal så dyker maskinen (adress 35)
När man programmerar till det här registet förändrar man andra bitvärden som har betydelse för sändningen av data för respektive kanal varför man får tänka sig för om man använder sig av ovanstående.

15.1 Register i SIO2 och DART

En mycket kort beskrivning av registren i SIO2 och DART kretsen. WR är register som man kan skriva till och RR är register man kan läsa.
Efter bit förklaring står en * för de bitar som bara har att göra med synkron överföring.





WR3

D7	D6	D5	D4	D3	D2	D1	D0	
							1	1-Rx enable
						1		Sync charakter load inhibit *
					1			Addres search mode (SDLC) *
				1				Rx CRC enable *
			1					Enter hunt phase *
		1						Auto enables
0	0							Rx 5 bits/character
0	1							Rx 7 bits/character
1	0							Rx 6 bits/character
1	1							Rx 8 bits/character

WR4

D7	D6	D5	D4	D3	D2	D1	D0		
							1	1-Parity Enable	
						1		Even parity	
						0		Odd parity	
				0	0			Sync mode	*
				0	1			1 stop bit/char	
				1	0			1 1/2 stop bit/char	
				1	1			2 stop bit/char	
		0	0					8 bit sync char.	*
		0	1					16 bit sync char.	*
		1	0					SDLC mode (01111110 flag)	*
		1	1					External sync mode	*
0	0							x 1 clock mode	
0	1							x 16 clock mode	
1	0							x 32 clock mode	
1	1							x 64 clock mode	

WR 5

D7	D6	D5	D4	D3	D2	D1		
						1	1-TX CRC Enable	*
					1		RTS (1=RTS pin low)	
				1			SDLC/CRC-16	*
			1				Tx enable	
		1					Send break	
0	0						Tx 5 bits/char	
0	1						Tx 7 bits/char	
1	0						Tx 6 bits/char	
1	0						Tx 8 bits/char	

WR 6

D7	D6	D5	D4	D3	D2	D1	D0
B7	B6	B5	B4	B3	B2	B1	B0-Sync bit *

WR 6

D7	D6	D5	D4	D3	D2	D1	D0
B15	B14	B13	B12	B11	B10	B9	B8 Sync bit *

RR 0

D7	D6	D5	D4	D3	D2	D1	D0	
							1	1-Recieve character available
						1		1——Interrupt pending (Ch A only)
					1			1——Transmitter buffer empty
				1				1——DCD (1=Low input)
			1					1——Sync/RI (1=Low input) Async. mode
		1						1——CTS (1=Low input)
	1							1——Tx underrun/Eom *
								——Break/Abort

RR 1

D7	D6	D5	D4	D3	D2	D1	D0	
							1	1-All sent (async. mode)
				0	0	0		0——I-field *
			1					1——Parity error
		1						1——Recieved overrun error
	1							1——CRC/Framing error
								——End of framing *

RR 2

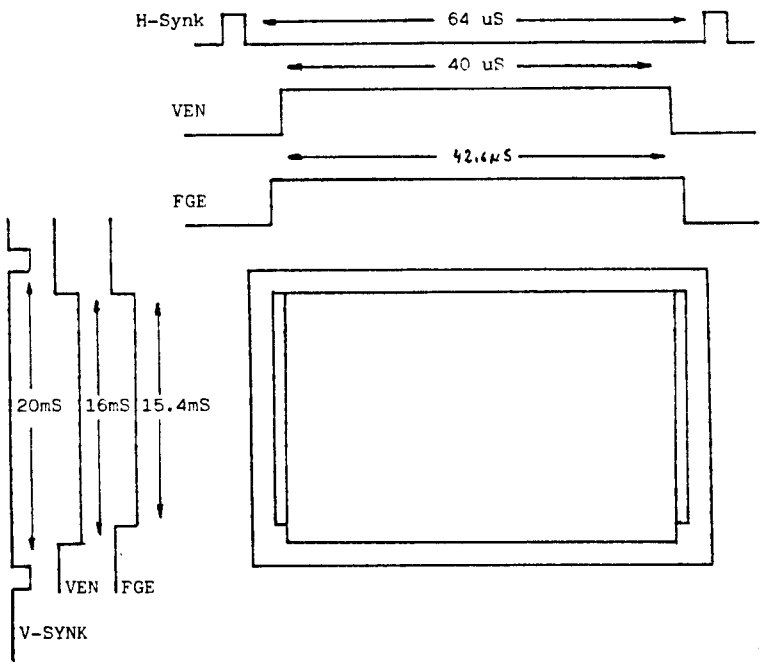
D7	D6	D5	D4	D3	D2	D1	D0	
V7	V6	V5	V4	V3	V2	V1	V0	V0-Interrupt vector
				0	0	0		0----If status affect vector
				0	0	0		0----CH B transmit Buffer emty
				0	0	1		1----CH B external/status change
				0	1	0		0----CH B receive character available
				0	1	1		1----CH B special receive condition
				1	0	0		0----CH A transmit Buffer emty
				1	0	1		1----CH A external/status change
				1	1	0		0----CH A receive character available
				1	1	1		1----CH A special receive condition

16 VU-KORT, TECKENGENERERING

På VU-kortet finns ett bildminne som CPU:n kan skriva och läsa information i, de data som skrivs kommer att visas som alfanumeriska/grafik tecken på den monitor som är ansluten till ABC806.
För att göra det möjligt finns det på VU-kortet elektronik som skapar en videosignal till monitorn med utgångs punkt från data i videominnet.

Signalerna som matas till monitoranslutningen är Y,R,G,B och en synk signal, Y-signalen används vid inkoppling till en monitor som bara har en färg t.ex ABC 815, R,G,B är signaler som används när en färgmonitor är ansluten,
R innehåller röd, B innehåller blå och G innehåller grön färginformation. Synksignalen består av vertikala och horisontella synkpulser.
Signalnivåerna ut ifrån ABC806 till monitorn är TTL nivå vilket gör att monitorn som signalerna kopplas till måste vara anpassad för det på sina signalingångar.

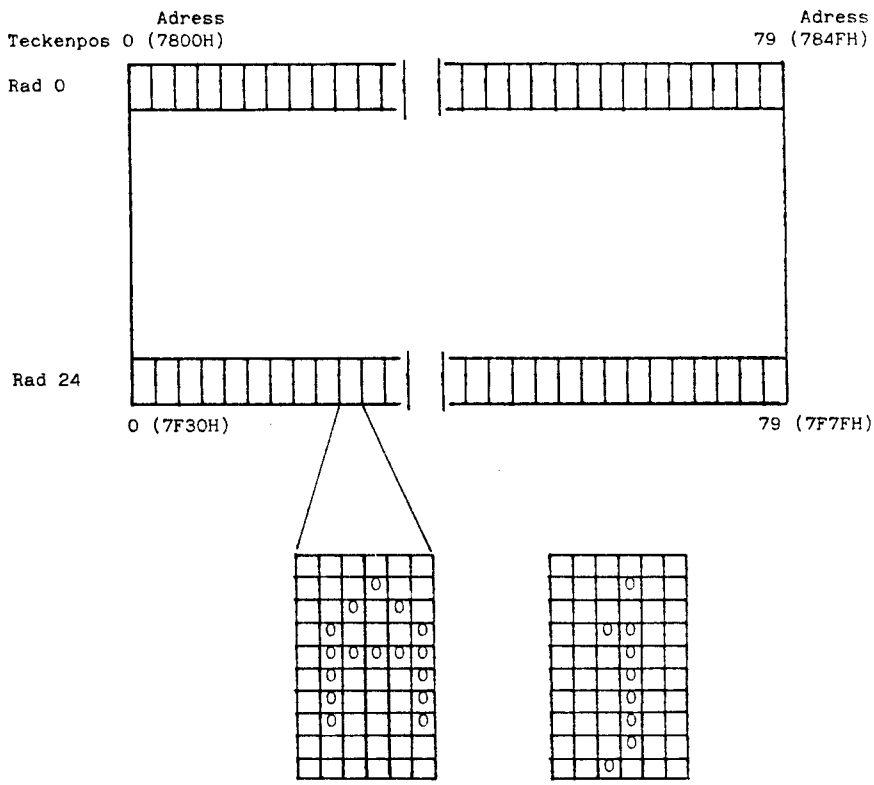
Den videosignal som sänds ut genererar 50 bilder/sek där var bild är uppdelad i 312 linjer, videosignalen ger inget radsprång, vilket innebär en flimmerfri bild.
Horisontalsynkfrekvensen blir med ovanstående 15625Hz/64 uS och vertikalsynkfrekvensen 50Hz/20mS.



V-H synk samt aktiv bildyta.
fig 46.

Den aktiva bildytan som används för tecken/grafik bilden är 40 uS horisontellt och 16 mS vertikalt (250 linjer), medan den bild som genereras från högupplösningsgrafiken har bildytan horisontellt 42.6 uS och vertikalt 15.4 mS (240 linjer). Generering av en högupplösningsgrafikbild beskrivs senare eftersom det är en separat funktion, skilt från den vanliga teckengenereringen.

Bildminnet är ett statiskt RAM-minne på 2kbyte där varje minnesposition motsvarar en teckenposition i en bild.
En bild är uppdelad i 25 rader med 80 tecken på varje rad, vilket ger att 25x80=2000 minnespositioner av 2048 utnyttjas för en bild. Adressområdet för bildminnet är 7800H till 7FFFH, där första minnespositionen motsvarar teckenposition 0 på rad 0 och sista teckenpositionen på rad 24 motsvarar adress 7F7FH.



Bildminne och teckenuppbyggnad.
fig 47.

Ett tecken byggs upp i en punktmatris om 6*10 punkter se fig. där den översta punktraden horisontellt och den första vertikalt alltid är tom för att ge ett avstånd mellan tecknen.

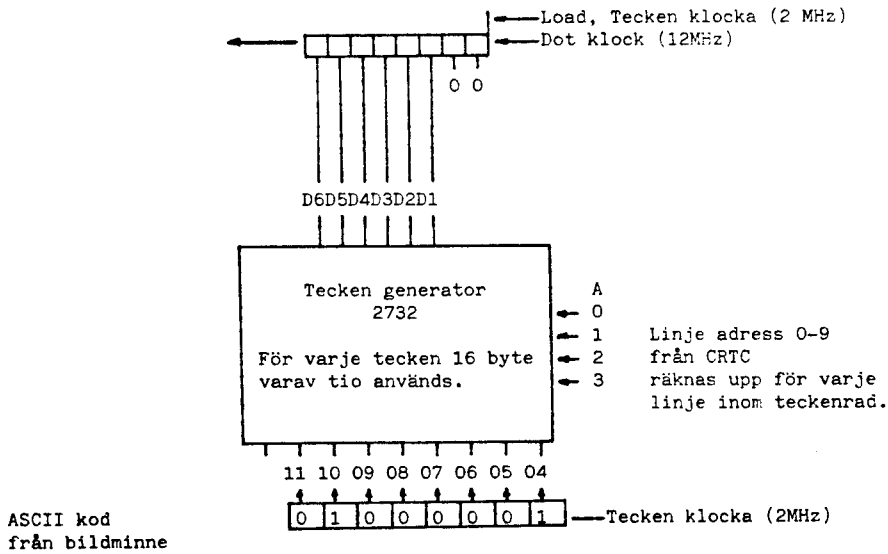
Ovanstående gäller när ABC806 används i 80 teckens mode, går man över i 40 teckens mode får man en bild om 25 rader och 40 tecken/ rad.

I bildminnet kommer då informationen för en rad att ta upp lika mycket av minnet, men man uttnyttjar bara varannan minnes position, vilket ger att rad 0 första teckenpositionen hamnar på 7800H andra teckenpositionen på adress 7802H tredje på 7806H osv. Teckenmatrisen för ett tecken ser likadan ut men får en bredd dubbelt så stor som ett 80 teckens tecken.

En teckenrad på skärmen är uppbyggd av 10 linjer och 480 punkter/ linje, ett tecken tar 6 punkter utmed linjen och med 480 punkter får det plats 80 tecken. Informationen i bildminnet som ska översättas till en videosignal består av data i form av ASCII kod där varje tecken har en egen binär kod, för att kunna skapa en bild av innehållet i bildminnet, måste innhållet i varje minnesposition läsas i tur och ordning och översättas till en videosignal som kan styra ut bildröret i monitorn.

En tecken rad består av tio linjer på bildskärmen.

```
          ! 0 ! 1 ! 2 ! 3      ! 76 ! 77 ! 78 ! 79
Linje 0   ...................../ /.....................
      1   ...o...oooo............./ /.............o...oooo.
      2   ..o.o..o...o.........../ /.............o.o..o...o
      3   .o...o.o...o.........../ /.............o...o.o...o
      4   .ooooo.oooo............/ /.............ooooo.oooo.
      5   .o...o.o...o.........../ /.............o...o.o...o
      6   .o...o.o...o.........../ /.............o...o.o...o
      7   .o...o.oooo............/ /.............o...o.oooo.
      8   ...................../ /.....................
      9   ...................../ /.....................
```



Omkodning från ASCII-kod till videosignal
fig 48.

Under den aktiva tiden på en linje (40 uS) ska 80 teckenpositioner i bildminnet läsas, från bildminnet kommer då en ASCII-kod med 0.5 uS mellanrum, dessa läggs fram till ett teckenprom (2732) och ger adressbitarna A11 till A4.
Till adressingångarna A3-A0 på teckenprommet finns samtidigt en adress som talar om vilken linje inom en teckenerad man befinner sig t.ex. linje 0.

Ut från prommet får man de första 6 punkterna i ett tecken (alla punkter släckta) dessa bitvärden laddas in i ett skiftregister där dom sedan skiftas ut med en klockfrekvens om 12 MHz.
När sex bitar skiftats ut har det gått 0.5 uS och ASCII-koden för nästa teckenposition laddas in i skiftregistret.

När en teckenrad i bildminnet lästs, börjar läsningssekvensen om igen på samma bildadress och ASCII-koden för teckenposition 0 på raden läggs fram till teckenprommet igen, men nu har linje adressen på adressingångarna A3-A0 räknats upp med 1 och nästa 6 punkter som ska bilda tecknet i pos. 0 laddas in i skiftregistret.

För att visa en teckenrad på skärmen avläses motsvarande minnespositioner i bildminnet 10 gånger medan linje adressen till tecken prommet räknas upp från linje 0 till 9 inom teckenraden.

I ABC806 kan tecknen ges olika egenskaper (attribute) t.ex olika färger, bakgrunds färg kan ändras, understrykning, elongate(förstorat), m.m.

För att hantera detta finns på VU-kortet ett attribute-minne som är av samma storlek som bildminnet (2 kbyte), attributeminnet avsöks parallelt med bildminnet vilket gör att det i attributeminnet finns en minnesposition som motsvarar varje teckenposition i bildminnet.

De data som finns i minnet är en kod som för respektive tecken i bildminnet anger den egenskap som tilldelats tecknet, koderna läses ut parallelt med bildminnets ASCII-koder och kopplas fram till en krets som tolkar koden och på olika sätt ger respektive tecken den egenskap som koden anger.

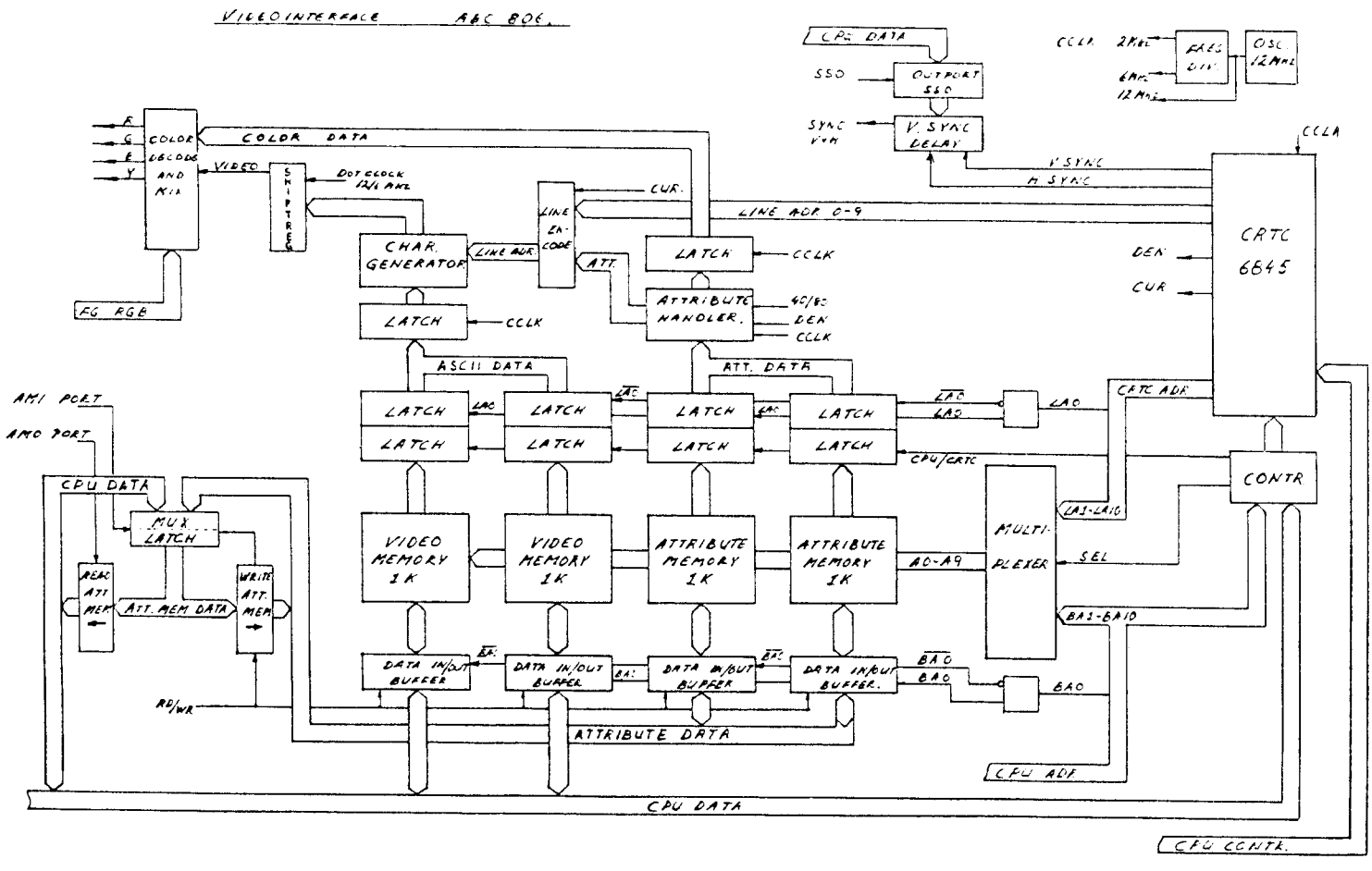
När CPU:n ska läsa eller skriva data i attribute minnet görs det med hjälp av I/O instruktioner.

16.1 Blockschema teckengenerering

Avsökningen av bildminnet, generering av synkpulser, linje adresser 0-9 och en del andra kontrollsignaler görs av en CRTC krets (Cathode Ray Tube Controller) 6845, kretsen är programmerbar och kan programmeras för att ge olika tider för synk och kontrollsignaler. Kretsen måste först programmeras upp av CPU:ns initierinsprogramm innan den kan börja generera kontrollsignaler och starta avsökningen av bildminnet.

CRTC:n lägger avsökningsadressen LA1-LA9 till bildminnet och attributeminnet via en multiplexerkrets. Adressbiten LA0, den minst signifikanta biten adresserar däremot inte minnet direkt utan kopplas via en inverterare och en buffertkrets till låskretsarna på bild och attributeminnets datautgångar.

Bild och attributeminnet (2 kbyte vardera) är uppdelat i fyra 1 kbytesblock och avsökningsadressen LA1 till LA9 kommer att läsa ut en byte ur vardera blocket till de efterföljande låskretsarna (Latch).



Blockschema videointerface
fig 49.

Den första låskretsen på varje datautgång kontrolleras av signalen CPU/CRTC medan de efterföljande kontrolleras av LAO och LAO/.

Låskretsarna som kontrolleras av signalen CPU/CRTC är normalt öppna och släpper fram data till de efterföljande låskretsarna, eftersom dessa styrs av LAO och LAO/ kommer bara två av låskretsarna att öppnas, en för att koppla fram data till teckenpromm:et och den andra för att koppla fram data till attribute avkodaren.

När CRTC kretsen ökar sin avsökningsadress med 1 kommer LAO att växla värde och de två låskretsarna som var öppna kommer att stängas medan de andra två kommer att öppnas och lägga fram sina data till teckenprom och attributehanteraren.

Ur bildminnet och attributeminnet läser man alltså ut information för två tecken samtidigt vilket gör att intervallet mellan varje läsning från CRTC-kretsen blir ca 1 uS, vilket ger tid för CPU:n att adressera en minnesposition i bildminnet och attributeminnet och läsa eller skriva data.

ASCII-data från bildminnet kopplas via ytterligare en låskrets fram till teckenprommet och från teckenprommets datautgångar laddas det efterföljande skiftregistret med ett bitmönster som skiftas ut som videosignal till den efterföljande färgavkodning/mixer kretsen. Vilket bitmönster som laddar skiftregistret bestämms av ASCII-koden och den linjeadress som adresserar teckenprommet.

Data från attributeminnet har samtidigt kopplats fram till kretsen som avkodar attribute data, där har data avkodats och som utsignal ger den bakgrundsfärg och färg på tecknet, dessa data kopplas fram till färgavkodning/mixerkretsen och kommer där att tillsammans med videosignalen att skapa RGB och Y signalen som sedan kopplas till bildskärmen.

Från attributehanteraren finns ytterligare utsignaler för att ge egenskaperna understrykning och förstorade tecken, detta görs bl.a. genom omkodning av linjeadresserna från CRTC-kretsen.

Klocksignaler som behövs för att kontrollera in och utläsningsförlopp genereras från en 12 MHz kristalloscillator och en frekvensdelare.

För teckengenerering skapas CCLK med frekvensen 2 MHz som ger utläsningshastigeheten för data från bild och attribut minne, CCLK används därför på ett flertal ställen för att klocka data.

12 MHz klockan används bl.a. som "Dot clock" för att skifta ut videosignalen från skiftregistret.

CRTC kretsen genererar också signalerna DEN (Display Enable) vilket är den signal som innhåller tidsfönstren för den aktivabildytan horisontellt och vertikalt, CUR (Cursor) är en signal som används för att generera en markör på bildskärmen.
Vertikal och Horisontal synkpulserna från CRTC kretsen kopplas till ett block där man skapar en kombinerad V+H syncpuls. I blocket kan man också fördröja vertikalpulserna ett antal linjeperioder, fördröjningen styrs via en Outport som benämns SSO, funktionen används när man har optionen Soft Scroll i ABC806.

För att programmera CRTC kretsen använder man sig av I/O instruktioner, därför finns det ett block som avkodar dessa och genererar kontrollsignaler till CRTC kretsen för in och utläsning av data.

När CPU adresserar videominnet kommer via kontrollbussen signalerna CPU/CRTC och SEL att aktiveras, vilket gör att de data som finns på utgångarna från bild och attributeminne lagras i de första låskretsarna, multiplexern växlas sedan av signalen SEL och kopplar fram adressbitarna BA1-BA10 från CPU:n till minnesblocken. Adressbit BA0 adresserar inte minnet direkt utan kopplas via en inverterare och buffert fram till de dubbelriktade databuffert kretsarna som är anslutna till dataingångarna på minneskretsarna.
En buffert till bildminnet och en till attribute minnet kommer att aktiveras, vilket block i respektive minne som adresseras bestämms av värdet på BA0.
Riktningen på bufferten bestämms av om en läsning ellere skrivning ska göras och kontrolleras av RD/WR signalen till buffertkretsarna.

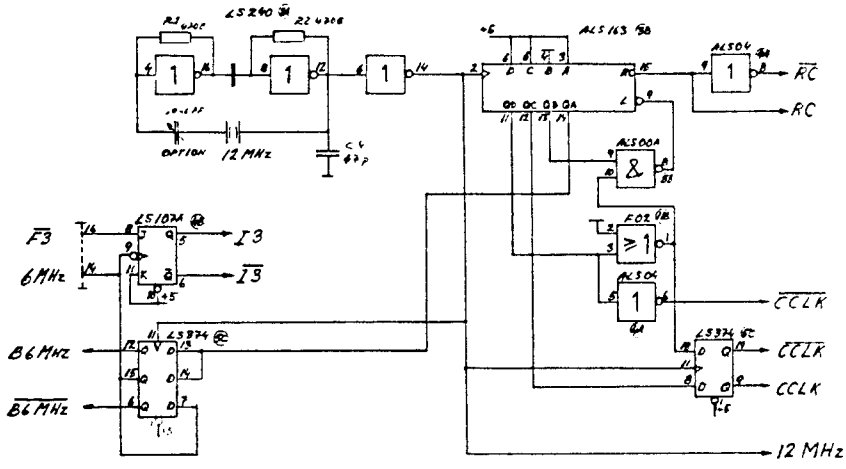
Data som ska skrivas i bildminnet läses in från CPU:ns databuss medan data som ska skrivas i attributminnet läses från en registerkrets, data till registerkretsen kan man skriva via en utport. Utporten som används för att sätta attributdata i registret benämns AMI (Attribute Memory In).
De data som finns på ingångarna på registerkretsen kommer att skrivas i attributeminnet på motsvarande position som CPU:n adresserar och skriver till i bildminnet.
Vid en läsning av data i bildminnet adresseras bild och attributeminnet på samma sätt som vid skrivning.
Data från bildminnet läggs ut på databussen medan data från motsvarande minnesposition i attributeminnet läggs fram till AMI porten via en multiplexer krets. Data från attribute minnet kan nu läsas via inporten som benämns AMO (Attribute Memory Out).

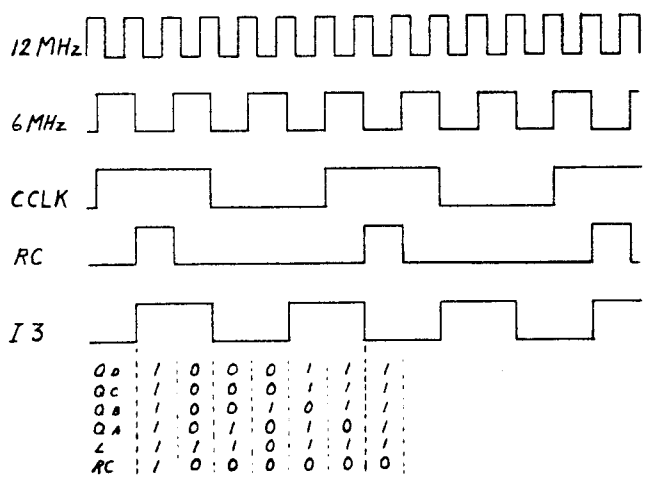
Det här gör att vid skrivning i bildminnet, kommer de data som finns i AMI porten att skrivas in som attributedata för för den teckenpositionen i attributeminnet.

Vid en läsning i bildminnet kommer attributdata för den lästa teckenpositionen att läggas i AMI porten och kopieras till det ställe man skriver på nästa gång i bildminnet.

16.2 Klocksignaler

Klocksignalerna för hela system ABC806 kommer från en kristalloscillator på 12 MHz bestående av en kristall och två inverterare, 12 MHz signalen buffras i ytterligare en inverterare innan den läggs fram till klockingången på en räknare, räknaren kommer att i takt med klockpulserna räkna upp värdet på utgångarna Qa-Qd. RC utgången från räknaren blir hög var gång den har det binära värdet 15 på Qa-Qd utgångarna, vid nästa klockpuls kommer utgångarna att få värdet 0, efter ytterligare två klockpulser kommer utgången Qb att bli hög och aktivera signalen L till räknaren, det binära värde som finns på ingångarna A-D kommer då att ge Qa-Qd utgången värdet 13 och uppräkningen sker från det värdet tills RC avges igen.





Klocksignaler och tidsdiagram
fig 50.

Klocksignaler som skapas är följande:

12 MHz som är den frekvens som punkterna i videosignalen skiftas ut med, signalen används också för att synkroniser olika tidsförlopp.

6 MHz som kopplas till PU-kortet och används för att generera 3MHz systemklocka.

B6MHz och B6MHz/ (B står för buffrad) som används på VU kortet i olika sammanhang.

CCLK (Character clock) är den frekvens som bestämmer hastigheten på utläsningen av tecknen från bildminnet m.m.

I3 och I3/ är en 3 MHz signal som anger fasläget på systemklockan till CPU:n,I3 kommer att få samma fasläge som CPU:ns klocksignal, signalen används på VU-kortet för att synkronisera CPU:n och CRTC kretsen.

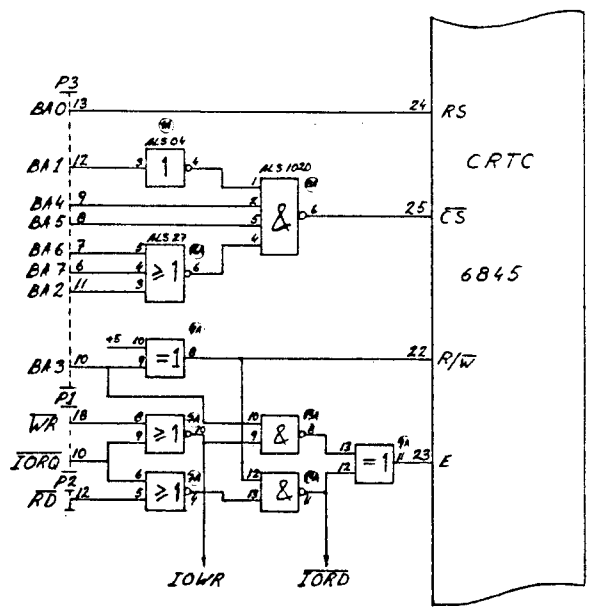
16.3 Adressering CRTC kretsen

CRTC kretsen adresseras med följande I/O adresserna:

Välj register =Adress 56 (00111000B,38H)
Skriv till register=Adress 57 (00111001B,39H)
Läs från register =Adress 49 (00110001B,31H)

Avkodningen visas i figuren, BA0 från CPU:n kontrollerar signalen RS till CRTC:n och anger, om den är låg, att de data som finns på databussen ska läsas in som en registeradress. BA1,BA2 och BA4 till BA7 avkodas i tre grindar och ger CS (Chip Selekt) till CRTC:n.
BA3 kontrollerar R/W ingången och anger skrivning eller läsning till kretsen.

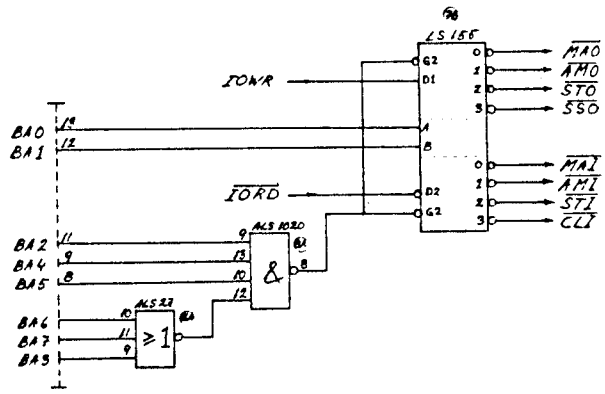
Enable till kretsen fås genom att avkoda IORQ, WR/ och RD/ signalerna från CPU:n.
Här bildas också två andra signaler som används för att avkoda några andra I/O addresser på VU-kortet. IORD/ är en kombination av signalerna IORQ/*RD/*BA3/ och används för att genera INP strobar.
IOWR är en kombination av signalerna IORQ/*WR och används för att generera OUT strobar.



I/O adressering CRTC-kretsen
fig 51.

16.4 Övrig I/O avkodning på VU-kort

I en dubbel 1 av 4 avkodare avkodas I/O adresserna från CPU:n för att ge I/O strobar till ett antal I/O portar på VU-kortet.



I/O avkodning på VU-kortet.
fig 52.

Adressbitarna BA0 och BA1 är kopplade till avkodarens adressingångar och väljer vilken av utgångarna som ska kopplas till respektive ingång, adress ingångarna kontrollerar båda avkodarna.
BA2-BA7 avkodas och aktiverar gate ingången till båda avkodarna så att värdet på D-ingången till respektive avkodare kommer att kopplas till den utgång som selektats med värdet på A och B ingången. Insignalerna till D1 och D2 ingången är IOWR och IORD/,

IOWR är aktiv när CPU gör en I/O skrivning och är en kombination av IORQ/*WR/. När en I/O skrivning görs på en adress som aktiverar gate ingångarna på kretsen, kommer signalen IOWR att aktivera någon av de fyra utgångarna som har IOWR som insignal, vilken bestäms av värdet på BA0 och BA1. Signalen IORD/ är i det här läget hög vilket gör att ingen av den andra avkodarens utsignaler blir aktiv.

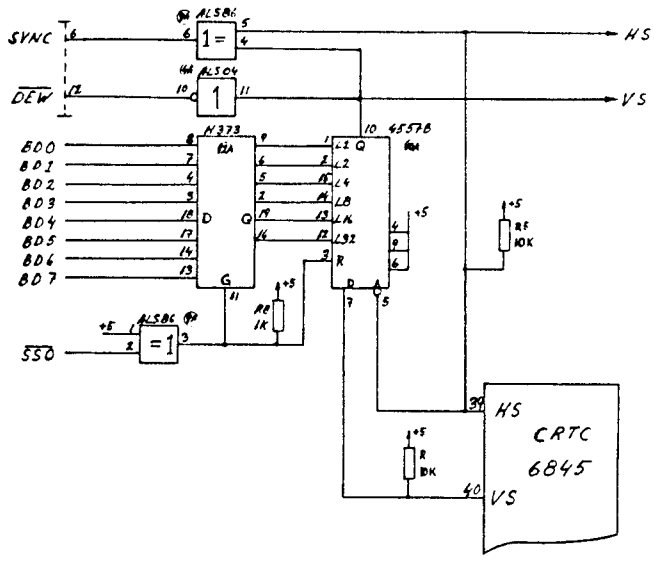
När en I/O read görs med en adress som aktiverar gateingångarna så kommer signalen IORD/ att läggas låg och aktivera någon av utgångarna som har IORD/ som insignal, signalen IOWR är nu låg vilket gör att ingen utsignal aktiveras på den avkodaren.

På den övre avkodare kommer man att få OUT strobar och på den undre INP strobar. I/O adresser som avkodas är följande:

```
MAO/ (write,Memory map. reg. file)=Adrses 52 (00110100B,34H)
AMO/ (write,Attribute Memory Out )=Adress 53 (00110101B,35H)
STO/ (write,Special              )=Adress 54 (00110110B,36H)
SSO/ (write,Sync Delay           )=Adress 55 (00110111B,37H)

MAI/ (read,Memory map. reg. file )=Adress 52 (00110100B,34H)
AMI/ (read,Attribute memory in   )=Adress 53 (00110101B,35H)
STI/ (read,                      )=Adress 54 (00110110B,36H)
CLI/ (read, fgctl prom and clock )=Adress 55 (00110111B,37H)
```

16.5 Vertikal och horisontal synk.

De vertikala synkpulserna från CRTC kretsen läggs fram till dataingången på ett skiftregister som fungerar som en fördröjning för vertikalsynkpulsen, födröjningstiden programmeras in på ingångarna L1-L32 som ett binärt tal som anger hur många klockpulser på klockingången som behövs innan värdet på D-ingången kommer ut på Q-Utgången, eftersom klockpulserna är horisontala synkpulser kan man här fördröja vertikalsynken ett antal linjer.
Värdet för fördröjningen skrivs till D-registret med I/O adress 55, vilken avkodas på VU-kortet och aktiverar signalen SSO/ som via XOR grinden läser in data från bussen till registret.
Porten används när man gör soft scroll på bildskärmen vilket innebär att bilden förflyttas linje för linje istället för en hel teckenrad som är det vanliga.
Fördröjningen sätts till 10 vid initiering.



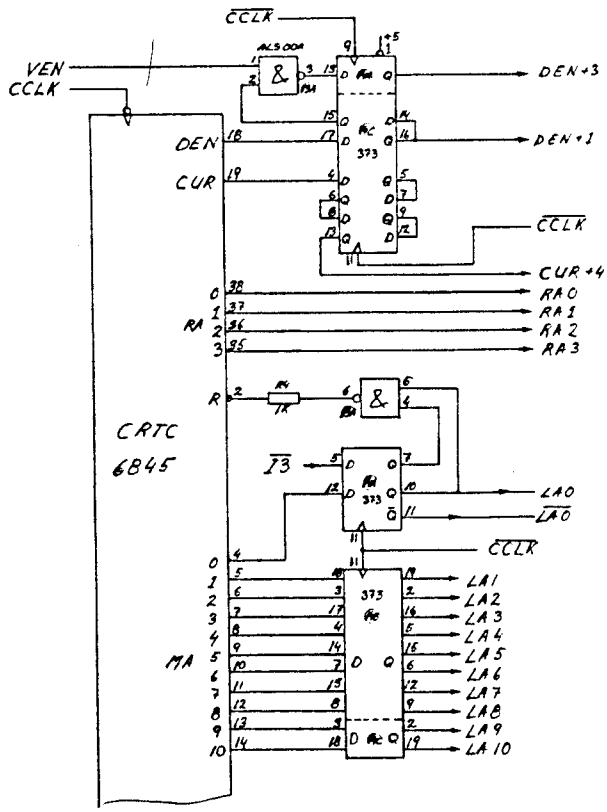
Vertikal och horisontal synk.
fig 53.

16.6 Övriga signaler från CRTC kretsen

CRTC kretsen kommer efter initiering att starta avsökningen av bild/attribute minnet genom att på adressutgångarna MA0-MA10 lägga ut adresser till bild/attribute minnet i takt med CCLK på klockingången till CRTC kretsen.
MA1-MA10 kopplas till en låskrets som klockas av CCKL och benäms sedan LA1-LA10, MA0 kopplas till en annan låskrets där man tar ut LA0 och LA0/ dessa adresserar sedan låskretsarna på bild och attribute minnets datautgångar.

För att synkronisera CPU:n (3MHz) med CRTC kretsen (2MHz) jämförs fasläget på signalen I3/ och LA0 i nand grinden i pos 13A, den föregående låskretsen känner av fasläget på de båda signalerna på den positiva flanken av CCLK.

Om I3/ är hög samtidigt som LA0 är hög kommer reseting-ången på CRTC kretsen att bli aktiv, vilket gör att avsökningsadressen ut från CRTC kretsen blir noll, reset signalen blir inaktiv så fort som I3/ och LA0 har motsatt värde när låskretsen klockas. På det här sättet får man att avsökningen av bild och attrbute minne startar i ett visst fasläge i förhållande till CPU:n vilket är nödvändigt för att CPU:n ska hinna läsa eller skriva i minnet utan att störa avsökningen.



CRTC signaler
fig 54.

RA0-RA3 är linjeadresserna som talar om för teckenprommet vilken linje inom teckenraden man befinner sig på.

DEN (Display Enable) är signalen som innhåller det horisontala och vertikala tidsfönstret som talar om var den aktiva bildytan ska placeras i video signalen. Signalen kopplas runt i ett antal D-vippor för att fördröja signalen ett antal teckenperioder för att den ska sammanfalla med aktuellt tecken i videosignalen. DEN+2 grindas med signalen VEN från högupplösningsgrafiken för att synkronisera bildstart av de bägge bilderna.

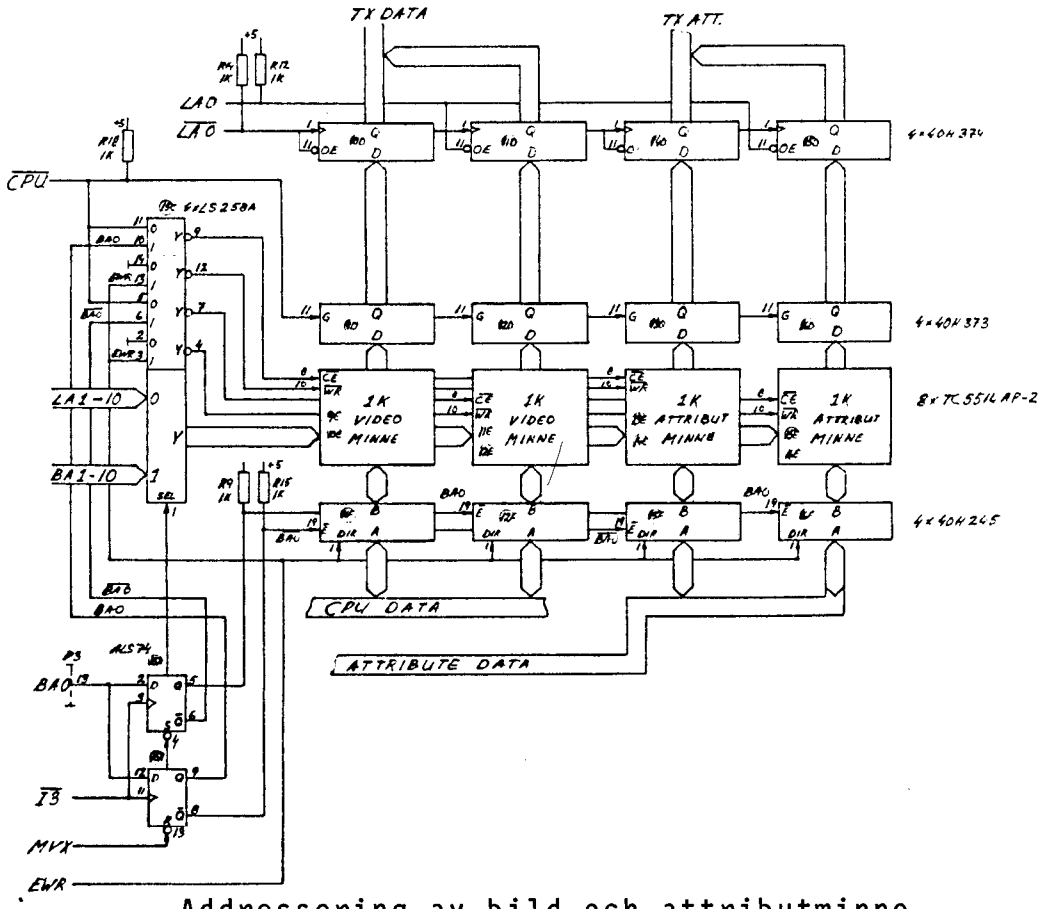
CUR signalen används för att generera en markör på skärmen, i CRTC kretsen finns två register som man kan skriva en adress i, när den adressen är samma som den som finns på MA0-MA10 kommer signalen att bli aktiv under den teckenperioden. Signalen fördröjs på samma sätt som DEN för att sammanfalla med aktuellt tecken i videosignalen.

16.7 Adressering av bild och attributeminne

Adressbitarna LA1-LA10 från CRTC kretsen kopplas fram till bild och attributeminnet via en multiplexerkrets vars selektingång styrs av signalen MVX

Minneskretsarnas CE/ hålls i avsökningsläge aktiva av signalen CPU/ via multiplexern, medan WR ingången hålls hög genom att ingångarna på multiplexern som kontrollerar WR är kopplad till signaljord.

För varje avsökningsadress kommer de fyra minnes blocken att lägga ut varsin byte av data till de efterföljande låskretsarna som hålls öppna av signalen CPU/ så att data läggs fram till nästa uppsättning låskretsar. Dessa låskretsars OE/ ingång kontrolleras av LA0 och LA0/ på så sätt att när LA0 är låg kommer en ena av låskretsarna efter bildminnet att släppa fram 1 byte av de 2 som lästs från bildminnet, den andra låskretsen hålls stängd av LA0/, när avsökningsadressen sedan räknas upp kommet LA0/ att öppna den låskrets som den kontrollerar och LA0 stänger den andra och den andra byten från bildminnet kopplas vidare.
Samma sak sker parallelt i de två låskretsar som lägger fram data från attributeminnet.
Data ut från bildminnet kopplas sedan vidare till tecken prommet och data från attributeminnet läggs fram till attributkretsen.



Addressering av bild och attributminne
fig 55.

Eftersom minnet adresseras med LA1 till LA10 och låskretsarna med LA0 och adresserna räknas upp med 2 MHz klocka kommer adressen till minnet att räknas upp med 1 uS tidsavstånd medan LA0 växlar med 0.5 uS mellanrum. Detta gör att CPU:n som behöver ca 0.66 uS på sig för att läsa eller skriva i bildminnet hinner att göra det utan att störa den avsökningen som kontinuerligt sker av bildminnet.

När CPU:n ska skriva eller läsa data i bildminnet blir signalen CPU/ aktivt låg och lägger Gate ingången till låskretsarna efter bild och attributeminne låga så att de data som fanns på ingången till kretsarna lagras på respektive utgång, samtidigt har signalen MVX som är kopplad till Reset och Set ingången på två D-vipporna samt till selektingången på multiplexern blivit hög och multiplexern kommer att koppla fram CPU:ns adressbitar BA1-BA10 till minnesblocken.

På de två D-vipporna hålls i avsökningsläge den enas Res ingång och den andras Set ingång låg av signalen MVX, vilket gör att Enable ingången till de dubbelriktade databuffert kretsarna ligger på hög nivå och håller kretsarna inaktiva.
De andra två utgångarna från D-vipporna (Q=BA0, Q/=BA0/) går upp till multiplexern och kontrollerar vid CPU adressering CE/ till minnesblocken.
När MVX blir aktiv kommer vipporna att läsa av läget på BA0 när signalen I3/ (inversen av CPU klockan) ger en positiv flank, vilket sker en halv klockperiod efter det att MVX blivit aktiv.

Signalerna ut från vipporna blir BAO och BAO/ och de som går till multiplexern kommer att göra CE/ på ett minnesblock vardera i bild och attributeminnet och de två som går till databufferterna kommer att aktivera bufferten för motsvarande minnesblock.

Signalen EWR (Early Write) är fördröjd Memory Read signal som är aktivt låg när CPU:n gör en läsning i bildminnet. Signalen kontrollerar WR ingången på minnesblocken samt direktion på buffert kretsarna till minnesblockens data in/ut ledningar. När CPU gör en läsning i bildminnet kommer data från bildminnet att läggas ut på CPU:ns data buss medan data från motsvarande minnesposition i attributeminnet läggs ut på attribute data bussen och läses in till ett register som kan läsas med en INP instruktion.

```
Tex 10 Asckod%=PEEK(30720) : Attkod%=INP(53) 20
    ;Asckod%:Attkod%
```

När CPU:n gör en skrivning kommer data till bildminnet att läsas in från CPU:ns databuss medan data till attributeminnet läses från attributedatabussen, data på attributedata bussen kommer från ett register och innehåller attributet för sist lästa tecken i bildminnet eller de data som man skrivit med en OUT instruktion till registret.

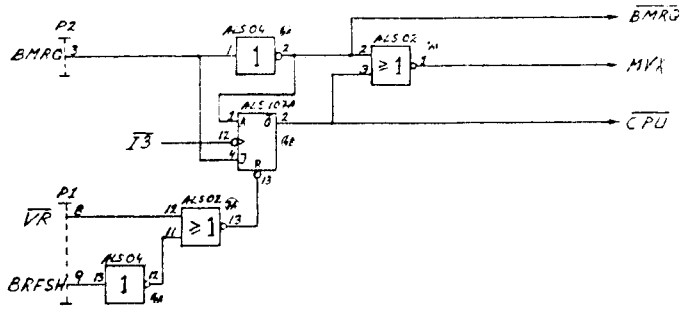
```
Ex 10 Asckod%=PEEK(30720) : POKE 30721%,Asckod%
```

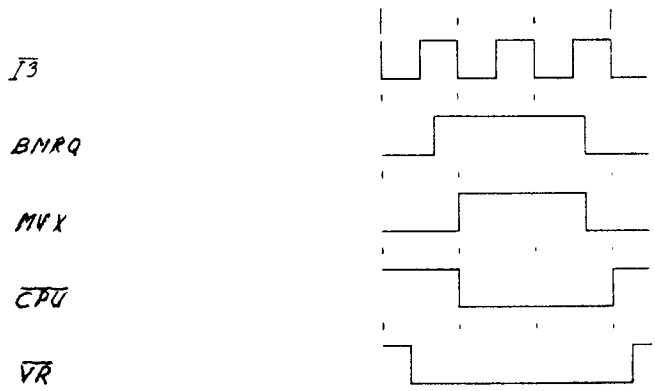
Gör att tecknet i minnesposition 30720 kopieras till minnesposition 30721 och samtidigt kommer attributedata att kopieras från minnesposition 30720 till 30721.

```
Ex 10  Out 53,Attkod% : POKE 30720,Asckod%
```

Gör att tecknet i minnesposition 30720 får det attribute som angetts i Attkod%.

Signalen CPU/ och MVX genereras av signalen VR/ tillsammans med BMRQ.
När VR/ signalen aktiveras från PU-kortet indikerar det att CPU:n vill läsa eller skriva i bildminnet.
VR/ grindas med RFSH/ signalen i en OR grind för att inte generera en läsning under refresh perioden, när VR/ blir aktiv kommer R ingången på JK vippan att bli hög och signalen BMRQ kommer att klockas av I3/ på den negativa flanken och aktivera signalen CPU/ som i sin tur via NOR grinden aktiverar signalen MVX.
BMRQ är inversen av CPU:ns MRQ signal, signalerna CPU/ och MVX fördröjs och synkroniseras på det här viset för att CPU:n ska kunna utföra en läsning/skrivning i bildminnet utan att störa den avsökning som CRTC kretsen gör.





Tidsdiagram för CPU läs/skriv.
fig 56.

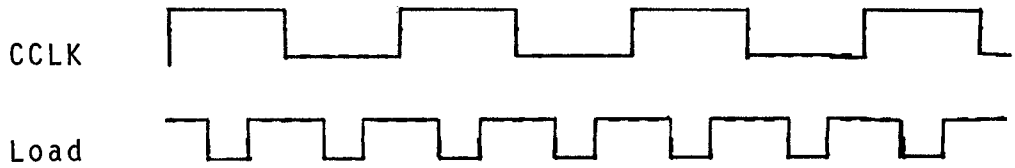
16.8 Teckenggenerering och attributeavkodning

ASCII-koderna från bildminnet kopplas via två stycken låskretsar som klockas av CCLK fram till teckenprommets adressingångar A11-A4.
Via en promkrets finns också en linje adress kopplad till teckenprommets adressingångar A3-A0, linjeadressen värde är aktuell linje inom en teckenrad.

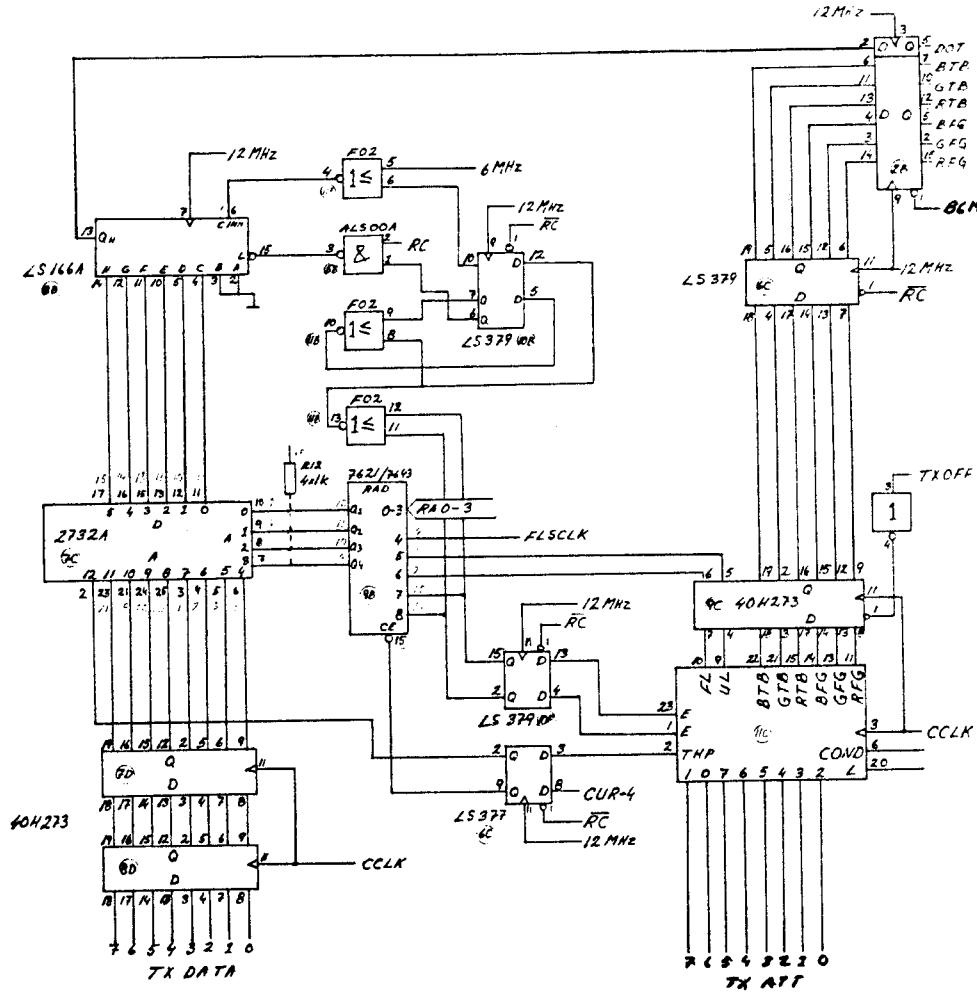
Ut från teckenprommet får man bitmönstret för aktuell linje och för det tecken som ASCII koden adresserar.

Bitmönstret laddas i ett skiftregister när Load signalen blir aktiv låg, och skiftas sedan ut i serieform på Qh utgången i takt med klocksignalen 12 MHz.

Load ingången styrs av signalen RC som har samma frekvens som CCLK vilket gör att ett bitmönster laddas med 0.5 uS mellanrum.



Promkretsen (RAD) som linjeadresserna till teckenprommet kommer från adresseras av linjeadresserna RA0-RA3 från CRTC kretsen samt av signaler från attribute hanteraren, om inget attribute har angetts för ett tecken kommer linjeadresserna ut från RAD prommet att överenstämma med de från CRTC kretsen.



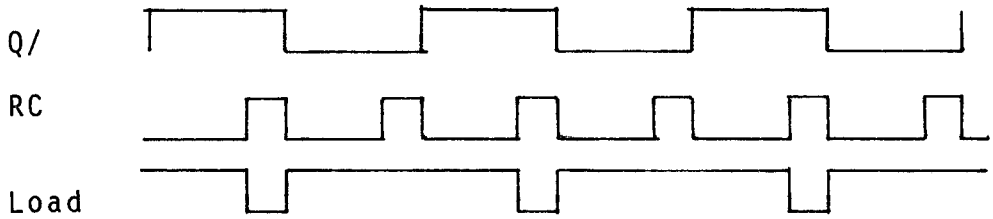
Teckengenerering.
fig 57.

Attribute data från attributeminnet läses parallelt med ASCII koden från bildminnet fram till attributehanterar kretsen som ska avkoda dessa data och ge tecknet olika egenskaper.

Attributekretsen har som insignaler DEN+3 som innehåller tidsfönstren för den aktiva bildytan, 40/80 signalen som väljer 40 eller 80 per rad och signalen CCLK för att klocka ut data i takt med tecknen från teckenprommet.

Signalerna ut från kretsen är BTB,GTB,RTB som anger en teckenpositions bakgrundsfärg, BTF,GTF,RTF som anger färgen på tecknet, dessa signaler kopplas via en låskrets som klockas av CCLK, (vilket ger fördröjning en teckenperiod), fram till ytterligare en låskrets som klockas med 12 MHz (Dot klock) och Enablas av 6MHz, för att sedan tillsammans med signalen från skiftregistret kopplas till en krets där signalerna R,G,B,Y skapas.

När kretsen detekterar en kod för ELongate (förstorade tecken) kommer utgångarna E5 och E6 att bli höga (normalläge låga) vilket gör att utgången på NOR-grinden i pos 11B blir låg, grindens utgången kopplas fram till ytterligare en NOR-grind som öppnas och gör att den understa D-vippan börjar "Togla" genom att den klockas med RC, frekvensen blir samma som RC men pulserna blir symmetriska.
Q/ utgången från vippan kopplas till nand-grinden som ger Load till skiftregistret och kommer att ta bort varannan Load puls eftersom det nu bara ska bli 40 tecken på en rad.



Den andra vippan i pos 10B har som insignal samma signal som startade den undre vippan, när den signalen blir låg och har klockats ut på Q utgången kommer den att öppna den efterföljande or-grinden och släppa fram en 6MHz signal till skiftregistrets CINH (Clock Inhibit) ingång vilket gör att varannan klockpuls till skiftregistret kommer att vara utan verkan.

Det som nu beskrivits gör att skiftregistret laddas med ett bitmönster från teckenprommet med 1 uS mellan rum och att bitarna som bildar videosignal skiftas ut med 6 MHz klocka.
En teckenposition på bildskärmen upptar nu två tidigare teckenpositioner därför laddas också skiftregistret bara med varannat bitmönster från teckenprommet.

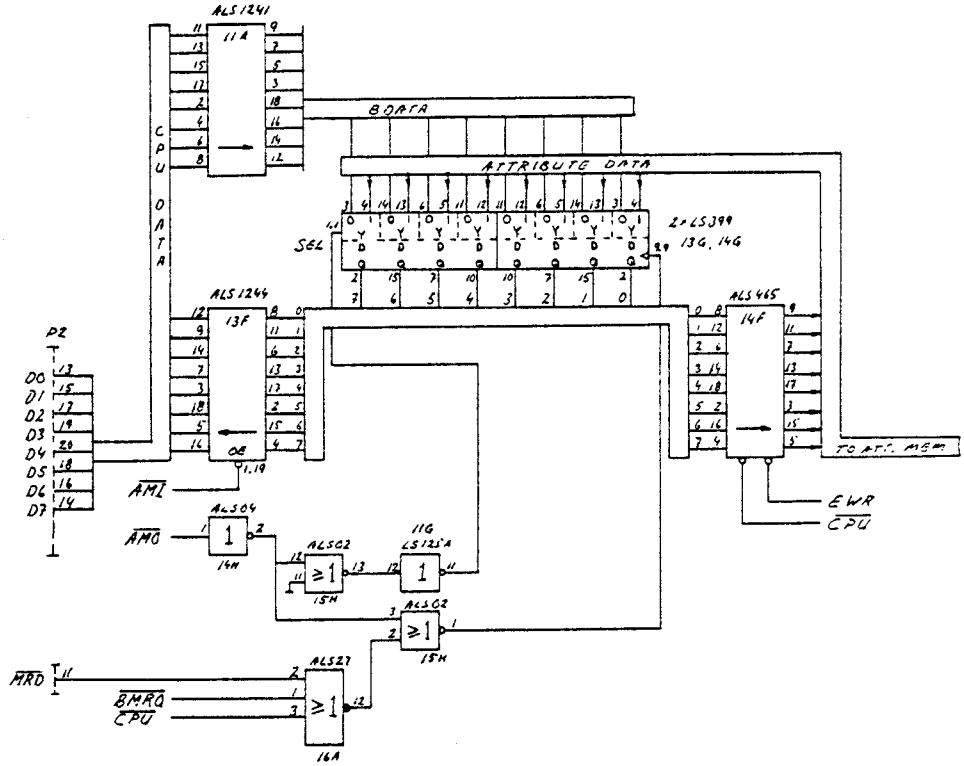
Eftersom avsökningen av bildminnet sker med samma takt som tidigare kommer varannan teckenposition i bildminnet inte att visas på skärmen, vilket också ger att vid skrivning i bildminnet i 40 teckens mode får man hoppa över varannan minnesposition.

Signalerna E5 och E6 fråm attribute kretsen påverkar också adresserna till RAD prommet men när man använder elongate kommer fortfarande linje adresserna till teckenprommet att vara samma som de som kommer från CRTC kretsen.

För attributet dubbel texthöjd kommer ett tecken att delas upp på två rader, på den övre raden ska den översta delen av ett tecken visas och på raden under ska den undre delen av tecknet visas.
För att få dubbel texthöjd får man skriva två attributekoder och samma ASCII-tecken i två teckenpositioner, i teckenpositionen på rad 1 ska attributekoden för övre teckendelen skrivas och i teckenpositionen under på rad 2 ska attributekoden för undre teckendelen skrivas. När attributekretsen detekterar koden för övre teckendelen kommer den att lägga E5 hög och E6 låg, vilket gör att samma sak händer som vid koden elongate, men nu kommer RAD prommet att koda om CRTC kretsens linje adresser så att linjeadress 0 läggs fram till teckenprommet för linje 0 och 1, adress 1 för linje 2-3 osv för tio linjer, när koden för undre tecken kommer på rad 2 blir E6 hög och E5 låg vilket gör att man får elongate och RAD prommet kodar nu om linje 0 och 1 till linjeadress 5, linje 2,3 till linjeadress 6 osv för tio linjer, på bildskärmen ser man nu ett tecken med dubbel höjd och elongate.

16.8.1 Skrivning och läsning av attribute data

När man skriver eller läser i attributeminnet använder man sig av två sticken I/O portar som kallas AMI (Attribute Memory In) och AMO (Attribute Memory Out).
När CPU:n adresserar bildminnet och läser en minnesposition kommer data från motsvarande minnesposition i attributeminnet att finnas på attributedata bussen, bussen är kopplad till kretsen LS399 som är en kombinerad multiplexer och låskrets.
Selectingången ligger normalt hög och anslutar data från attribute-minnet till låskretsen, låskretsen klockas från en nangrind vars utgång aktiveras av signalerna MRD/*BMRQ/*CPU/ eller AMI stroben. När CPU:n läser i bildminnet kommer signalerna MRD/*BMRQ/*CPU/ att bli aktiva vilket gör att klocksignalen blir låg, när signalen sedan går hög klockas data från attributedata bussen in till låskretsen. Om man nu vill läsa av data från låskretsen som innehåller attributedata läser man bara AMI porten.
AMI porten öppnas av AMI signalen som är en avkodat inadress 53.



Skrivning eller läsning av data i attributeminne.
fig 58.

Write registret som också har attributedata som insignal används för att skriva data till attributeminnet, när CPU:n addresserar en minnesposition i bildminnet för skrivning aktiveras signalerna CPU/och EWR och lägger ut skrivregistrets data till attributeminnet, dessa data kommer att skrivas i attributeminnet på samma position som CPU:n addresserade i bildminnet. Data i det här fallet var attributedata från sist lästa minnesposition i bildminnet.
Om man vill byta attributedata skriver man attributedata till AMO porten, när signalen AMO aktiveras (avkodad Out adress 53) kommer Selectsignalen till multiplexern att läggas låg och lägga fram CPU:ns databuss till AMO porten, klockingången till AMO porten läser sedan in data till låskretsen när AMO signalen går hög (positiva flanken). Nästa skrivning i bildminnet skriver data via skrivregistret till attributeminnet.

17 HÖGUPPLÖSNINGSGRAFIK

På VU-kortet finns en funktion som genererar en grafikbild med utgångspunkt från data som finns i ett 128 KByte stort RAM-minne. En grafikbild tar upp 30Kbytes i minnet, vilken del av ramminnet som ska generera grafikbild kan väljas i block om 32Kbyte.

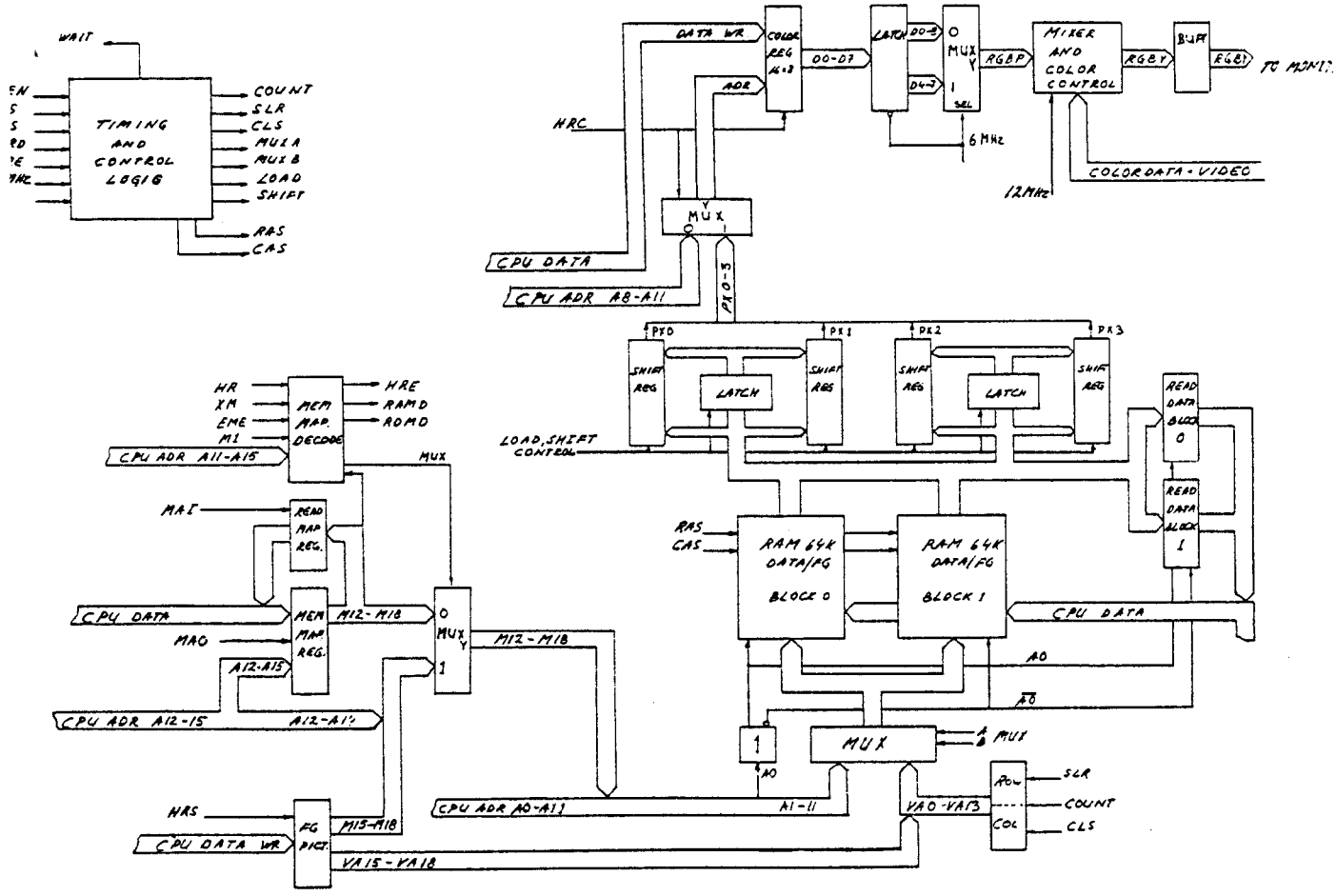
För att generera en grafikbild måste RAM-minnet avsökas kontinuerligt i det 30K område som ska generera bilden. Data i minnet innhåller information om max 512x240 grafikpunkter där varje punkt kan ha färgnr 0-3. Den informationen måste läsas ut från minnet och konverteras så att det på bildskärmen på rätt position tänds en punkt med färg enligt färgnr.

När man använder grafiken i 256x240 mode får man ett mindre antal punkter men varje punkt kan ha ett färgnr 0-15.

I 256x240 grafik mode motsvaras en grafikpunkt på skärmen av två "dottar" i 512x240 grafik mode.

17.1 Blockschemabeskrivning

RAM minnet på 128 Kbyte är uppdelat i två block om vardera 64 Kbyte minnet adresseras via en multiplexerkrets antingen av CPU:n eller av en räknare, räknaren är den som sköter om den kontinuerliga avsökningen av minnet, så det är bara när CPU:n vill läsa eller skriva i minnet som multiplexer kretsen kopplar CPU:ns adresser till minnet.



Blockschema HR-grafik och extra minne.
fig 59.

När man använder ramminnet som grafikminne är det organiserat som ett 64K minne med ordlängden 16 bit. Räknaren som avsöker den del av minnet som innehåller grafikdata är egentligen två stycken, den ena ger adressen för linjen som avsöks och den andra ger kolumnposition utmed linjen.

Adresserna från räknaren kopplas via multiplexern fram till de två minnesblocken och adresserar med samma adress båda blocken och ett ord om 16 bitar läses ut från minnet.
Ordet läggs fram till de båda låskretsarna som ligger mellan de fyra skiftregistren och läses in till dessa, räknaren stegar under tiden upp adressen med ett och nästa ord läggs fram till låskretsarna, nu görs en Load på skiftregistren så att dessa laddas med de två orden som lästs ut från grafikminnet, från skiftregistret skiftas bitarna sedan ut som ett färgvärde för två grafikpunkter i grafikmode 512x240, i grafikmode 256x240 motsvarar det värdet för en punkt (2 dot).

Informationen ligger lagrat på följande vis i de data som lästs ut från grafikminnet.

512x240
256x240
Färgvärde
Ett 16 bits ord från grafik minnet.

Dot 0		Dot 1		Dot 2		Dot 3		Dot 4		Dot 5		Dot 6		Dot 7	
Dot 0				Dot 1				Dot 2				Dot 3			
3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Block 0								Block 1							

För att omvandla två 16 bits ord till 8 färgvärden om 4 bit delas data upp på följande vis innan de laddas i skiftregistren:

PX3	PX2	PX1	PX0
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7

Ord 1	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
	Block 0								Block 1							
Ord 2	4	4	4	4	5	5	5	5	6	6	6	6	7	7	7	7

Dessa färgvärden för varje punkt skiftas sedan med 6 MHz klockfrekvens ur skiftregistren och adresserar via en multiplexerkrets ett 16x8 bits register, i registret omvandlas färgvärdet till färginformation för två dots i följd på skärmen.
Färginformationen läggs fram via en låskrets som delar upp värdet så att 4 bitar ger färgen för dot 1 och de andra 4 ger färgen för dot 2, informationen för dot 1 läggs sedan till ingång 0 och och informationen för dot 2 läggs till ingång 1 på en multiplexer som sedan selektas av 6 Mhz klockan vilket gör att 0 ingångarna kopplar ut värdet för första "dotten" under ena halvperioden och 1 ingångarna kopplar värdet för andra "dotten" under den andra halvperioden.
I den efterföljande färgkontroll/mixerkretsen mixas sedan RGB signalen från färgregistret med videosignalen från bildminnet och klockas ut med 12MHz klockfrekvens (Dot clock).

Videosignalen från videominnet ges i den här kretsen färginformation efter de färgdata som kommer från attribute minnet.

Registret som gjorde om färgvärdet till en RGB signal initieras från BASIC av FGCTL BLK+BLU+RED+GRN osv, om inte detta har gjorts står det 0 i alla positioner i registret viket gör att ingen grafikbild genereras.

CPU:n adresserar registret med en OUT instruktion för att skriva färgvärden till registret, Out stroben HRC gör att multiplexern som kopplar fram PX0-3 selektas och kopplar fram CPU:ns adressbitar A8-A11 fram till registerkretsens adressingångar för att välja register som data ska hamna i. För att kunna skriva i registret på det här sättet måste en speciell out instruktion användas (OUT (C),reg) vilket beskrivs mer i detalj senare.

Vilken minnesarea 0-32K i extraminnet som ska adresseras av räknaren som avsöker grafik minnet väljs med adressbitarna VA14-VA17 som sätts via en Out port.
De åtta bitarna från outporten anger med de 4 lägst signifikanta bitarna vilken minnesarea som ska avsökas och med de fyra mest signifikanta bitarna vilken minnesarea CPU:n ska adressera när den läser eller skriver i grafikminnet.
Värdena i out porten sätts av BASIC funtionen FGPICTURE

När CPU:n ska skriva eller läsa i grafikminnet kommer signalen HRE från minnesdelnings kretsen (Mem. Map Decoding) att aktiveras vilket gör att blocket som benäms Timing and Control Logic kommer att generera signaler som växlar adressmultiplexern så att CPU:n adresserar ramminet.

Samtidigt genereras signalen ROMDIS från minnesdelnings kretsen vilket gör att adressområde 0 till 32K på PU--kortet kopplas bort.

CPU:n kan nu adressera adressområde 0 till 32k i grafikminnet (en bild) via multiplexern. CPU:ns adresser A0 till A11 adresserar minnet direkt, medan A12-A14 kopplas via en multiplexer krets som styrs från minnesdelnings kretsen, samma multiplexer kopplar också fram adresserna M15 och M18 från från Out porten som angav vilken minnesarea 0-32k i extraminnet som skulle adresseras.

Tillsammans ger detta en adress som kan adressera ett adressområde på 512Kbyte men för tillfället använd bara A0 till M16 vilket ger ett adressområde på 128Kbyte.

Adressbit A0 kopplas inte fram via multiplexern utan inverteras först och ger A0/.
A0 och A0/ väljer vid adresseringen vilket block som ska adresseras och vid läsning öppnar de också respektive låskrets på data utgångarna från minnet.
För att inte störa avsökningen av grafikbilden genereras WAIT till CPU:n när den adresserar grafikminnet, hur lång WAIT signal som genereras beror på var i avsökningscykeln som HRE aktiveras, värsta fallet ger 4 WAIT cykler.

Minnes delningen av minnet på PU-kortet och 128K arean kan också hanteras i mindre block än 32 K via ett 16x8 bitars register(Memory Map Register).
Registret addresseras av CPU:ns adressbitar A12-A15 som avkodar 4Kbytes areor i CPU:ns 64Kbytes adressområde.

För varje 4Kbyte block som CPU:n adresserar kommer registret att lägga ut 8 bitars data på sina dataut-gångar, där bit 0-6 kopplas till den multiplexern som ger adressbitarna M12-M18 till 128K minnet, bit 7 kopp-las till minnesdelnings kretsen.

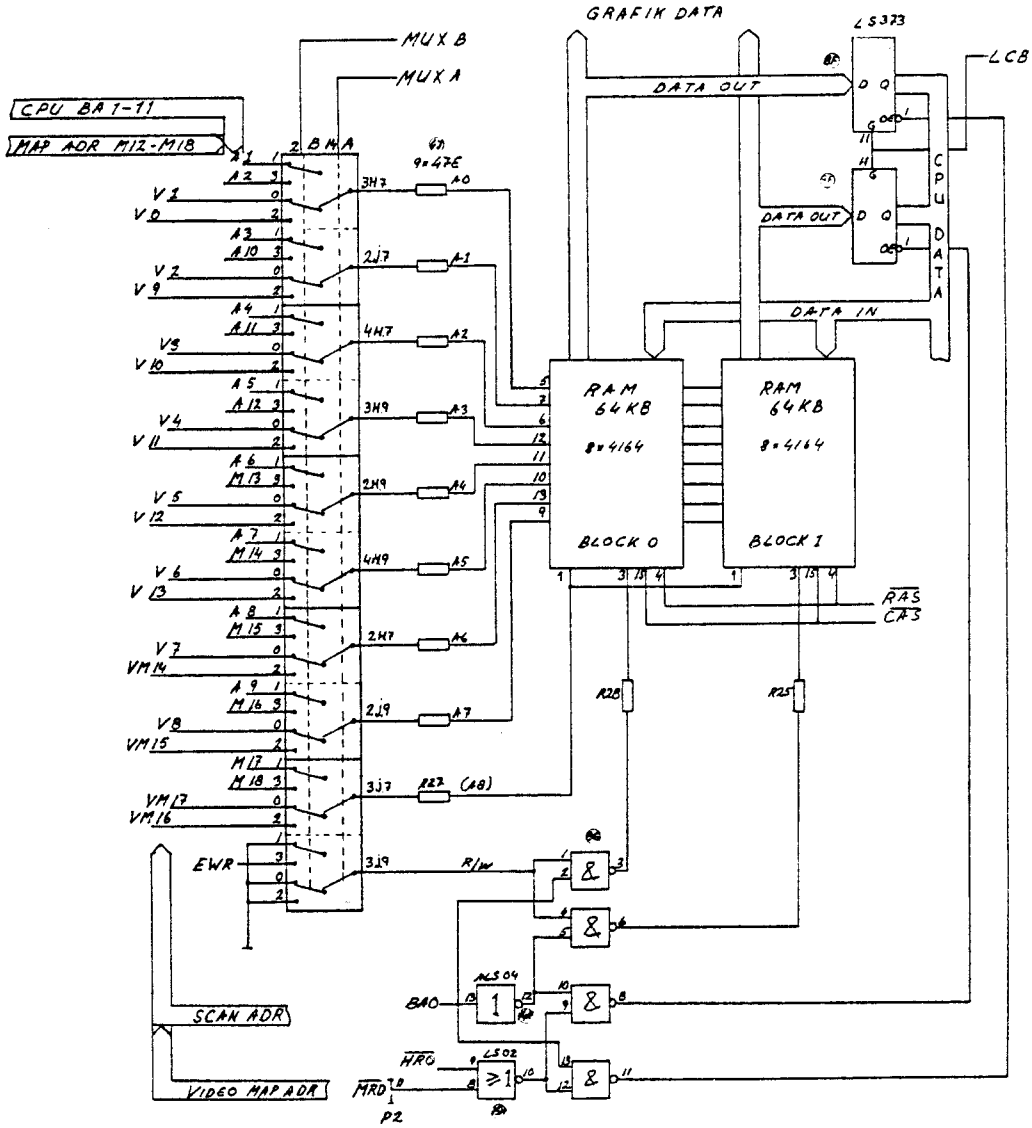
Om bit 7 är hög avkodas detta i kretsen som aktiverar signalen MUX så att bitarna 0-6 kommer att kopplas fram som adressbitar M12-M18 till 128K minnet, samtidigt genereras signalen HRE och RAMDIS eller ROMDIS.
HRE aktiverar adressering av 128K minnet och RAMDIS eller ROMDIS kopplar bort minnet på PU-kortet.
Vilken av ROMDIS/RAMDIS signalerna som aktiveras beror på vilken adress det var som aktiverade förloppet.

A12-A15 som adresserar registret anger ett 4Kbyte block i CPU:ns minnesarea där det 4K block i 128K minnet som addresseras av M12-M18 ska placeras, vilket ger valfri-het att koppla in vilken som helst av 32 st 4K block från extraminnet till vilket som helst av CPU:ns 16 stycken 4K block.

17.2 Adressering av 128Kbyte dynamiskt RAM

RAM-minnet är uppbyggd av 16 stycken kretsar 4164 som innehåller 64Kbit i varje krets, kretsarna är monterade så att de bildar 2 stycken block om 64Kbyte vardera.

Kretsarna har bara 8 adressingångar varför en 16 bitars adress läses in i två steg via multiplexern, först läggs adressbitarna V1-V15,V17 fram till adressingång-arna och läses in som radadress till båda minnesblocken när signalen RAS aktiveras, selekt ingång B till mul-tiplexern läggs sedan låg och kopplar fram adressbitar-na V0,V9-V16 till adressingångarna som läses in som kolumnadress när CAS signalen aktiveras, data läggs sedan ut från minnet fram till låskretsarna och till grafikavkodningen.



Adressering 128 Kbyte RAM.
fig 60.

När CPU:n adresserar minnet kommer selektingång A att välja adressbitarna BA1-M18 för att adressera minnet. Adressbit BAO kopplas till ett antal grindar för att välja minnesblock som ska lägga ut data till CPU:ns databuss eller läsa av data från CPU:ns databuss.

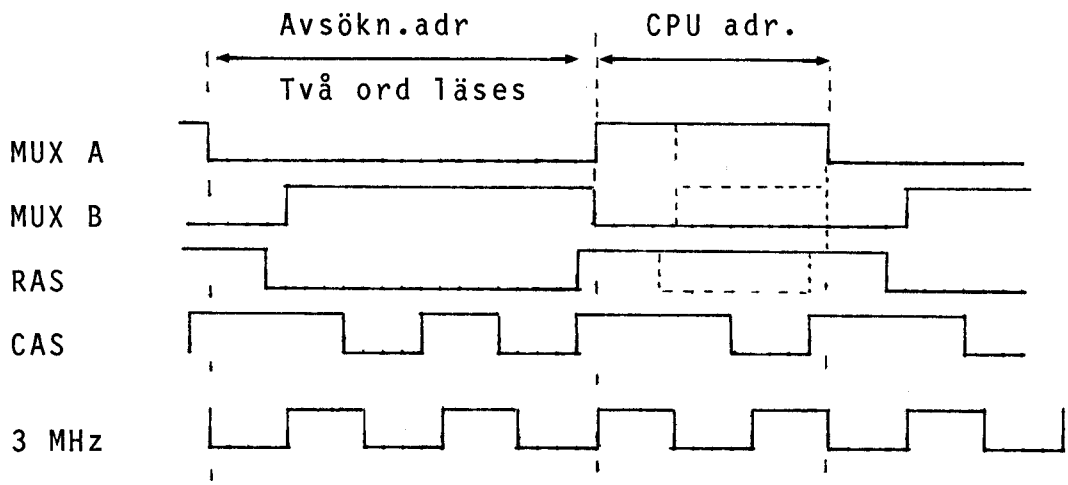
Signalen HRO/ grindas tillsammans med signalen MRD/ i en NOR-grind för att öppna de två efterföljande NAND-grindarna som avkodar värdet på BAO och BAO/ och gör att bara en av låskretsarna öppnas och lägger ut data till CPU:ns databuss vid en läsning (båda blocken adresserades parallelt).

HRO/ är en signal som aktiverats av HRE från minnesdelnings kretsen och MRD/ är en kombination av CPU:ns MREQ/ och RD/.
Signalen LCB till låskretsarna läser in data från minnet till kretsarna eftersom det inte finns tid för minnet att ligga och vänta tills CPU:n har läst av data.

När CPU:n skriver i extraminnet kommer signalen EWR att kopplas till de två andra NAND-grindarna som har BA0 och BA0/ som insignal för att aktivera WR till respektive minnesblock beroende på värde av BA0. EWR är hög när CPU:n gör en skrivoperation och tillsammans med BA0 eller BA0/ kommer den att lägga R/W ingången låg till adresserat minnesblock, när sedan MUX:en växlar tillbaka kommer den att bli låg och läsa in data från bussen till adresserad minnesposition.

Med adressbitarna V0-V13 adresseras en 16 Kbytes area i vardera minnesblocket kontinuerligt och läser ut 1 byte ur vardera och bildar ett ord om 16 bitar som sedan avkodas för att generera en grafikbild Adressbitarna VM14-VM17 sätts via en Outport och bestämmer vilken 16K area i minnet adressbitarna V0-V13 ska adressera för att generera en bild.
Eftersom läsningen av minnet sker kontinuerligt och oavsett om grafiken används eller ej, får RAM-minnet den refresh som behövs för att det ska behålla sin information.

Avsökningen av minnet sker i en sekvens där CPU:n ges möjlighet att (under ett tidsintervall där den inte stör avsökningen) gå in och läsa eller skriva i minnet. Nedan visas hur RAS, CAS, MUX A och B signalerna ser ut under en avsökningscykel, de streckade linjerna visar hur signalerna förändras om CPU:n adresserar minnet.

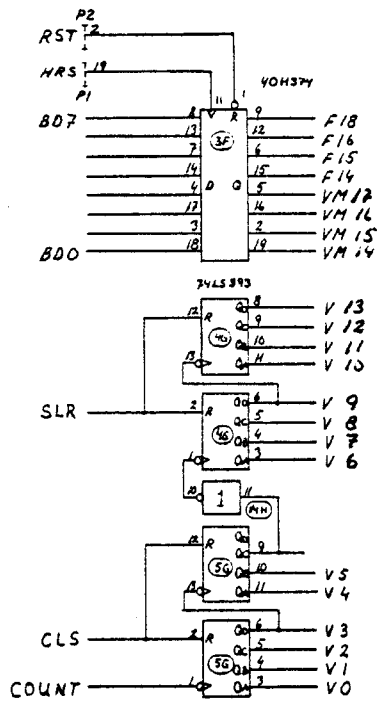


17.3 Räknare för avsökningsadresser

För att skapa adresser som avsöker en 30K area (16K*16-bit) av 128Kbyte minnet finns det två räknare en som räknar antalet avsökta linjer och en som räknar kolumnpositioner (16 bitars ord) utmed en linje, tillsammans ger de en 14 bitars adress V0-V13.
De övriga adressbitarna som behövs för att addressera en 64K area (16 bitars ord) fås från en 8 bitars outport på bitarna 0-3.

VM14-VM17 anger vilken 16K area som ska avsökas för att ge grafikdata. Adressbitarna VM16-VM17 används inte eftersom minnet i dag inte är större än 128Kbyte.
De andra bitarna i outporten används när CPU:n adresserar minnet vid bearbetning av grafikdata och ger vilken 30Kbytes area (bild) som ska adresseras (bearbetas).
Registret adresseras med I/O instruktionen Out 7,Data.
I/O adressen 7 avkodas på pu-kortet och aktiverar signalen HRS och Data skrivs in i registret, vid systemreset nollställs registret. Registret används av BASIC funktionen FGPICTURE X,X,X.

Signaler in till räknarna är SLR som nollställer den övre räknaren vid varje bildstart, CLS som nollställer den undre räknaren vid varje linjestart samt COUNT som räknar upp kolumnräknaren.



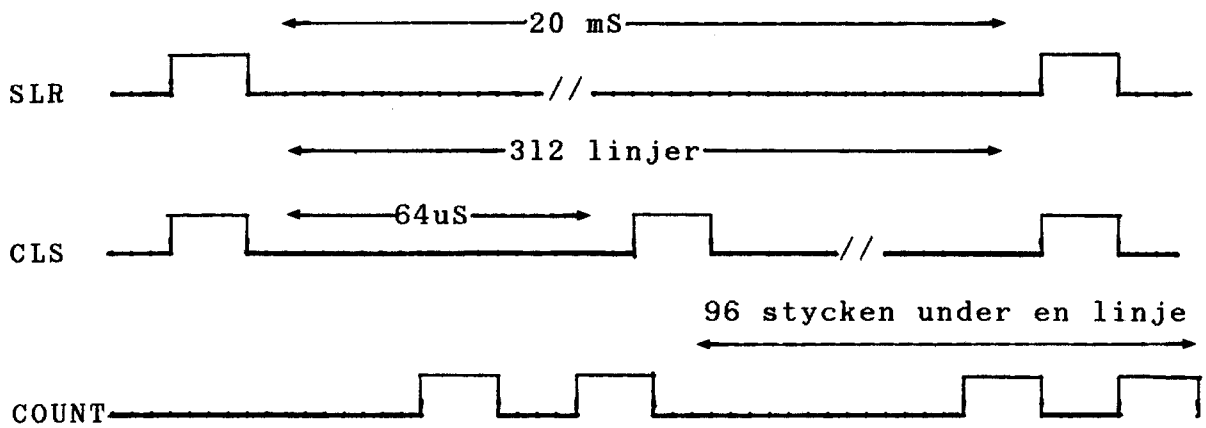
Avsökningsadresser till grafikminnet.
fig 61.

Under avsökningen av en linje kommer signalen COUNT att räkna upp kolumnräknaren, för varje adress till minnet läses en ord om 16 bitar, i 16 bitar finns information om 8 grafikpunkter, 512/8 blir 64 vilket innebär att kolumnräknaren ska lägga ut 64 adresser för att avsöka en linje, sedan ska den räkna upp linjeräknaren och börja om på 0 igen.

När linjeräknaren räknat till 63 kommer bit 6 från kolumnräknaren att gå hög vilket gör att klocksignalen till linjeräknaren blir låg och klockar linjeräknaren så att den räknar upp en linje, kolumnräknaren fortsätter att räkna upp till CLS blir aktiv och nollställer räknaren.

När en bild avsökts och nästa startas kommer signalen SLR att aktiveras och nollställa linjeräknaren.

När 4 linjer avsökts har hela RAM minnet fått en RFSH läsning. För att refreshen ska göras kontinuerligt räknas adresserna upp även utanför aktiv bildyta vilket inte påverkar grafik genereringen eftersom det finns kontrollsignaler som bestämmer när videosignal ska genereras.

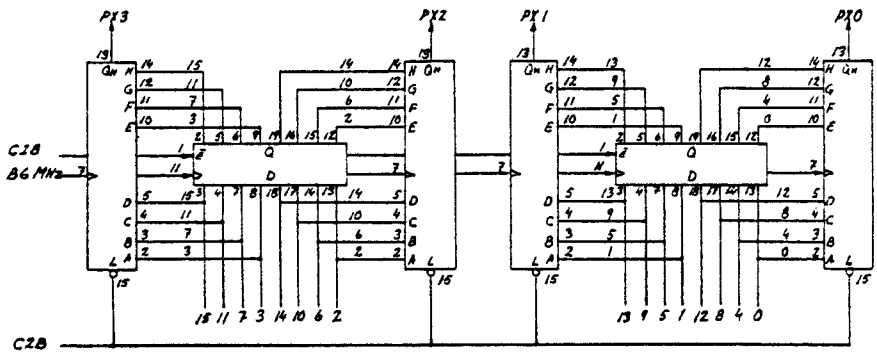


Vid avsökningen ser man att räknaren räknar upp kolumnadressen två gånger ganska tätt och sedan gör ett upphåll innan nästa två pulser kommer, som beskrivits tidigare läses i en avsökningssekvens två ord om 16 bit som laddade skiftregistren som konverterade data till färgvärden.

Vid addressering av grafikminnet används en adresserings metod som kallas "Page Mode Read Cykle" vilket innebär att man adresserar minnet med en rad adress och en kolumn adress och läser ut ord, sedan ökar man kolumn adressen med 1 och läser bara in en ny kolumnadress för att läsa ut nästa ord (se RAS och CAS signalerna), förfarandet gör att det går snabbare att läsa ut data från två minnespositioner.

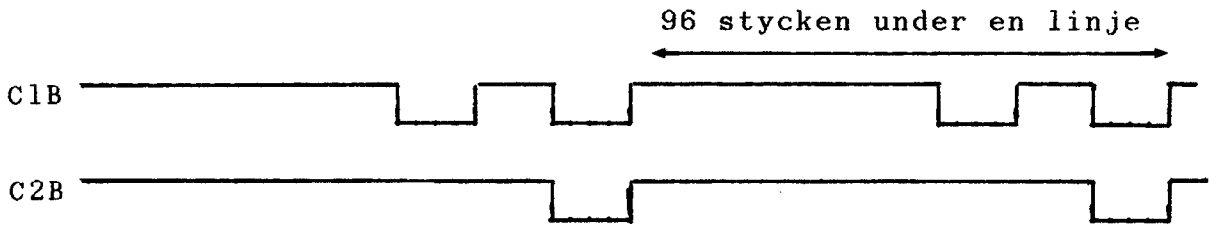
17.4 Konvertering till färgvärden

Data ut ifrån grafikminnet som ska konverteras till ett färgvärde (värdet för en eller två grafikpunkter) är ett ord med 16 bitar, dessa läggs fram till två låskretsar och läses in till dessa av signalen C1B och klockan B6MHz, på låskretsarnas utgångar delas data upp och läggs fram till skiftregistrens ingångar H,G,F och E, eftersom två ord om 16 bitar behövs för att fylla skiftregistret med data som ska skiftas ut och bilda färgvärden läses ytterligare ett ord och när det ordet finns på bussen laddas skiftregistren med signalen C2B.



Konverering av grafikdata till färgvärden.
fig 62.

Sekvensen på signalerna C1B och C2B visas nedan.



Data som laddats skiftas sedan ut med B6MHz klockan så att ett nytt färgvärde läggs ut för varannan dot som visas på skärmen.

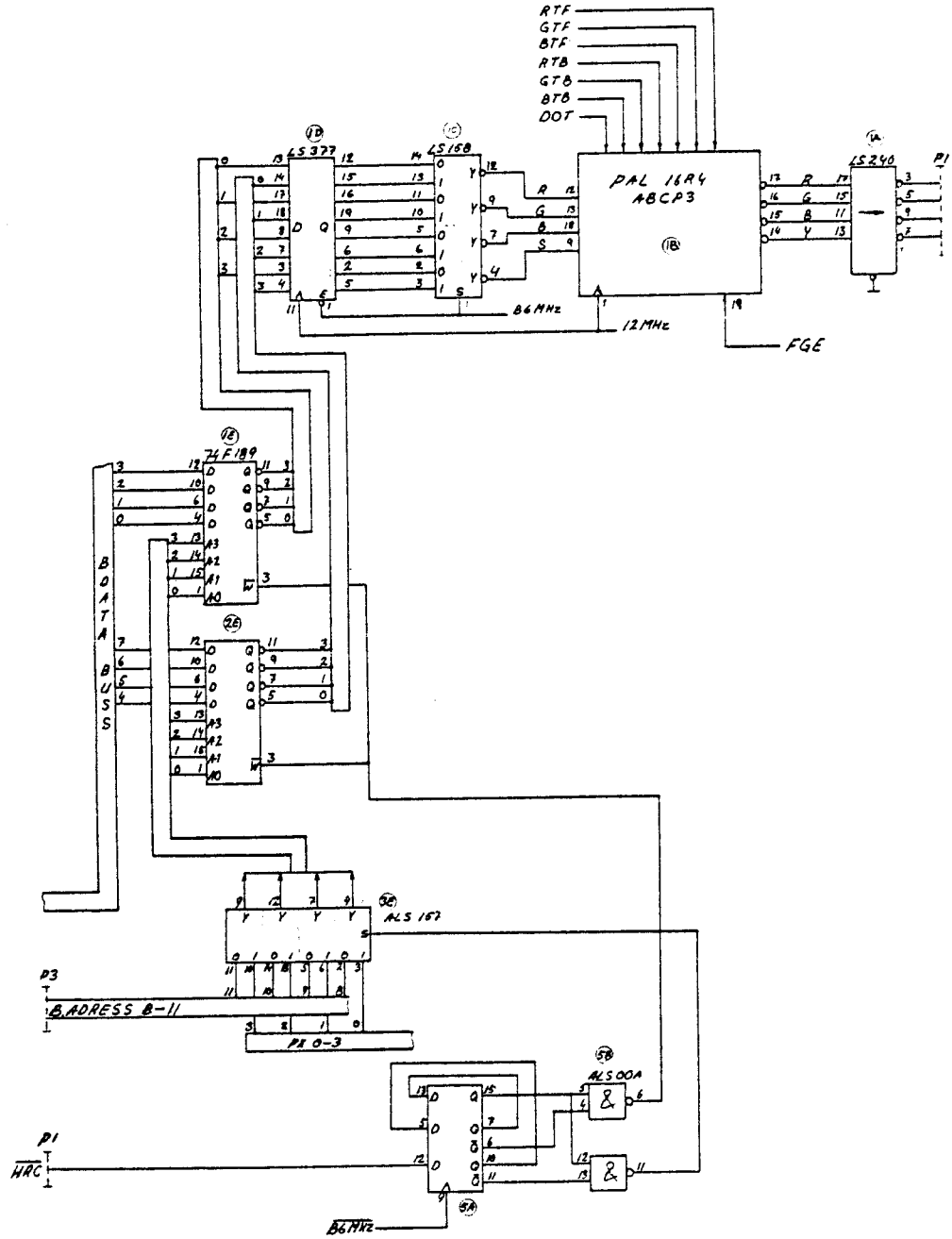
17.5 Avkodning av färgvärden till RGB signal

Färgvärdena från skiftregistren kopplas via en multiplexer fram till adressingångarna på två 16*4 bits register, för varje färgvärde 0-15 från skiftregistret kommer det på registrens data utgångar att läggas ut 2 stycken 4 bitars värden, den ena anger R,G,B och P för dot 1 och den andra R,G,B och P för dot 2.

R är röd, G grön och B blå färginformation för en dot, antalet färgkombinationer blir med 3 bitar max 8 färger. Den fjärde biten anger prioritet för RGB signalen från grafikminnet, om P är 1 i ett dotvärde så innebär det att informationen från videominnet kommer att skrivas över av den färgen från grafiksignalen.

De två fyrabitars RGB värdena kopplas via en låskrets fram till en multiplexer där de delas upp så att värdet för dot 1 läggs till multiplexerns 1 ingångar och värdet för dot 2 till 0 ingångarna.

Data till låskretsen läses in när B6MHz är låg och en positiv flank finns på 12MHz klockan, när B6Mhz blir hög kopplas värdet för dot 1 fram till ABCP3 kretsen och läses in till den på den positiva flanken i 12MHz klockan, när sedan B6MHz klockan blir låg kopplas värdet för dot 2 fram till ABCP3 och klockas in till den samtidigt som det första dotvärdet skickas ut från kretsen som RGB signal till monitorn.



Konvertering av färgvärden till RGB värden.
fig 63.

I kretsen ABCP3 som är konstruerad med en PAL-krets (Programable Array Logic) avkodas signalerna BTB,GTB,RTB och BTF,GTF,RTF från bildminnet. Dessa anger teckenbakgrund och teckenfärg och om dot-signalen är aktiv kommer den att generera RGB signaler enligt teckenfärgvärdet medan teckenbakgrund signalerna genererar RGB signaler för bakgrundsfärg, dessa blandas sedan i kretsen och skickas ut som RGB och Y signal. I kretsen blandas också grafiksignalen med bildsignalen efter angivet värde på P biten.

Signalen FGE innehåller det vertikala och horisontala tidsfönstret som visar den aktiva bildytan för grafikbilden.

Registret som adresserades av PX0-3 och kodade om dessa till RGB signaler, initieras av CPU:n när man i BASIC anger mode och färgvärden med funktionen FGCTL BLK+RED+GRN+YEL+BLU+MAG+WHT+CYA osv. registret kommer att programmeras efter den ordning som färgerna anges i FGCTL.
När 8 färger angetts enligt ovan så skrivs i båda registren på adress 0 värdet 0000B (Svart), på adress 1 skrivs 0001B (Röd), på adress 2 skrivs 0010B(Grön) och på adress 3 0110B (Grön och BLå=Gul) osv. Färgvärde 3 från grafikminnet tolkas sedan som en gul grafikpunkt.

I 256*240 grafikmode kommer registren att laddas med värden enligt nedan om följande angetts med FGCTL:
BLK+RED+BLU+GRN+WHT+YEL+MAG+CYA
+GBLK+GRED+GBLU+GGRN+GWHT+GYEL+GMAG+GCYA

	1 Grafikpunkt								
	Dot 1				Dot 2				
	7	6	5	4	3	2	1	0	
PX 15	P	0	G	R	P	0	G	R	CYA+CYA
PX 14	P	B	0	R	P	B	0	G	MAG+MAG
PX 13	P	B	G	0	P	B	G	0	YEL+YEL
PX 12	P	B	G	R	P	B	G	R	WHT+WHT
PX 11	P	0	G	0	P	0	G	0	GRN+GRN
PX 10	P	B	0	0	P	B	0	0	BLU+BLU
PX 9	P	0	0	R	P	0	0	R	RED+RED
PX 8	P	0	0	0	P	0	0	0	BLK+BLK
PX 7	0	0	G	R	0	0	G	R	CYA+CYA
PX 6	0	B	0	R	0	B	0	R	MAG+MAG
PX 5	0	B	G	0	0	B	G	0	YEL+YEL
PX 4	0	B	G	R	0	B	G	R	WHT+WHT
PX 3	0	0	G	0	0	0	G	0	GRN+GRN
PX 2	0	B	0	0	0	B	0	0	BLU+BLU
PX 1	0	0	0	R	0	0	0	R	RED+RED
PX 0	0	0	0	0	0	0	0	0	BLK+BLK

I 512 mode kommer registerinnehållet att se ut på följande sätt om FGCTL BLK+RED+GRN+BLU angetts i BASIC:

			Dot 1				Dot 2					
			7	6	5	4	3	2	1	0		
Färgnr	xx	xx										
PX 15	11	11	0	B	0	0	0	B	0	0	BLU	BLU
PX 14	11	10	0	B	0	0	0	0	G	0	BLU	GRN
PX 13	11	01	0	B	0	0	0	0	0	R	BLU	RED
PX 12	11	00	0	B	0	0	0	0	0	0	BLU	BLK
PX 11	10	11	0	0	G	0	0	B	0	0	GRN	BLU
PX 10	10	10	0	0	G	0	0	0	G	0	GRN	GRN
PX 9	10	01	0	0	G	0	0	0	0	R	GRN	RED
PX 8	10	00	0	0	G	0	0	0	0	0	GRN	BLK
PX 7	01	11	0	0	0	R	0	B	0	0	RED	BLU
PX 6	01	10	0	0	0	R	0	0	G	0	RED	GRN
PX 5	01	01	0	0	0	R	0	0	0	R	RED	RED
PX 4	01	00	0	0	0	R	0	0	0	0	RED	BLK
PX 3	00	11	0	0	0	0	0	B	0	0	BLK	BLU
PX 2	00	10	0	0	0	0	0	0	G	0	BLK	GRN
PX 1	00	01	0	0	0	0	0	0	0	R	BLK	RED
PX 0	00	00	0	0	0	0	0	0	0	0	BLK	BLK

För att skriva till registret måste maskinspråksinstruktionen OUT (C),r användas, CPU:n kommer under exekvering av den instruktionen att lägga ut innehållet i register B på adressbitarna A8-A15 och innehållet i C registret på adressbitarna A0-A7 som en I/O adress.

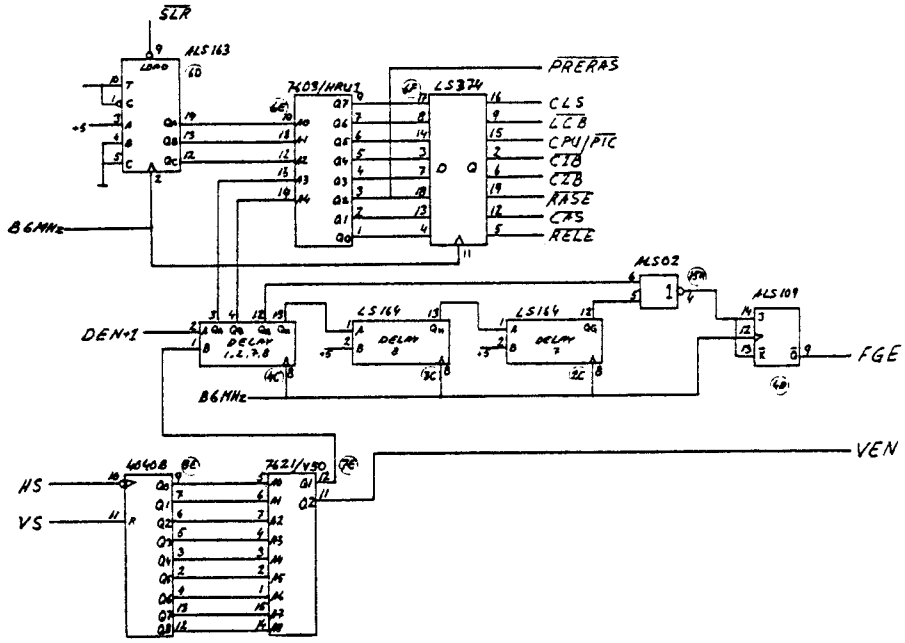
Adressbitarna A0-A7 kommer att aktivera Out stroben HRC/ som via D-vippan i pos 5A och den efterföljande nand grinden kommer att selekta multiplexern så att adressbitarna A8-A11 kopplas fram till registren och adresserar en position i dessa.
Data från register r i CPU:n finns nu på dataingångarna till registren och läses in när WR/ signalen aktiveras, vilket görs av HRC/ stroben via en fördröjning i några D-vippor.

För att tilldela ett färgvärde (färgnummer) från grafikminnet en färg laddas B registret med färgnummer 0-15, register r med färgdata enligt P B G R P B G R och C registret med I/O adress 7.
D7 6 5 4 3 2 1 0

Programexempel som initierar registret och visar hur man i 256*240 mode kan få 16 färger på skärmen finns i appendix.

17.6 Kontrollsignaler för extraminne och grafikgenerering

För att generera kontroll och synkroniserings signaler på VU-kortet används en PROM krets 7603 (32*8bit), kretsen addresseras på adressingångarna A0-A2 av en räknare som klockas med B6Mhz och kommer därför kontinuerligt att lägga ut data på sina utgångar.
Prom kretsen får också adressbitarna A3-A4 från ett skiftregister, dessa används för att ur promkretsen generera pulser vid linjestart och linjeslut.



Generering av kontrollsignaler för 128K minne.
fig 64.

Räknaren som adresserar Prom kretsens adressingångar A0-A2 synkroniseras vid bildstart av signalen SLR/ på Load ingången, sedan kommer den att räkna kontinuerligt 0-7 under avsökningen av en bild.
A3-A4 kommer från skiftregistret som har signalen DEN+1 som insignal på A ingången, B ingången får sin insignal från en PROM krets 7621 (512*4bit), signalen på B ingången grindar signalen på A ingången så att om inte B är hög kopplas inte DEN+1 vidare i skiftregistret.

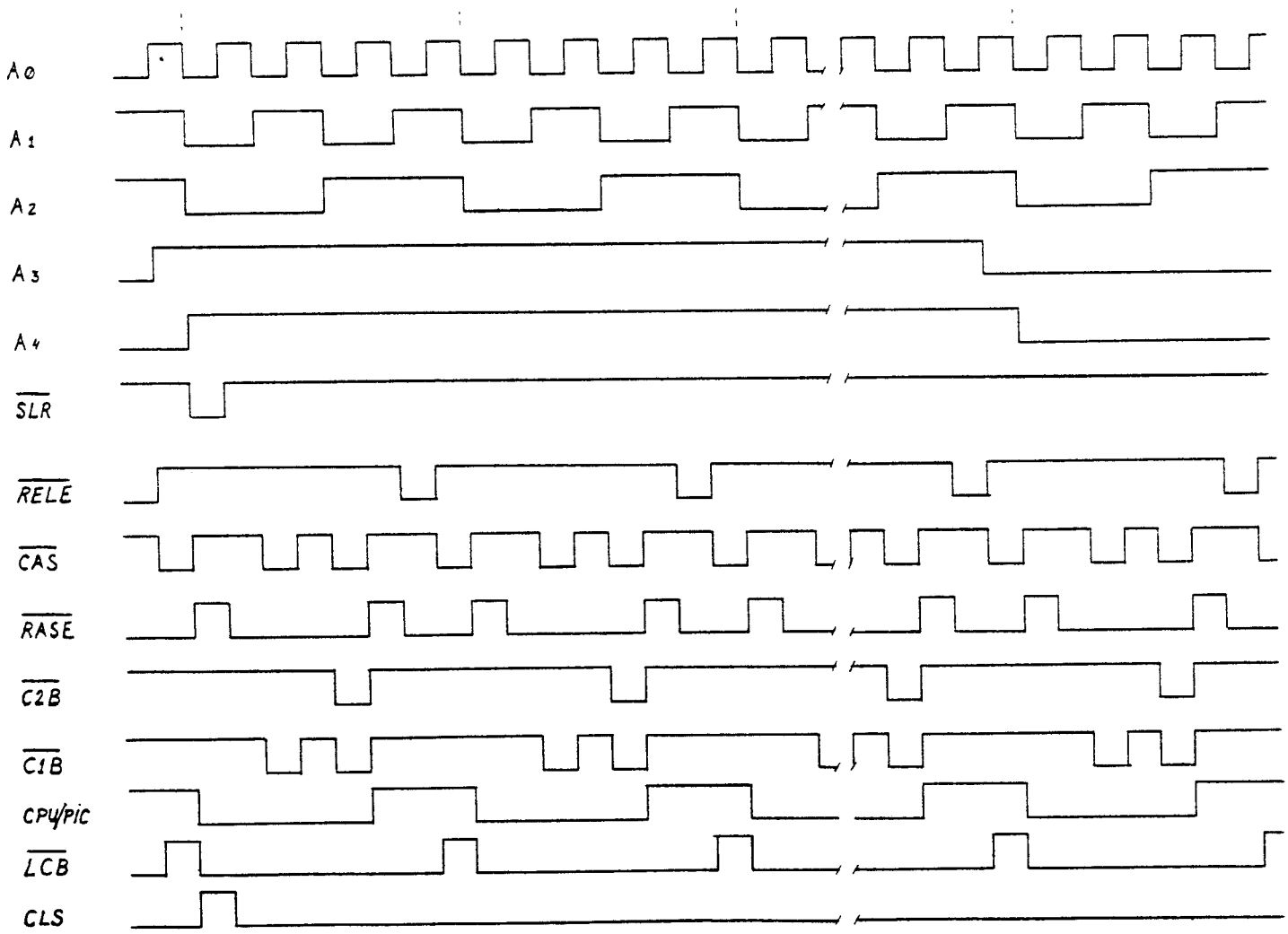
Promkretsen som benäms V50, adresseras av en räknare som kontrolleras av Vertikala och Horisontala synkpulser, vertikal pulsen resetar räknaren vid bildstart och horisontalpulsen räknar sedan upp räknaren för varje linje som avsökts.
V50 kretsen har programmerats så att det på sin Q1 utgång kommer att ge det aktiva bildytan vertikalt för grafikbilden, medan Q2 utgången ger det vertikala tidsfönstret för bilden från bildminnet (VEN Video Enable).
Signalen DEN+1 kommer från CRTC kretsen och innehåller det horisontella tidsfönstret som markerar var den aktiva bildytan under en linje ska vara.

När signalen till B ingången från V50 kretsen blir aktivt kommer den att koppla fram DEN+1 signalen till skiftregistret som sedan kommer att skifta ut den på sina pararella data utgångar i takt med B6Mhz klockan.

På de utgångar som ger A3 och A4 signalerna till HRU1 kretsen kommer man därför att vid linjestart få värdet A3=1, A4=0 och vid linjeslut få A3=0 och A4=1 vilket kan användas för att ge vissa pulser ur HRU1 prommet bara vid start och slut på en linje.

DEN+1 signalen fördröjs sedan ytterligare i skiftregistret för att sedan via JK-vippan ge signalen FGE (Full Graphic Enable) som används för att ge den aktiva bildytan för grafikbilden.

Med V50 prommet och skiftregistren synkroniseras förhållandet mellan signalerna FGE och VEN så att bilderna från grafikminnet och bildminnet får rätt placering i förhållande till varandra.
Om bildfrekvensen från CRTC kretsen ändras till 60 HZ måste V50 kretsen bytas till en som är anpassad för detta annars kommer bildernas placering på skärmen att förändras.



Signaler från HRU1 kretsen.

HRU1 kretsens datautgångar kopplas via en låskrets, som klockas med B6MHz vidare ut på VU-kortet och ger följande kontrollsignaler:

CLS — Används för att reseta räknaren som ger kolumnadresser för avsökning av en grafikbild. (Column Line Strob)

LCB — När CPU:n adresserar 128K minnet för att läsa data kommer den här signalen att läsa in data till låskretsarna som finns på minnets datautgångar.

CPU/PIC — Signalen markerar när den är låg att en avsökningadressering pågår, när den är hög indikerar den att CPU: kan adressera 128K minnet.

C1B/ — Läser in ett ord från 128K minnet till låskretsarna som sedan förser skiftregistren med data som ska omvandlas till färgvärden.

C2B/ — Laddar två 16 bitars ord till skiftregistren som omvandlar grafikdata till färgvärden.

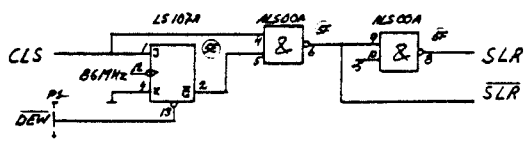
RASE/ — Tillsammans med signalen CPU/PIC och PRERAS/ kontrollerar den här signalen Rad Adress Stroben till 128K minnet.

CAS/ — Ger Column Adress Strob till 128K minnet.

RELE/ — När CPU:n adresserar 128K minnet genereras alltid Wait signal, den här signalen deaktiverar den wait signalen.

PRERAS/ — Kontrollerar tillsammnas med RASE/ och CPU/PIC aktiveringen av Rad Adress Stroben till 128K minnet.

Signalen SLR till räknarens Load ingång synkroniserar räknaren med bildstart, signalen skapas av CLS som är aktiv i början på varje linje under den aktiva bildtiden samt DEW som är vertikalsynkpulsen, dessa signaler är kopplade till en JK-vippa och en nand grind så att de kommer att aktivera SLR/signalen bara när den första CLS signalen kommer under en bildavsökning.
OBS. om man mäter SLR/ signalen med ett oscilloskåp kan det vara svårt att se pulsen eftersom den bara är 160 nS lång och kommer med 20 mS mellanrum.

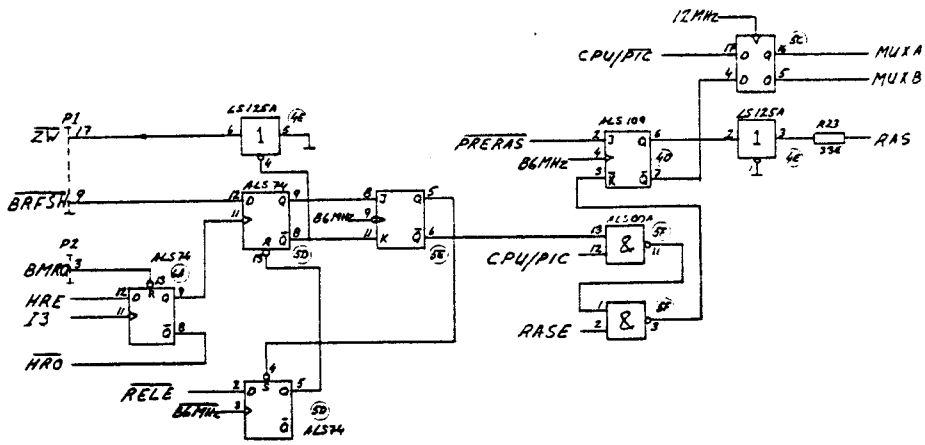


SLR/ signal.
fig 65.

Signalerna RASE/, CPU/PIC, och PRERAS/ kontrollerar som tidigare nämts RAS/ (Rad Adress Strob) till 128K minnet, samma signaler styr också signalen MUXB som växlar mellan radadresser och kolumnadresser till 128K minnet.

Signalen CPU/PIC som kontrollerar var i en avsökningscykel CPU:n kan få adressera 128K minnet kopplas via en D-vippa (som klockas med 12MHz) fram som signalen MUXA.
MUXA signalen kommer under varje avsökningscykel att växla adressmultiplexern så att CPU:ns adressbitar BA2-BA9 läggs fram som radadress till 128K minnet.

Om RAS/ ska genereras i det här läget för att läsa in radadressen bestämms av signalen HRE från minnesdelningskretsen, om den är låg kommer inte någon RAS/ att aktiveras vilket är normalfallet.



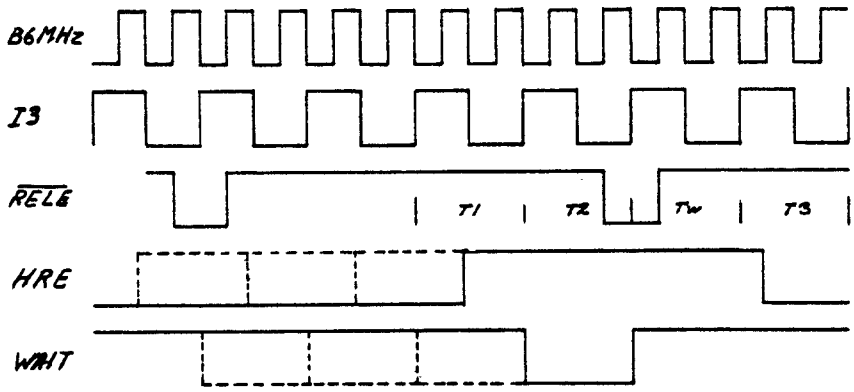
Växling mellan CPU adress och avsöknings adress.
fig 66.

Signalen HRE aktiveras när CPU:n ska läsa eller skriva i 128K minnet samtidigt har motsvarande minnesarea som CPU:n adresserar kopplats bort på PU-kortet.
HRE signalen kopplas fram via en D-vippa där man synkroniserar den med CPU klockan (I3), när Q utgången på vippan går hög klockas den efterföljande vippan som på sin D-ingång har RFSH/ signalen som insignal, om RFSH/ signalen har hög nivå (ingen refresh addressering) kommer Q/ utgången att bli låg och via tristate bufferten att generera Wait till CPU:n.

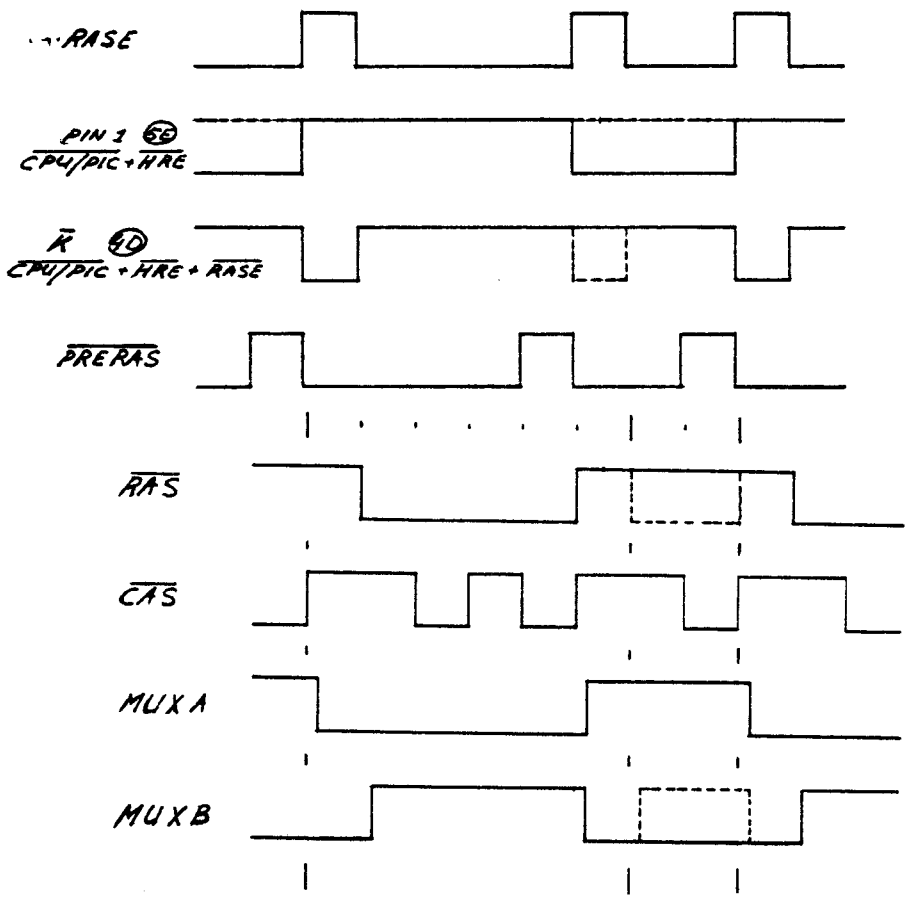
D-vippan i pos 5E kommer att på sin Q utgång att få hög nivå vilket gör att den D-vippa som har signalen RELE/ som insignal frikopplas och kan känna av värdet på RELE/ signalen, när RELE/ blir låg kommer den att reseta vippan som genererade Wait till CPU:n, så att CPU:n släpps ur wait läget.

Q/ invers utgången på D-vippan i pos 5E kommer under tiden innan RELE signalen blev aktiv att generera en RAS/ signal samt en växling av MUXB signalen, vilket gör att CPU:n adresserar 128K minnet.
Beroende på i vilket tidsögonblick som HRE aktiveras genereras olika antal wait cykler till CPU:n, minst 1 och max 4 stycken.

Nedan följer tidsdiagram som visar förloppet på de olika signalerna.



Wait signal till CPU.
fig.67



Kontrollsignaler minnesadressering
fig 68.

17.7 Minnesdelning 128K byte

Eftersom CPU:n bara kan adressera en minnes area på 64K byte och det på VU kortet finns ytterligare 128K byte som ska kunna adresseras måste en minnesdelning göras mellan de båda minnesareorna.

CPU:n har på PU kortet en minnes area på 64K byte uppdelad i två block, en ROM area 0-32K och en RAM area 32-64k, den minnesarean kan kopplas bort på olika vis och istället kopplas delar av extraminnet in på dessa adressområden.

Hur stor del av extraminnet och var i CPU:ns adressområde denna del ska kopplas in kan man på olika sätt bestämma genom programstyrning.

128K minnet kan delas in i olika stora block, och de möjligheter som finns är följande 4*30K, 4*32K eller 32*4K block.
Beroende på vilken delning man använder kopplas minnet in över olika adressområden, vid 30K och 32K delningen kopplas blocken in över CPU:ns adressområde 0-30K,0-32K, med 4K block kan man välja ett 4K block i CPU:ns 64K minnesarea och placera 1 av 32 block där, fler än ett block kan kopplas in samtidigt.

Avkodningen av när extraminnet ska kopplas in och hur det ska adresseras görs av en krets på VU-kortet, kretsen benämns ABCP4 och är konstruerad i en PAL (Programable Array Logic).

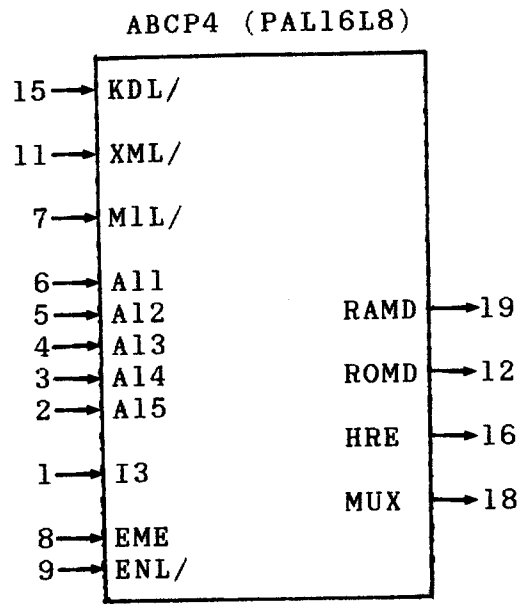


fig 69.
Minnesdelningskretes ABCP4

Utsignaler från kretsen är följande:

RAMD — Ram Disable, aktiv hög.
Kopplar bort RAM minnet på adressområde 32 till 64k på PU-kortet.

ROMD — Rom Disable, aktiv hög.
Kopplar bort ROM minnet på adressområde 0 till 32k på PU-kortet.

HRE — High Res Enable, aktiv hög.
Aktiverar inkoppling av extraminnet på VU-kortet.
Är aktiv tillsammans med RAMD eller ROMD

MUX — Multiplexer selekt.
Väljer varifrån adresserna M12-M18 som adresserar block i extraminnet ska genereras.

Insignaler som avkodas i kretsen är följande:

KDL — Key DTR low,aktiv låg.
När KDL är aktiv kommer den att aktivera utsignalerna ROMD och HRE samt sätta MUX signalen låg, när CPU:n adresserar adressområde 0-32k.
Signalen används vid 32k blockning av extraminnet.
KDL kontrolleras av signalen KEYDTR som kommer från DART kretsen och kan programmässigt kontrolleras via ett register i DART ketsen.

XML — External Memory Low,aktiv låg.
Förhindrar aktivering av signalerna HRE,-RAMD eller ROMD. XML kontrolleras av signalen XM/ i busskontakten som indikerar att externt extraminne adresseras.
När signalen XM är aktiv ska allt internt minne kopplas bort.

M1 — Machine Cykel 1,aktiv låg.
M1 kommer från CPU:n och indikerar att den gör en instruktionshämtning. I kretsen avkodas den tillsammans med adressbitarna A11-A15 för att aktivera inkoppling av extraminnet vid ett specialfall.
Specialfallet inträffar om CPU:n hämtar en instruktion på adressområde 7800H-7FFFH och den instruktionen innebär en operand läsning/skrivning på adressområde 0 till 30K. I det fallet kommer ett 30K block i extraminnet att adresseras (HRE,ROMD och låg MUX signal).

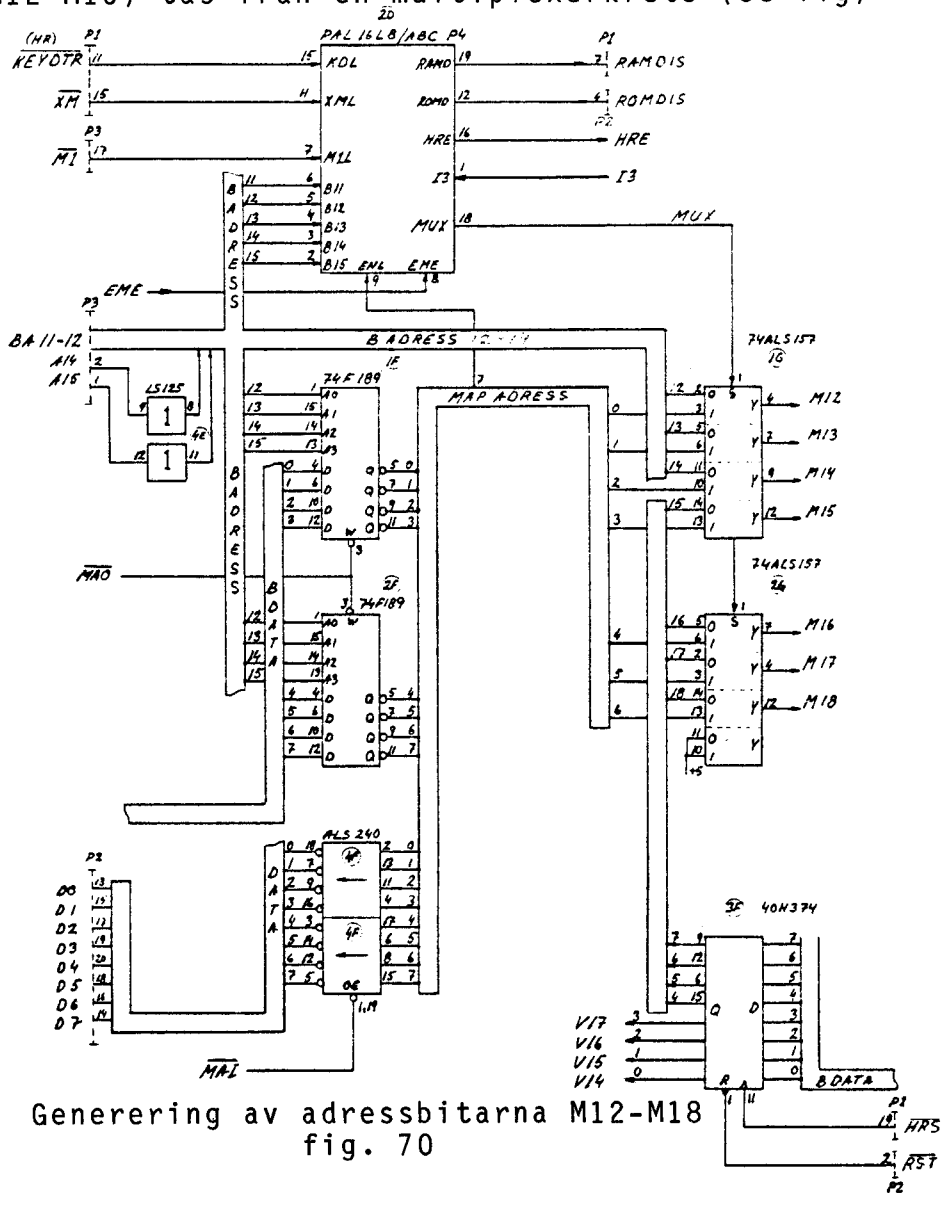
A11-A15 Avkodas i kretsen för att välja om RAMD eller ROMD ska aktiveras tillsammans med HRE, samt tillsammans med M1 för att aktivera inkoppling av extraminnet.

I3 Synkroniserad CPU:klocka.
Används för att synkronisera signaler med CPU:n.

EME External Memory Enable,aktiv hög.
För att kunna koppla in extra minnet i 4K block måste den här signalen vara aktiv.

ENL Enable low, aktiv låg.
Signalen kommer när den är aktiv och om EME är hög att aktivera signalerna HRE,-RAMD eller ROMD samt sätta MUX signalen hög. Används vid inkoppling av extraminnet i 4K bytes block.

När extraminnet adresseras kopplas adressbitarna A0-A11 från CPU:n fram och adresserar minnet direkt, medan de övriga adressbitarna som behövs för att adressera 128K bytes (M12-M16) tas från en multiplexerkrets (se fig)



Generering av adressbitarna M12-M18
fig. 70

Multiplexer kretsen kontrolleras på selektingången av signalen MUX från minnesdelningskretsen och väljer varifrån adressbitarna M12-M18 ska genereras.
M17 och M18 används inte när man har 128K bytes minne, utan är avsedda för en framtida utvidgning av minnet.

När MUX signalen är låg kommer CPU:ns adressbitar A12-A14 att ge M12-M14 medan M15-M18 tas från en registerkrets i pos 3F.
Registerkretsen är ansluten till CPU:n som en Out port på adress 6.
M15-M18 sätts med D4-D7 i porten och kommer att välja vilken 32K area i extraminnet som CPU:n kommer att adressera.

Bitarna D0-D3 i samma registerkrets ger adressbitarna V14-V17 som väljer vilket block i extraminnet som en grafikbild ska genereras från. Registret kontrolleras av basicfunktionen FGPICTURE. Registret nollställs vid systemreset.

Den här adresseringen används när extraminnet kopplas in över CPU:ns minnesarea i block om 30/32K bytes, och används bl.a.vid bearbetning av grafikbilder från basic av (FGPOINT,FGFILL osv). Programexempel på inkoppling från basic ges i appendix.

Vid inkoppling av extraminnet i block om 4K byte kommer MUX signalen att sättas hög, vilket gör att adressbitarna M12-M18 hämtas från de två 16*4 bits registren i pos 1Foch 2F.

De två registerkretsarnas adressingångar är anslutna till CPU:ns adressbitar A12-A15, vilket gör att för varje 4K block i CPU:ns 64kminnesarea får man från registerkretsarna ut ett 8 bitars datavärde där bit 7 ger signalen ENL till minnesdelnings kretsen och aktiverar HRE,RAMD/ROMD och sätter MUX signalen till hög nivå vilket gör att bitarna 0-6 från registren kopplas fram via multiplexern och adresserar ett 4K block i extraminnet.

Vilket 4K block i extraminnet (0-31,med 128KB) som ska kopplas in bestäms med bit 0-6, medan bit 7 anger att det ska kopplas in, var i CPU:ns minnesarea blocket ska kopplas in bestäms av vilken registerposition (0-15) det är som ger data ut.

För att signalen ENL ska aktivera inkoppling av extraminnet måste signalen EME vara aktiv, EME signalen kontrolleras via en out port och sätts låg vid systemreset och sätts aktiv först efter att registren har nollställts av systemprogramvaran.

Vid läsning och skrivning av data till registren adresseras dessa som en I/O krets, för att det ska fungera måste man använda sig av I/O instruktionen OUT (C),r och IN r,(C), CPU:n kommer under exekveringen av dessa instruktionen att lägga ut innehållet i register B på adressbitarna A8-A15 och innehållet i C registret på adressbitarna A0-A7 som en I/O adress.

Adressbitarna A12-A15 kommer vid en skrivning att adressera en registerposition medan A0-A7 aktiverar I/O stroben MA0/ till writeingången och data från CPU:ns register r skrivs in på adresserad position.
För att koppla in ett 4K block från extraminnet får man först ladda register B med ett värde 0-15*16 (*16 för att skifta upp värdet till A12-A15) som anger var i CPU:ns minnes area blocket ska kopplas in, register C laddas med out adress 52 och register r laddas med ett värde 0-31+128 (+128 för att aktivera inkoppling) som anger vilket 4K block i extraminnet som ska kopplas in.

Vid läsning av en registerposition laddas register B och C på samma vis som vid skrivning, A12 till A15 kommer att adressera en registerposition och A0-A7 aktiverar I/O stroben MAI/ som öppnar inporten i pos 4F och lägger ut data från adresserad position i registerkretsen till CPU:n.

OBS. att data ut på registerkretsarna är inverterade. Inverteringen gör att extraminnets inblockning sker från varsitt håll beroende på vilket inkopplingsätt man använder (32K block eller 4K block).

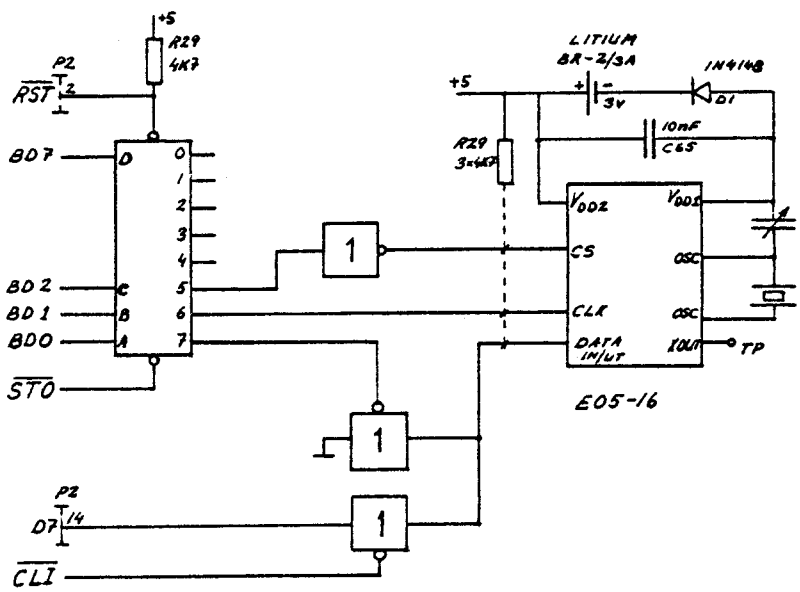
Programexempel på hantering från BASIC ges i appendix.

17.8 Realtids klocka

Kretsen som innehåller klockfunktionen är uppbyggd i CMOS teknik vilket gör att den inte drar mycket effekt. Kretsen innehåller ett antal räknare som drivs från en kristalloscillator på 32768 Hz, från räknarna kan sedan värden hämtas som anger sekunder,minuter,timmar, dag,-månad och år.

Klockan har spännings backup från ett batteri vilket gör att räknarna i kretsen fortsätter att räkna även om systemet slås av.
När systemet slås på tas spänningsmatning från +5 volt. För att klockan ska fungera när spänningen är avslagen ska batterispänningen vara 1.5-3 volt, strömförbrukningen är vid batteridrift max 15uA.

Klockkretsen är ansluten till CPU:n som en I/O krets, för att läsa och skriva data till kretsen sker kommunikationen via I/O portarna med adresser 54 och 55.

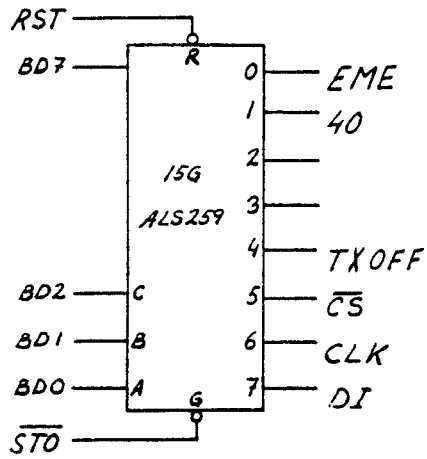


CMOS-Klocka
fig 71.

Via I/O port 54 som är en åtta bitars adresserbar registerkrets, kontrolleras Chip Select, klocka och data in till klockkretsen.

I samma port får man också några andra kontrollsignaler, External Memory Enable, 40/80 och Adress bit 8.

Kretsen adresseras med I/O adress 54 och med databitarna 0-3 väljer man register medan bit 8 är data till addresserad bit.



I/O port adress 54
fig 72.

Out 54,128	EME	Gör att 4K minnesblockning kan aktiveras via MAP register kretsarna.
Out 54,0	EME/	Gör att 4K minnesblockning inte kan aktiveras via MAP registerkretsarna.
Out 54,129	40	40 teckens mode. Går till attribut hanterar kretsen (11C) och bestämmer vilket mode den ska starta i vid radstart, 40 eller 80 tecken.
Out 54,1	80/	80 teckens mode
Out 54,130	A8	Adressbit 8 sätts hög. Är kopplad till adressbit 8 på ett 512*4 bitars PROM (12G), prom:et läses med I/O instruktioner och innehåller tabeller för färgvärden i olika moder.
Out 54,2	A8/	Adressbit 8 sätts låg
Out 54,131 Out 54,3		Används ej.
Out 54,132	TX	Text på. Går till reset ingången på krets 9C och används för att stänga av och på videoinformationen.
Out 54,4	TX/	Text av.
Out 54,133	CS/	Card Select till CMOS klockan sätts låg.
Out 54,5	CS	Card Select till CMOS klockan sätts hög.

Out 54,134	CL	Klockingången till CMOS klockan sätts hög.
Out 54,6	CL/	klockingången till CMOS klockan sätts låg.
Out 54,135	DI	Data in till CMOS klockan sätts hög.
Out 54,7	DI/	Data in till CMOS klockan sätts låg

Kretsen utgångar nollställs vid reset.

För att läsa data från CMOS klockan användeer man I/O instruktionen INP(55) där CMOS klockans data finns på bit 7 och HRUII prommets data på bitarna 0-3.

I appendix finns program ex. som visar hur man läser och skriver till klockkretsen och hur man läser av HRUII Prom:et.

18 APPENDIX

Här följer beskrivningar om hur man programmässigt styr och kontroller vissa funktioner.

I de programexempel som används, finns ingen felhantering införd, detta för att programmen ska bli så överskådliga som möjligt.

En del av programmen är utförda som testprogram men kan inte anses som fullständiga test.

Appendix	A	Färgregister
	B	Attribute koder
	C	Realtids klocka
	D	Test av HRUII-Prom
	E	Test av MAP-register
	F	Minnesblockning (4K)
	G	Minnesblockning (32K)
	H	Minnes blockning,HR grafik (30K)
	I	Kretsscheman

APPENDIX A FÄRGREGISTER

I ABC806 finns det 16 st 8 bitars register som används för att ge videoinformation från högupplösnings grafiken.
Registren addresseras med färgvärden från grafikminnet, för varje färgvärde får man från registren ut färginformation om två ´dottar´ på skärmen.

I registren ligger informationen lagrad på följande vis:

D7	D6	D5	D4	D3	D2	D1	D0
P	R	G	B	P	R	G	B
	DOT1				DOT2		

Registren programmeras upp av systemprogramvaran när man i BASIC använder FGCTL BLK+RED+BLU+GRN+YEL+ osv.

När man använder 256*240 mode, d.v.s att man angett fler färger än fyra i FGCTL, programmeras registren så att dot1 och dot2 tänds i samma färg och bildar en grafikpunkt på skärmen.
I den här moden kan man ha 8 färger + 8 färger med prioritet, med prioritet anges om videoinformationen från från högupplösningsgrafiken ska skriva över videoinformationen från teckengenereringen.

Genom att programmera registren lite annorlunda än vad som görs ifrån basic kan man samtidigt få 16 olika färger på högupplösningsgrafiken.

Om man programmerar så att DOT1 och DOT2 får olika färger får man en grafikpunkt på bildskärmen som en kombination av två färger vilket ögat uppfattar som en blandfärg.

På så sätt kan man få 16 olika färger samtidigt på skärmen, i programmexemplet som följer kan man testa de olika färgkombinationerna.

I programmet är det funktionen FNFgctl som skriver in önskade data i färgregistren, funktionen består av en maskinspråksdel som utför instruktionen OUT(C),r som måste användas för att adressera färgregistren (se teknisk beskr.), till funktionen skickas parametrar med som talar om vilken färgkombination som ska skrivas in samt vilket färgnummer den ska motsvaras av.
Färgvärdena för DOT1 och DOT2 räknas fram i funktionen.

Färgvärdet och färnumret laddas i DE-registret vid anropet av maskinspråksrutinen, där sedan färgnumret laddas i B registret och C-registret laddas med outadress 7.

Om funktionen anropas med följande parametrar:

FNFgctl (´RED´,´WHT´,1)

kommer det att innbära att i färgregister 1 kommer följande informationen att skrivas (binärt) 0 1 0 0 0 1 1 1, om sedan en grafikpunkt på skärmen tänds med färgnr 1 (FGPOINT 239,239,1) så kommer den att bestå av en röd och vit dot.
Innan funktionen används måste man använda FGCTL för att sätta HRhanteringen i 256*240 mode.
OBS. att färgparametrarna måste anges med stora bokstäver (RED,GRED osv). Program exempel nästa sida.

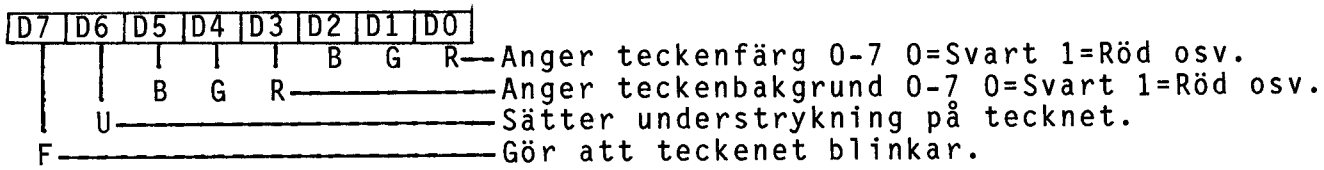
Program för att visa 16 färg.

```
100 ! PROGRAM FÖR ATT TESTA UTÖKADE FÄRGKOMBINATIONER I 256*240 GRAFIKMODE
110 FGCTL BLK+RED+GRN+YEL+BLU+MAG+CYA+WHT+BLK+RED+GRN+YEL+BLU+MAG+CYA+WHT
120 !
130 ; CHR$(12)
140 ; FNHrerase%(255,239,0)
150 !
160 Inramn%=FNRekt%(0,119,241,239,14)
170 Färgbalk%=FNBalk%
180 ; FNText%;
190 !
200 !
210 ; CUR(22,0);
220 INPUT Färg1%,Färg2%,Färgnr
230 ; FNFgct1%(Färg1%,Färg2%,Färgnr);
240 GOTO 210
250 END
1180 !
1190 !
1200 DEF FNText%
1210   ; CUR(19,0) 'Ange färgkombination och färgnummer enligt Färg1,Färg2,Färgn
ummer'
1220   ; CUR(20,0) '  (Färg=BLU,RED,GRN,WHT,BLK,YEL,CYA,MAG  Färgnummer 0-15)'
1240   RETURN ''
1250 FNEND
1260 !
1270 !
1280 DEF FNBalk%
1290   Ant=0
1300   WHILE Ant<16
1310     ; FNFrekt%(Ant*15+1,120,Ant*15+15,238,Ant)
1320     ; CUR(13,5*Ant-2) Ant
1330   Ant=Ant+1 : WEND
1340   RETURN ''
1350 FNEND
1360 !
1370 !
1380 DEF FNHrerase%(X,Y,Xy)
1390   FGPOINT Xy,Xy,0
1400   FGFILL X,Y
1410   RETURN ''
1420 FNEND
1430 !
1440 !
1450 DEF FNFrekt%(Xu,Yu,Xö,Yö,F)
1460   FGPOINT Xu,Yu,F
1470   FGFILL Xö,Yö
1480   RETURN ''
1490 FNEND
1500 !
1510 DEF FNRekt%(Xu,Yu,Xö,Yö,Färg)
1520   FGPOINT Xu,Yu,Färg
1530   FGLINE Xö,Yu
1540   FGLINE Xö,Yö
1550   FGLINE Xu,Yö
1560   FGLINE Xu,Yu
1570   RETURN ''
1580 FNEND
7000 !
7010 !
7020 DEF FNFgct1%(Färg1%,Färg2%,Färgnr) LOCAL Färger%=56,Out7%=6,Dot1,Dot2,Hrc
7030   ! LD   B,D
7040   ! LD   C,7
7050   ! OUT  (C),E
7060   ! RET
7070   Out7%=CHR$(66,14,7,237,89,201)
7080   Färger%="BLKREDGRNYELBLUMAGCYAWHTGBLKGREDGGRNGYELGBLUGMAGGCYAGWHT"
7090   Dot1=INT(INSTR(1,Färger%,Färg1%)/3)*16
7100   Dot2=INT(INSTR(1,Färger%,Färg2%)/3)
7110   Hrc=CALL(VARPTR(Out7%),256*Färgnr+(Dot1+Dot2))
7120   RETURN ''
7130 FNEND
```

APPENDIX B ATTRIBUTEKODER

I ABC 806 finns ett attribute minne som avsöks parallelt med videominnet,för varje teckenposition i videominnet finns en motsvarande position i attributeminnet. Data i attributeminnet består av koder som kommer att påverka motsvarande tecken i videominnet. Ett attribute data består av åtta bitar med följande betydelse:

Normalfallet.



Specialfallet, när bakgrundsfärg är samma som teckenfärgen.

D7	D6	D5	D4	D3	D2	D1	D0		Bgr	Fgr
0	0	X	X	X	X	X	X	--Programmerarens kod,Attributet=Sist valda		
0	1	X	X	X	X	X	X	--Reserverad för framtida behov.		
1	0	X	X	X	X	X	X	--Blank		
1	1	1	1	1	1	1	1	--Elongate, normal höjd	7	7
1	1	1	1	0	1	1	0	--Elongate, dubbel höjd, övre raden.	6	6
1	1	1	0	1	1	0	1	--Elongate, dubbel höjd, undre raden.	5	5

Ett elongerad tecken täcker två teckenpositioner, attributet i tecken position två används för att ange blink,understrykning och färger för det elongerade teckenet.

Dessa koder avkodas i attributavkodarkretsen, vid varje radstart återställs kretsen till 40/80 teckens mode samt vit text på svart bakgrund.
40 teckens mode sätts med OUT 54,129, 80 teckens mode med OUT 54,1, i BASIC ger detta ingen synbar skillnad eftersom drivrutinen för monitorn genom att skriva en attributkod ställer 40/80 tecken.

För att läsa av ett attribut på en positon i attribut minnet får man först läsa motsvarande minnespositionen i video minnet.

Ex. Tecken=PEEK(Vmemadr) : Attribut=INP(53)

Vid varje läsning i videominnet kommer attributdata för den positionen att läggas i en registerkrets (som kan läsas med INP(53)), om man skriver på en position i videominnet kommer de data som finns i samma registerkrets att automatiskt skrivas på motsvarande position i attributminnet.

För att skriva data till registerkretsen används OUT 53, attribut.

Ex. OUT 53,attribut : POKE Vmemadr,Tecken

OBS!
Om man av någon anledning tex. gör en läsning på en adress i bildminnet och sedan skriver på en annan adress i bildminnet kopieras automatiskt attributet från den lästa adressen till den adress man skrev till.

En förteckning över alla attributekoder finns på de efterföljande sidorna.

Attribute koder.
* anger special attribute

Binärt	Dec	Hex	Att.	Bakgr,Förgr.
0 0 0 0 0 0 0 0	0	0	*Sist valda	Bakgr=Förgr
0 0 0 0 0 0 0 1	1	1	STDY+NULN	Blk + Red
0 0 0 0 0 0 1 0	2	2		Grn
0 0 0 0 0 0 1 1	3	3		Yel
0 0 0 0 0 1 0 0	4	4		Blu
0 0 0 0 0 1 0 1	5	5		Mag
0 0 0 0 0 1 1 0	6	6		Cya
0 0 0 0 0 1 1 1	7	7		WHT
0 0 0 0 1 0 0 0	8	8	STDY+NULN	Red + Blk
0 0 0 0 1 0 0 1	9	9	*Sist valda	Bakgr=Förgr
0 0 0 0 1 0 1 0	10	A		Grn
0 0 0 0 1 0 1 1	11	B		Yel
0 0 0 0 1 1 0 0	12	C		Blu
0 0 0 0 1 1 0 1	13	D		Mag
0 0 0 0 1 1 1 0	14	E		Cya
0 0 0 0 1 1 1 1	15	F		WHT
0 0 0 1 0 0 0 0	16	10	STDY+NULN	GRN + Blk
0 0 0 1 0 0 0 1	17	11		Red
0 0 0 1 0 0 1 0	18	12	*Sist valda	Bakgr=Förgr
0 0 0 1 0 0 1 1	19	13		Yel
0 0 0 1 0 1 0 0	20	14		Blu
0 0 0 1 0 1 0 1	21	15		Mag
0 0 0 1 0 1 1 0	22	16		Cya
0 0 0 1 0 1 1 1	23	17		WHT
0 0 0 1 1 0 0 0	24	18	STDY+NULN	YEL + Blk
0 0 0 1 1 0 0 1	25	19		Red
0 0 0 1 1 0 1 0	26	1A		Grn
0 0 0 1 1 0 1 1	27	1B	*Sist valda	Bakgr=Förgr
0 0 0 1 1 1 0 0	28	1C		Blu
0 0 0 1 1 1 0 1	29	1D		Mag
0 0 0 1 1 1 1 0	30	1E		Cya
0 0 0 1 1 1 1 1	31	1F		WHT
0 0 1 0 0 0 0 0	32	20	STDY+NULN	BLU + Blk
0 0 1 0 0 0 0 1	33	21		Red
0 0 1 0 0 0 1 0	34	22		Grn
0 0 1 0 0 0 1 1	35	23		Yel
0 0 1 0 0 1 0 0	36	24	*Sist valda	Bakgr=Förgr
0 0 1 0 0 1 0 1	37	25		Mag
0 0 1 0 0 1 1 0	38	26		Cya
0 0 1 0 0 1 1 1	39	27		Wht
0 0 1 0 1 0 0 0	40	28	STDY+NULN	MAG + Blk
0 0 1 0 1 0 0 1	41	29		Red
0 0 1 0 1 0 1 0	42	2A		Grn
0 0 1 0 1 0 1 1	43	2B		Yel
0 0 1 0 1 1 0 0	44	2C		Blu
0 0 1 0 1 1 0 1	45	2D	*Sist valda	Bakgr=Förgr
0 0 1 0 1 1 1 0	46	2E		Cya
0 0 1 0 1 1 1 1	47	2F		Wht
0 0 1 1 0 0 0 0	48	30	STDY+NULN	YEL + Blk
0 0 1 1 0 0 0 1	49	31		Red
0 0 1 1 0 0 1 0	50	32		Grn
0 0 1 1 0 0 1 1	51	33		Yel
0 0 1 1 0 1 0 0	52	34		Blu
0 0 1 1 0 1 0 1	53	35		Mag
0 0 1 1 0 1 1 0	54	36	*Sist valda	Bakgr=Förgr
0 0 1 1 0 1 1 1	55	37		WHT

Attribute koder.
* anger special attribute

Binärt	Dec	Hex	Att.	Bakgr.	Förgr.
0 0 1 1 1 0 0 0	56	38	STDY+NULN	WHT +	Blk
0 0 1 1 1 0 0 1	57	39			Red
0 0 1 1 1 0 1 0	58	3A			Grn
0 0 1 1 1 0 1 1	59	3B			Yel
0 0 1 1 1 1 0 0	60	3C			Blu
0 0 1 1 1 1 0 1	61	3D			Mag
0 0 1 1 1 1 1 0	62	3E			Cya
0 0 1 1 1 1 1 1	63	3F	*Sist valda	Bakgr=Förgr	
0 1 0 0 0 0 0 0	64	40	*Reserv	Bakgr=Förgr	
0 1 0 0 0 0 0 1	65	41	STDY+ULN	Blk +	Red
0 1 0 0 0 0 1 0	66	42			Grn
0 1 0 0 0 0 1 1	67	43			Yel
0 1 0 0 0 1 0 0	68	44			Blu
0 1 0 0 0 1 0 1	69	45			Mag
0 1 0 0 0 1 1 0	70	46			Cya
0 1 0 0 0 1 1 1	71	47			WHT
0 1 0 0 1 0 0 0	72	48	STDY+ULN	Red +	Blk
0 1 0 0 1 0 0 1	73	49	*Reserv	Bakgr=Förgr	
0 1 0 0 1 0 1 0	74	4A			Grn
0 1 0 0 1 0 1 1	75	4B			Yel
0 1 0 0 1 1 0 0	76	4C			Blu
0 1 0 0 1 1 0 1	77	4D			Mag
0 1 0 0 1 1 1 0	78	4E			Cya
0 1 0 0 1 1 1 1	79	4F			WHT
0 1 0 1 0 0 0 0	80	50	STDY+ULN	GRN +	Blk
0 1 0 1 0 0 0 1	81	51			Red
0 1 0 1 0 0 1 0	82	52	*Reserv	Bakgr=Förgr	
0 1 0 1 0 0 1 1	83	53			Yel
0 1 0 1 0 1 0 0	84	54			Blu
0 1 0 1 0 1 0 1	85	55			Mag
0 1 0 1 0 1 1 0	86	56			Cya
0 1 0 1 0 1 1 1	87	57			WHT
0 1 0 1 1 0 0 0	88	58	STDY+ULN	YEL +	Blk
0 1 0 1 1 0 0 1	89	59			Red
0 1 0 1 1 0 1 0	90	5A			Grn
0 1 0 1 1 0 1 1	91	5B	*Reserv	Bakgr=Förgr	
0 1 0 1 1 1 0 0	92	5C			Blu
0 1 0 1 1 1 0 1	93	5D			Mag
0 1 0 1 1 1 1 0	94	5E			Cya
0 1 0 1 1 1 1 1	95	5F			WHT
0 1 1 0 0 0 0 0	96	60	STDY+ULN	BLU +	Blk
0 1 1 0 0 0 0 1	97	61			Red
0 1 1 0 0 0 1 0	98	62			Grn
0 1 1 0 0 0 1 1	99	63			Yel
0 1 1 0 0 1 0 0	100	64	*Reserv	Bakgr=Förgr	
0 1 1 0 0 1 0 1	101	65			Mag
0 1 1 0 0 1 1 0	102	66			Cya
0 1 1 0 0 1 1 1	103	67			WHT
0 1 1 0 1 0 0 0	104	68	STDY+ULN	MAG +	Blk
0 1 1 0 1 0 0 1	105	69			Red
0 1 1 0 1 0 1 0	106	6A			Grn
0 1 1 0 1 0 1 1	107	6B			Yel
0 1 1 0 1 1 0 0	108	6C			Blu
0 1 1 0 1 1 0 1	109	6D	*Reserv	Bakgr=Förgr	
0 1 1 0 1 1 1 0	110	6E			Cya
0 1 1 0 1 1 1 1	111	6F			WHT

Attribute koder.
* anger special attribute

Binärt	Dec	Hex	Att.	Bakgr. Förgr.
0 1 1 1 0 0 0 0	112	70	STDY+ULN	YEL + Blk
0 1 1 1 0 0 0 1	113	71		Red
0 1 1 1 0 0 1 0	114	72		Grn
0 1 1 1 0 0 1 1	115	73		Yel
0 1 1 1 0 1 0 0	116	74		Blu
0 1 1 1 0 1 0 1	117	75		Mag
0 1 1 1 0 1 1 0	118	76	*Reserv	Bakgr=Förgr
0 1 1 1 0 1 1 1	119	77		WHT
0 1 1 1 1 0 0 0	120	78	STDY+ULN	WHT + Blk
0 1 1 1 1 0 0 1	121	79		Red
0 1 1 1 1 0 1 0	122	7A		Grn
0 1 1 1 1 0 1 1	123	7B		Yel
0 1 1 1 1 1 0 0	124	7C		Blu
0 1 1 1 1 1 0 1	125	7D		Mag
0 1 1 1 1 1 1 0	126	7E		Cya
0 1 1 1 1 1 1 1	127	7F	*Reserv	Bakgr=Förgr
1 0 0 0 0 0 0 0	128	80	*Blank	Bakgr=Förgr
1 0 0 0 0 0 0 1	129	81	FLSH+NULN	Blk + Red
1 0 0 0 0 0 1 0	130	82		Grn
1 0 0 0 0 0 1 1	131	83		Yel
1 0 0 0 0 1 0 0	132	84		Blu
1 0 0 0 0 1 0 1	133	85		Mag
1 0 0 0 0 1 1 0	134	86		Cya
1 0 0 0 0 1 1 1	135	87		WHT
1 0 0 0 1 0 0 0	136	88	FLSH+NULN	Red + Blk
1 0 0 0 1 0 0 1	137	89	*Blank	Bakgr=Förgr
1 0 0 0 1 0 1 0	138	8A		Grn
1 0 0 0 1 0 1 1	139	8B		Yel
1 0 0 0 1 1 0 0	140	8C		Blu
1 0 0 0 1 1 0 1	141	8D		Mag
1 0 0 0 1 1 1 0	142	8E		Cya
1 0 0 0 1 1 1 1	143	8F		WHT
1 0 0 1 0 0 0 0	144	90	FLSH+NULN	Grn + Blk
1 0 0 1 0 0 0 1	145	91		Red
1 0 0 1 0 0 1 0	146	92	*Blank	Bakgr=Förgr
1 0 0 1 0 0 1 1	147	93		Yel
1 0 0 1 0 1 0 0	148	94		Blu
1 0 0 1 0 1 0 1	149	95		Mag
1 0 0 1 0 1 1 0	150	96		Cya
1 0 0 1 0 1 1 1	151	97		WHT
1 0 0 1 1 0 0 0	152	98	FLSH+NULN	YEL + Blk
1 0 0 1 1 0 0 1	153	99		Red
1 0 0 1 1 0 1 0	154	9A		Grn
1 0 0 1 1 0 1 1	155	9B	*Blank	Bakgr=Förgr
1 0 0 1 1 1 0 0	156	9C		Blu
1 0 0 1 1 1 0 1	157	9D		Mag
1 0 0 1 1 1 1 0	158	9E		Cya
1 0 0 1 1 1 1 1	159	9F		WHT
1 0 1 0 0 0 0 0	160	A0	FLSH+NULN	BLU + Blk
1 0 1 0 0 0 0 1	161	A1		Red
1 0 1 0 0 0 1 0	162	A2		Grn
1 0 1 0 0 0 1 1	163	A3		Yel
1 0 1 0 0 1 0 0	164	A4	*Blank	Bakgr=Förgr
1 0 1 0 0 1 0 1	165	A5		Mag
1 0 1 0 0 1 1 0	166	A6		Cya
1 0 1 0 0 1 1 1	167	A7		WHT

Attribute koder.
* anger special attribute

Binärt	Dec	Hex	Att.	Bakgr.	Förgr.
1 0 1 0 1 0 0 0	168	A8	FLSH+NULN	MAG +	Blk
1 0 1 0 1 0 0 1	169	A9			Red
1 0 1 0 1 0 1 0	170	AA			Grn
1 0 1 0 1 0 1 1	171	AB			Yel
1 0 1 0 1 1 0 0	172	AC			Blu
1 0 1 0 1 1 0 1	173	AD	*Blank	Bakgr=Förgr	
1 0 1 0 1 1 1 0	174	AE			Cya
1 0 1 0 1 1 1 1	175	AF			WHT
1 0 1 1 0 0 0 0	176	B0	FLSH+NULN	YEL +	Blk
1 0 1 1 0 0 0 1	177	B1			Red
1 0 1 1 0 0 1 0	178	B2			Grn
1 0 1 1 0 0 1 1	179	B3			Yel
1 0 1 1 0 1 0 0	180	B4			Blu
1 0 1 1 0 1 0 1	181	B5			Mag
1 0 1 1 0 1 1 0	182	B6	*Blank	Bakgr=Förgr	
1 0 1 1 0 1 1 1	183	B7			WHT
1 0 1 1 1 0 0 0	184	B8	FLSH+NULN	WHT +	Blk
1 0 1 1 1 0 0 1	185	B9			Red
1 0 1 1 1 0 1 0	186	BA			Grn
1 0 1 1 1 0 1 1	187	BB			Yel
1 0 1 1 1 1 0 0	188	BC			Blu
1 0 1 1 1 1 0 1	189	BD			Mag
1 0 1 1 1 1 1 0	190	BE			Cya
1 0 1 1 1 1 1 1	191	BF	*Blank	Bakgr=Förgr	
1 1 0 0 0 0 0 0	192	C0	*Används ej	Bakgr=Förgr	
1 1 0 0 0 0 0 1	193	C1	FLSH+ULN	Blk +	Red
1 1 0 0 0 0 1 0	194	C2			Grn
1 1 0 0 0 0 1 1	195	C3			Yel
1 1 0 0 0 1 0 0	196	C4			Blu
1 1 0 0 0 1 0 1	197	C5			Mag
1 1 0 0 0 1 1 0	198	C6			Cya
1 1 0 0 0 1 1 1	199	C7			WHT
1 1 0 0 1 0 0 0	200	C8		Red +	Blk
1 1 0 0 1 0 0 1	201	C9	*Används ej	Bakgr=Förgr	
1 1 0 0 1 0 1 0	202	CA			Grn
1 1 0 0 1 0 1 1	203	CB			Yel
1 1 0 0 1 1 0 0	204	CC			Blu
1 1 0 0 1 1 0 1	205	CD			Mag
1 1 0 0 1 1 1 0	206	CE			Cya
1 1 0 0 1 1 1 1	207	CF			WHT
1 1 0 1 0 0 0 0	208	D0		GRN +	Blk
1 1 0 1 0 0 0 1	209	D1			Red
1 1 0 1 0 0 1 0	210	D2	*Används ej	Bakgr=Förgr	
1 1 0 1 0 0 1 1	211	D3			Yel
1 1 0 1 0 1 0 0	212	D4			Blu
1 1 0 1 0 1 0 1	213	D5			Mag
1 1 0 1 0 1 1 0	214	D6			Cya
1 1 0 1 0 1 1 1	215	D7			WHT
1 1 0 1 1 0 0 0	216	D8		YEL +	Blk
1 1 0 1 1 0 0 1	217	D9			Red
1 1 0 1 1 0 1 0	218	DA			Grn
1 1 0 1 1 0 1 1	219	DB	*Används ej	Bakgr=Förgr	
1 1 0 1 1 1 0 0	220	DC			Blu
1 1 0 1 1 1 0 1	221	DD			Mag
1 1 0 1 1 1 1 0	222	DE			Yel
1 1 0 1 1 1 1 1	223	DF			WHT

Attribute koder.
* ager spaecial attribute

Binärt	Dec	Hex	Att.	Bakgr. Förgr.
1 1 1 0 0 0 0 0	224	E0	FLSH+ULN	BLU + Blk
1 1 1 0 0 0 0 1	225	E1		Red
1 1 1 0 0 0 1 0	226	E2		Grn
1 1 1 0 0 0 1 1	227	E3		Yel
1 1 1 0 0 1 0 0	228	E4	*Används ej	Bakgr=Förgr
1 1 1 0 0 1 0 1	229	E5		Mag
1 1 1 0 0 1 1 0	230	E6		Cya
1 1 1 0 0 1 1 1	231	E7		WHT
1 1 1 0 1 0 0 0	232	E8		MAG + Blk
1 1 1 0 1 0 0 1	233	E9		Red
1 1 1 0 1 0 1 0	234	EA		Grn
1 1 1 0 1 0 1 1	235	EB		Yel
1 1 1 0 1 1 0 0	236	EC		Blu
1 1 1 0 1 1 0 1	237	ED	*DBLE undre	Bakgr=Förgr
1 1 1 0 1 1 1 0	238	EE		Cya
1 1 1 0 1 1 1 1	239	EF		WHT
1 1 1 1 0 0 0 0	240	F0		YEL + Blk
1 1 1 1 0 0 0 1	241	F1		Red
1 1 1 1 0 0 1 0	242	F2		Grn
1 1 1 1 0 0 1 1	243	F3		Yel
1 1 1 1 0 1 0 0	244	F4		Blu
1 1 1 1 0 1 0 1	245	F5		Mag
1 1 1 1 0 1 1 0	246	F6	*DBLE övre	Bakgr=Förgr
1 1 1 1 0 1 1 1	247	F7		WHT
1 1 1 1 1 0 0 0	248	F8	FLSH+ULN	WHT + Blk
1 1 1 1 1 0 0 1	249	F9		Red
1 1 1 1 1 0 1 0	250	FA		Grn
1 1 1 1 1 0 1 1	251	FB		Yel
1 1 1 1 1 1 0 0	252	FC		Blu
1 1 1 1 1 1 0 1	253	FD		Mag
1 1 1 1 1 1 1 0	254	FE		Cya
1 1 1 1 1 1 1 1	255	FF	EL	Bakgr=Förgr

APPENDIX C REALTIDS KLOCKA

För kommunikation med omvärlden är klockkretsen försedd med en seriell data in/ut ledning, data skiftas ut/in i takt med en extern klocksignal på kretsens klockingång.

Överföringen av information går till på följande sätt, efter att ha aktiverat Chip Select signalen till kretsen måste man alltid sända ett 4 bitars ord där dom tre första bitarna är en adress som väljer vilket eller vilka register som avses, den fjärde biten anger om det är läsning eller skrivning som ska utföras.

Informationen i varje register är åtta bitar och består av två BCD kodade siffror.

Registren har följande adresser och innehåll:

Adress	Binär Msb Lsb	Information	Min-Max värde
0	0 0 0	Sekunder	00..59
1	0 0 1	Minuter	00..59
2	0 1 0	Timmar	00..23
3	0 1 1	Datum	01..28/29.30/31
4	1 0 0	Månad	01..12
5	1 0 1	Dag i veckan	01..07
6	1 1 0	År	00..99
7	1 1 1	Alla registren läses eller skrivs i följande ordning, t im,min,dat, mån,år,dag,sek.	

	Bit 4
Läs	1
Skriv	0

För att kontrollera Chip selekt,Klocksignalen och Data in till kretsen använder man I/O adress 54 där följande data används för att kontrollera ovanstående signaler:

```
Out 54,128+5    ger låg CS
Out 54,5        ger hög CS

Out 54,128+6    ger positiv flank på klockingången
Out 54,6        ger negativ flank på klockingången

Out 54,128+7    ger 1 på data in/ut
Out 54,7        ger 0 på data in/ut
```

Vid läsning av data från kretsen använder man I/O adress 55.

```
Inp(55)         ger värdet på data in/ut på bit 7
```

Vid skrivning klockas värdet på data in/ut in till kretsen på den positiva flanken på klocksignalen.

Vid läsning klockas data ut på data in/ut på den negativa flanken på klocksignalen.

I program exemplet är det funktionen FNKlcmd som skriver en 3 bitars register adress och den fjärde biten som anger läsning eller skrivning.

Parametrar som skickas med vid anropet av funktionen är ett heltal mellan 0-6 som anger register adress samt ett heltal som anger läsning eller skrivning 1=läsning och 0 är skrivning.

Om en läsning ska göras anropas sedan funktionen FNTidin som läser av adresserat register och omvandlar BCD värdena till två ASCII värden och returnerar dem i en sträng.

Om en skrivning ska göras anropas istället funktionen FNTidout som skriver till adresserat register, vad som skrivs är den strängparameter som skickas med vid anropet av funktionen, strängen ska bestå av två stycken ASCII siffror som omvandlas till BCD kod i funktionen innan de skrivs till kretsen.

```
Ex ;FNKlcmd (1,1) ; : ;´Min=´ FNTidin
```

Skriver ut aktuellt innehåll i registret som räknar minuter.

```
Ex ;FNKLcmd (1,0) ; : ; FNTidout (´22´)
```

Sätter minutregistret till värdet 22

I programexemplet testas klockan på så sätt att den först sätts till max värdena i varje register (utom sekunder som sätts till 56) sedan väntar programmet tills fyra sekunder har gått för att sedan läsa av registren och se att de innehåller min. värdena.

OBS. om klockan inte fungerar kommer programmet att låsa sig på rad 340, samt att efter det att programmet körts måste klockan sättas till aktuell tid.

I systemprogrammen som följer med på systemdisketten finns det ett program som heter SETCAL som kan användas för att sätta tiden ifall ni inte vill göra ett eget program.

Test av CMOS klocka.

```
100 ; CHR$(12)
110 ; '******************* Test av CMOS-Klocka *******************'
120 ! ------------------------------------------------------------------------
130 Sto=54 ! ------ WRITE SPECIAL PORT
140 Cli=55 ! ------ SERIELL DATA IN FRAN KLOCKKRETS (PA BIT 7)
150 Csl=128+5 ! --- AKTIVERAR CHIP SELEKT TILL KLOCKKRETS
160 Csh=5 ! ------- DEAKTIVERAR CHIP SELEKT TILL KLOCKKRETS
170 Clh=128+6 ! --- KLOCKA FöR DATA IN/UT FRAN KLOCKKRETS (HöG)
180 Cll=6 ! ------- KLOCKA FöR DATA IN/UT FRAN KLOCKKRETS (LAG)
190 Doh=128+7 ! --- SERIELLA DATA OUT TILL KLOCKKRETS (HöG)
200 Dol=7 ! ------- SERIELLA DATA OUT TILL KLOCKKRETS (LAG)
210 Läs=1 : Skriv=0 ! Anger läsning eller skrivning
220 ! ------------- Adress som väljer register att skriva eller läsa till
230 Asec=0 : Amin=1 : Atim=2 : Adat=3 : Amån=4 : Adag=5 : Aår=6 : Aallt=7
240 ! ------------------------------------------------------------------------
250 !
260 DATA 56,59,23,31,12,7,99
270 Reg=0
280 WHILE Reg<7
290   READ Tdat$
300   Setreg$=FNTime$(Reg,Skriv,Tdat$)
310   Reg=Reg+1
320 WEND
330 !
331 ! Om klockan står still blir du hängande här!
340 WHILE FNTime$(Asec,Läs,'')<>'00' : WEND
350 !
360 DATA 00,00,01,01,01,00
370 Reg=1
380 WHILE Reg<7
390   READ Tdat$
400   IF FNTime$(Reg,Läs,'')<>Tdat$ THEN ; 'Fel på klockkretsen i reg. ?  Reg EL
SE ; 'Register' Reg 'OK'
410   Reg=Reg+1
420 WEND
430 !
440 END
1000 !
1010 !
1020 DEF FNTime$(Reg,Rw,Tid$) LOCAL Send$=1
1030   Send$=FNKlcmd$(Reg,Rw)
1040   IF Rw THEN RETURN FNTidin$ ELSE Send$=FNTidout$(Tid$)
1050   RETURN ''
1060 FNEND
1070 !
1080 !
1090 DEF FNKlcmd$(Reg,Rw) LOCAL Adr,Bit
1100   Adr=(Reg*2)+Rw
1110   OUT Sto,Csl
1120   Bit=4
1130   WHILE Bit<8
1140     OUT Sto,Cll,Sto,7+((Adr*2^Bit) AND 128),Sto,Clh
1150     Bit=Bit+1
1160   WEND
1170   RETURN ''
1180 FNEND
1190 !
1200 !
1210 DEF FNTidin$ LOCAL Tid,Bit,Tid$=2
1220   Bit=0
1230   WHILE Bit<8
1250     OUT Sto,Cll : Tid=Tid/2+(INP(Cli) AND 128) : OUT Sto,Clh
1270     Bit=Bit+1
1280   WEND
1290   Tid$=CHR$((Tid AND 240)/16+48)+CHR$((Tid AND 15)+48)
1300   OUT Sto,Csh
1310   RETURN Tid$
1320 FNEND
1330 !
1340 !
1350 DEF FNTidout$(Tid$) LOCAL Tid,Bit
1360   Tid=(ASCII(Tid$)-48)*16+(ASCII(RIGHT$(Tid$,2))-48)
1370   Bit=0
1380   WHILE Bit<8
1390     OUT Sto,Cll,Sto,7+((Tid*2^Bit) AND 128),Sto,Clh
1400     Bit=Bit+1
1410   WEND
1420   OUT Sto,Csh
1430   RETURN ''
1440 FNEND
```

APPENDIX D TEST AV HRUII PROM

I ABC806 finns ett prom HRUII (pos 12G) som innehåller tabeller för de färgvärden som laddas i färgregistren när man kör HR grafiken i ABC800 kompatibel mode (FGCTL 0-255).

Prom:et är på 512*4bit och läses med hjälp av I/O instruktioner.

För att kunna läsa med hjälp av I/O instruktioner måste en speciell I/O instruktion användas, IN r,(C), när CPU:n exekverar den lägger den på adressbitarna A8-A15 ut innehållet i B registret och dessa är anslutna till prom:ets adressingångar A0-A7, register C ska innehåller I/O adress 55 som läggs ut på adressbitarna A0-A7 och avkodas vilket ger CS till prom:et och addresserad minnes position läses till register r.

I/O adress som används är 55 där man får data från prom:et på D3-D0

Med 8 adressbitar kan man bara adressera 256 byte varför A8 på prom:et är kopplat till I/O port 54 och kan sättas låg eller hög för att välja sida i prom:et

```
Ex Out 54,128+2   sätter A8 till 1
   Out 54,2       sätter A8 till 0
```

I pogramexemplet görs en checksummering av prom:et som sedan jämförs med ett känt värde.
För att läsa data från prom:et används funktionen FNInc som består av en maskinspråksdel som utför instruktionen IN r,(C).

```
Maskinkoden är LD B,D
               LD C,E
               IN L,(C)
               RET
```

Vid anropet av funktionen skickas två heltal med, C ska vara den I/O adress som man ska ladda i C-registret samt B ska vara det värde som ska laddas i B-registret (läggs ut på A8-A15, som adresserar prom:ets A0-A7), i retur får man ett heltal som innehåller det värde som lästs in från databussen.

Test av HRUII prom.

```
100 ; CHR$(12)
110 ; '********** Program för att testa HRUII-Prom. **********
120 !
130 Cli=55 ! ----Portadress för HRU-prom, laddas i C-reg,B-reg laddas med A0-A7
140 Sto=54 ! ----Write Special
150 A8l=2 ! -----Adressbit A8 till HRUII-prom sätts låg
160 A8h=128+2 ! -Adressbit A8 till HRUII-prom sätts hög
170 !
180 A8=0
190 OUT Sto,A8l
200 WHILE A8<2
210   Adr=0
220   WHILE Adr<256
230     Chsum=Chsum+(FNInc(Cli,Adr) AND 15)
260     Adr=Adr+1
270   WEND
280   A8=A8+1
290   OUT Sto,A8h
291 WEND
300 IF Chsum=4385. THEN ; 'HRUII Prom OK.' ELSE ; ' Fel i HRUII  Prom'
301 END
2000 !
2010 !
2020 DEF FNInc(C,B) LOCAL Inc$=7,Kadr,Regc,Regb,Regde,Inc,Dat
2030   Inc$=CHR$(197,66,75,237,104,193,201)
2040   Kadr=VARPTR(Inc$)
2050   Regc=C AND 255 : Regb=B AND 255
2060   Regde=(Regb*256)+Regc
2070   Inc=CALL(Kadr,Regde)
2080   Dat=Inc AND 255
2090   RETURN Dat
2100 FNEND
2110 !
2120 !
```

APPENDIX E TEST AV MAP REGISTREN

Vid inblockning av extraminnet i 4K bytes block använder man sig av ett 16*8 bitars register för att välja block i extraminnet som ska kopplas in över CPU:ns minnesarea.

Data kan både skrivas och läsas i registren, vid läsning använder man sig av maskinspråksinstruktionen IN r,(C) och vid skrivning OUT (C),r.

CPU:n lägger vid exekveringen av den instruktionen ut innehållet i register B på adressbitarna A8-A15 samtidigt som C-registrets innehåll läggs ut som en I/O adress på adressbitarna A0-A7.

I det här fallet addresserar A12-A15 position i registren medan A0-A7 avkodas och aktiverar läsning eller skrivning.

I testprogrammet skrivs först data till registren sedan läses dessa tillbaka och jämförs, först med data 170 och sedan medan data 85 för att därefter nollställas.

För att inte inkoppling av extraminnet ska aktiveras vid testningen disablas först minnesblockningen med signalen Externt Memory Enable, Out 54,0 och efter testet när registren nollställts enablas minnesblockningen igen med Out 54,128.

I programmet används funktionen FNOutc(C,B,Dat) för att skriva data till registren, maskinspråksdelen utför instruktionen OUT (C),r
och ser ut så här:

```
LD B,D
LD C,E
LD A,00
OUT (C),A
RET
```

Parametrarna som skickas med är 3 heltal där C ska vara outadress 53 B ska vara registerposition (0-15)*16 (skifta upp registerpos till A12-A15) och Dat de data som ska skrivas till registerpositionen.

För att läsa data från registren används funktionen FNInc(C,B) som beskrivits i appendix D.
Vid anropet av funktionen ska parameter C vara I/O adress 53 och B vara registerposition (0-15)*16, i retur får man ett heltal innehållande data från adresserad registerposition.

Test av MAP-register.

```
100 ; CHR$(12)
110 ; '************************* TEST AV MAP REGISTREN ************************'
: ;
120 !
130 Mao=52 ! ---Mapregistrens outadress, laddas i C-reg, reg.pos. laddas i B-reg
140 Sto=54 ! ---Write Special
150 Dis=0 ! ----Disable minnesblockning (4K)
160 Ene=128 ! --Enable minnesblockning (4K)
170 !
180 !
190 OUT Sto,Dis ! Disable memory mapping
200 Dat=170 : W=FNRegskr
210 IF FNRegläs THEN ; 'Fel i krets ' FNKrets$ ELSE ; 'Första test OK'
220 Dat=85 : W=FNRegskr
230 IF FNRegläs THEN ; 'Fel i krets ' FNKrets$ ELSE ; 'Andra test OK'
240 Dat=0 : W=FNRegskr
250 OUT Sto,Ene ! enable memory mapping
260 END
270 !
280 !
290 DEF FNKrets$
300   Bit=Dat XOR Datläs
310   IF Bit<16 THEN Krets$='1F' ELSE Krets$='2F'
320   RETURN Krets$
330 FNEND
340 !
350 !
360 DEF FNRegskr LOCAL Reg,Rskriv
370   Reg=0
380   WHILE Reg<16
390     Rskriv=FNOutc(Mao,Reg*16,Dat)
400     Reg=Reg+1
410   WEND
420   RETURN 0
430 FNEND
440 !
450 !
460 DEF FNRegläs LOCAL Reg
470   Reg=0
480   WHILE Reg<16
490     Datläs=FNInc(Mao,Reg*16)
500   IF Dat=Datläs THEN Reg=Reg+1 : WEND : RETURN 0
510   RETURN -1
520 FNEND
1000 !
1010 !
1020 DEF FNOutc(C,B,Dat) LOCAL Outc$=9,Kadr,Regc,Regb,Regde,Outc
1030   Outc$=CHR$(197,66,75,62,0,237,121,193,201)
1040   Kadr=VARPTR(Outc$)
1050   POKE Kadr+4,Dat
1060   Regc=C AND 255 : Regb=B AND 255
1070   Regde=(Regb*256)+Regc
1080   Rskr=CALL(Kadr,Regde)
1090   RETURN 0
1100 FNEND
2000 !
2010 !
2020 DEF FNInc(C,B) LOCAL Inc$=7,Kadr,Regc,Regb,Regde,Inc,Dat
2030   Inc$=CHR$(197,66,75,237,104,193,201)
2040   Kadr=VARPTR(Inc$)
2050   Regc=C AND 255 : Regb=B AND 255
2060   Regde=(Regb*256)+Regc
2070   Inc=CALL(Kadr,Regda) AND 255
2090   RETURN Inc
2100 FNEND
2110 !
2120 !
```

APPENDIX F MINNESBLOCKNING 4K

Vid inblockning av extraminnet i 4K block använder man sig av ett 16*8 bitars register för att välja vilket 4K block från extraminnet som ska kopplas in och var blocket ska kopplas in i CPU:ns minnesarea.

Registret adresseras av CPU:ns adressbitar A12-A15 vilket gör att för varje 4K block CPU:n adresserar, adresseras en av 16 positioner i registerkretsen.
Data som läses ut från registret bildar adressbitarna M12-M18 till extraminnet och adresserar ett 4K block i extraminnet och en bit bestämmer om inkoppling ska aktiveras eller ej.

	D7	D6	D5	D4	D3	D2	D1	D0
Registerinnehåll	På/Av	M18	M17	M16	M15	M14	M13	M12

Vid läsning och skrivning av data till registren adresseras dessa som en I/O krets, för att det ska fungera måste man använda sig av en speciell I/O instruktion OUT (C),r och IN r,(C), CPU:n kommer under exekveringen av dessa instruktioner att lägga ut innehållet i register B på adressbitarna A8-A15 och innehållet i C-registret på adressbitarna A0-A7 som en I/O adress.

Adressbitarna A12-A15 kommer vid en skrivning eller läsning att adressera en registerposition medan A0-A7 aktiverar registren för läsning eller skrivning. Register r innehåller de data som ska skrivas eller de data som läst från registret.

Om man t.ex. har skrivit värdet 128+3 till registerposition 0, kommer block 3 i extraminnet att kopplas in över CPU:ns adressområde 0-4095 (block 0), om man skrivit samma värde till register position 6 hade extraminnets block 3 kopplats in i CPU:ns adressområde 28678-30719 (block 6).

Om man från basic vill överföra data till och från extraminnet måste man göra det via en maskinspråksrutin eftersom inkoppling av extraminnet kopplar bort delar av det ordinarie minnet.

I de följande programlistningarna finns en funktion som kan användas för överföring av data mellan extraminne och t.ex en variabel.

Funktionen kallas FNXm4 och innehåller en maskinspråksdel som sköter om inkoppling av extraminnet och överför data med en så kallad blockmove instruktion.
Parametrar in till funktionen är tre heltal där Från ska vara en adress varifrån data ska hämtas, Till ska vara den adress dit data ska skrivas och Xmembl väljer vilket block i extraminnet som ska kopplas in.

I funktionen bestämmer Cpubl vilket CPU block som extraminnet ska kopplas in över (block 0) och Ant hur många data som ska förflyttas (256), dessa värden kan ändras efter behov.

```
Ex 10 DIM Data =256
   20 Data =string¤(256,0)
   30 Läs=FNXm4(0,varptr(data ),0)
   40 Skriv=FNXm4(varptr(data ),256,0)
```

Den adress som adresserar extra minnet får man själv hålla reda på eftersom den är beroende av var i CPU:ns minnes area man kopplar in extraminnet. I exemplet läses 256 bytes från adress 0 i block 0 och flyttas upp till adress 256 och uppåt i block 0.

```
6000 !
6010 !
6020 DEF FNXm4(Från,Till,Xmembl) LOCAL Adress,Mkod%=35,Ant,Cpubl,Move
6030   Mkod%=CHR$(243,197,6,0,62,0,14,52,203,255,237,121,197,33,0,0,17,0,0)
6040   Mkod%=Mkod%+CHR$(1,0,0,237,176,193,237,120,203,191,237,121,193,251,201)
6050   Adress=VARPTR(Mkod%)
6060   Ant=256
6070   Cpubl=0
6080   POKE Adress+3,Cpubl*16
6090   POKE Adress+5,Xmembl
6100   POKE Adress+14,Från,SWAP%(Från)
6110   POKE Adress+17,Till,SWAP%(Till)
6120   POKE Adress+20,Ant,SWAP%(Ant)
6130   Move=CALL(Adress)
6140   RETURN 0
6150 FNEND
6160 !
6170 !
```

Funktionen FNXm4.

Maskinkodsprogrammet som används i funktionen finns listat på nästa sida.

I det efterfäljande testprogrammet skrivs först data till ett block i extraminnet, sedan läses data tillbaka och jämförs med det skrivna, om data är olika skrivs adress och vilken krets felet finns i ut.

Testet är mycket enkelt och talar egentligen bara om vilken krets det är fel i, om felet ligger i minne-skretsarna.
Ligger felet i adresseringen eller dataläsningen får man indikering på fel i minneskretsarna fast det inte är det.

Minneskartan visar organisationen av minnet vid 4K blockning.

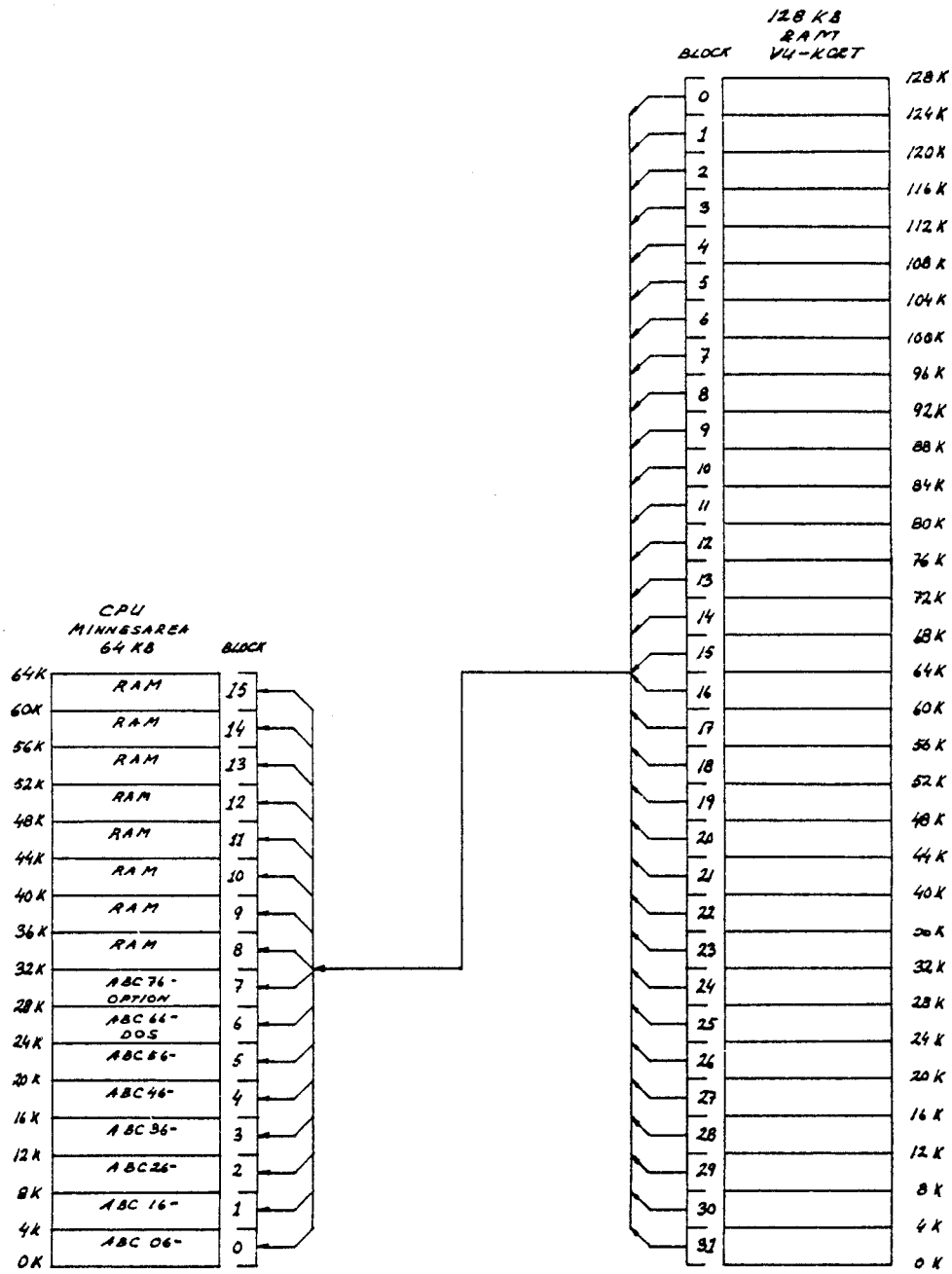
Maskinkod i funktion FNXm4.

```
ASMZ-4.46                                        ERRORS:   0    84-06-28    11:30:56    PAGE  1

   LOCATION PLC CODE      ARG     LC  SOURCE STATEMENT ----------------------------------------------------

   0000                           1          ZPROG
                                  2 ;INKOPPLING AV EXTRAMINNE ABC806 MED 4K BLOCKNING
   0000                           3          *PAGE   72
   0000                0000       4          ORG 0
                                  5 ;
                                  6 ;
   0000                0034       7 MAD      EQU  52              ;ADRESS TILL MAP REGISTER
                                  8 ;
                                  9 ;
                                 10 ;
                                 11 ;
   0000                0000      12 CPUBL    EQU  00H             ;LADDAS FRAN BASIC,VALJER ETT 4 KB
                                 13                               ;BLOCK I CPU:NS MINNES AREA (0-15)
                                 14                               ;VID INBLOCKNING FRAN BASIC FAR
                                 15                               ;BARA BLOCK 0-7 ANVANDAS.
                                 16 ;
   0000                0000      17 XMEMBL   EQU  00H             ;LADDAS FRAN BASIC,VALJER ETT 4 KB
                                 18                               ;BLOCK I 128 KB MINNET. (0-31)
                                 19 ;
                                 20 ;
   0000                0000      21 FRAN     EQU  0000H           ;ADRESS DAR DATA HAMTAS FRAN
   0000                0000      22 TILL     EQU  0000H           ;ADRESS DIT DATA FLYTTAS
   0000                0000      23 ANT      EQU  0000H           ;ANTAL DATA SOM SKA FLYTTAS
                                 24                               ;ALLA TRE VARDENA LADDAS FRAN BASIC
                                 25 ;
                                 26 ;
   0000      F3                 27          DI
   0001      C5                 28          PUSH BC
   0002      0600               29          LD   B,CPUBL         ;*16 FRAN BASIC
   0004      3E00               30          LD   A,XMEMBL
   0006      0E34               31          LD   C,MAD
   0008      CBFF               32          SET  7,A
   000A      ED79               33          OUT  (C),A           ;MINNESBLOCK INKOPPLAT
                                 34 ;
   000C      C5                 35          PUSH BC
   000D      210000             36          LD   HL,FRAN
   0010      110000             37          LD   DE,TILL
   0013      010000             38          LD   BC,ANT
   0016      EDB0               39          LDIR                 ;ÖVERFÖR DATA
   0018      C1                 40          POP  BC
                                 41 ;
   0019      ED78               42          IN   A,(C)
   001B      CBBF               43          RES  7,A
   001D      ED79               44          OUT  (C),A           ;MINNESBLOCK BORTKOPPLAT
   001F      C1                 45          POP  BC
   0020      FB                 46          EI
   0021      C9                 47          RET
                                 48 .
   0022      0000               49          END  0

   ERRORS :   0   WARNINGS :   0
```

Testprogram med 4K Blockning.

```
100 ; CHR$(12)
110 ; '***********  TEST AV 128K BYTE MINNET PA VU-KORTET ABC806 ***************

120 ; '***********            med 4K blockning                  ***************
' : ;
130 !
140 DIM Ts$=4096,Ts1$=4096
150 Ts$=STRING$(4096,0) : Ts1$=STRING$(4096,0)
160 POKE VAROOT(Ts$)+2,0,0
170 !
180 !
190 IF FNMemtest THEN ; 'Fel på adress' FNAdr. 'i krets(?)' FNKrets$ '   Prova m
ed att byta.' ELSE ; : ; 'Minnestest OK.'
200 END
1000 !
1010 !
1020 DEF FNKrets$ LOCAL Raw$=2,Col$=21
1030   IF Adr AND 1 THEN Raw$='I' ELSE Raw$='H'
1040   IF Bit AND 128 THEN Col$='.12'
1050   IF Bit AND 64 THEN Col$=Col$+',11'
1060   IF Bit AND 32 THEN Col$=Col$+',10'
1070   IF Bit AND 16 THEN Col$=Col$+',9'
1080   IF Bit AND 8 THEN Col$=Col$+',8'
1090   IF Bit AND 4 THEN Col$=Col$+',7'
1100   IF Bit AND 2 THEN Col$=Col$+',6'
1110   IF Bit AND 1 THEN Col$=Col$+',5'
1120   RETURN Raw$+Col$
1130 FNEND
2000 !
2010 !
2020 DEF FNAdr. LOCAL Sdata,Ldata
2030   Adr=1
2040   WHILE Adr<=4096
2050   IF MID$(Ts$,Adr,1)=MID$(Ts1$,Adr,1) THEN Adr=Adr+1 : WEND : RETURN 0
2060   Sdata=ASCII(MID$(Ts$,Adr,1)) : Ldata=ASCII(MID$(Ts1$,Adr,1))
2070   Adr=Adr-1
2080   Bit=Sdata XOR Ldata AND 255
2090   RETURN (4096.*Bladr)+Adr
2100 FNEND
3000 !
3010 !
3020 DEF FNMemtest LOCAL Xmskriv,Xmläs
3030   Bladr=0
3040   WHILE Bladr<32.
3050     Xmskriv=FNXm4(VARPTR(Ts$),4096,Bladr)
3060     Xmläs=FNXm4(4096,VARPTR(Ts1$),Bladr)
3070   IF Ts$=Ts1$ THEN ; 'Block' Bladr, : Bladr=Bladr+1. : WEND : RETURN 0
3080   RETURN -1
3090 FNEND
6000 !
6010 !
6020 DEF FNXm4(Från,Till,Xmembl) LOCAL Adress,Mkod$=35,Ant,Cpubl
6030   Mkod$=CHR$(243,197,6,0,62,0,14,52,203,255,237,121,197,33,0,0,17,0,0)
6040   Mkod$=Mkod$+CHR$(1,0,0,237,176,193,237,120,203,191,237,121,193,251,201)
6050   Adress=VARPTR(Mkod$)
6060   Ant=4096
6070   Cpubl=1
6080   POKE Adress+3,Cpubl*16
6090   POKE Adress+5,Xmembl
6100   POKE Adress+14,Från,SWAP%(Från)
6110   POKE Adress+17,Till,SWAP%(Till)
6120   POKE Adress+20,Ant,SWAP%(Ant)
6130   Z=CALL(Adress)
6140   RETURN 0
6150 FNEND
6160 !
6170 !
```

Minneskarta för 4K blockning.



APPENDIX G MINNESBLOCKNING 32K

Vid 32k blockning kopplas extraminnet in över CPU:ns adressområde 0-32k, vilket innebär att även videominnet kopplas bort.
Extraminnet delas in i fyra block om 32K bytes, vilket block som ska kopplas in bestäms via en register krets som programmeras med OUT 6,Data.
Data till registret har följande innehåll och betydelse:

D7	D6	D5	D4	D3	D2	D1	D0
M18	M17	M16	M15	V17	V16	V15	V14

M15-M18 väljer vilket block från extraminnet som ska kopplas in över CPU:ns adrsesområde 0-32K och V14-V17 väljer vilket block i extraminnet som en grafikbild ska genereras från.

Med fyra bitar kan man välja 16 block och är avsett för att klara av en framtida minnesutökning, för tillfället används bara två bitar för att välja ett av fyra block (M15-M16, V14-V15).

Organisationen av minnet vid 32K blockning visas i minneskartan på nästa sida.

För att aktivara inkopplingen av valt minnesblock använder man sig av en av kontrollsignalerna (DTR) på DART kretsen. Signalen kontrolleras via ett register i DART kretsen.

Dart kretsen har ett antal register som kan programmeras för olika funktioner, för att välja ut det register som kontrollerar DTR signalen (register 5) skriver man OUT 35,5 och för att skriva data till register 5 skriver man sedan OUT 35,Data.
Den bit i registret som kontrollerar DTR signalen är Bit 7, om den sätts till 1 kommer extraminnet att kopplas in och om den sätts låg kopplas extraminnet bort.

Default värde för register 5 är 68H (104D) och bör inte förändras eftersom det påverkar sändning till tangentbordet.

In och urkoppling av extraminnet kan inte göras från basic eftersom hela rom arean kopplas bort vid inkopplingen av extra minnet.

Hur man från basic överför data till och från extraminnet med hjälp av 32K blockning visas i funktionen FNXm32.

Funktionen innehåller en maskinspråksdel som sköter om inkoppling och överföring av data.

Parametrar in till funktionen är tre heltal, där Från anger adress där data hämtas, Till anger en adress dit data flyttas och Xmbl anger vilket block från extraminnet som ska kopplas in (0-3).
Antalet data som flyttas har i funktionen angetts till 256 och kan ändras efter behov. Adress till extraminnet blir här 0-32K i varje block som kopplas in.

```
Ex 10 DIM Dat =256
   20 Dat =STRING¤(256,0)
   30 Läs =FNXm32(0,VARPTR(Dat ),0)
   40 Skriv=FNXm32(VARPTR(Dat ),256,0)
```

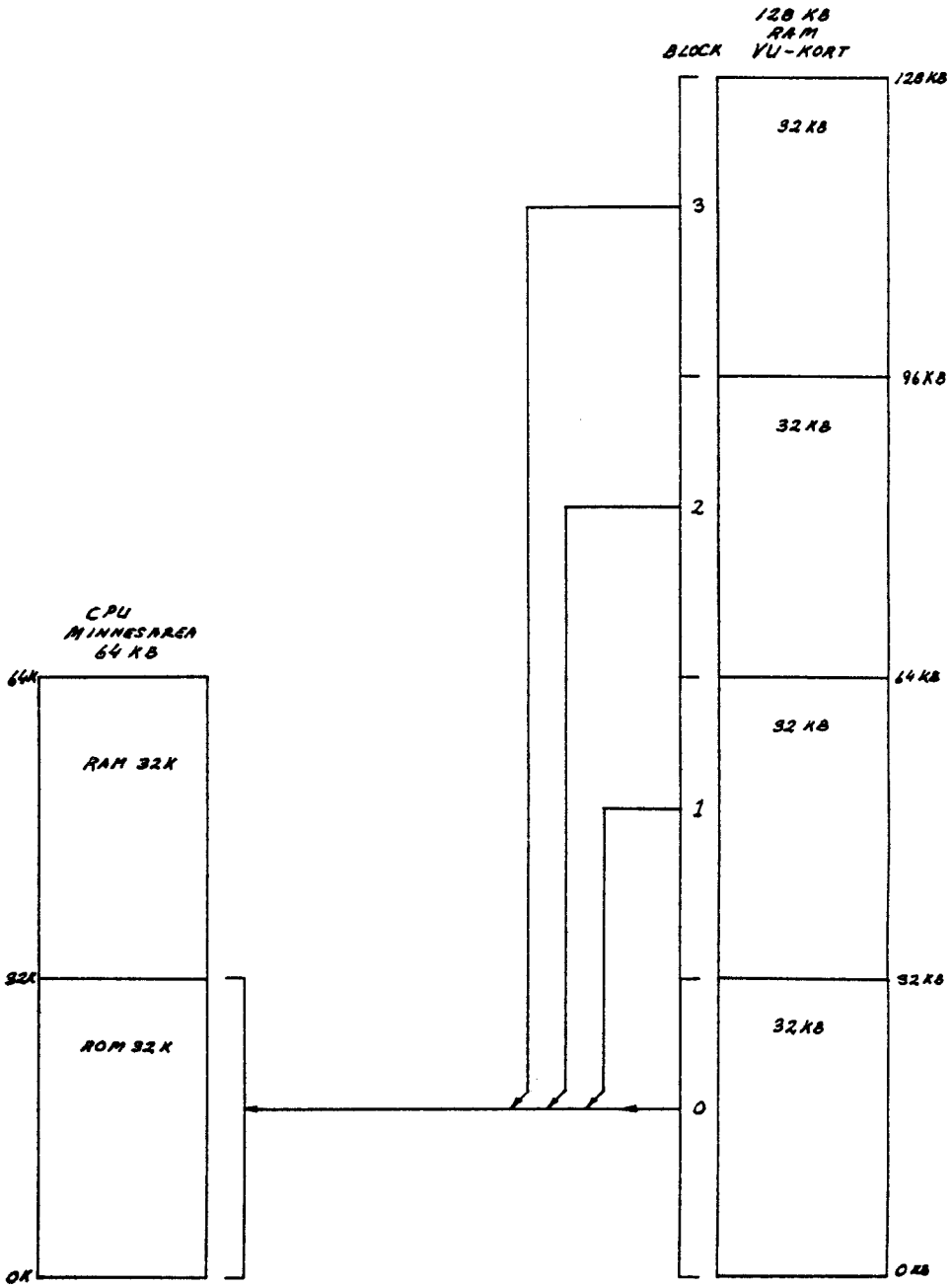
I exemplet flyttas data från adress 0-255 upp till adress 256-511 i extraminnets block 0.

I det efterföljande testprogrammet testas extraminnet på samma sätt som beskrivits för testprogrammet i appendix F.

```
5000 !
5010 !
5020 DEF FNXm32(Från,Till,Xmbl) LOCAL Mkod%=38,Ant,Adress,Move
5030   Mkod%=CHR%(243,197,62,5,6,104,14,35,203,248,237,121,237,65,197,33,0,0)
5040   Mkod%=Mkod%+CHR%(17,0,0,1,0,0,237,176,193,62,5,203,184,237,121,237,65)
5050   Mkod%=Mkod%+CHR%(193,251,201)
5060   Adress=VARPTR(Mkod%)
5070   Ant=256
5080   OUT 6,Xmbl*16
5090   POKE Adress+17,Från,SWAP%(Från)
5100   POKE Adress+20,Till,SWAP%(Till)
5110   POKE Adress+23,Ant,SWAP%(Ant)
5120   Move=CALL(Adress)
5130   RETURN 0
5140 FNEND
5150 !
```

Funktionen FNXm32.

Minneskarta 32K blockning.



Maskinspråksrutin i FNxm32.

```
                                                ERRORS:    0    84-07-02    13:24:28    PAGE   1

LOCATION PLC CODE      ARG     LC  SOURCE STATEMENT ----------------------------------------------------------

 0000                           1          ZPRO6
                                2 ;INKOPPLING AV EXTRAMINNE ABCB06 MED 32K BLOCKNING
 0000                           3          *PAGE 72
 0000                0000       4          ORG 0
                                5 ;
                                6 ;
 0000                0023       7 DARTC    EQU   35              ;I/O ADRESS FÖR DART KONTROLL
 0000                0068       8 DV5      EQU   068H            ;DEFAULTVÄRDE FÖR REG5 I DART
 0000                0005       9 REG5     EQU   5               ;ADRESS TILL REG5
 0000                0007      10 DTR      EQU   7               ;BIT I REG5 SOM STYR DTR
                               11 ;
                               12 ;
 0000                0000      13 FRAN     EQU   0000H           ;ADRESS DÄR DATA HÄMTAS FRÅN
 0000                0000      14 TILL     EQU   0000H           ;ADRESS DIT DATA FLYTTAS
 0000                0000      15 ANT      EQU   0000H           ;ANTAL DATA SOM SKA FLYTTAS
                               16                                ;ALLA TRE VÄRDENA LADDAS FRÅN BASIC
                               17 ;
                               18 ;
 0000       F3              19          DI
 0001       C5              20          PUSH  BC
 0002       3E05            21          LD    A,REG5
 0004       0668            22          LD    B,DV5
 0006       0E23            23          LD    C,DARTC
 0008       CBF8            24          SET   DTR,B
 000A       ED79            25          OUT   (C),A
 000C       ED41            26          OUT   (C),B             ;MINNESBLOCK INKOPPLAT
                               27 ;
 000E       C5              28          PUSH  BC
 000F       210000          29          LD    HL,FRAN
 0012       110000          30          LD    DE,TILL
 0015       010000          31          LD    BC,ANT
 0018       EDB0            32          LDIR                    ;ÖVERFÖR DATA
 001A       C1              33          POP   BC
                               34 ;
 001B       3E05            35          LD    A,REG5
 001D       CB88            36          RES   DTR,B
 001F       ED79            37          OUT   (C),A
 0021       ED41            38          OUT   (C),B             ;MINNESBLOCK BORTKOPPLAT
 0023       C1              39          POP   BC
 0024       FB              40          EI
 0025       C9              41          RET
                               42 .
 0026       0000            43          END   0

 ERRORS :   0   WARNINGS :   0
```

Testprogram minnesdelning 32K block.

```
100 : CHR$(12)
110 ; '***********  TEST AV 128K BYTE MINNET PA VU-KORTET ABC806 **************'
111 ; '***********            med 32K blockning                  **************'
 : ;
120 !
130 DIM Ts$=4096,Ts1$=4096
140 Ts$=STRING$(4096,128+32+8+2) : Ts1$=STRING$(4096,0)
150 !
160 !
170 IF FNMemtest THEN ; 'Fel på adress' FNAdr. 'i krets(?)' FNKrets$ '   Prova m
ed att byta.' ELSE ; : ; 'Minnestest OK.'
180 END
1000 !
1010 !
1020 DEF FNKrets$ LOCAL Raw$=2,Col$=21
1030   IF Adr AND 1 THEN Raw$='I' ELSE Raw$='H'
1040   IF Bit AND 128 THEN Col$=',12'
1050   IF Bit AND 64 THEN Col$=Col$+',11'
1060   IF Bit AND 32 THEN Col$=Col$+',10'
1070   IF Bit AND 16 THEN Col$=Col$+',9'
1080   IF Bit AND 8 THEN Col$=Col$+',8'
1090   IF Bit AND 4 THEN Col$=Col$+',7'
1100   IF Bit AND 2 THEN Col$=Col$+',6'
1110   IF Bit AND 1 THEN Col$=Col$+',5'
1120   RETURN Raw$+Col$
1130 FNEND
2000 !
2010 !
2020 DEF FNAdr. LOCAL Sdata,Ldata
2030   Adr=1
2040   WHILE Adr<=4096
2050   IF MID$(Ts$,Adr,1)=MID$(Ts1$,Adr,1) THEN Adr=Adr+1 : WEND : RETURN 0
2060   Sdata=ASCII(MID$(Ts$,Adr,1)) : Ldata=ASCII(MID$(Ts1$,Adr,1))
2070   Adr=Adr-1
2080   Bit=Sdata XOR Ldata AND 255
2090   RETURN (4096.*Bladr)+Adr
2100 FNEND
3000 !
3010 !
3020 DEF FNMemtest LOCAL Xmskriv,Xmläs
3030   Xmbl=0
3040   WHILE Xmbl<4
3050     Bladr=0
3060     WHILE Bladr<8
3070       Xmskriv=FNXm32(VARPTR(Ts$),Bladr*4096,Xmbl)
3080       Xmläs=FNXm32(Bladr*4096,VARPTR(Ts1$),Xmbl)
3090       IF Ts$<>Ts1$ THEN RETURN -1
3100   ; 'Block' Xmbl 'Adress' 4096*Bladr, : Bladr=Bladr+1 : WEND : Xmbl=Xmbl+1
: ; : WEND : RETURN 0
3110 FNEND
5000 !
5010 !
5020 DEF FNXm32(Från,Till,Xmbl) LOCAL Mkod$=38,Ant,Adress,Move
5030   Mkod$=CHR$(243,197,62,5,6,104,14,35,203,248,237,121,237,65,197,33,0,0)
5040   Mkod$=Mkod$+CHR$(17,0,0,1,0,0,237,176,193,62,5,203,184,237,121,237,65)
5050   Mkod$=Mkod$+CHR$(193,251,201)
5060   Adress=VARPTR(Mkod$)
5070   Ant=4096
5080   OUT 6,Xmbl*16
5090   POKE Adress+16,Från,SWAP%(Från)
5100   POKE Adress+19,Till,SWAP%(Till)
5110   POKE Adress+22,Ant,SWAP%(Ant)
5120   Move=CALL(Adress)
5130   RETURN 0
5140 FNEND
5150 !
5160 !
```

APPENDIX H MINNESBLOCKNING 30K, HR-GRAFIK

Vid 30K blockning kopplas extraminnet in över CPU:ns adressområde 0-30K, minnet blir här indelat i 4 block om 30K och mellan varje block finns ett block på 2K byte, se minneskarta för 30K blockning.

Vilket block som kopplas in över CPU:ns minnesarea bestäms på samma sätt som vid 32K blockning med OUT 6,Data.

Inkopplingen av extraminnet görs här genom att CPU:n gör en instruktionshämtning på adresserna 7800H-7FFFH och om den instruktionen innebär en operand läsning eller skrivning på adressområde 0-30K kommer valt block i extraminnet att addresseras.

För att överföra data mellan extraminne och primärminnet kan man utnyttja en subrutin som finns inlagd med start på adress 7FFDH (32765D), subrutinen består av instruktionerna LDIR och RET.

Innan man anropar subrutinen laddar man register HL med en adress där data hämtas, register DE med en adress dit data ska flyttas och register BC med antalet data.

Ett programexempel som använder sig av detta inkopplingssätt visas på en av de efterföljande sidorna, i programmet är det funktionen FNHrmove som utför överföringen av data.

Eftersom 30K blockningen används vid bearbetning av grafikbilder, har funktionen utformats så att man kan läsa eller skriva data till en grafiklinje.
Vid anropet ska man ange en adress som pekar på en buffert, vilken linje som adresseras (0-239) samt om om data ska skrivas eller läsas (läs=1,skriv=0)

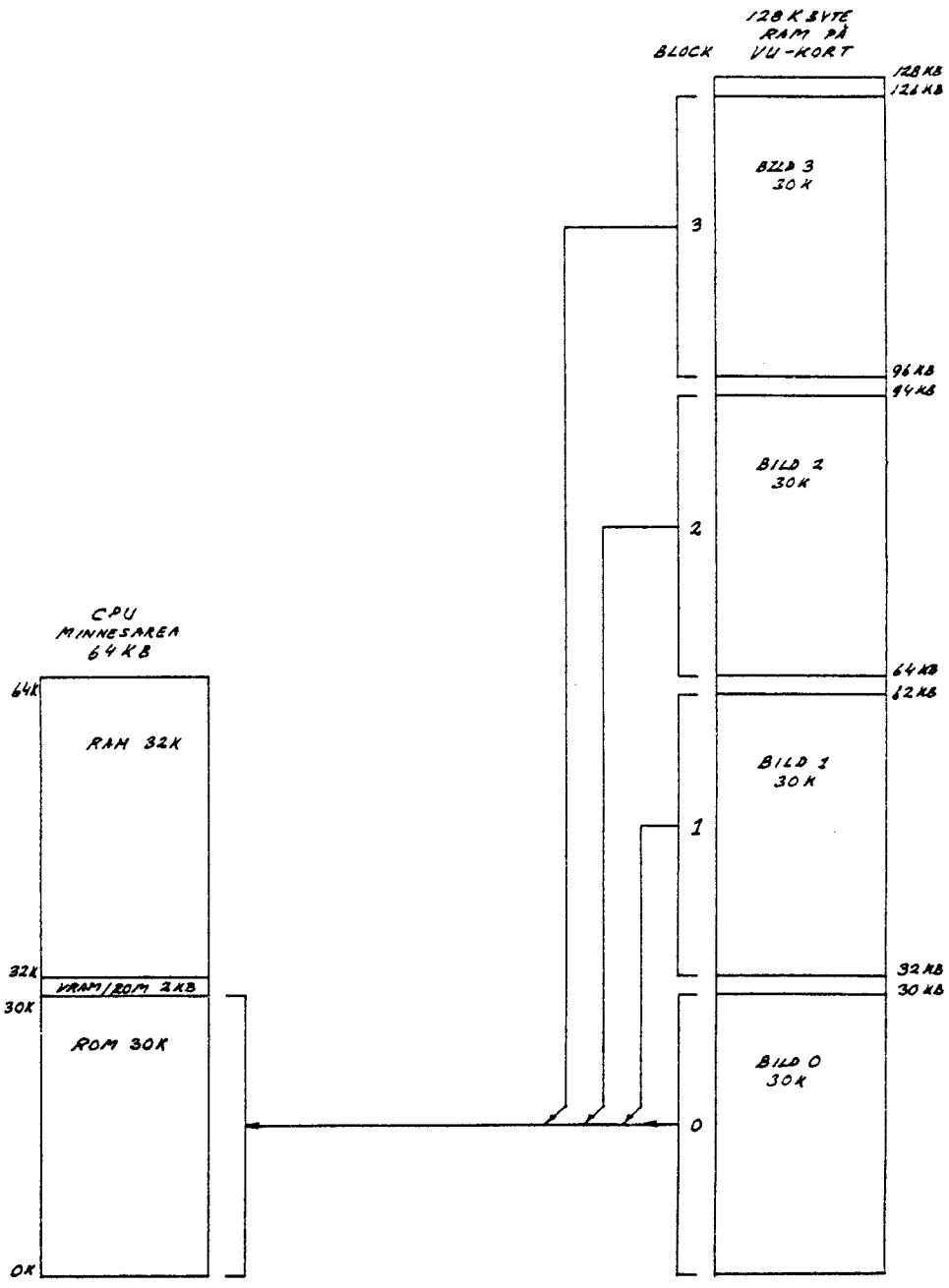
```
Ex 10 DIM Buffert =128
   20 Buffert =STRING (128,0) 30 Läs=1:Skriv=0
   40 Badr=VARPTR(Buffert )
   50 Ll=FNHrmove(Badr,1,Läs) !Läs linje 1
   60 Sl=FNHrmove(Badr,2,Skriv) ! Skriv till linje 2
```

För att välja vilken bild som adresseras använder man antingen FGPICTURE eller OUT 6,Badr*16+Vadr, Badr är den bild som ska bearbetas och Vadr är den bild som ska visas.

Programexemplet kan användas för att spara bilder på en fil samt för att läsa tillbaka filen till HR-grafiken.

För att spara en bild anropar man funktionen FNHrsave(-Filnamn) och för att hämta en bild anropar man funktionen FNHrload(Filnamn).

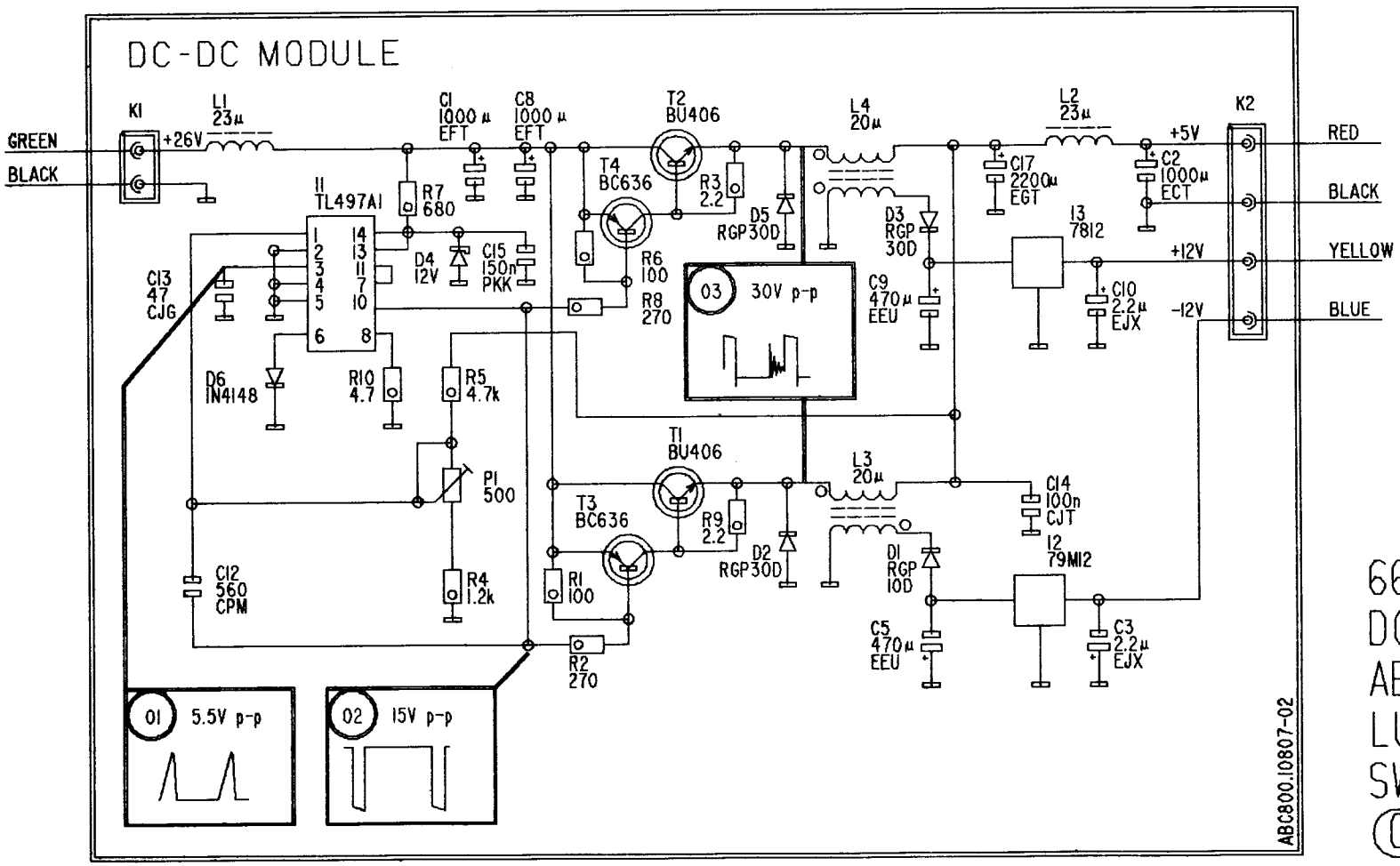
```
EX 10 FGPICTURE 0,1,3
   20 FGCTL BLK+RED+GRN+BLU
   30 FGPOINT 0,0,1
   40 FGFILL 511,239
   50 Sparabild=FNHrsave("TEST.PIC")
   60 FGPOINT 0,0,0
   70 FGFILL 511,239
   80 Hämtabild=FNHrload("TEST.PIC")
   90 END
```



Minneskarta för 30K blockning (eller HR-Bilder).

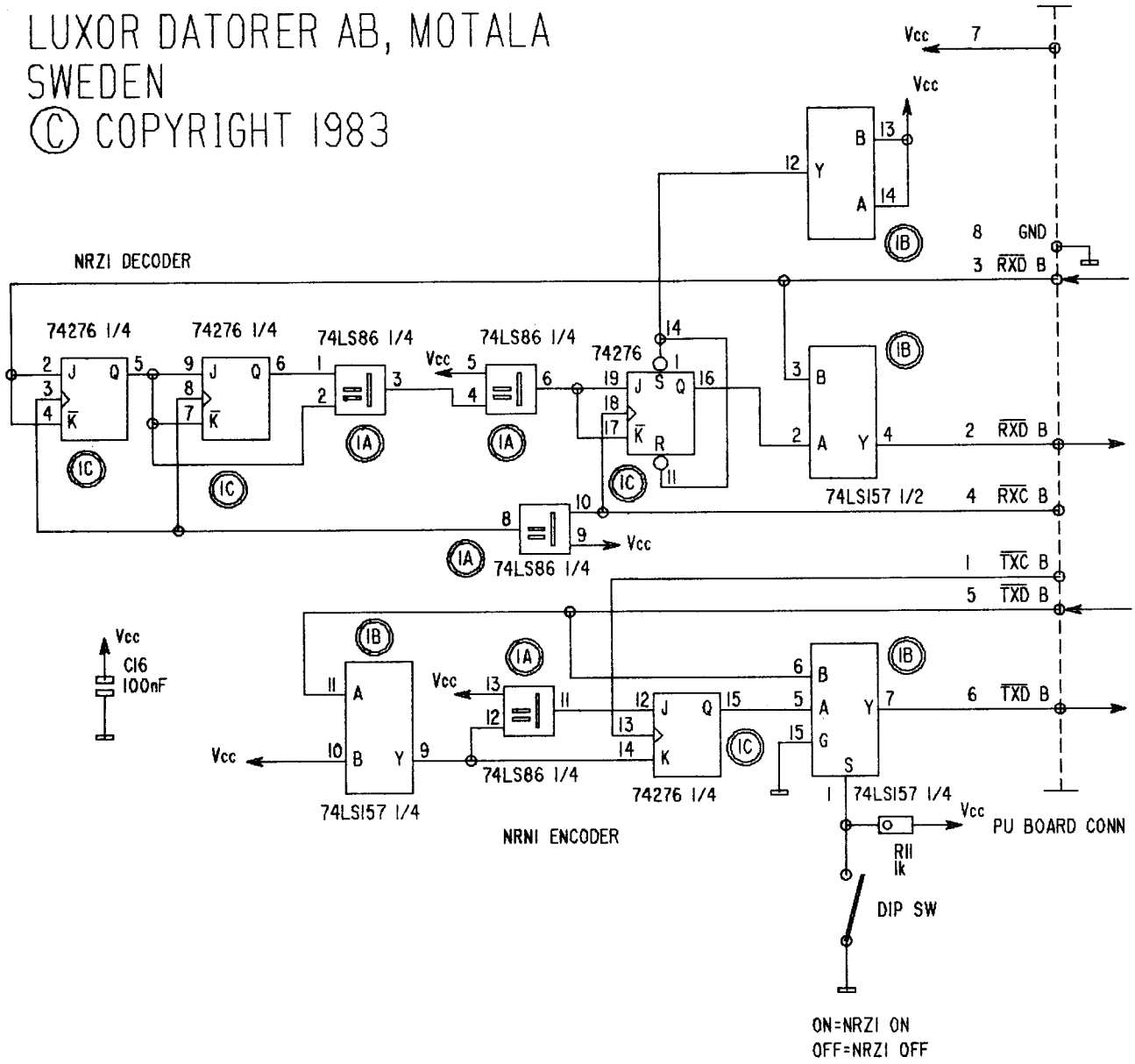
Programexempel med 30K blockning.

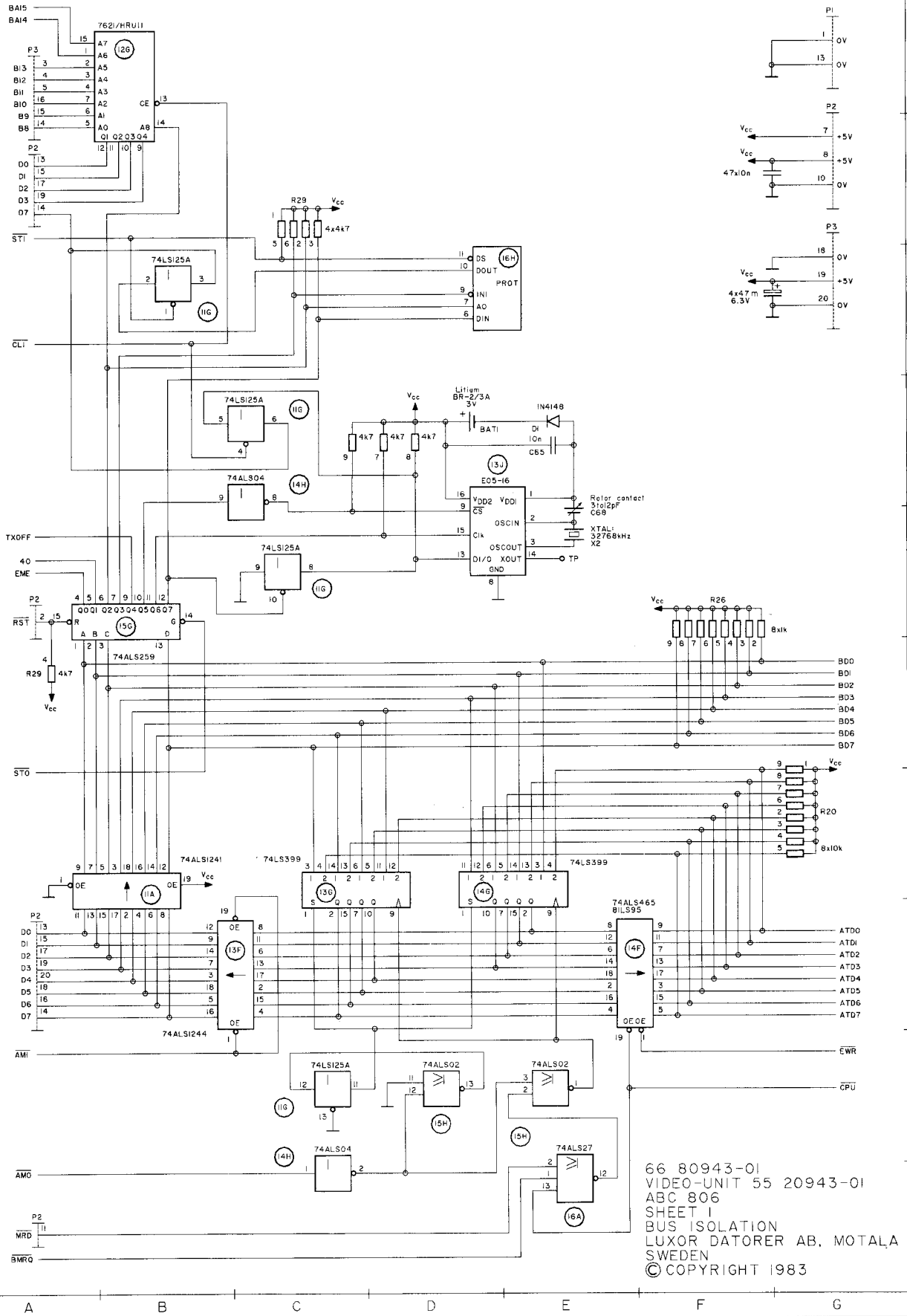
```
5000 DEF FNHrload%(Filnamn$) LOCAL Nr%,Putline%,Linje%,Buffert$=128
5010   Nr%=99%
5020   IF FNOpen%(Filnamn$,Nr%) THEN RETURN ERRCODE
5030   Buffert$=STRING$(128%,0%)
5040   Linje%=0%
5060   WHILE Linje%<240%
5070     GET #Nr% Buffert$ COUNT 128%
5080     Putline%=FNHrmove%(VARPTR(Buffert$),Linje%,0%)
5090     Linje%=Linje%+1%
5100   WEND
5110   CLOSE Nr%
5120   RETURN 0%
5130   RETURN ERRCODE
5140 FNEND
5150 !
5160 !
6000 DEF FNHrsave%(Filnamn$) LOCAL Nr%,Getline%,Linje%,Buffert$=128
6010   Nr%=99%
6020   IF FNPrepare%(Filnamn$,Nr%) THEN RETURN ERRCODE
6030   Buffert$=STRING$(128%,0%)
6040   Linje%=0%
6055   WHILE Linje%<240%
6060     Getline%=FNHrmove%(VARPTR(Buffert$),Linje%,1%)
6070     PUT #Nr%,Buffert$
6080     Linje%=Linje%+1%
6090   WEND
6100   CLOSE Nr%
6110   RETURN 0%
6120   RETURN ERRCODE
6130 FNEND
6998 !
6999 !
7000 DEF FNOpen%(Fil$,Filnr%)
7010   ON ERROR GOTO 7040
7020   OPEN Fil$ AS FILE Filnr%
7030   RETURN 0%
7040   RETURN ERRCODE
7050 FNEND
7060 !
7070 !
8000 DEF FNPrepare%(Fil$,Filnr%)
8010   ON ERROR GOTO 8040
8020   PREPARE Fil$ AS FILE Filnr%
8030   RETURN 0%
8040   RETURN ERRCODE
8050 FNEND
8060 !
8070 !
9000 DEF FNHrmove%(Bufadr%,Linje%,Rw%) LOCAL Stadr%,Från%,Till%,Ant%,Move%,Mkod$
=12
9010   Mkod$=CHR$(33%,0%,0%,17%,0%,0%,1%,0%,0%,195%,253%,127%)
9020   Stadr%=VARPTR(Mkod$) : Ant%=128%
9030   IF Rw%=0% THEN Från%=Bufadr% : Till%=Linje%*128%
9040   IF Rw%=1% THEN Till%=Bufadr% : Från%=Linje%*128%
9050   POKE Stadr%+1%,Från%,SWAP%(Från%)
9060   POKE Stadr%+4%,Till%,SWAP%(Till%)
9070   POKE Stadr%+7%,Ant%,SWAP%(Ant%)
9080   Move%=CALL(Stadr%)
9090   RETURN 0%
9100 FNEND
```



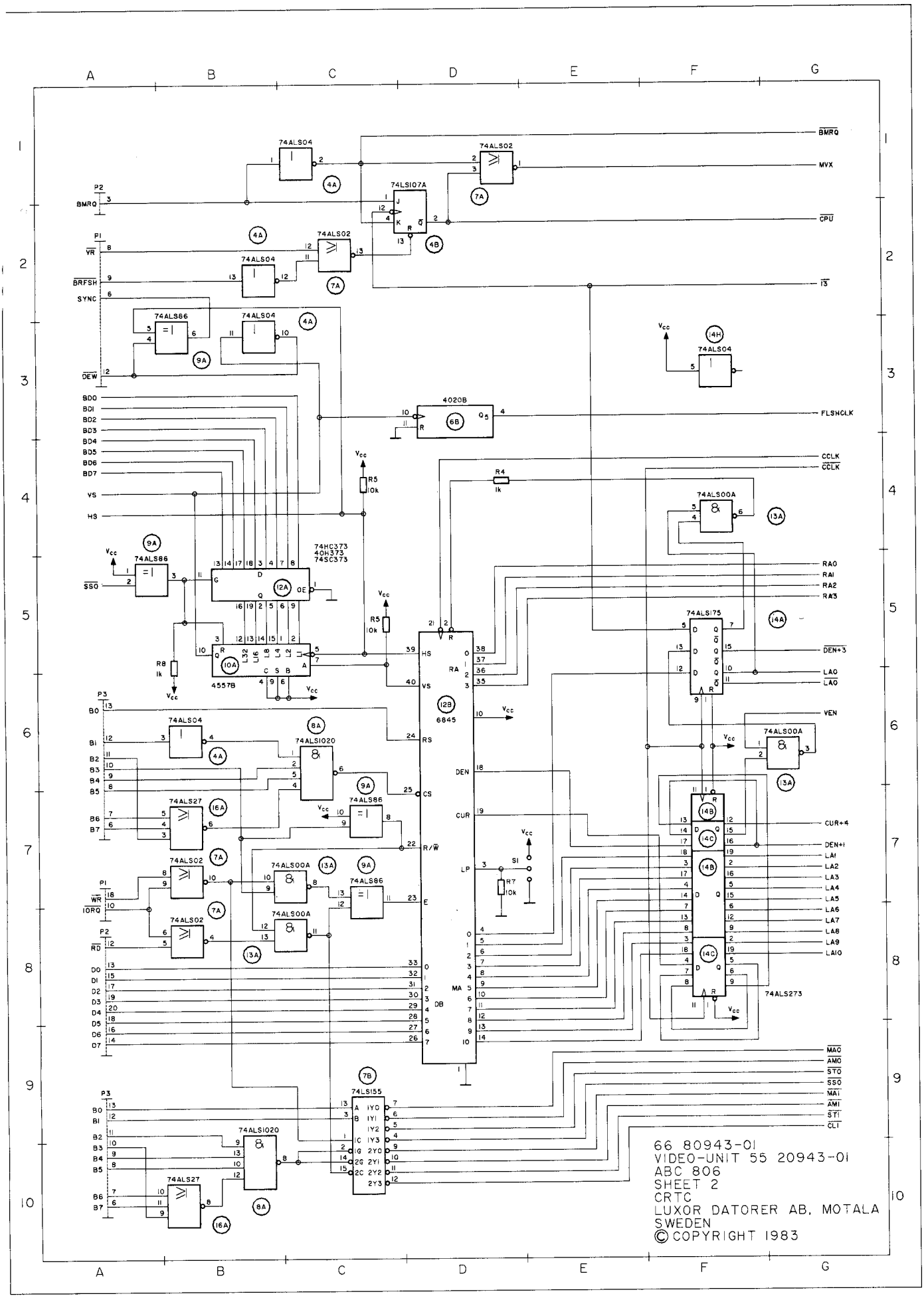
66 80807-02
DC/DC MODULE 55 20807-02
ABC806
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983

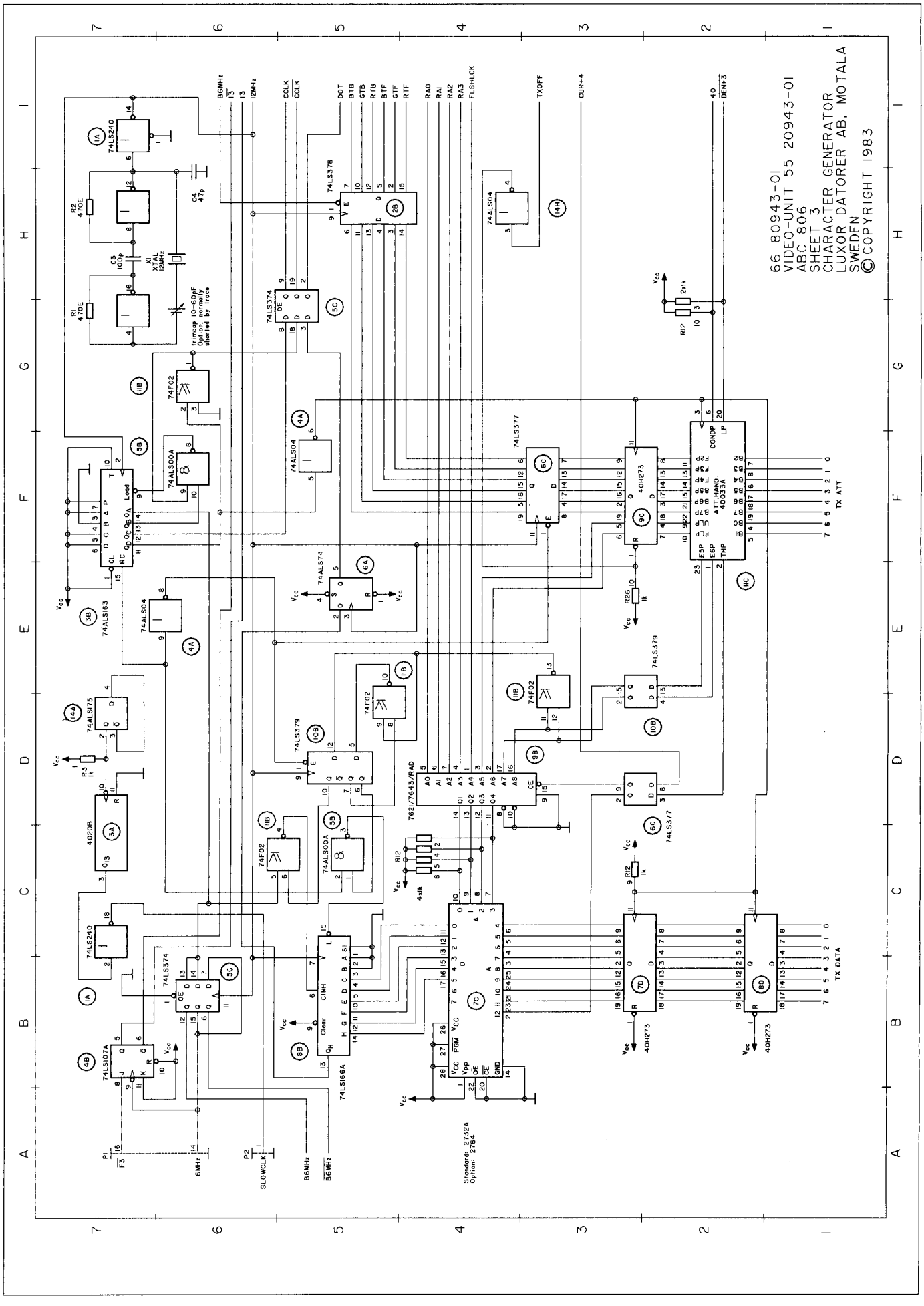
66 80941-01
NRZI COM MODULE 55 20941-01
ABC806
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983

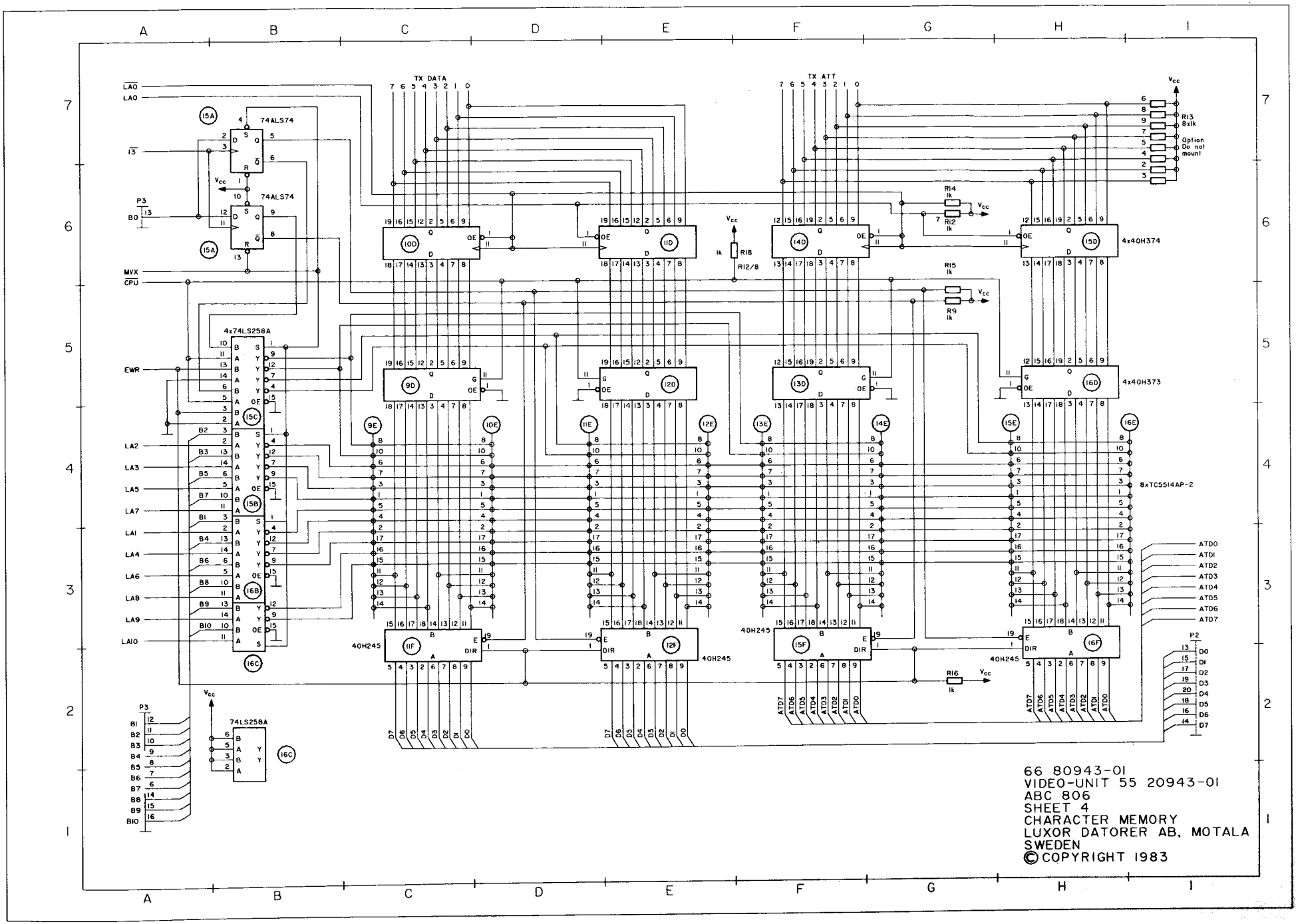


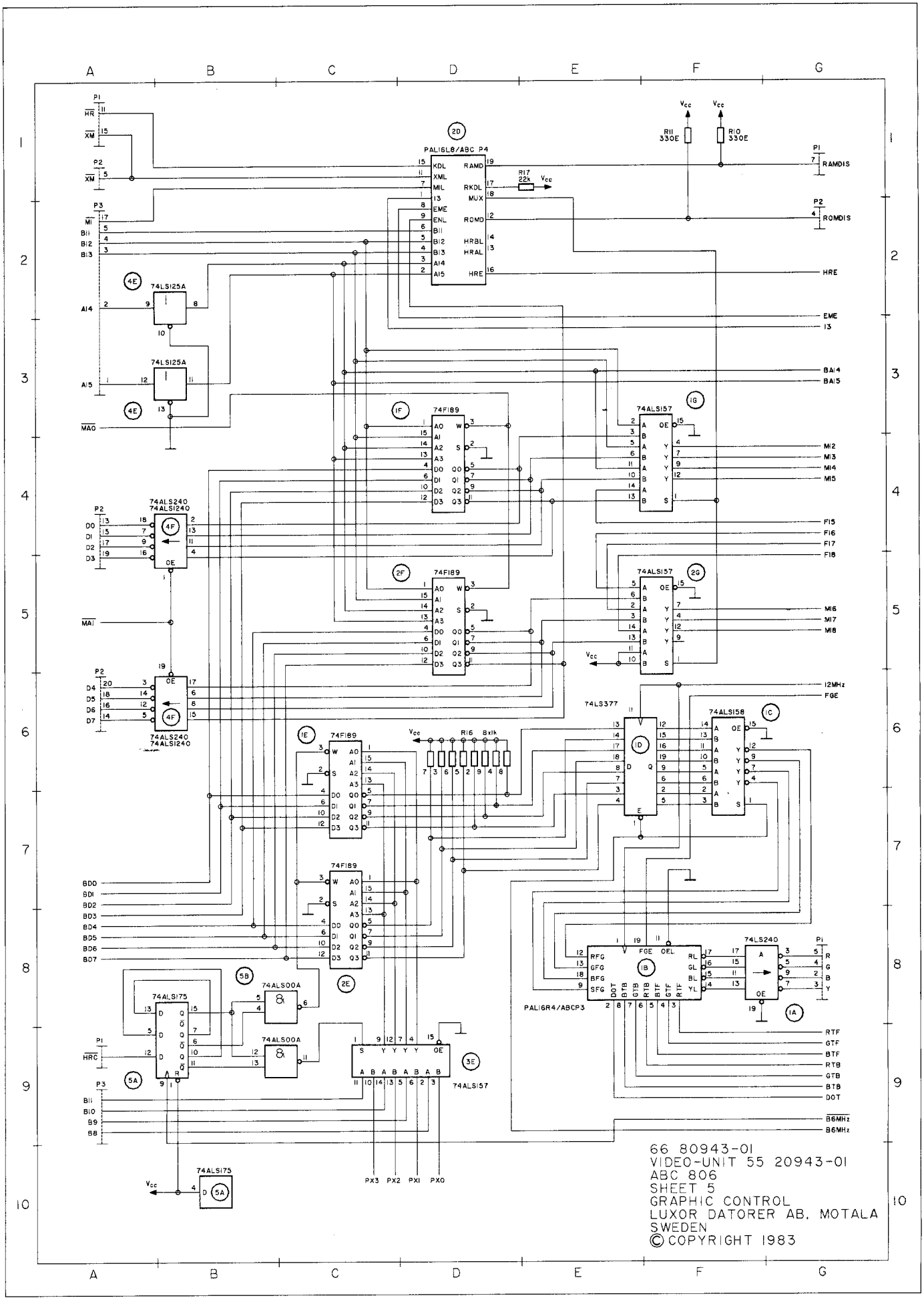


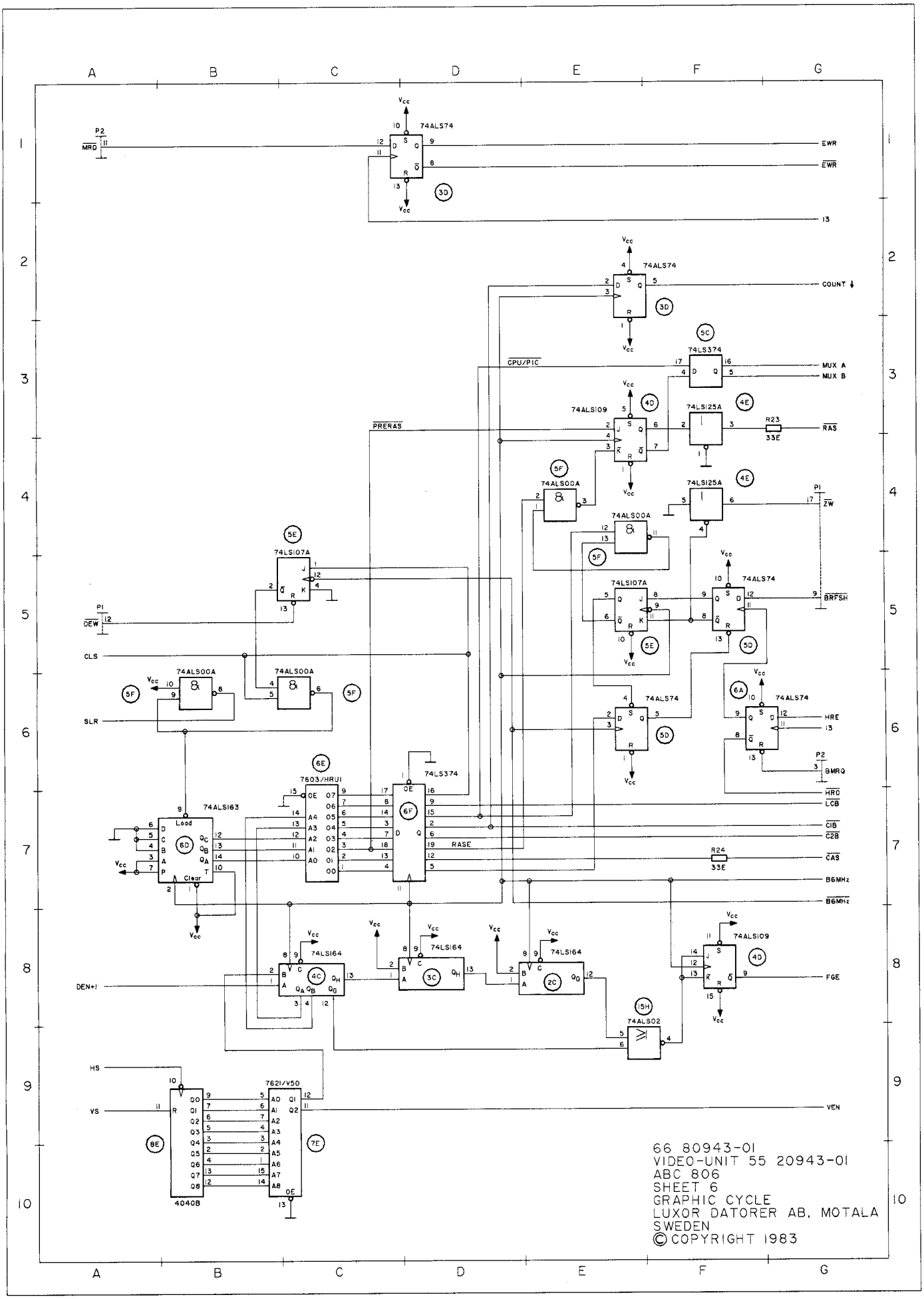
66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 1
BUS ISOLATION
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983

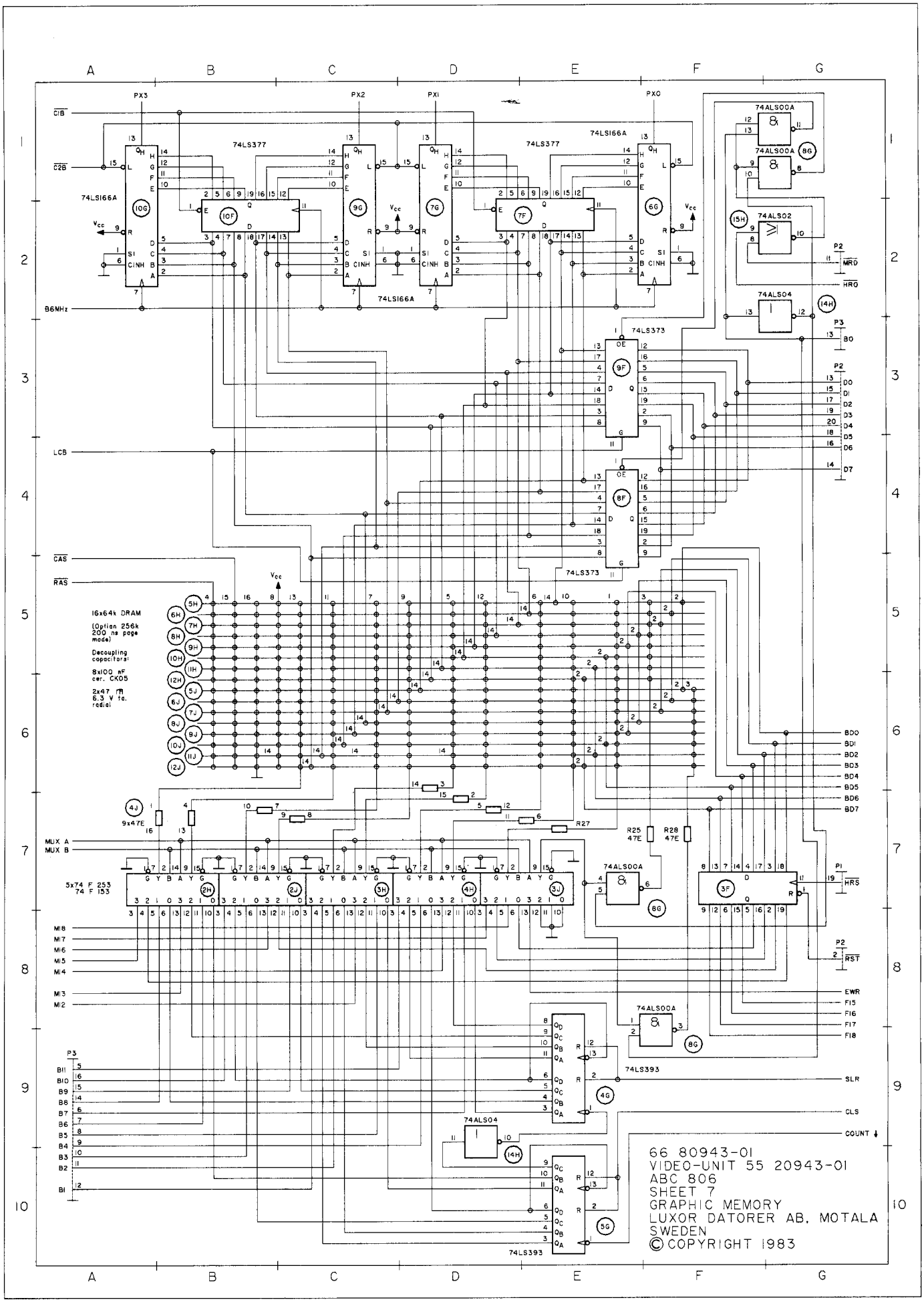

74ALS04
74ALS02
74ALS107A
74ALS86
74ALS00A
74ALS175
74ALS273
74ALS27
74ALS155
74ALS1020
4020B
74HC373
40H373
74SC373
45578
6845
BMRQ
MVX
CPU
VR
BRFSH
SYNC
DEW
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
VS
HS
FLSHCLK
CCLK
SSO
RA0
RA1
RA2
RA3
DEN+3
LA0
VEN
CUR+4
DEN+1
LA1
LA2
LA3
LA4
LA5
LA6
LA7
LA8
LA9
LA10
WR
IORQ
RD
MA0
AM0
ST0
SS0
MA1
AM1
ST1
CL1
R4 1k
R5 10k
R7 10k
R8 1k
66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 2
CRTC
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983

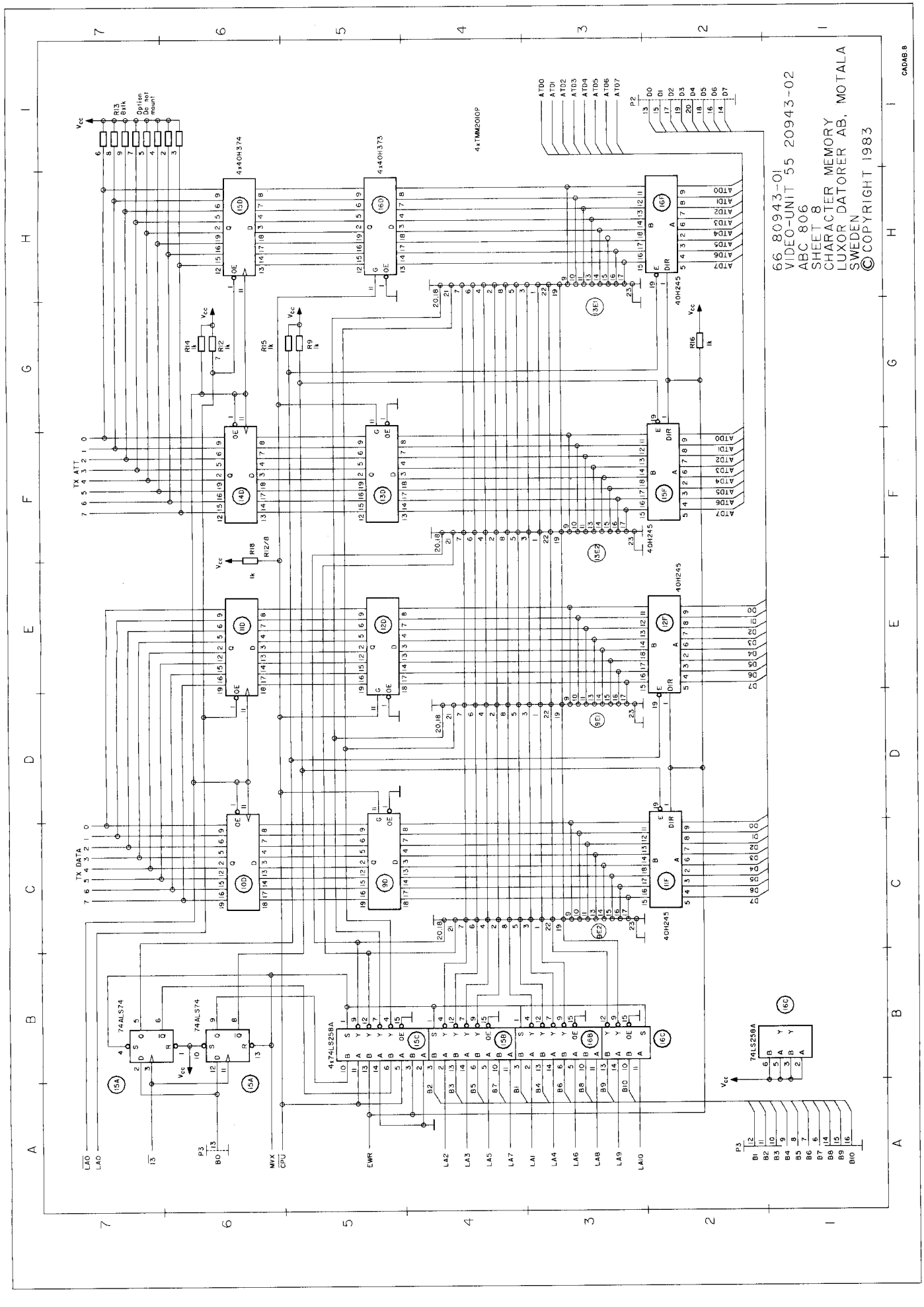

66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 3
CHARACTER GENERATOR
LUXOR DATORER AB, MOTALA
SWEDEN
©COPYRIGHT 1983

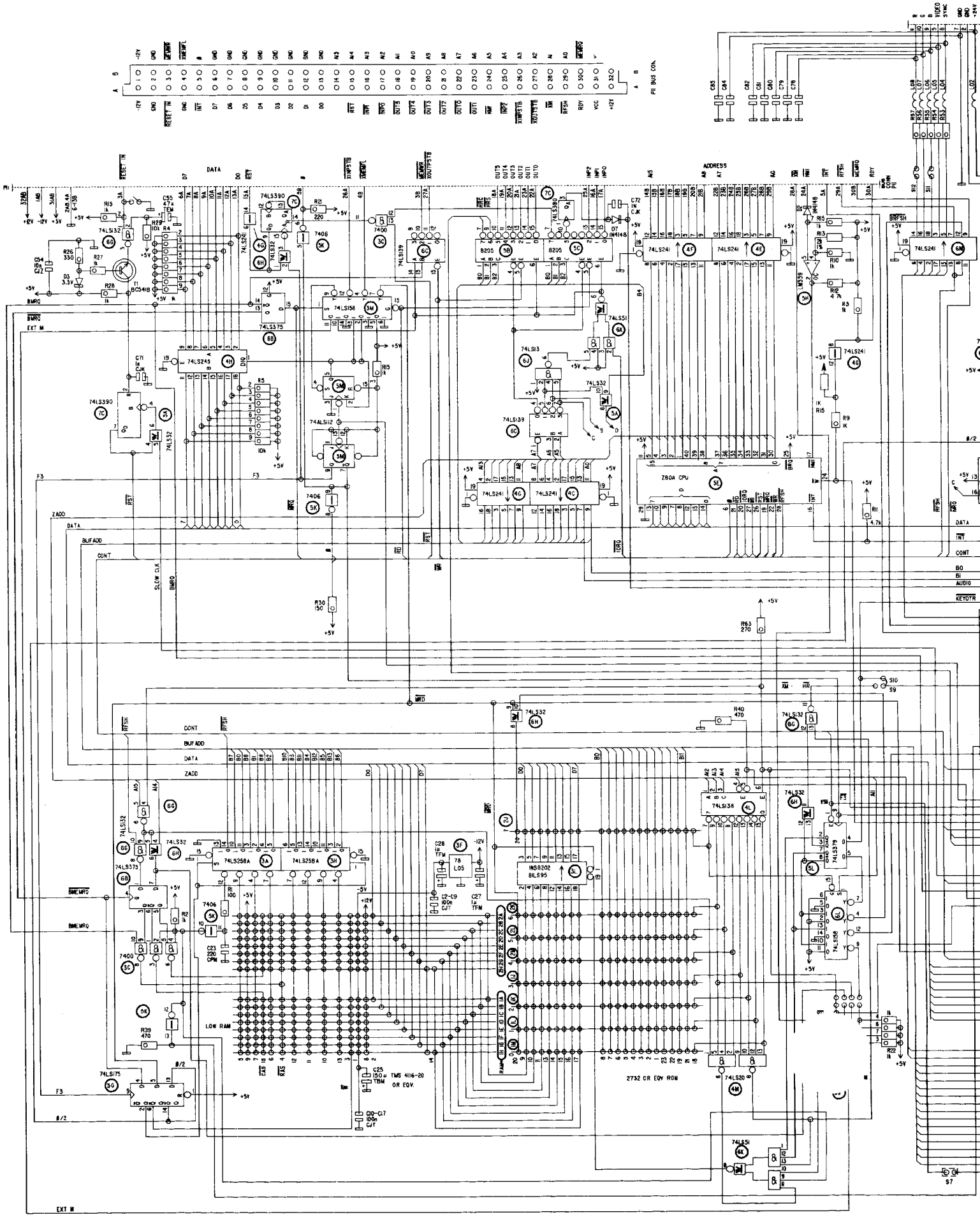

TX DATA
TX ATT
74ALS74
74ALS74
4x74LS258A
74LS258A
4x40H374
4x40H373
8xTC5514AP-2
40H245
40H245
40H245
LA0
LA0
MVX
CPU
EWR
LA2
LA3
LA5
LA7
LA1
LA4
LA6
LA8
LA9
LA10
ATD0
ATD1
ATD2
ATD3
ATD4
ATD5
ATD6
ATD7
R13 8x1k
Option Do not mount
R14 1k
R12 1k
R18 R12/8
R15 1k
R9 1k
R16 1k
66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 4
CHARACTER MEMORY
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983

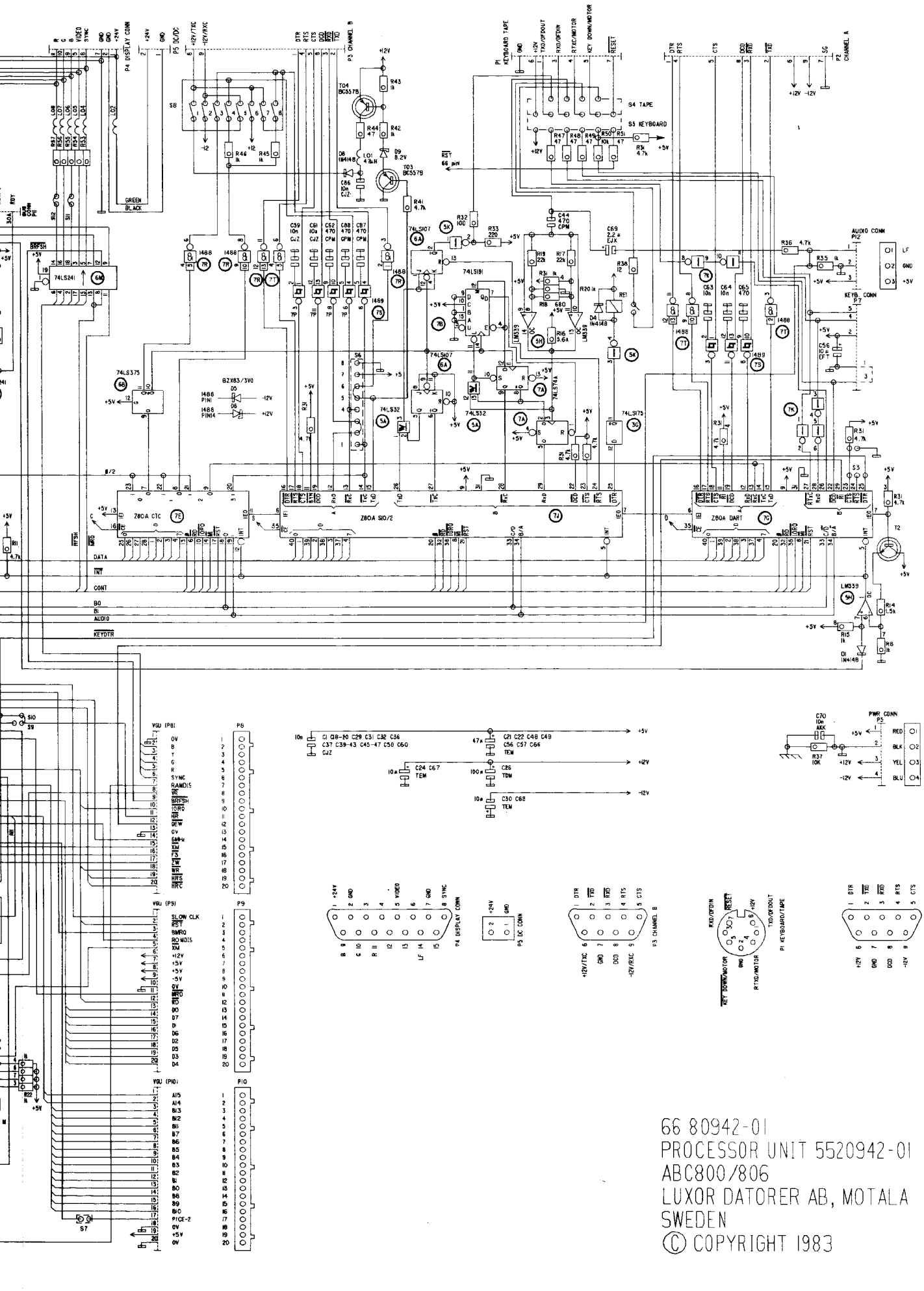

PAL16L8/ABC P4
74LS125A
74LS125A
74ALS240
74ALS1240
74ALS240
74ALS1240
74F189
74F189
74F189
74ALS157
74ALS157
74ALS157
74LS377
74ALS158
74LS240
74ALS175
74ALS175
74ALS00A
74ALS00A
74ALS157
PAL16R4/ABCP3
RAMDIS
ROMDIS
HRE
EME
12MHz
FQE
RTF
GTF
BTF
RTB
GTB
BTB
DOT
B6MHz
PX3 PX2 PX1 PX0
66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 5
GRAPHIC CONTROL
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983


74ALS74
74LS374
74LS107A
74ALS00A
74LS125A
74ALS109
74ALS163
74LS164
74ALS02
7603/HRUI
7621/V50
4040B
MRD
EWR
COUNT
CPU/PIC
MUX A
MUX B
PRERAS
RAS
R23
33E
ZW
BRFSH
DEW
CLS
SLR
HRE
BMRQ
HRQ
LCB
CIB
C28
RASE
CAS
R24
B6MHz
DEN+I
FGE
HS
VS
VEN
Load
Clear
66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 6
GRAPHIC CYCLE
LUXOR DATORER AB, MOTALA
SWEDEN
©COPYRIGHT 1983


PX3
PX2
PXI
PX0
C1B
C2B
74LS377
74LS166A
74LS373
74ALS00A
74ALS02
74ALS04
74LS393
B6MHz
LCB
CAS
RAS
16x64k DRAM
(Option 256k 200 ns page mode)
Decoupling capacitors:
8x100 nF cer. CK05
2x47 nF 6.3 V ta. radial
9x47E
MUX A
MUX B
5x74 F 253
74 F 153
R27
R25 47E
R28 47E
MI8
MI7
MI6
MI5
MI4
MI3
MI2
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
MRD
HRQ
HRS
RST
EWR
FI5
FI6
FI7
FI8
SLR
CLS
COUNT
B0
D0
D1
D2
D3
D4
D5
D6
D7
P1
P2
P3
BII
BI0
B9
B8
B7
B6
B5
B4
B3
B2
BI
66 80943-01
VIDEO-UNIT 55 20943-01
ABC 806
SHEET 7
GRAPHIC MEMORY
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983


66 80943-01
VIDEO-UNIT 55 20943-02
ABC 806
SHEET 8
CHARACTER MEMORY
LUXOR DATORER AB, MOTALA
SWEDEN
©COPYRIGHT 1983
CADAB 8
4x40H374
4x40H373
4xTMM2010P
40H245
74ALS74
4x74LS258A
74LS258A
TX ATT
TX DATA
ATD0
ATD1
ATD2
ATD3
ATD4
ATD5
ATD6
ATD7
EWR
MVX
CPU


PII BUS CON.
DATA
ADDRESS
Z80A CPU
2732 OR EQV ROM
LOW RAM
TMS 4116-20 OR EQV.
EXT M
BMRQ
BMEMRQ
SLOW CLK
CONT
BUFADD
DATA
ZADD
KEYDTR



66 80942-01
PROCESSOR UNIT 5520942-01
ABC800/806
LUXOR DATORER AB, MOTALA
SWEDEN
© COPYRIGHT 1983

ABC800.6680942-01

LUXOR
Datorer

MOTALA 854640