Viktigt Viktigt Viktigt Viktigt Viktigt Viktigt Viktigt
==========

Viktigt information till köparen av den här manualen.

Den här manualen är avsedd som ett uppslagsverk för den som redan är bekant med maskinspråksprogrammering och är inte tillräckligt pedagogisk och utförlig för att användas som grundkurs i maskinspråksprogrammering.
För den som vill lära sig programmera Z-80 förslår jag därför någon av följande böcker:

"Z80 Assembly language programming" av Lance A. Leventhal
"Programming the Z80" av Rodney Zaks
"the Z-80 microcomputer handbook" av William Barden,Jr

Dessa och flera andra böcker finns hos välsorterade data och bokhandlare.
Manualen tillsammans med någon av de nämnda böckerna och Spectravideos monitor utgör ett kraftfullt paket för maskinspråksprogrammering på Spectravideo.

Denna manual är skriven på en SV328 med Wordstar.

Andrzej Felczak

P.S. Många har frågat om denna manual saknar sidor. Det är inte så utan manualen är sidnumrerad på "Basic-sätt" d.v.s. (nästan) varje kapitel börjar på jämnt tiotal sidnummer.

Viktigt Viktigt Viktigt Viktigt Viktigt Viktigt Viktigt
==========

Innehållsförteckning
====================

Detaljerad beskrivning av Spectravideos bus-signaler för
expander.

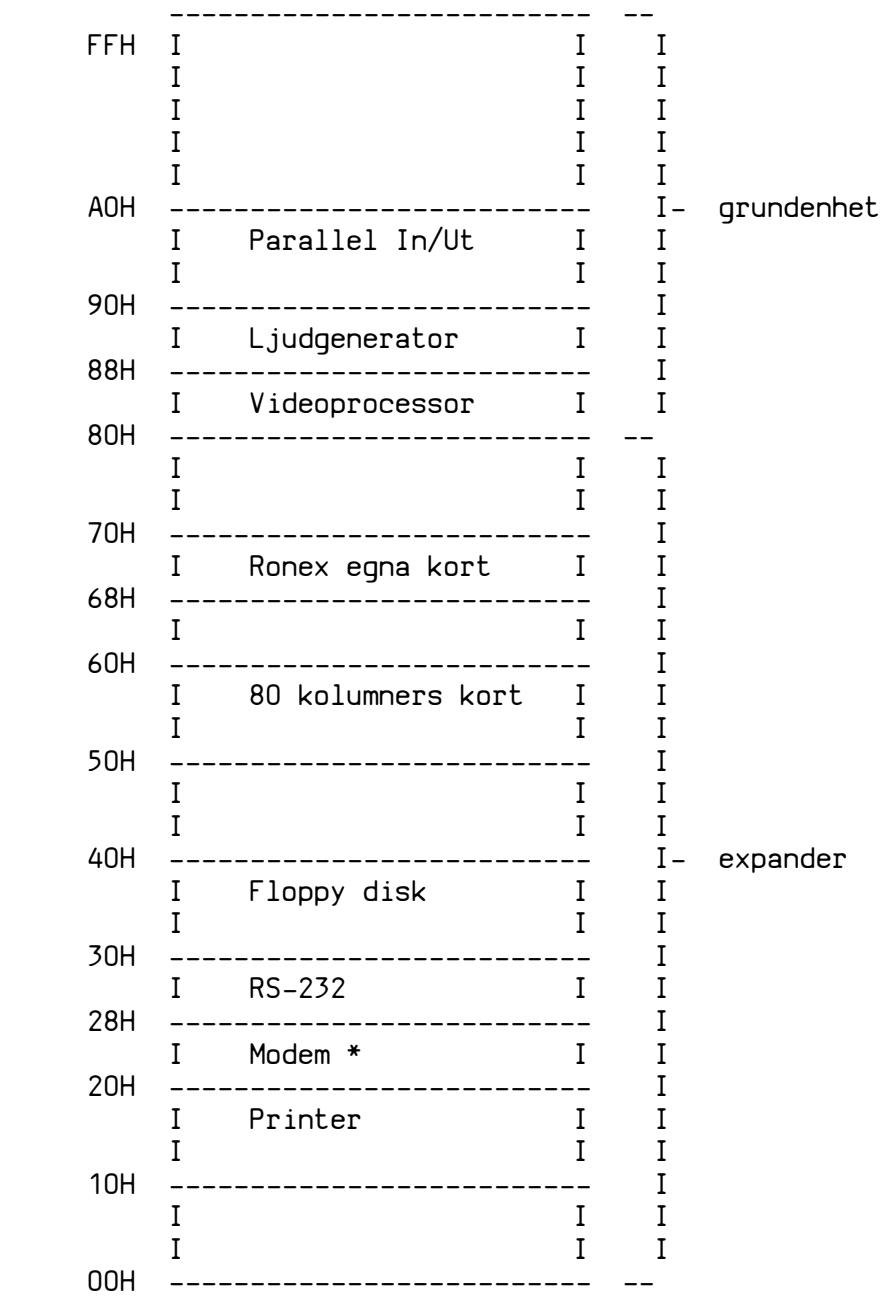
Stift	Namn	I/O	Beskrivning
1	+5V	O	+5V spänningsmatning. Kan belastas med max 300 mA.
2	$\overline{\text{CNTRL2}}$	I	Kontrollsignal för Spectravideo Coleco adapter (normalt hållen hög via ett 3.3 Kohm motsånd till +5V). Denna signal kontrollerar dataöverföringen mellan CPU:n och adaptern under adressering av yttre In/Ut enheter.
3	+12V	O	+12V spänningsmatning. Kan belastas med max 100 mA.
4	-12V	O	-12V spänningsmatning Kan belastas med max 50 mA.
5	$\overline{\text{CNTRL1}}$	I	Kontrollsignal för Spectravideo Coleco adapter (normalt hållen hög via ett 1 Kohm motstånd till +5V). När denna signal dras låg kopplas all intern avkodning av In/Ut enheterna bort och A15 inverteras.
6	$\overline{\text{WAIT}}$	I	När denna signal dras låg indikerar den för Z80 CPU:n att adresserade enheter inte är färdiga för dataöverföring.
7	$\overline{\text{RST}}$	I	När denna signal går låg sätts CPU:ns adress, data och kontrollsignaler i frisvävande läge. När signalen gör hög igen startar datorn upp igen som efter spänningstillslag.
8	CPUCLK	O	Buffrad systemklocka med frekvensen 3.58 MHz.
9-24	A15-A0	O	Buffrad adressbus. Den här 16-bitars bussen skickar ut adress för dataöverföring mellan CPU och In/Ut enheter eller minne.
25	$\overline{\text{RFSH}}$	O	Buffrad REFRESH signal för det dynamiska RAM-et i expandern. När denna signal får låg indikerar den att de 8 lägsta bitarna av adressbussen innehåller en refresh-adress.

Stift	Namn	I/O	Beskrivning
26	$\overline{\text{EXCSR}}$	I	Det här är en yttre CPU-från-VDP LÄS signal som används av Spectravideos Colecoadapter.
27	$\overline{\text{M1}}$	O	Buffrad MACHINE CYCLE ONE signal. Denna signal indikerar att CPU:n hämtar en instruktion från minnet.
28	$\overline{\text{EXCSW}}$	I	Det här är en yttre CPU-från-VDP SKRIV signal som används av Spectravideos Colecoadapter.
29	$\overline{\text{WR}}$	O	Buffrad WRITE signal. Indikerar att CPU:ns databus innehåller giltig data för skrivning i det adresserade minnet eller den adresserade Ut-porten.
30	$\overline{\text{MREQ}}$	O	Buffrad MEMORY REQUEST signal. Denna signal indikerar att adressbussen innehåller en giltig minnesadress.
31	$\overline{\text{IORQ}}$	O	Buffrad INPUT/OUTPUT REQUEST. Denna signal indikerar att adressbussen innehåller en giltig In/Ut adress.
32	$\overline{\text{RD}}$	O	Buffrad READ signal. Denna signal indikerar att CPU:n vill läsa data från det adresserade minnet eller den adresserade In-porten.
33-40	D0-D7	I/O	Buffrad dubbelriktad DATA bus. Det här är en 8-bitars dubbelriktad bus för överföring av data mellan CPU och minne eller In/Ut enheter.
41	CSOUND	I	Audioingång från Spectravideo Colecoadapter.
42	$\overline{\text{INT}}$	I	Denna signal (INTERRUPT) indikerar för CPU:n att In/Ut enheter begär avbrott.
43	$\overline{\text{RAMDIS}}$	I	Dras denna signal låg kopplas Spectravideos användar-RAM-minne bort. Denna signal hålls normalt hög via ett 1 Kohm motstånd till +5V.
44	$\overline{\text{ROMDIS}}$	I	Dras denna signal låg kopplas Spectravideos BASIC-ROM-minne bort.

Stift	Namn	I/O	Beskrivning
45	$\overline{\text{BK32}}$	O	Buffrad MEMORY BANK CONTROL signal. Går denna signal låg kopplas minnesbank 32 in och det inbyggda RAM-minnet kopplas bort.
46	$\overline{\text{BK31}}$	O	Buffrad MEMORY BANK CONTROL signal. Går denna signal låg kopplas minnesbank 31 in och det inbyggda BASIC-ROMet kopplas bort.
47	$\overline{\text{BK22}}$	O	Buffrad MEMORY BANK CONTROL signal. Går denna signal låg kopplas minnesbank 22 in och det inbyggda RAM-minnet kopplas bort.
48	$\overline{\text{BK21}}$	O	Buffrad MEMORY BANK CONTROL signal. Går denna signal låg kopplas minnesbank 21 in och det inbyggda BASIC-ROMet kopplas bort.
49-50	GND		Elektrisk jord.

Översiktskarta för In/Ut portarna i SV-318/SV-328 samt expander
==========

Adress (Hex) Enhet



* Modemet finns inte i Sverige eftersom det fungerar enligt amerikansk standard och kan därför inte användas här.

Detaljerad beskrivning av portarna i expandern

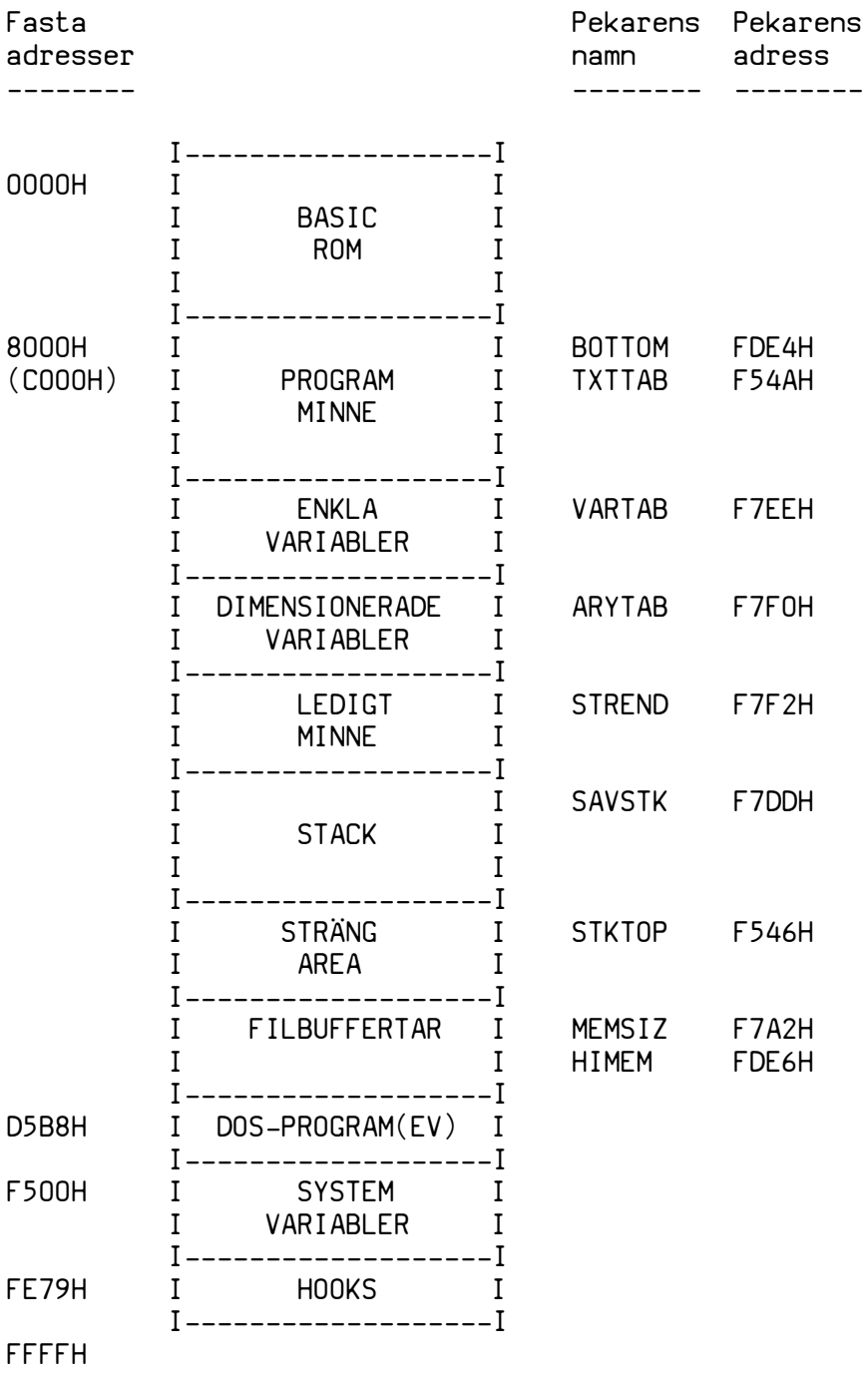
Adress Hex	Dec	R/W	Beskrivning	Enhet
10H	16	W	Write data port	Printer
11H	17	W	Data strobe	Printer
12H	18	R	Status (Bit 0="0" for ready)	Printer
20H	32	R	Receiver buffer register	Modem
		W	Divisor latch (least significant)	Modem
		W	Transmitter holding buffer reg.	Modem
21H	33	W	Divisor latch (most significant)	Modem
		W	Interrupt enable register	Modem
22H	34	W	Interrupt identification register	Modem
23H	35	W	Line control register	Modem
24H	36	W	Read modem control register	Modem
25H	37	R	Line status register	Modem
26H	38	R	Read modem status register	Modem
28H	40	R	Receiver buffer register	Modem
		W	Divisor latch (least significant)	Modem
		W	Transmitter holding buffer reg.	Modem
29H	41	W	Divisor latch (most significant)	Modem
		W	Interrupt enable register	Modem
2AH	42	W	Interrupt identification register	Modem
2BH	43	W	Line control register	Modem
2CH	44	W	Read modem control register	Modem
2DH	45	R	Line status register	Modem
2EH	46	R	Read modem status register	Modem
30H	48	R	FD-1793 Status register	Floppy disk
31H	49	R/W	FD-1793 Command register	Floppy disk
32H	50	R/W	FD-1793 Track register	Floppy disk
33H	51	R/W	FD-1793 Sector register	Floppy disk
34H	42	R	Read INTRQ and DRQ from FD-1793	Floppy disk
		W	Disk select register	Floppy disk
			(Bit 0 = "0" to select disk 1	
			Bit 1 = "0" to select disk 2)	
38H	56	W	Density select register	Floppy disk
			(Bit 0 = "0" for double density	
			Bit 0 = "1" for single density)	
50H	80	W	Adress register select	80-column card
51H	81	W	CRT controller register (R0-R17)	80-column card
58H	88	W	CRT bank control	80-column card
			(Bit 0 = "0" for bank off	
			Bit 0 = "1" for bank on)	
68H	104	R/W	Datavision status & programmering	Eget RS-232
69H	105	R/W	Datavision receive/transmit buffer	Eget RS-232
6AH	106	R/W	Datavision status & programmering	Modem
6BH	107	R/W	Datavision receive/transmit buffer	Modem
6CH	108	R/W	PIO-kort lägsta 8 bitarna	PIO-kort
6DH	109	R/W	PIO-kort högsta 8 bitarna	PIO-kort

Detaljerad beskrivning av portarna i expandern

Adress Hex	Dec	R/W	Beskrivning	Enhet
80H	128	W	TMS-9918A Write Mode=0	Video processor
81H	129	W	TMS-9918A Write Mode=1	Video processor
84H	132	R	TMS-9918A Read Mode=0	Video processor
85H	133	R	TMS-9918A Read Mode=1	Video processor
88H	136	W	AY-3-8910 Latch adress	Ljudgenerator
8CH	140	W	AY-3-8910 Write	Ljudgenerator
90H	144	R	AY-3-8910 Read	Ljudgenerator
96H	150	W	Write 8255 port C	Parallel In/Out
97H	151	W	Write 8255 control word register	Parallel In/Out
98H	152	R	Read 8255 port A	Parallel In/Out
99H	153	R	Read 8255 port B	Parallel In/Out

Körning av maskinspråksprogram ihop med Basic

Om du tittar på minneskartan nedanför ser du hur Basic-tolken använder RAM-minnet.
Ska du kunna köra ett maskinspråksprogram ihop med ett Basic-program måste du se till att programmen inte kolliderar med varandra och måste därför reservera plats.
Lyckligtvis är detta relativt lätt gjort genom att ställa om några pekare.



Reservering av plats för egna maskinspråksprogram

Det finns två sätt att reservera utrymme för egen användning.
Antingen kan du höja "golvet" för RAM-minnet eller sänka "taket".
Metoden att höja golvet passar bäst om programmet ska användas på en SV328 med och utan disk medan metoden att sänka taket är bäst om programmet ska passa både SV318 och SV328.
De två följande exemplen visar båda principerna.

Reservering av minnesutrymme genom höjning av "golvet"

De pekare som bestämmer golvet är BOTTOM (adress FDE4H) och TXTTAB (F54AH). Dessa pekare består av två byte så när jag säger att en pekare finns på adress FDE4H menar jag egentligen att den lägre halvan finns på adress FDE4H och den högre halvan finns ett steg högre, alltså på adress FDE5H.
Börja med att titta vilka värden pekarna har från början.
Följande rad skriver ut värdet på pekaren BOTTOM.

```
PRINT HEX$(PEEK(&HFDE4H) + 256*PEEK(&HFDE4+1))
```

Studera raden noga och försök förstå hur den fungerar.
Orsaken till att jag multiplicerar den övre halvan med 256 är att den ska ha rätt "viktning".
BOTTOM har värdet 8000H (C000H). Värdet inom parantes är för SV318.
Skriv därefter ut värdet av TXTTAB. Det ska vara 8001H (C001H).
Titta därefter vad som finns på adress 8000H (C000H) och adress 8001H (C001H).
Följande rad skriver ut innehållet på adress 8000H (C000H).

```
PRINT PEEK(&H8000)    ( PRINT PEEK(&HC000) )
```

Värdet på båda adresserna ska vara 0.

Säg nu att vi behöver reservera 1024 bytes för något ändamål.
Då måste vi flytta båda pekarna 1024 steg upp.
1024 decimalt är 400 hexadecimalt. BOTTOM ska därför få värdet 8400H (C400H) och TXTTAB 8401H (C401H).
Följande rader ändrar pekarnas värden.

```
POKE &HFDE4,&H00            ' lägsta adresshalvan BOTTOM
POKE &HFDE5,&H84   (C4)     ' högsta adresshalvan BOTTOM
POKE &HF54A,&H01            ' lägsta adresshalvan TXTTAB
POKE &HF54B,&H84   (C4)     ' högsta adresshalvan TXTTAB
```

Dessutom måste du nollställa de två byten som dessa pekare pekar på. Det gör du lätt med följande rad.

```
POKE &H8400,0 : POKE &H8401,0 (POKE &HC400,0 : POKE &HC401,0)
```

Till sist får du ge kommandot NEW för att datorn själv ska ändra pekarna VARTAB och ARYTAB.

Inmatning av maskinspråksprogram i minnet

Vi ska nu mata in nedanstående program i minnet. Studera programmet och jämför koderna med dem som står i någon av de Z-80 läroböcker som du (förhoppningsvis) har köpt.

Adress (Hex)	Kod (Hex)	Mnemonic	Kommentar
C000	C5	PUSH BC	Sparar BC-registerparet
C001	E5	PUSH HL	Sparar HL-registerparet
C002	F5	PUSH AF	Sparar Accumulator-registret
C003	06	LD B,20	Laddar B-registret med 20
C004	14		B används här som loop räknare
C005	3E	LD A,"C"	Laddar A med ASCII-värdet
C006	43		av "C"
C007	F5	PUSH AF	Sparar A:s värde
C008	CD	CALL OUTDO	Mata ut ASCII-koden i A
C009	18		på skärmen
C00A	00		
C00B	21	LD HL,FFFFH	Laddar HL med antalet varv
C00C	FF		
C00D	FF		
C00E	2B	DEC HL	Minskar HL ett steg
C00F	7C	LD A,H	Enda gången (H OR L) är noll
C010	B5	OR L	är när HL är noll
C011	C2	JP NZ,C00EH	Fortsätt om HL<>0
C012	0E		
C013	C0		
C014	F1	POP AF	Återställ A:s värde
C015	3C	INC A	Öka ASCII-koden i A
C016	05	DEC B	Minska loopräknaren
C017	C2	JP NZ,C007H	Fortsätt om inte 20 tkn än
C018	07		
C019	C0		
C01A	F1	POP AF	Återställ A
C01B	E1	POP HL	Återställ HL
C01C	C1	POP BC	Återställ BC
C01D	C9	RET	Återvänd till Basic

Programmet är placerat på adress C000H för att passa både SV318 och SV328 med och utan disk. Detta medför att du inte kan använda allt minne på SV328 men det är väldigt lätt att ändra programmet så att det ligger på adress 8000H. Försök själv.
Glöm inte att ändra hoppadresserna.
Programmet fungerar så att det laddar A-registret med ASCII-värdet för C. Därefter skrivs bokstaven ut.
På adress C00BH är en fördröjning på c:a 1/2 sekund är insatt.
På adress C015H ökas ASCII-koden. Till sist hoppar programmet tillbaka till adress C007 så länge B inte är noll. Programmet kör slingan totalt 20 ggr och skriver ut bokstäverna C till V med paus. Därefter återvänder programmet till Basic:en.

Det finns tre sätt att få in ett maskinspråksprogram i minnet.

1 Genom användning av POKE.
2 Med kommandot BLOAD.
3 Med Spectravideos monitor. Detta sätt beskrivs i bruksanvisningen på sid 110.

Följande Basc-program POKE:ar in det föregående maskinspråks-programmet, ändrar pekarna och raderar sig själv.
Har du en SV318 börjar Basic-programmet på adress C000H och då skriver programmet faktiskt över sig själv. Om du i så fall utesluter fr.o.m. rad 20, d.v.s. delen som ändrar pekarna och försöker lista programmet händer det underliga saker. Det är list-rutinen som enbart är avsedd att lista Basic-program och reagerar konstigt om den stötet på maskinspråk.

```
10 DATA C5,E5,F5,06,14,3E,43,F5
11 DATA CD,18,00,21,FF,FF,2B,7C
12 DATA B5,C2,0E,C0,F1,3C,05,C2
13 DATA 07,C0,F1,E1,C1,C9,*
14 ADR%=&HC000
15   READ A$
16   IF A$="*" GOTO 20
17   POKE ADR%,VAL("&H"+A$)
18   ADR%=ADR%+1
19   GOTO 15
20 POKE &HFDE4,&H20    ' lägsta adresshalvan BOTTOM
20 POKE &HFDE5,&HC0    ' högsta adresshalvan BOTTOM
20 POKE &HF54A,&H21    ' lägsta adresshalvan TXTTAB
20 POKE &HF54B,&HC0    ' högsta adresshalvan TXTTAB
24 POKE &HC020,0       ' nollställ (BOTTOM)
25 POKE &HC021,0       ' nollställ (TXTTAB)
26 NEW
```

Om du jämför Basic-programmets data-satser med de hex-koder som skrevs på sid 33 ser du att dom är identiska.
Du kan nu spara maskinspråksprogrammet genom att ge kommandot

```
BSAVE "1:DEMO",&HC000,&HC01D
```

Nästa gång du vill köra programmet behöver du bara kör rad 20-26 av Basic-programmet som du reserverar den plats som behövs. Själva maskinspråksprogrammet laddar du snabbt in med följande:

```
BLOAD "1:DEMO"
```

Att använda BLOAD är speciellt fördelaktigt att använda om man har väldigt långa program eftersom det går mycket snabbare att ladda in och tar mindre plats på diskett.

Anrop av egna maskinspråksprogram

När du kört det lilla Basic-programmet ligger maskinspråks-programmet dolt i minnet på adress C000H-C01DH.
Du anropar programmet genom sekvensen:

```
DEF USR0=&HC000   ' definiera USR0 att börja på adress C000H
A=USR0(0)         ' kalla på programmet
```

Då ser du att tecknena C till V skrivs ut med en kort paus mellan och därefter visar datorn att den är färdig.
Du har nu kört ditt (kanske) första maskinspråksprogram på Spectravideo. Läs noga igenom texten, jämför med din Z-80 lärobok och försök förstå. Fortsättnigen är svårare och kräver att du verkligen kan och begriper vad som skrivits tills nu.

Överföring av parametrar mellan Basic och maskinspråk

Du kan skicka med parametrar (variabler) från Basic:en till maskinspråksprogrammet och vice versa.
Det medför att du kan skicka med en variabel av valfri typ (heltal, enkel och dubbel precision eller sträng) direkt i anropet av maskinspråksprogrammet. Du kan också få tillbaka en variabel av valfri typ. Se på exemplet nedan.

```
A%=USR0(B%)
```

Där skickar du med variabeln B% och får tillbaka variabeln A%.
Jag ska nu förklara hur maskinspråksprogrammet tolkar de medföljande variablerna.
När Basic:en kallar på maskinspråksprogrammet så ligger det ett värde i A-registret som säger vilken typ av variabel det är. T.ex. betyder värde 2 heltal. Adressen till variabeln ligger dessutom i HL-registret.
Se på sammanfattningen på sid 38 för att få en komplett uppställning hur överföringen går till.
På nästa sida finns ett nytt maskinspråksprogram som demonstrerar överföringen av variabler. Det är mycket likt det förra men studera noga skillnaderna.

```
Adress    Kod       Mnemonic       Kommentar
(Hex)     (Hex)
------    -----     --------       ---------

C000      C5        PUSH BC        Sparar BC-registerparet
C001      D4        PUSH DE        Sparar DE-registerparet
C002      E5        PUSH HL        Sparar HL-registerparet
C003      F5        PUSH AF        Sparar Accumulator-registret
C004      23        INC  HL        Flyttar pekaren till
C005      23        INC  HL        eventuellt heltal
C006      FE        CP   2         Testa om heltal
C007      02
C008      C2        JP   NZ,C00FH  Hoppa om inte heltal
C009      0F
C00A      C0
C00B      46        LD   B,(HL)    Laddar B med överförd variabel
C00C      C3        JP   C011H     Hoppa för att inte ladda B igen
C00D      11
C00E      C0
C00F      06        LD   B,20      Laddar B-registret med 20.
C010      14                       B används här som loopräknare
C011      3E        LD   A,"C"     Laddar A med ASCII-värdet
C012      43                       av "C"
C013      F5        PUSH AF        Sparar A:s värde
C014      CD        CALL OUTDO     Matar ut ASCII-koden i A
C015      18                       på skärmen
C016      00                       (Se Kernel för mer info)
C017      11        LD   DE,8000H  Laddar DE med antalet varv
C018      00
C019      80
C01A      1B        DEC  DE        Minskar DE ett steg
C01B      7A        LD   A,D       Enda gången (D OR E) är noll
C01C      B3        OR   E         är när DE är noll
C01D      C2        JP   NZ,C01AH  Forsätt om DE<>0
C01E      1A
C01F      C0
C020      F1        POP  AF        Återställ A:s värde
C021      3C        INC  A         Öka ASCII-koden i A
C022      05        DEC  B         Minska loopräknaren
C023      C2        JP   NZ,C013H  Fortsätt om inte slut än
C024      13
C025      C0
C026      36        LD   (HL),FFH  Ladda lägsta halvan av talet
C027      FF                       med 255
C028      23        INC  HL        Flytta pekaren till övre halvan
C029      36        LD   (HL),FFH  Ladda högsta halvan av talet
C02A      FF                       med 255
C02B      F1        POP  AF        Återställ A
C02C      E1        POP  HL        Återställ HL
C02D      D1        POP  DE        Återställ DE
C02E      C1        POP  BC        Återställ BC
C02F      C9        RET            Återvänd till Basic
```

När du har förstått hur programmet fungerar så ändrar du de gamla DATA-satserna i det föregående Basic-programmet så att du istället matar in det nya programmet.
Glöm inte att ändra pekarna till adress C030H istället för adress C020H eftersom detta nya program är längre än det gamla.
När du skrivit det nya Basic-programmet och kört det ligger maskinspråksprogrammet dolt på adress C000H-C02FH.
Du kallar på det genom följande Basic-sekvens.

```
10 DEF USR0=&HC000
20 INPUT B%
30 IF B%>255 GOTO 20
40 A%=USR0(B%)
50 PRINT
60 PRINT A%,B%
```

Du ser då att maskinspråksprogrammet skriver ut så många tecken som finns i B%. Dessutom sätts värdet av den utgående variabeln A% till -1 (försök med PRINT &HFFFF).

Om du ändrar programmet så att det ser ut som nedanstående märker du att oberoende vilket värde du ger B# skriver den ut 20 tecken. Detta därför att maskinspråksprogrammet testar om den ingående variabeln är ett heltal, om inte så sätter den värdet till 20.

```
10 DEF USR0=&HC000
20 INPUT B#
30 A#=USR0(B#)
50 PRINT
60 PRINT A#,B#
```

Som du ser blir A#:s värde lite konstigt.
Det beror på att du har byggt upp talet på ett felaktigt sätt.

Du ska efter den här genomgången kunna länka in och använda dina egna maskinspråksprogram. Har du haft problem med att förstå vad som står här bör du läsa på din Z-80 lärobok.
Vill du ha flera exempel på program så finns programmet för att komma åt de 32K RAM som finns dolt i maskinen.
Programmet står på sid 90 och framåt.

Sammanfattning av parameteröverföring med USR

Vid anrop med USR ligger variabeltypskoden i A.

2 Heltal
3 Sträng
4 Flyttal enkel precision
5 Flyttal dubbel precision

Dessutom ligger adress till variabeln i HL enligt följande system.

Heltal :	HL+2	LSB
	HL+3	MSB
Sträng :	HL+2	LSB adress till strängbeskrivning
	HL+3	MSB " " "
Sträng-beskrivning:	adr	stränglängd
	adr+1	LSB adress till sträng
	adr+2	MSB " " "
Flyttal enkel prec:	HL	exponent
	HL+1	2 högsta siffror i BCD-kod
	..HL+4	2 lägsta " i "
Flyttal dubbel prec:	HL	exponent
	HL+1	2 högsta siffror i BCD-kod
	..HL+4	2 lägsta " i "

Hopptabell (Kernel)

Kernel "Kärna" är den benämning man ger den hopptabell som brukar ligga i början på varje större program. I denna hopptabell finns hopp till de flesta rutiner som man kan ha nytta av separat, t.ex. diskhanteringsrutiner, skärm och tangentbordsrutiner m.m. Fördelen med att ha alla anrop till rutinerna samlade till ett ställe är att man kan stuva om resten av programmet så mycket som man önskar utan att program som använder dessa rutiner behöver skrivas om. Kernel-tabellen ligger nämligen kvar på samma ställe hela tiden.

Beskrivning av hopptabellen som ligger i början av Spectravideons BASIC-tolk.

Listan är upplagd på följande sätt:
Hexadecimal adress, namn och kort beskrivning.

Förklaring till vissa förkortningar:

Z,C	CPU-flaggor
B,C,D,E,H,L,A	CPU-register
BC,DE,HL	CPU-registerpar
Tgb	Tangentbord
Lpt	Skrivare (centronicsinterface)
FAC	Accumulator för flyttal (F923H)
FACLO	Accumulator för heltal (F925H)

	0008 SYNCHK	Looks at the current (HL) character to check if it equals the character after SYNCHK, otherwise "Syntax error"
	0010 CHRGET	Incr HL, get character into A and set flags: C=numeric, Z=EOL (":" or 0)
	0018 OUTDO	Output char in A using PRTFLG, TTYPOS etc.
	0020 COMPAR	Compares HL & DE : HL>DE sets C, HL=DE sets Z
	0028 SIGN	Returns A=-1 if FAC<0, 0 if FAC=0, 1 if FAC>0, works on single and double precision constants.
	0030 GETYPR	Get valtyp & set condition codes (flags) INT=2,SIGN STR=3,Z SNG=4,ODD P DBL=8,NC
0038	3CC2 KEYINT	Interrupt handler
003B	3DCA CHSNS	Testa om tecken från Tgb finns, Z om finns ej
003E	403D CHGET	Väntar på tecken från Tgb. Tecknet hamnar i A
0041	3938 CHPSTT	Testar om Lpt Ready. Z om ej Ready.
0044	3915 CHPLPT	Skriver ut tecknet i A på Lpt
0047	3541 INITXT	Initiera screen 0
004A	3610 INIGRP	Initiera screen 1
004D	3665 INIMLT	Initiera screen 2
0050	3B95 FNKSB	Visa funktionstangenterna om screen 0 och flagga (CNSDF) satt.
0053	3B86 ERAFNK	Radera funktionstangentvisningen
0056	3B9F DSPFNK	Visa funktionstangenterna
0059	3498 RSTFNK	Initiera funktionstangenterna

```
005C 3512 BREKX   Kolla om CTRL-STOP. C om nertryckt
005F 3476 JMPBNK  Hoppa till minnesbank
0062 3480 CALBNK  Kalla på subrutin i en annan minnesbank
0066 0138 NMI     NMI-handler
0069 203A CSRDON  Sätt på kassett-motor och vänta SYNC
                  och HEADER
006C 2016 CASIN   Läs data byte för byte till A från
                  kassett-bandspelaren
006F 207C CTOFF   Stäng av kassett-motor
0072 2059 CWRTON  Sätt på kassett-motor och skriv SYNC
                  och HEADER
0075 2026 CASOUT  Skriv data byte för byte från A
                  på kassett-bandspelaren
0078 206C CTWOFF  Skriva nolla på kassett och stanna motorn
007B 6474 CRDO    Put CR to output device
007E 6463 CDRONZ  Put CR to output device if not at left position
0081 6415 OUTDLP  Put char in A to Lpt and decode TAB, CR and skip
                  rest of the Ctrl-characters
0084 692C STRINI
0087 6959 PUTNEW
008A 6AD5 FRESTR
008D 697D STROUT
0090 08C9 READYR  Reset stack when BASIC externally stopped & then
                  restarted
0093 08ED SNERR   Print "Syntax error"
0096 0907 ERROR   Print error number in E
0099 0981 NTDERR  ?
009C 09AF READY   Warmstart
009F 0AE5 LINKER  Goes through the program and fixes links
00A2 0E3E NEWSTT  New statement fetcher
00A5 0F9E FCERR   Function call error
00A8 643D FINLPT  Reset output device to video and move Lpt head
                  to left margin
00AB 162D EVAL    Evaluate variable, constant, function call
                  Put result in FAC
00AE 14CA FRMEVL  Formula evaluation routine
00B1 1AA6 GETBYT  ?
00B4 1CB9 FRMQNT  Get integer value from formula
00B7 1AA9 CONINT  Convert FAC to an integer in DE
00BA 183C SNGFLT  ?
00BD 1A99 GETIN2  ?
00C0 0B44 CRUNCH  Translate all reserved words into tokens
00C3 0AE9 CHEAD   Goes through program and fixes links from
                  position in DE
00C6 55C9 CONIA   Convert signed number in A to integer
00C9 5968 INEG2   Convert integer into single precision, put in DAC
00CC 56C4 MAKINT  Put HL in FACLO, set valtyp to int
00CF 204D DIOERR
00D2 3966 POPALL
00D5 74B0 EOF     Close file with # in FAC
00D8 59C9 POPHRT
00DB 5B44 LINPRT
```

```
00DE 6545 OMERR   Fixes program links, resets SAVSTK and prints
                  "OUT OF MEMORY"
00E1 656A RUNC    Intitiates for run
00E4 6FD3 NAMSCN  Scan filename and device
00E7 702F SCNBLK  Scan block
00EA 7033 GETFLP  Get pointer to file with number in FAC
00ED 7036 GETPTR  Same as above but number in A
00F0 7073 SETFIL  Tests if file wit # in a A is open and returns
                  pointer
00F3 70C3 NULOPN  Open file: Drive in D, mode in E (4=filemode=
                  no "FOR", 1="INPUT", 2="OUTPUT", 8="APPEND"),
                  file # in A
00F6 70EA CLSFIL  Close file # in A
00F9 710A NOCLSB
00FC 737D CLSALL  Close all files if NLONLY<>0
00FF 73CA FILOU1  Writes character in (A) onto disc
0102 73F1 INDSKC  Gets a char from disk into A. Carry if EOF
0105 7460 CLRBUF  Clears data buffer of selected file. On return HL
                  points at start of buffer
0108 7474 DOCLR   Clear B # of bytes starting at HL
010B 75D0 NOSKCR
010E 75FA DERBFN  "BAD FILE NAME"
0111 75FD DERFAO  "FILE ALREADY OPEN"
0114 7603 DERFNF  "FILE NOT FOUND"
0117 7607 DERFNO  "FILE NOT OPEN"
011A 760F DERIER  "INTERNAL ERROR"
011D 7612 DERRPE  "READ PAST EOF"
0120 170B MAKUPL
0123 0BCA CRN2ND
0126 346A PUTBNK
0129 3463 GETBNK
012C 721A GETDEV  Returns device # in A from PTRFIL (F997H)
012F 7210 NOROOM  No room for program loaded from disk: erase, close
                  files and print "OUT OF MEMORY"
0132 7209 CHKTOP  Check if enough room, else fall into NOROOM
0135 747D GETBF1
0138 7609 DERFOV  "FIELD OVERFLOW"
013B 6FD6 NAMSC1
013E 7615 DERSAP
0141 7067 FILSCN
0144 747A BETBUF
0147 40BE BEEP
014A 3C5A CNVCDO
014D 3CBC GETLEN
0150 3CA7 GETTRM
0153 3C39 GETCOD
0156 3CB5 SETTRM
0159 3CB3 TERMIN
015C 1365 FINPRT
015F 1AA5 GTBYTC
0162 0F99 INTIDX
0165 09FB INILIN
0168 5783 FRCSTR
016B 6993 GETSPA
016E 405D CKCNTC
```

```
0171 6557 SCRTCH  New command
0174 56B5 FRCINT
0177 7402 INDSKE
017A 6066 PTRGET  Get pointer to variable
017D 71AB SPSVEX
```

Namn : EVAL Adress : 162DH Kernel : 00ABH

Rutinen beräknar (evaluate) värdet av variabeln, konstanten eller funktionsanropet som HL pekar på. Efter anropet hamnar typen av variabel i VALTYP (F793H) där 2=heltal, 3=sträng, 4=enkel precision och 8=dubbel precision.
Tal i enkel & dubbel precision hamnar i FAC (F923H), heltal i FACLO (F925H) och för strängar hamnar adress till en sträng-beskrivning i FACLO.
Strängbeskrivningen är uppbyggd av 3 byte, 1 byte stränglängd och 2 byte strängadress.
Efteråt pekar HL på tecknet efter den uträknade funktionen.

Namn : PTRGET Adress : 6066H Kernel : 017AH

Hämtar adressen till variabeln vars namn HL pekar på. Adressen hamnar i DE och HL pekar på positionen efter variabelnamnet vid uthopp. Finns inte variabeln får DE värdet noll.

Här följer en lista på var rutinerna för att exekvera kommandon och instruktioner ligger i BASIC-tolken Rev 1.0.
Dessutom visas internkoden (Token) för alla kommandon.

Kommando	Token	Adress
ABS	06	55B1
AND	F8	1804
ASC	15	6B10
ATN	0E	5139
ATTR$	E9	34D3
AUTO	A9	11F5
BEEP	C0	40BE
BIN$	1D	6904
BLOAD	CD	7684
BSAVE	CE	7624
CDBL	20	5765
CHR$	16	6B20
CINT	1E	56B5
CIRCLE	BC	2652
CLEAR	92	67A6
CLICK	C8	31AF
CLOAD	9B	1EAA
CLOSE	B4	7375
CLS	9F	3777
CMD	D7	34C4
COLOR	BD	4552
CONT	99	671B
COPY	D6	34BF
COS	0C	50B8
CSAVE	9A	1E15
CSNG	1F	56DD
CSRLIN	E8	31C7
CVD	2A	7331
CVI	28	732B
CVS	29	732E
DATA	84	109B
DEF	97	188A
DEFDBL	AE	0F65
DEFINT	AC	0F5F
DEFSNG	AD	0F62
DEFSTR	AB	0F5C
DELETE	A8	1C6C
DIAL	D0	79C2
DIM	86	6061
DRAW	BE	29DA
DSKF	26	34C9
DSKI$	EA	34CE
DSKO$	D1	34A6
ELSE	A1	109D
END	81	66CF
EOF	2B	74B0
EQV	FB	181C
ERASE	A5	676E

Kommando	Token	Adress
ERL	E1	1671
ERR	E2	1663
ERROR	A6	11EA
EXP	0B	526B
FIELD	B1	72CD
FILES	B7	73B2
FIX	21	57E9
FN	DE	18AD
FOR	82	0D65
FPOS	27	74C6
FREE	0F	6CF7
GET	B2	2FB4
GOSUB	8D	0FF6
GOTO	89	1028
HEX$	1B	68FF
IF	8B	1225
IMP	FC	1824
INKEY$	EC	64F3
INP	10	1A37
INPUT	85	13D2
INSTR	E5	6BF0
INT	05	3048
IPL	D5	34BA
KEY	C7	3120
KILL	D4	34B5
LEFT$	01	6B66
LEN	12	6B04
LET	88	10C0
LFILES	BB	73AD
LINE	AF	1374
LIST	93	1AB8
LLIST	9E	1AB3
LOAD	B5	7121
LOC	2C	7484
LOCATE	D8	2FD1
LOF	2D	749A
LOG	0A	5197
LPOS	1C	1834
LPRINT	9D	125D
LSET	B8	7228
MAX	CA	7CBA
MDM	CF	3036
MERGE	B6	7122
MID$	03	6C73
MKD$	30	7318
MKI$	2E	7312
MKS$	2F	7315
MOD	FD	32BD
MON	CB	7B44
MOTOR	CC	2BE5
NAME	D3	34B0
NEW	94	6556
NEXT	83	6821
NOT	E0	????
OCT$	1A	68FA

Kommando	Token	Adress
OFF	EB	6909
ON	95	1124
OPEN	B0	7080
OR	F9	17FE
OUT	9C	1A4C
PAD	25	32BD
PAINT	BF	24FC
PDL	24	3280
PEEK	17	1CA6
PLAY	C1	2C24
POINT	ED	2346
POKE	98	1CAD
POSE	11	1839
PRESET	C3	2328
PRINT	91	1265
PSET	C2	232D
PUT	B3	2FB1
READ	87	1405
REM	8F	109D
RENUM	AA	1CF0
RESTORE	8C	66AE
RESUME	A7	119D
RETURN	8E	1061
RIGHT$	02	6B96
RND	08	5300
RSET	B9	7227
RUN	8A	0FE2
SAVE	BA	7167
SCREEN	C5	459A
SET	D2	34AB
SGN	04	55C6
SIN	09	50D1
SOUND	C4	2BFD
SPACE$	19	6B4D
SPRITE	EE	45D2
SQR	07	5222
STEP	DC	????
STICK	22	3206
STOP	90	66C8
STR$	13	6909
STRIG	23	3263
STRING$	E3	6B2E
SWAP	A4	6735
SWITCH	C9	337F
TAB	DB	????
TAN	0D	5120
THEN	DA	????
TIME	EF	31D3
=TIME		31BD
TO	D9	????
TROFF	A3	6730
TRON	A2	672F
USING	E4	????
USR	DD	1842
VAL	14	6BC0

Kommando	Token	Adress
VARPTR	E7	6066
VPEEK	18	46F2
VPOKE	C6	46D8
WAIT	96	1A52
WIDTH	A0	1AC6
XOR	FA	1812
>	F0	
=	F1	
<	F2	
+	F3	
-	F4	
*	F5	
/	F6	
Ü	F7	

Memory-mapp

Lista på användbara variabler i Spectravideons BASIC-tolk.

Lista är upplagd på följande sätt:
Hexadecimal adress, antal byte, namn och kort förklaring.

B=byte (1) W=Word (2) I=Instruktion (3)

```
F500   RAMLOW
F504   RNDCNT
F506   RNDTAB
F52B   USRTAB
F53F B ERRFLG  Error number
F540   LPTLST  Last Lpt operation 0=LF, 0<>print
F541   LPTPOS  Position of Lpt print head
F542   PRTFLG  =0  output device = CRT
               =1     "     "    = LPT
F543   LINLEN  Line length (39,40,80)
F545 B RUBSW   Rubout switch=1 means inside the processing of a
               rubout (inlin)
F546 W STKTOP  Top of stack, start of strigarea, 2:nd parameter in
               CLEAR statement
F548 W CURLIN  Current line #, FFFFH at direct statement in execution
F54A W TXTTAB  Start of Basic-program
F54C W VLZADR  Adress of character replaced by VAL
F54E B         ":": a colon for restarting input
F54F   KBUF    This is the CRUNCH buffer
F68D   BUFMIN
F68E   BUF     Input line buffer
F790 B ENDBUF  Make sure overrun stops
F791 B TTYPOS  Store terminal position
F792 B DIMFLG
F794 B VALTYP  Type of variable
F798   CONSAV
F7A2 W MEMSIZ  End of string area, start of file-buffer area
F7A4 W TEMPPT  Temporary pointer
F7A6   TEMPST
F7C4 B DSCTMP  Character count ) temporary string description
F7C5 W DSCPTR  pointer         )
F7C7 W FRETOP
F7C9   TEMP3
F7CB   TEMP8
F7CD   ENDFOR
F7D1   SUBFLG
F7D2   USFLG
F7D3   TEMP
F7D5   PTRFLG
F7D6 B AUTFLG  <>0 means AUTO mode
F7D7 W AUTLIN  Initial line for auto
F7D9 W AUTINC  Increment in auto
F7DB W SAVTXT
F7DD W SAVSTK  Stack temporary saved during command execution
F7DF W ERRLIN  Line number where last error occurred
```