Table of Contents

Manual#

- Atari_XEP80_Interface_Module_Owners_Manual.pdf ; size: 2 MB ; Original Atari XEP80 Manual (C) 1987 Atari

ATR-Image#

- XEP80_Boot_Disk-DX5087.atr ; Original Atari XEP80 Boot Disk DX5087 for NTSC systems (C) 1987 Atari
- XEP80_Boot_Disk-DX5087-PAL.atr ; patched driver for PAL systems
- Altirra-Additions.atr ; alternate XEP80 driver from the Altirra Additions Diskette ; faster than the original one from Atari!

Drivers#

- xep80han.arc ; XEP80 handler - .com-file
- xep80bxe.arc ; XEP80 driver to work with OSS BASIC XE
- xep80-new_stuff.zip ; archive with 3 atr-images containing XEP80 specific stuff
- ALTXEP80.SYS ; alternate XEP80 driver from the Altirra Additions Diskette ; faster than the original one from Atari!

Images#

Original Atari XEP80 Boot Disk DX5087 Label (C) 1987 Atari

The XEP 80 - Part 1#

by Erhard Pütz (aka Atreju aka Floppydoc)
prepared and translated by Mathy v. Nisselroy

Ever since the 1993 ABBUC Annual meeting (Jahreshauptversammlung) I've been working with the XEP 80 and gained quite a bit of knowledge and experience. I've used the XEP under BASIC, MAC65, BobTerm and from DOS to read texts.

At this time I want to ask everybody who's got information not mentioned in the manual, to send it to me. I'm especially interested in Information about the Hardware. Like, what graphicsprocessor as used?

While working with the XEP some very annoying things cought my attention. First there is this constant need to switch video plugs. Really nerve recking is readjusting of screen height, horizontal screen position and brightness. The height of the screen is almost right when the XEP is working in the American TV-standard with 50 Hz. But then the screen flickers and who knows how long the monitor will put up with the imposed 60 Hz. I once read that the higher screen frequency increases the current in the coils (Comment by Mathy: I don't quite know what these coils are called in English, but the direct the electronic beams that draw the picture on your cathode ray tube (according to this booklet I have)). This increases the load on both the electronics, which might not be designed handle this higher current, and the coils, which might burn out.

After switching to 50 Hz though, the screen is significantly higher than the standard XL/XE screen, pushing both upper and lower parts of the graphics off the screen. And the characters look really course. Once readjusted, one is rewarded with a good 80 character screen.

Speed

I've heard people say, the XEP would be slow. Well, when I print text to the XEP screen (X:) (original device driver), it rushes by so fast I hardly have time to press Ctrl-1 to halt the screen. Slow you say? Absolutely not, rather fast actually, in my opinion much faster then the original screen editor (E:).
So I tried out some demo's. So this is it, they really are slow. But why? For is purpose I printed out one of the BASIC-listings and read through it. And read it again, and again, and again... Oh my God, who committed this crime...Oops, programmed this. At that time I didn't understand the programm, but it was one big chaos and even on the first glance one would find the slowest BASIC solutions.
So, I'm gonna cut this thing short and state: the XEP 80 is fast. And thiss is mentioned in the manual really: Data transfer speed between XEP and computer takes place at 15.7 kBit. That's quite fast, but why this rare number? Why not 19.2 kBit? In that case the XEP would be able to keep up with the highest transfer rate of BobTerm. And why on the other hand, can the XEP only keep up with speeds of 4800 baud under BobTerm, where it should at least get 9600 baud? Let's start with the number 15.7 kBit. This means 15700 bits per second. The duration of one bit in this case is approximately 63.7us (micro seconds = 10^-6 or 0.000001 sec.). The time the microcessor in our computer needs to execute a command is measured in cycles. With the Atari 8bit every cycle takes approximately .564 us. One bit therefore takes about 113 cycles. The execution of command takes 2 to 7 cycles. While one bit is send, the computer could execute 16 to 56 commands. Let's use an average of 35. Then we can say one thing for sure: our 8 bitter can handle a transfer rate of 15.7 kBit hands down. Back to the question why it only works up to 4800 baud under BobTerm. Here's part of the XEP driver (V1.2) for BobTerm

```
LDX #$00          ; Load X with zero
SEI               ; don't execute IRQ's
STX $D40E         ; disable NMI's
STA WSYNC         ; pause
STX PORTA         ; send startbit
```

```
STA WSYNC         ; pause
STA PORTA         ; send bit 0
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 1
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 2
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 3
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 4
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 5
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 6
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send bit 7
ROR               ; next bit
STA WSYNC         ; pause
STA PORTA         ; send Mode bit
LDA #$FF
STA WSYNC         ; pause
STA PORTA         ; send stopbit
STA $D40E         ; enable NMI's
CLI               ; IRQ's permitted
LDX #$20          ; wait 90 us
DEX
BPL LOOP
LDA PORTA         ; read XEP status
AND #Mask         ; ($02 or $20)
BEQ Busy          ; XEP not ready yet
RTS               ; ready, 1 byte sent
```

Everywere, where it reads "pause" the computer does the following: NOTHING. This command tells the computer it should halt the microprocessor until one screen line is processed. This seems to take up as much time as one bit at 15.7 kBit, or 63.7 microseconds. This command was issued 11 times, add to that the 166 cycles in the 90 us waiting loop, where the computer can't do much either. This makes 866 cycles or maximum 431, on average 250 commands, the computer can not execute, just because it is halted. The data BobTerm receives via the RS232 interface however, come in faster then it can be processed. The data not processed in time are lost. And this only because the computer is busy doing nothing while transfering data to the XE. How could one change this and why could the XEP print to screen even faster yet? The computer has to the whole job, like for instance preparing the next byte that should be send to the XEP, in the time between 2 bytes. But what if the next byte to be send would instantaneously be available? The computer should get it from the buffer or wherever, while 1 byte is send to the XEP. To achieve this, the byte should be automatically - without the CPU having anything to do with it - be send and this is done by HARDWARE. Such a device is built into the computer and calls itself POKEY. That's the chip that, among other things, handles the datatraffic with your disk drive, printer and cassette. If we then think about how fast the data transfer with a disk drive can be, namely 51 kBit/s th a Speedy or even 78 kBit/s with the HSS Copier (Mathy: both are German hardware upgrades for the 1050 disk drive), it becomes clear what advantages there are to use a hardware solution.

As far I can see, POKEY can not be used for datatrafficing with the XEP, because the XEP uses 9 bit data words during data transfer, where POKEY only uses 8 bits. A standard SIO chip, like the 6551, do it. The nineth bit, used to distinguish between data and commands, could be emulated by the parity bit. And here the 6551 has five possibilities: None, odd, even, mark and space parity. How we would handle the nineth bit when the computer receives it is one of the things I have to think about, but it should be possible. So we take an EPROM, in which we place the new XEP 80 driver, a bit of electronics to decode some signals, a 6551 and a triple DIP switch. The DIP switch is used to select a device number (Mathy: Hm, nice idea! Where have I heard that one before :-)) and connect the whole thing to the parallel bus of the XL/XE. For the electrical engineers among you I have to add that the XL/XE first of al needs a couple of powerfull bus drivers in the shape of two 74AS244 and one 74AS245, especially if you're running a BlackBox!

The memory map of the driver:#

The driver for the XEP 80 basically exists of two parts. One of those parts is a so called Relocator, meaning, it moves the main programm to the top of the memory. Now I've recently worked with TurboWord+ und while reading the manual, I bumped into something strange. The driver of the XEP 80 should be called AUTORUN.SYS under TurboWord+ and SpartaDOS, because otherwise the MEMHI would be too low. Read it, tried it, seen it. But why? With AUTORUN.SYS, MEMHI is set at $95FF, when run by hand with XEP80 (RETURN) or from a batch file, it's set at $8FFF. I accepted it for a long time, but since it kept bugging me, I checked into this. I'm not gonna tell you how. But what came out will knock you off your sock When run by hand, the driver will install itself twice. But only the last one installed will be used. And it installs itself twice, because the file XEP80 ends with an INIT address, instead of a RUN address. DOS jumps to the INIT address ($2E2) after loading the file and when returning, tries to jump to the RUN address. But since there is none, it jumps to the beginning of the file, which starts the relocator once again. Using a disk monitor, I changed INIT to RUN and that was it. I was rewarded with 1.5 kByte more free memory.

RESET#

One thing you can not do is press RESET or otherwise cause a warm start. A warm start will reinitialise the device drivers, including the XEP 80 driver. One would think that all values are entere again, but the value for MEMHI is NOT entered. Sigh, those programmers at ATARI, real pro's. But even they can forget some things. However, after a warm start, MEMHI is set to the start of the play list. With BASIC this is $9C1F. If you now fill up the memory nicely, you'll eventually overwrite the XEP driver, which is located between $9600 and $9C1F. If at this point you press RESET again, not even DOS will be reinitialised and you will be unable to save anything. And that's by far not all. Small error, big results. Well, that's about all I want to say about the XEP 80 for now, but the text is over 10 kByte anyhow. Maybe it's even usefull to somebody. I'm always happy to get an echo.

Greetings, your Floppydoc - Erhard Puetz

The XEP 80 part two#

When I said in my previous article that I would be please about an echo, I meant, I would be pleased with ANY echo, not ONE echo. But I shouldn't complain, better ONE echo than NO echo at all. The echo said to me: Erhard, after I read you article about the XEP 80 in the ABBUC magazine (Mathy: number 38), I pulled my XEP 80 out of the drawer. I've always wanted to use it with BobTerm, but since the XEP driver for BobTerm doesn't switch the XEP to the German (Mathy: ahem, I thought it was called the European or PAL) standard, I've always had a flickering, too small image. Can't we do something about that?

Sure, I say, one takes the XIO command that sets the XEP 80 to the 50 Hz standard, turns that into a small machine language program and loads it as AUTORUN.SYS before BobTerm - Just a moment,

this wont gonna work this way. There is no XEP driver initialised yet, making the XIO command disappear into thin air. OK, first the XEP driver, then the XIO command, then BobTerm... Hmmm, won't work either. The XEP driver for BobTerm isn't hooked to the normal editor.

And, BobTerm loads and initializes the driver. - Think, Think, Think - The driver contains the subroutine for sending data to the XEP. That's the part we talked about in the previous article (Mathy: ABBUC magazine 38). Here is the address data for the driver: memory from $4000 to $422E and RUN $4001. Ah yes, this should work. At the end of the files, before the RUN address, I append a small routine and redirect the RUN address to it.

This is what that looks like:

```
$422F   SEC           ; A command is send
$4230   LDA #$D7      ; the number $D7 sets the XEP to 50 Hz in textmode
$4232   JSR $4145     ; That's the point where the Send-routine starts
$4235   JMP $4001     ; driver installed
       .ORG $02E0     ; RUN vector
       .WORD $422F    ; RUN address
```

Try it, it works. During all these try outs, I've searched through the XEP driver several times, and that's when I saw that the Send-outine starts a bit earlier than I told you last month. It starts like this:

```
$4139  X_DAT CLC          ; Data will be send
$413A        .BYTE
$413B  X_DAT SEC          ; A command is send
$413C        PHP          ; Get Carry status
$413D  WAIT_3 LDX VCOUNT  ; Number of just created image divided by two
$4140  CPX #$6B           ; Equal or greater than 107
$4142  BCS WAIT_3         ; Yes, then wait
$4144  PLP                ; return Carry Status
$4145  ROR                ; shift byte to the right until
$4146  ROR                ; bit 0 is send first.
$4147  ROR
$4148  ROR
$4149  ROR
```

... And this is where the part from the article in magzine 8 continues.

What do my square eyes see there? An other way that's that suppose to do? Well, boldly said, it's not allowed to send data during the Vertical Blank. But it's not clear to me why. The VBI is disabled anyhow. Might the WSYNC register in the "austastluecke" (Mathy: some kind of interval. Maybe I'll know what this word means when I translate the next article about the XEP, should Erhard write more of them. :-)) not deliver the apropriate timing? Let's try that. Starting with $413D I'll just write the NOP command 7 times. We're still losing time, but at least we're not waiting 50 times a second. So, loaded BobTerm with the changed driver and - Look at that. All of a sudden BobTerm works with the XEP 80 at 9600 baud, not just 4800. I've tried it out a bit more, but could not find any negative side effects. So I called the ABBUC BBS and looky here, works like a charm. With my chest raised in pride I called the ABBUCcian that had laid this task upon me. I've ooen it, I've done it, I said, not just the 50 Hz, but even 9600 baud.

The day before yesterday was the day that I decided to show off my newly gained knowledge in ABBUC magazine. So I called Wolfgang (Mathy: Wolfgang Burger is the president of ABBUC and previous SysOp of the ABBUC BBS) to ask him, if he would in interested in a sequel to the XEP 80 article. Sure, he said, opening up the way to help the ever information hungry ABBUCcians. And here's a little more, just to make sure you don't starve.

For $40E0 on, we have 13 bytes, with which the XEP is initialized:

value	function	
$D9	Cursor always on	
$D4	Use ATASCII characterset	
$DE	Bright characters on dark background	
$D3	Burst mode, no cursor data is send back	
$60	Left Margin 0 low nibble	
$70	Left Margin 0 Hi nibble	
$AF	Right Margin low nibble 15 + 4*16 Hi nibble = 79	
$B4	Right Margin hi nibble	
$00	Horizontal cursor position = 0	
$80	vertical cursor position = 0	
$D0	Erase List flag	
$C5	Fill RAM with SPACE-characters	
$DC	Set Scroll window to X-cursor position	

Since all this trail and error stuff started to annoy me, I "just for a minute" reassembled the driver, so I now a fairly readable file to work with. This means I'm gonna write more articles about the XEP 80. Since I was able to delve up more information about the XEP 80, I can already tell you: the 19200 baud are in reachable distance and I seem to be on a hot trail with my suspiciouns about the 6551 SIO chip from the first article (I think).

DEEPER DUNGEONS#

The videocontroller inside the XEP 80 is a 48 pin chip. It's data bus is 16 bits wide on the video side. This way, two RAMs could be read simultaniously, one containing the text that should be displayed and another containing the text attributes. There are eight possible attributes:

attribute	display	
Inverse	A character plus it's surrounding area is display "Inverse"	
Halve Bright	The character in displayed with a lower brightness. Other options are a higher brightness and even color.	
Flickering	On, Off, On, Off	
Double Height	Text in double height, internally some conditions have to be met though.	
Double Width	Text in double wide. The next line can't be displayed though, so we have to leave a blank line between two lines of character	
Underlined	An apparently freely defined character is mixed in. This can be the "underline" character or an "overline" or "strait through the middle" character or whatever you want	
Hidden	The text is not displayed at all. This attribute should also be set when using a double height character, so the controller knows that this is the bottom halve of the double height haracter.	
Graphics	Since this is an attribute, text and blockgraphics can be mixed at will. Except double height, all other attributes can be used at the same time	

And then there is the Graphics mode of the XEP 80. Just like the subject of block graphics, it's a very broad subject and will not be discussed here.

THE LIGHT PEN#

The controller offers the option of connecting a lightpen to it. Via interrupts, the horizontal and vertical positions are stored into two registers. HPEN is 7 bit wide (128 positions) and VPEN is 5 bit wide (32 positions). Since this does not provide us with a very high resolution, we can forget about drawing with a light pen.

THE SERIAL INTERFACE#

Here we find a UART (Universal Asynchronus Receiver Transmitter), that, in it's addresses and functions, equals the good old 6551 so much, that I'm assuming it is. With a 4 bit pre-devider, the system clock (12MHz in the XEP 80) is devided into one of 16 steps. These steps are 3.4, 4, 4.5, ... ,11. The actual baudrate devider is 11 bits wide and is further devided by 16 at the output. In the XEP 80 the baudrate is set to 15625. The mathematics could however also look like this:

```
12 MHz     : 6.5 = approximately 1.8432 MHz
12 MHz     : 6.5 = approximately 1.8432 MHz
```

```
1.8432 MHz : 6   = 307200 Hz
1.8432 MHz : 3   = 614400 Hz

307200 Hz  : 16  = 19.200 kHz
614400 Hz  : 16  = 38.400 kHz
```

Those 19.200 kHz lies within manufacturers specifications. If 38.400kHz works, is something someone should try out. But since the dividers aren't used up yet (we can even go much faster), why won't it work.

That's it for this time.

Greetings, your FloppyDoc

The Original ATARI XEP 80 Handler Source#

Handler Source#

```
0100 ;.OPT NOLIST
0110  .OPT NOEJECT
0120 ;
0130 BASE=$6000
0140 ;
0150 ;CHAR EQUATES
0160 ;
0170 LF=10
0180 CR=13
0190 ESC=$1B
0200 SPACE=$20
0210 CNTL=$5E
0220 CLS=$7D
0230 EOL=$9B
0240 ;
0250 ;80 COL COMMANDS
0260 ;
0270 XCH80=$50
0280 LMG80=$60
0290 LMH80=$70
0300 YCR80=$80
0310 SGR80=$99
0320 PAG80=$9A
0330 RMG80=$A0
0340 RMH80=$B0
0350 GET80=$C0
0360 CUR80=$C1
0370 RST80=$C2
0380 PST80=$C3
0390 CLR80=$C4
0400 LIS80=$D0
0410 SCR80=$D2
0420 SCB80=$D3
0430 GRF80=$D4
0440 ICM80=$D5
0450 PAL80=$D7
0460 CRS80=$D9
0470 MCF80=$DB
0480 PNT80=$DD
```

```
0490 ;
0500 ;MEMORY EQUATES
0510 ;
0520 DOSINI=$0C
0530 ICDNOZ=$21
0540 ICCOMZ=$22
0550 ICAX1Z=$2A
0560 ICAX2Z=$2B
0570 ICIDNO=$2E
0580 LMARGN=$52
0590 RMARGN=$53
0600 VCP=$54
0610 HCP=$55
0620 IN=$CC
0630 CDTMV3=$21C
0640 SDMCTL=$22F
0650 KEYDEL=$2D9
0660 KEYREP=$2DA
0670 DVSTAT=$2EA
0680 CRSINH=$2F0
0690 CHBAS=$2F4
0700 LISTF=$2FE
0710 SFLAG=$2FF
0720 HATABS=$31A
0730 ICDNO=$341
0740 ICCOM=$342
0750 PAL=$D014
0760 IRQEN=$D20E
0770 SKSTAT=$D20F
0780 PORTA=$D300
0790 PACTL=$D302
0800 DMACTL=$D400
0810 WSYNC=$D40A
0820 VCOUNT=$D40B
0830 NMIEN=$D40E
0840 ;
0850  *=BASE-2
0860 ;
0870  .WORD CEND-BEGIN ;RELOCATER INFO
0880 ;
0890 BEGIN JSR ERTS ;DOSINI VECTOR
0900  JMP CINIT ;RELOCATER JUMP
0910 ;
0920 PAUX1 .BYTE 0
0930 PAUX2 .BYTE 0
0940 ;
0950 READ JSR DISAB;DISABLE IRQ INTS
0960  LDA #GET80
0970  JSR CINP ;REQUEST, GET CHAR
0980  PHA ;SAVE CHAR
0990  JSR INPUT ;GET CURS
1000  JSR CURCK ;CHECK FOR X>$4F
1010  PLA ;RESTORE CHAR
1020  JMP ENAB
1030 ;
1040 CINP JSR CMD
1050 INPUT LDA #00 ;TIME CRITICAL CODE
1060  TAX ;MUST NOT CROSS A
1070  LDY #31 ;PAGE BOUNDARY
```

```
1080  STA DATIN
1090 IN0 LDA PORTA ;4
1100  AND INMSK ;4
1110  BEQ IN01 ;3 IF A 0, 2 IF NOT
1120  DEX
1130  BNE IN0
1140  DEY ;TIMEOUT LOOPS
1150  BNE IN0
1160  SEC ;NO RESPONSE
1170  RTS
1180 IN01 LDX #08
1190  LDY #12 ;2
1200 IN1 DEY
1210  BNE IN1 ;5*Y-1
1220  NOP ;2
1230 IN10 LDY #15 ;2 MAIN DLY COUNT
1240 IN2 DEY
1250  BNE IN2 ;5*Y-1
1260  LDA PORTA ;4 GET BYTE
1270  AND INMSK ;4 GET BIT
1280  CLC ;2
1290  BEQ IN25 ;0=3,1=2
1300  SEC ;1=2
1310 IN25 BCC IN26 ;0=3,1=2
1320 IN26 DEX ;2 DEC COUNT
1330  BMI IN3 ;2 (3 DONE)
1340  ROR DATIN ;6 SHIFT IN BIT
1350  BCC IN10 ;3 ALWAYS
1360 IN3 LDY #15 ;DELAY 1/2 BIT
1370 IN33 DEY
1380  BNE IN33
1390  LDA DATIN ;GET CHAR (Y=0)
1400  BCC I5 ;RETURN IF CHAR
1410  BPL I0 ;HORIZ WITH NO VERT
1420  AND #$7F ;CLEAR UPPER FLAG
1430  CMP #$51 ;TEST HORIZ/VERT
1440  BCC I00 ;HORIZONTAL
1450  AND #$1F ;CLEAR MID FLAG
1460  BCS I01 ;SAVE VERT
1470 I00 JSR I0 ;SAVE HORIZ
1480  BCC INPUT ;GET VERT
1490 I0 INY ;OFFSET FOR HORIZ
1500 I01 STA VCP,Y ;CURS POSITION
1510  STA VCS,Y ;CURS SHADOW
1520  CLC ;INDICATE RESPONSE
1530 I5 RTS
1540 ;
1550 CURCK LDA HCP ;CHECK HORIZ CURSOR
1560  CMP #$50 ;FOR >$4F
1570  BCC I5 ;IF NOT
1580  LDA #CUR80 ;GO GET REAL VALUE
1590  JSR CINP
1600  JMP I0 ;AND STORE IT (Y=0)
1610 ;
1620 CMD SEC ;THIS CODE MUST NOT
1630  BCS OUT ;CROSS A PAGE BOUNDARY
1640 OUTPUT CLC ;CMD FLAG=0 FOR CHAR
1650 OUT LDY #00
1660  JSR SEND ;SEND START BIT
```

```
1670  LDX #08 ;SETUP BIT COUNT OF 9
1680  NOP
1690  NOP
1700  NOP ;2+2+2+2=8
1710 OUT0 ROR A ;PUT BIT INTO CARRY
1720  BCS HI
1730  BCC LO ;2+3=5 CYCLES TO LO
1740 LO LDY #00 ;5+2 CYCLES TO JSR
1750  JSR SEND ;SEND A 0
1760  BCC OUT1 ;3 CYCLES
1770 HI LDY OUTMS ;3+4 CYCLES TO JSR
1780  JSR SEND ;SEND A 1
1790  BCS OUT1 ;3 CYCLES
1800 OUT1 DEX ;NEXT BIT 2 CYC
1810  BPL OUT0 ;MORE 3 OR 2 CYC
1820  BMI OUT2 ;SEND STOP BIT 3 CYC
1830 OUT2 LDY OUTMS ;SEND A 1
1840  BNE OUT3
1850 OUT3 JSR SEND ;2+3+4+3=12
1860  RTS
1870 SEND STY PORTA ;OUTPUT BIT
1880  LDY #12 ;TIMER FOR 15.7KB
1890 S1 DEY
1900  BNE S1 ;5*Y-1 CYCLES
1910  BEQ S2 ;3
1920 S2 NOP
1930  NOP
1940  NOP
1950  NOP ;2+2+2+2=8
1960 S3 RTS ;6 CYCLES
1970 ;
1980 COM LDA ICCOMZ ;GET COM BYTE
1990  CMP #$14 ;CHECK DEBUG OUT
2000  BNE COM1 ;TRY NEXT XIO
2010  LDA ICAX2Z ;GET AUX 2
2020 COMSD JSR DISAB ;STOP INTERRUPTS
2030  JSR CMD ;GO SEND
2040  JMP ENAB ;ENABLE AND NORM EXIT
2050 COM1 CMP #$15 ;TEST VALID
2060  BNE COM2 ;NEXT
2070  LDA ICAX2Z ;GET AUX 2
2080  BNE COMBR ;GO DO BURST
2090  STA MODE ;MAKE NORMAL
2100  LDA #SCR80 ;GET CMD
2110  BNE COMSD ;GO SEND
2120 COMBR STA MODE ;MAKE BURST
2130  LDA #SCB80 ;GET CMD
2140  BNE COMSD ;GO SEND
2150 COM2 CMP #$16 ;CHECK DEBUG IN
2160  BNE COM3 ;NEXT
2170  LDA ICAX2Z ;GET BYTE TO SEND
2180  JSR DISAB
2190  JSR CINP ;REQUEST, GET CHAR
2200  STA DVSTAT+1 ;FOR NOW
2210  JMP ENAB
2220 COM3 CMP #$19 ;CHECK 80/40
2230  BNE COM4
2240  JMP XIO19 ;DO IT
2250 COM4 RTS
```

```
2260 ;
2270 PCOM LDA ICCOMZ ;GET CMD
2280  CMP #$17 ;TEST VALID
2290  BNE S3 ;NO MORE FOR NOW
2300  LDA ICAX2Z ;GET AUX 2
2310  CMP #08 ;CHECK RESERVED
2320  BCS S3 ;NO GOOD
2330  AND #03 ;CHECK 3 AND 7
2340  EOR #03
2350  BEQ S3 ;NOT ALLOWED
2360  LDA ICAX1Z ;GET AUX1
2370  CMP #08 ;CHECK UPPER LIMIT
2380  BCS S3 ;NO GOOD
2390  STA PAUX1
2400  LDA ICAX2Z ;GET AUX 2
2410  STA PAUX2
2420  JMP EXIT
2430 ;
2440 WRITE LDY SFLAG ;CHECK CNTL 1
2450  BNE WRITE ;IF ON
2460  JSR DISAB
2470  LDY DEV ;ARE WE SCREEN?
2480  BEQ WR2 ;YES
2490  PHA
2500  LDA #00
2510  STA DEV
2520  LDA #SCR80
2530  JSR CMD
2540  PLA
2550 WR2 LDY LISTF ;CHECK LIST FLAG
2560  CPY LISTS
2570  BEQ WR3
2580  STY LISTS ;SAVE NEW VALUE
2590  PHA
2600  TYA
2610  BEQ WR25
2620  LDA #01 ;FORCE LSB
2630 WR25 ORA #LIS80
2640  JSR CMD ;SEND NEW VALUE
2650  PLA
2660 WR3 JSR ALIGN ;SET PARMS
2670  LDY CHBAS ;CHECK CHAR SET
2680  CPY CHSH
2690  BEQ WR5
2700  CPY #$E0
2710  BNE WR4
2720  STY CHSH
2730  PHA
2740  LDA #GRF80
2750 WR35 JSR CMD
2760  PLA
2770  JMP WR5
2780 WR4 CPY #$CC
2790  BNE WR5
2800  STY CHSH
2810  PHA
2820  LDA #ICM80
2830  BNE WR35
2840 WR5 LDY CRSINH ;CHECK CURS FLAG
```

```
2850  CPY CRSS
2860  BEQ WR6
2870  STY CRSS
2880  PHA
2890  TYA
2900  BEQ WR55
2910  LDA #01
2920 WR55 EOR #CRS80 ;CURSOR ON/OFF
2930  JSR CMD
2940  PLA
2950 WR6 JSR OUTPUT ;SEND CHAR
2960  LDA MODE ;TEST FOR BURST
2970  BNE WWAT ;IF SO
2980  JSR INPUT ;GET NEW CURSOR
2990  JSR CURCK ;CHECK FOR X>$4F
3000  JMP ENAB
3010 WWAT JSR ENAB ;ENABLE INTS
3020  LDY #25 ;OR SUCH
3030  JSR S1
3040 WW1 LDA PORTA
3050  AND INMSK
3060  BEQ WW1
3070 WW2 LDY #01
3080  RTS
3090 ;
3100 POPEN STX TIOCB
3110  LDX ICDNOZ
3120  JSR MATRIX
3130  BCS HANDGO
3140  LDY SDMCTL
3150  BNE WW2
3160  LDA #PST80
3170  JSR DISAB
3180  JSR CINP ;REQUEST, GET CHAR
3190  BNE POP1
3200  LDA #139 ;NOBODY HOME
3210 POP1 JSR ENAB
3220  TAY
3230  RTS
3240 ;
3250 HANDGO LDA ICDNOZ
3260  STX ICDNOZ
3270  PHA
3280  LDA ICCOMZ
3290  AND #08
3300  TAX
3310  JSR HAND
3320  PLA
3330  STA ICDNOZ
3340  RTS
3350 ;
3360 PWRT TAY ;SAVE CHAR
3370  STX TIOCB
3380  LDA ICDNO,X
3390  TAX
3400  JSR MATRIX
3410  TYA
3420  BCC PWP
3430  STA TCHAR ;SAVE CHAR FOR CALL
```

```
3440  LDY TIOCB ;GET UNIT #
3450  LDA ICDNO,Y
3460  PHA ;SAVE UNIT #
3470  TXA ;GET NEW VALUE
3480  STA ICDNO,Y ;REPLACE WITH NEW
3490  STA ICDNOZ ;AND ZERO PAGE
3500  LDX #06
3510  JSR HAND ;GO PRINT
3520  PLA ;RESTORE UNIT #
3530  LDX TIOCB ;GET POINTER
3540  STA ICDNO,X ;RESTORE OLD
3550  STA ICDNOZ
3560  RTS
3570 PWP LDY DEV ;CHECK OUTPUT DEV
3580  BNE PW0
3590  PHA
3600  LDY SDMCTL
3610  BEQ PW2
3620 PW1 LDY VCOUNT
3630  CPY #129
3640  BNE PW1
3650 PW2 JSR DISAB
3660  LDA #PNT80
3670  STA DEV
3680  JSR CMD
3690  JSR ENAB
3700  PLA ;RESTORE CHAR
3710 PW0 TAY ;SAVE CHAR
3720  LDA PAUX2 ;GET CNTL
3730  ROR A ;CHECK NO XLATE
3740  TYA ;RESTORE CHAR
3750  BCS DOIT ;DONT XLATE
3760  CMP #EOL ;CHECK EOL
3770  BNE XLATE ;XLATE IF NOT
3780  LDA #CR ;REPLACE WITH CR
3790  JSR DOIT ;SEND
3800  LDA PAUX2 ;GET CNTL
3810  AND #04 ;CHECK NO APPEND
3820  BNE WGDS ;DONT APPEND
3830  LDA #LF ;GET LF
3840  BNE DOIT ;SEND
3850 XLATE LDA PAUX2 ;GET CNTL
3860  CMP #02 ;CHECK LIGHT XLATE
3870  TYA ;RESTORE CHAR
3880  BCC DOIT ;DONE WITH XLATE
3890  AND #$7F ;REMOVE MSB
3900  CMP #$20 ;CHECK ASCII CHAR
3910  BCS DOIT ;GO PRINT ASCII
3920  PHA ;SAVE CHAR
3930  LDA #CNTL ;GET "CNTL" CHAR
3940  JSR DOIT ;SEND
3950  PLA ;RESTORE CHAR
3960  ORA #$40 ;MAKE ALPHA
3970 DOIT LDY SDMCTL
3980  BEQ DO1
3990 DO0 LDY VCOUNT
4000  CPY #129
4010  BNE DO0
4020 DO1 JSR DISAB
```

```
4030  JSR OUTPUT
4040  JSR ENAB
4050 WAIT LDY #25 ;FOR NOW
4060  JSR S1
4070  LDY #02
4080 W0 LDX #255 ;FOR NOW
4090  STX CDTMV3 ;SETUP VBLANK COUNT
4100 W1 LDA PORTA
4110  AND INMSK
4120  BNE WGDS ;AVAILABLE
4130  LDA CDTMV3 ;CHECK COUNTDOWN
4140  BNE W1
4150  DEY
4160  BNE W0
4170 WTMO LDY #138 ;DO TIMEOUT
4180  BNE WRTS ;COULD DO BRKKEY ALSO
4190 WGDS LDY #01
4200 WRTS RTS
4210 ;
4220 MATRIX CPX #02
4230  BEQ PNEXT
4240  BCS POVER
4250  LDA PAUX1
4260  LSR A
4270 POVER RTS
4280 PNEXT LDA #03
4290  CMP PAUX1
4300  BCC POVER
4310  LDA PAUX1
4320  AND #02
4330  BEQ POVER
4340  DEX
4350  RTS
4360 ;
4370 HAND LDA $E431,X
4380  PHA
4390  LDA $E430,X
4400  PHA
4410  LDA TCHAR ;RESTORE CHAR
4420  LDX TIOCB ;GET IOCB POINTER
4430  RTS ;CALL PRINTER HANDLER
4440 ;
4450 FORCOM LDA ICCOMZ
4460  CMP #$18
4470  BEQ XIO18
4480  RTS
4490 ;
4500 CINIT LDA #00
4510  STA TOGGLE
4520  JSR JINIT
4530  LDA #$50
4540  LDY #02
4550  JSR FSET
4560  LDA #$53
4570  JSR FIND
4580  LDA HATABS+1,X
4590  STA TEMPSV
4600  LDA HATABS+2,X
4610  STA TEMPSV+1
```

```
4620  LDA #$45
4630  JSR FIND
4640  LDA HATABS+1,X
4650  STA IN
4660  LDA HATABS+2,X
4670  STA IN+1
4680  LDY #15
4690 C003 LDA (IN),Y
4700  STA TMTAB,Y
4710  DEY
4720  BPL C003
4730  LDA #FORCOM-1&$FF
4740  STA TMTAB+10
4750  LDA #FORCOM-1/256
4760  STA TMTAB+11
4770  LDA SKSTAT
4780  AND #08
4790  BEQ C004
4800 XIO18 LDA #$45
4810  LDY #00
4820  JSR FSET
4830  LDA #$53
4840  LDY #01
4850  JSR FSET
4860  LDA #WRITE-1&$FF
4870  STA $346
4880  LDA #WRITE-1/256
4890  STA $347
4900 EOPEN LDX #00
4910  JSR FESUB
4920  LDA ICAX1Z ;GET AUX 1
4930  AND #32 ;CHECK CLEAR BIT
4940  BNE C005 ;DONT DO RESET
4950  LDA #00
4960  LDX #06
4970 C0035 STA VCS,X
4980  DEX
4990  BPL C0035
5000  LDA #$E0
5010  STA CHSH
5020  LDA #$4F
5030  STA RMARGS
5040  STA COMPOS
5050  JSR DISAB
5060 IO00 LDA #RST80 ;RESET 80 COL
5070  JSR CINP ;REQUEST, GET CHAR
5080  BCC IO01 ;GOT IT
5090  JSR JTOGL ;SWITCH PORTS
5100  BNE IO00 ;DO IT AGAIN
5110 IO01 LDA PAL ;CHECK COMPUTER TYPE
5120  AND #$0E
5130  BNE IOP1
5140  LDA #PAL80
5150  JSR CMD ;SET 80 COL TO 50HZ
5160 IOP1 JMP ENAB
5170 ;
5180 XIO19 LDX #06
5190  JSR FESUB
5200  LDA #$53
```

```
5210  JSR FIND
5220  LDA TEMPSV
5230  STA HATABS+1,X
5240  LDA TEMPSV+1
5250  STA HATABS+2,X
5260  LDA TMTAB+6
5270  STA $346
5280  LDA TMTAB+7
5290  STA $347
5300 C004 LDA #$45
5310  LDY #03
5320  JSR FSET
5330  JMP EXIT
5340 C005 LDA #SCR80 ;IN CASE A PRINT
5350  JMP COMSD ;HAS OCCURED
5360 ;
5370 EDTAB .WORD EOPEN-1 ;OPEN
5380  .WORD EXIT-1 ;CLOSE
5390  .WORD EGET-1 ;GET LINE OF TEXT
5400  .WORD WRITE-1 ;PUT (NO CURS)
5410  .WORD EXIT-1 ;STATUS
5420  .WORD COM-1 ;SPECIAL (CMD OUT)
5430  JMP EXIT ;INIT
5440  .BYTE 0
5450 ;
5460 PRTAB .WORD POPEN-1 ;OPEN
5470  .WORD EXIT-1 ;CLOSE
5480  .WORD ERTS-1 ;GET
5490  .WORD PWRT-1 ;PUT
5500  .WORD POPEN-1 ;STATUS
5510  .WORD PCOM-1 ;SPECIAL
5520  JMP EXIT ;INIT
5530  .BYTE 0
5540 ;
5550 SCTAB .WORD SOPEN-1 ;OPEN
5560  .WORD EXIT-1 ;CLOSE
5570  .WORD SREAD-1 ;GET-LOCATE
5580  .WORD SWRIT-1 ;PUT-PLOT
5590  .WORD EXIT-1 ;STATUS
5600  .WORD ERTS-1 ;SPECIAL
5610 SCT1 JMP EXIT ;INIT
5620  .BYTE 0
5630 ;
5640 TMTAB .WORD EXIT-1,EXIT-1
5650  .WORD EXIT-1,EXIT-1
5660  .WORD EXIT-1,EXIT-1
5670  JMP EXIT
5680  .BYTE 0
5690 ;
5700 SOPEN LDA ICAX2Z
5710  AND #08
5720  BEQ SCT1
5730  LDA ICAX1Z
5740  AND #16
5750  BNE SCT1
5760  LDA #00 ;SEND 0
5770  JSR DISAB
5780  JSR OUTPUT
5790  JSR INPUT
```

```
5800  LDA #SGR80 ;SET GRAPHICS
5810  JSR CMD
5820  LDA PAL
5830  AND #$0E
5840  BNE SOP1
5850  LDA #PAG80
5860  JSR CMD
5870 SOP1 LDA #CLR80 ;FILL WITH 0 SENT
5880  JSR CINP ;REQUEST, GET CHAR
5890  LDA #01
5900  JMP COMBR
5910 ;
5920 SREAD JSR DISAB
5930  JSR ALIGN ;SET PARMS
5940  JSR READ ;GET CHAR
5950  CMP #EOL ;CHECK EOL
5960  BNE SCT1 ;RETURN NORMAL
5970  LDA #SPACE ;REPLACE WITH SPACE
5980  BNE SCT1 ;RETURN NORMAL
5990 ;
6000 SWRIT PHA ;SAVE CHAR
6010  LDA #ESC ;FORCE PRINT
6020  JSR WRITE
6030  PLA ;RESTORE CHAR
6040  JMP WRITE ;SEND IT
6050 ;
6060 EGET LDA COMPOS
6070  BEQ EBACK
6080  LDA HCP
6090  STA HCPS
6094  STA HCPE
6100 EG1 JSR KCALL ;GET KB BYTE
6110  CMP #EOL
6120  BEQ EGBAK
6130  JSR WRITE ;SEND TO 80 COL
6132  LDY HCP ;THIS CODE IS FOR
6134  CPY HCPE ;SPECIAL CASE LINES
6140  BCC EG1 ;DONT UPDATE IF LESS
6142  STY HCPE
6144  BCS EG1
6150 EGBAK STY KSTAT ;SAVE STATUS
6160  CPY #$80 ;CHECK STAT
6170  BCS EBA0 ;DO EOL IF EOF/BREAK
6180  JSR DISAB ;DISAB FOR CMD
6190  LDA #00
6200  STA COMPOS
6210  LDA MODE
6220  BEQ EG2
6230  LDA #00
6240  BEQ EG3
6250 EG2 LDA HCPS
6260 EG3 JSR CMD ;X CURS TO OLD VAL
6270  LDA #MCF80 ;Y CURS TO FIRST
6280  JSR CMD
6290 EBACK JSR READ ;GO GET A CHAR
6300  CMP #EOL
6310  BNE EGXT ;NOT DONE YET
6312  LDY HCP
6314  CPY HCPE ;CHECK RIGHTMOST
```

```
6315  BCS EBA0 ;CURSOR POSITION
6316  LDA #SPACE ;IF NOT THERE
6318  BNE EGXT ;THEN FAKE SPACE
6320 EBA0 STA COMPOS ;SET NON 0
6330  JSR WRITE
6340  LDA #EOL ;RETURN WITH EOL
6350 EGXT LDY KSTAT ;GET STATUS
6360  RTS
6370 ;
6380 KCALL LDA $E425
6390  PHA
6400  LDA $E424
6410  PHA
6420  RTS
6430 ;
6440 DISAB LDY #00
6450  STY NMIEN
6460  SEI
6470  RTS
6480 ;
6490 ENAB LDY #$C0
6500  STY NMIEN
6510  CLI
6520 EXIT LDY #01
6530 ERTS RTS
6540 ;
6550 VCS .BYTE 0
6560 HCS .BYTE 0
6570 DEV .BYTE 0
6580 LMARGS .BYTE 0
6590 LISTS .BYTE 0
6600 MODE .BYTE 0
6610 CRSS .BYTE 0
6620 ;
6630 CHSH .BYTE 0
6640 RMARGS .BYTE 0
6650 COMPOS .BYTE 0
6660 ;
6670 DATIN .BYTE 0
6680 HCPS .BYTE 0
6685 HCPE .BYTE 0
6690 KSTAT .BYTE 0
6700 INMSK .BYTE 0
6710 OUTMS .BYTE 0
6720 TOGGLE .BYTE 0
6730 TIOCB .BYTE 0
6740 TCHAR .BYTE 0
6750 ;
6760 TEMPSV .WORD 0
6770 ;
6780 INMST .BYTE 02,$20
6790 OUTMT .BYTE 01,$10
6800 ;
6810 FETAB .BYTE 0,0,0,$4F,24,3
6820  .BYTE 62,0,2,39,30,6
6830 ;
6840 LOOKUP .BYTE "ESP"
6850 LOWAD .BYTE EDTAB&$FF,SCTAB&$FF,PRTAB&$FF,TMTAB&$FF
6860 ;
```

```
6870 ALIGN LDY HCP ;GET HCURS
6880  CPY HCS ;COMPARE TO SHADOW
6890  BEQ A1 ;NO CHANGE
6900  STY HCS ;SAVE NEW VALUE
6910  PHA ;SAVE CHAR
6920  TYA
6930  CMP #$50
6940  BCC A00
6950  LSR A
6960  LSR A
6970  LSR A
6980  LSR A
6990  ORA #XCH80
7000  PHA
7010  TYA
7020  AND #$0F
7030  JSR CMD
7040  PLA
7050 A00 JSR CMD ;SEND NEW CURSOR
7060  PLA
7070 A1 LDY VCP ;GET VCURS
7080  CPY #25 ;CHECK UPPER LIMIT
7090  BCC A15
7100  LDY #24 ;STATUS LINE
7110 A15 CPY VCS ;COMPARE TO SHADOW
7120  BEQ A2 ;NO CHANGE
7130  STY VCS ;SAVE NEW VALUE
7140  PHA ;SAVE CHAR
7150  TYA
7160  ORA #YCR80 ;SET CMD BIT
7170  JSR CMD ;SEND NEW CURSOR
7180  PLA
7190 A2 LDY LMARGN
7200  CPY RMARGN
7210  BCC A24
7220  LDY #00
7230  STY LMARGN
7240 A24 CPY LMARGS
7250  BEQ A3
7260  STY LMARGS
7270  PHA
7280  TYA
7290  AND #$0F
7300  ORA #LMG80
7310  JSR CMD
7320  LDA LMARGN
7330  LSR A
7340  LSR A
7350  LSR A
7360  LSR A
7370  BEQ A25
7380  ORA #LMH80
7390  JSR CMD
7400 A25 PLA
7410 A3 LDY RMARGN
7420  CPY RMARGS
7430  BEQ A4
7440  STY RMARGS
7450  PHA
```

```
7460  TYA
7470  AND #$0F
7480  ORA #RMG80
7490  JSR CMD
7500  LDA RMARGN
7510  LSR A
7520  LSR A
7530  LSR A
7540  LSR A
7550  CMP #04
7560  BEQ A35
7570  ORA #RMH80
7580  JSR CMD
7590 A35 PLA
7600 A4 RTS
7610 ;
7620 FESUB LDA FETAB,X
7630  STA SDMCTL
7640  STA DMACTL
7650  LDA FETAB+1,X
7660  STA VCP
7670  LDA FETAB+2,X
7680  STA HCP
7690  STA LMARGN
7700  LDA FETAB+3,X
7710  STA RMARGN
7720  LDA FETAB+4,X
7730  STA KEYDEL
7740  LDA FETAB+5,X
7750  STA KEYREP
7760  RTS
7770 ;
7780 FIND LDX #00
7790 F1 CMP HATABS,X
7800  BEQ F2
7810  INX
7820  INX
7830  INX
7840  BNE F1
7850 ;
7860 FSET JSR FIND
7870 SET LDA LOWAD,Y
7880  STA HATABS+1,X
7890  LDA #EDTAB/256
7900  STA HATABS+2,X
7910 F2 RTS
7920 ;
7930 JTOGL LDA #01
7940  EOR TOGGLE
7950  STA TOGGLE
7960 JINIT LDX TOGGLE
7970  LDY INMST,X
7980  STY INMSK
7990  LDY OUTMT,X
8000  STY OUTMS
8010  LDA #$FF
8020  STA PORTA
8030  LDX #$38
8040  STX PACTL
```

```
8050  STY PORTA
8060  LDX #$3C
8070  STX PACTL
8080  RTS
8090 ;
8100 CEND=*
8110 ;
8120  .END
```

Relocator Source#

```
0100  .OPT NOEJECT
0110 DOSINI=$0C
0120 IN=$CC
0130 OUT=$CE
0140 OFFSET=$D0
0150 MEMTOP=$2E5
0160 CBEGIN=LENGTH+2
0170  *=$3000
0180 ENTER SEC
0190  LDA MEMTOP
0200  SBC LENGTH
0210  LDA MEMTOP+1
0220  SBC LENGTH+1
0230  STA OUT+1
0240  STA OFFSET
0250  TAY
0260  DEY
0270  STY MEMTOP+1
0280  LDY #$FF
0290  STY MEMTOP
0300  INY
0310  STY OUT
0320  LDX #00
0330  LDA #CBEGIN&$FF
0340  STA IN
0350  LDA #CBEGIN/256
0360  STA IN+1
0370  JSR MOVI
0380 HERE TAX
0390  BMI MORE
0400  JSR MOVI
0410  CLC
0420  ADC OFFSET
0430  LDX #00
0440  JSR MOV
0450  BMI HERE
0460 MORE INX
0470  BEQ OVER
0480  DEX
0490  JSR MOVI
0500  BMI HERE
0510 MOV STA (OUT),Y
0520  INC OUT
0530  BNE MOVI
0540  INC OUT+1
0550 MOVI LDA (IN),Y
0560  INC IN
```

```
0570  BNE MOVO
0580  INC IN+1
0590 MOVO DEX
0600  BPL MOV
0610 ARND RTS
0620 OVER LDA #01
0630  STA OUT
0640  LDA OFFSET
0650  STA OUT+1
0660  LDA DOSINI
0670  STA (OUT),Y
0680  INC OUT
0690  LDA DOSINI+1
0700  STA (OUT),Y
0710  INC OUT
0720  TYA
0730  STA DOSINI
0740  LDA OFFSET
0750  STA DOSINI+1
0760  JMP (OUT)
0770 LENGTH=*
```

BASIC Relocator#

```
10 REM RELOCATING PROGRAM FOR XEP80
20 REM TO USE THIS PROGRAM THE FOLLOWING CRITERIA MUST BE MET:
30 REM .  THE HANDLER SOURCE CODE MUST BE ASSEMBLED TWICE, THE FIRST
40 REM .  ASSEMBLY AT ANY CHOSEN START ADDRESS AND THE SECOND AT
50 REM .  THAT ADDRESS+$100. THE FIRST OBJECT CODE WILL BE CALLED
60 REM .  H6000.OBJ AND THE SECOND OBJECT CALLED H6100.OBJ. THE
70 REM .  RELOCATER CODE MUST BE ASSEMBLED AS FILE MOVE.OBJ. THIS
80 REM .  PROGRAM WILL READ IN MOVE.OBJ THEN COMPARE BYTES BETWEEN
90 REM .  H6000.OBJ AND H6100.OBJ AND CREATE A LINKED RELOCATABLE
100 REM . STRING. THIS STRING IS APPENDED TO THE MOVE.OBJ STRING
105 REM . THEN WRITTEN TO DISK AS AUTORUN.SYS
110 CLR :DIM A$(2048)
120 DIM REL$(14),OB1$(14),OB2$(14),OUT$(14)
130 REL$="D:MOVE.OBJ"
131 OB1$="D:H6000.OBJ"
132 OB2$="D:H6100.OBJ"
133 OUT$="D:AUTORUN.SYS"
200 PULL=3000:REND=3500:WRITE=4000
205 ADD=5000
210 TREND=32768:IN=1
300 OPEN #1,4,0,REL$:FILE=1
310 TRAP REND:TEND=0
320 GOSUB PULL
322 IF TEND=1 THEN GOTO 340
324 A$(IN,IN)=CHR$(DAT)
325 GOSUB ADD
330 GOTO 320
340 CLOSE #1
350 TRAP TREND
400 OPEN #2,4,0,OB1$
410 OPEN #3,4,0,OB2$
415 TRAP REND:TEND=0
417 FOR X=0 TO 5:FILE=2:GOSUB PULL:FILE=3:GOSUB PULL:NEXT X
418 FOR X=0 TO 1:FILE=2:GOSUB PULL:FILE=3:GOSUB PULL
419   A$(IN,IN)=CHR$(DAT):GOSUB ADD:NEXT X
```

```
420 COUNT=0
430 SPOT=IN
440 GOSUB ADD
445 IF COUNT>127 THEN GOTO 487
450 FILE=2:GOSUB PULL:IF TEND=1 THEN GOTO 550
455 T1=DAT
460 FILE=3:GOSUB PULL:T2=DAT
470 IF T1<>T2 THEN GOTO 500
480 A$(IN,IN)=CHR$(T1):COUNT=COUNT+1
485 GOTO 440
487 A$(SPOT,SPOT)=CHR$(128)
490 GOTO 420
500 A$(SPOT,SPOT)=CHR$(COUNT)
510 TR=T1-96:REM SHOULD READ FROM FILE
520 A$(IN,IN)=CHR$(TR)
530 GOSUB ADD
540 GOTO 420
550 A$(SPOT,SPOT)=CHR$(COUNT)
551 A$(IN,IN)=CHR$(0)
552 GOSUB ADD
553 A$(IN,IN)=CHR$(255)
554 CLOSE #2:CLOSE #3
560 START=ASC(A$(3,3))+ASC(A$(4,4))*256
570 FEND=START+IN-7
580 HI=INT(FEND/256):LO=FEND-HI*256
590 A$(5,5)=CHR$(LO):A$(6,6)=CHR$(HI)
591 GOSUB ADD:A$(IN,IN+6)=""
600 REM STOP
700 OPEN #4,8,0,OUT$
710 ? #4;A$;
720 CLOSE #4
800 STOP
900 REM THIS SECTION CAN BE ENABLED TO OUTPUT A HEX VERSION OF THE FILE
1000 DIM T$(16):T$="0123456789ABCDEF"
1002 FOR X=0 TO LEN(A$)-1
1003   M=INT(X/256):MH=X-M*256:H=INT(MH/16):L=MH-H*16:M=M+1:H=H+1:L=L+1
1004   ? T$(M,M);T$(H,H);T$(L,L);"  ";
1010   R=ASC(A$(X+1,X+1))
1020   H=INT(R/16):L=R-H*16:H=H+1:L=L+1
1030   ? T$(H,H);T$(L,L)
1040 NEXT X
1050 STOP
3000 GET #FILE,DAT
3010 RETURN
3500 TEND=1:RETURN
```

Diagrams#

Atari XEP80 diagram 1 ; thank you so much Jerzy Sobola for your help! Greatly appreciated!

Atari XEP80 diagram 2

References#

- The XEP 80 - part one from Mathy's collection of special stuff ; thank you so much Mathy and Floppydoc! Your help is greatly appreciated. :-)
- The XEP 80 - part two from Mathy's collection of special stuff ; thank you so much Mathy and Floppydoc! Your help is greatly appreciated. :-)

XEP 80 Demos in Basic#

Attributes Example#

```
100 REM ATTRIBUTES EXAMPLE
110 REM TO SET ATTRIBUTES USE FOLLOWING DEFINITIONS TO FORM COMMAND
120 REM .   THE XEP80 CONTAINS TWO ATTRIBUTE REGISTERS
130 REM .  REGISTER A IS USED WHEN A CHARACTER ASCII VALUE IS BELOW 128
140 REM .  REGISTER B IS USED WHEN A CHARACTER ASCII VALUE IS ABOVE 127
150 REM .  EACH REGISTER REMAINS CONSTANT ACROSS THE SCREEN
160 REM .  THAT IS, NO MATTER WHERE A CHARACTER IS PLACED ON THE SCREEN
170 REM .  ITS ATTRIBUTES WILL REMAIN THE SAME
180 REM .   REGISTER BIT DEFINITIONS:
190 REM .     BIT     EFFECT IF A 0
200 REM .      0      REVERSE VIDEO
```

```
210 REM .      1      NO EFFECT
220 REM .      2      BLINKING
230 REM .      3      NO EFFECT
240 REM .      4      DOUBLE WIDE
250 REM .      5      UNDERLINE
260 REM .      6      BLANK
270 REM .      7      SPECIAL TEXT GRAPHICS
280 REM .   TO ISSUE A COMMAND DO THE FOLLOWING:
290 REM .   1)  PRINT CHR$(27);
300 REM .   2)  PRINT CHR$(VALUE DERIVED FROM ABOVE BIT DEFINITIONS);
310 REM .   3)  XIO 20,#1,12,A,"E:"
320 REM .       WHERE A=244 FOR REGISTER A AND A=245 FOR REGISTER B
330 REM .   THEN PRINT CHARACTERS ON THE SCREEN, EITHER WITH BIT 7 OFF
340 REM .   TO USE REGISTER A, OR WITH BIT 7 ON TO USE REGISTER B
400 ? "}";:POSITION 28,11:? "E X A M P L E   T E X T ":POKE 752,1
410 GOSUB 2000
500 REM SOME ATTRIBUTE COMBINATIONS
510 ? CHR$(27);CHR$(255-1);:XIO 20,#1,12,245,"E:"
520 ? CHR$(27);CHR$(255-1);:XIO 20,#1,12,245,"E:"
530 GOSUB 2000:REM REVERSE VIDEO
550 ? CHR$(27);CHR$(255-4);:XIO 20,#1,12,244,"E:"
560 ? CHR$(27);CHR$(255-1);:XIO 20,#1,12,245,"E:"
580 GOSUB 2000:REM BLINKING
600 ? CHR$(27);CHR$(255-16);:XIO 20,#1,12,244,"E:"
610 ? CHR$(27);CHR$(255-1);:XIO 20,#1,12,245,"E:"
630 GOSUB 2000:REM SOME DOUDLE WIDE
700 ? CHR$(27);CHR$(255);:XIO 20,#1,12,244,"E:"
710 ? CHR$(27);CHR$(255-16-1);:XIO 20,#1,12,245,"E:"
730 GOSUB 2000:REM OTHERS DOUBLE WIDE
750 ? CHR$(27);CHR$(255-16);:XIO 20,#1,12,244,"E:"
760 ? CHR$(27);CHR$(255-16-1);:XIO 20,#1,12,245,"E:"
780 GOSUB 2000:REM ALL DOUBLE WIDE
800 ? CHR$(27);CHR$(255-128);:XIO 20,#1,12,244,"E:"
810 ? CHR$(27);CHR$(255-128-1);:XIO 20,#1,12,245,"E:"
830 GOSUB 2000:REM ALL GRAPHICS
850 ? CHR$(27);CHR$(255);:XIO 20,#1,12,244,"E:"
860 ? CHR$(27);CHR$(255);:XIO 20,#1,12,245,"E:"
880 GOSUB 2000:REM BACK TO NORMAL
990 POKE 752,0:END
2000 ? "~~";:FOR X=0 TO 999:NEXT X:RETURN
```

Scrolling Window Example#

```
100 REM SCROLLING WINDOW EXAMPLE
105 REM SCROLL LEFT AND RIGHT WITH JOYSTICK, WHEN FINISHED PRESS TRIGGER
107 POKE 53774,64:REM DISABLE BREAK KEY
110 POKE 83,255:POKE 752,1:REM SET RIGHT MARGIN, CURSOR OFF
112 FOR COL=0 TO 3*57 STEP 57:GOSUB 1050:REM CLEAR ALL WINDOWS
114   ? CHR$(125);:NEXT COL:COL=0:GOSUB 1050:REM SET FIRST WINDOW
120 ? CHR$(125);:POSITION 0,20:? "USE JOYSTICK TO SCROLL",,,
125 ? "PRESS TRIGGER BUTTON TO EXIT"
130 FOR X=0 TO 15:POSITION X*16,1:? "COLUMN ";X;
140   GOSUB 1000:FOR Y=3 TO 16:REM ALLOW SCROLLING
150     POSITION X*16+2,Y:? INT(RND(0)*100);:REM MAKE UP A VALUE
160     GOSUB 1000:REM ALLOW SCROLLING WHILE PRINTING
170   NEXT Y:NEXT X
180 GOSUB 1000:GOTO 190-STRIG(1)*10:REM ALLOW SCROLLING, CHECK TRIGGER
190 POSITION 0,20:? CHR$(253);:XIO 20,#1,12,220,"E:":POKE 83,79:POKE 752,0
195 POKE 53774,192:REM ENABLE BREAK KEY
```

```
200 END :REM WINDOW TO LEFT EDGE, RESET RIGHT MARGIN, CURSOR ON, EXIT
1000 REM JOYSTICK ROUTINE
1010 A=STICK(1):COL=COL+SGN((A=7)-(A=11)):REM GET DIRECTION AND MOVE
1030 COL=COL*(COL>-1 AND COL<177)+(COL=177)*176:REM LIMITS OF SCROLLING
1050 POSITION COL,20:? CHR$(253);:XIO 20,#1,12,220,"E:"
1060 REM LINE 1050 SETS THE NEW CURSOR AND ISSUES THE SCROLL COMMAND
1090 RETURN
```

Graphics Example (Circle)#

```
0 GOTO 100:REM KEEP TIME DEPENDENT CODE NEAR BEGINNING
10 FOR M=0 TO 12:V=0:S=128:FOR N=0 TO 7:X=P+N:REM LOOPS FOR CIRCLE LIMITS
20     V=V+S*(X*X+Y*Y*2.3<2500):S=S/2:NEXT N:P=P+8:REM CIRCLE CALCULATION
30   IF V THEN G$(M+1,M+1)=CHR$(V):REM SET BITS FOR OUTPUT STRING
40 NEXT M:? G$;:GOTO 170:REM PRINT OUTPUT STRING
100 REM GRAPHICS DISPLAY EXAMPLE TO DRAW CIRCLE
110 POKE 752,1:? CHR$(125);:DIM G$(40),D$(40):REM CURSOR OFF
111 ? :? "  PRESS ANY KEY TO BEGIN GRAPHICS PLOTTING (TOTAL TIME 5 MIN)"
112 ? :? "  WHEN PLOT IS FINISHED PRESS ANY KEY TO RETURN TO TEXT SCREEN"
113 IF PEEK(764)=255 THEN 113
114 POKE 764,255:REM RESET CHARACTER INPUT BYTE
115 D$=CHR$(0):D$(40)=CHR$(0):D$(2)=D$:GRAPHICS 8+16:REM INIT STRING
117 POKE 53774,64:REM DISABLE BREAK KEY
120 FOR Y=-32 TO 32:REM VERTICAL AXIS LOOP
130   G$=D$:P=-50:GOTO 10:REM GO TO TIME CRITICAL CODE
170 NEXT Y
180 IF PEEK(764)=255 THEN 180:REM HOLD SCREEN
185 POKE 53774,192:REM ENABLE BREAK KEY
190 POKE 764,255:OPEN #1,12,0,"E:":POKE 752,0:END :REM RESTORE TEXT SCREEN
```

Printer Configuration Program#

```
100 REM PRINTER CONFIGURATION PROGRAM
110 DIM A$(2048),IN$(16),OUT$(16):REM ALLOCATE STRINGS
120 IN$="D:AUTORUN.SYS":REM READ THIS FILE
130 OUT$="D:PRINT.SYS":REM CREATE THIS FILE
140 TRAP 32000:REM IF NO AUTORUN.SYS
150 OPEN #1,4,0,IN$
160 TRAP 180:REM TO CATCH EOF
170 A=A+1:GET #1,B:A$(A,A)=CHR$(B):GOTO 170:REM BUILD INPUT STRING
180 CLOSE #1
190 I=136:? "AUX1 VALUE";:INPUT A:REM PRINTER AUX1 BYTE
200 A$(I,I)=CHR$(A)
210 I=137:? "AUX2 VALUE";:INPUT A:REM PRINTER AUX2 BYTE
220 A$(I,I)=CHR$(A)
230 OPEN #2,8,0,OUT$
240 ? #1;A$;:CLOSE #2:REM WRITE MODIFIED STRING TO FILE
250 END
32000 ? "FILE NOT FOUND"
```