TECHNICAL INFO

Memory Map

Start Address	End Address	Usage
0x0000	0x1fff	ROM area
0x2000	0x3fff	Shadow of ROM area
0x4000	0xffff	48k of RAM; at startup, the ZX81 at most 16k of RAM to BASIC; the remaining 32k is kept free for data storage; more can be allocated to BASIC by moving(system variable at 4004h) upwards.
0x4000	0x407c	system variables
0x407d	(0x400c)-1	BASIC program area
(0x400c)	(0x4010)-1	display file (text mode!)
(0x4010)	(0x4014)-1	BASIC variables area
(0x4014)	(0x401c)-1	work space
(0x401c)	(0x4004)-1	free memory available for BASIC; this area cleared by NEW; the machine code stack grows down from the top address of this area
(0x4004)	0xffff	free memory that cannot be used by BASIC and that is also not influenced by NEW, useful for resident utilities or RAMdisk; provided (0x4004) is above 0x7fff, this area is also not altered by RST 0.

Note : (addr) = PEEK (addr) + 256 * PEEK (addr+1)

* Register I points to the ZX81 character set. Default value is I=$1E which means that the character set starts at address $1E00 (last half K of ROM). If you want to change the characterset, first POKE the new set somewhere in memory (at an address which is a multiple of 512 bytes) and then alter register I accordingly (so: only even values for I are allowed). This will instantaneously change the display. The reversed sequence of this procedure will not work the way you would expect! I advise not to change the character set present in ROM.

* According to docs, IX is used to hold the NMI. (IX is only set in the ROM via a pop(ix) at 0x292!!)

* IM 1 seems to be the set interrupt mode

INPUT PORTS

Address	Usage
0xfe	Keyboard (high byte defines which half-row to read). Also turns off HSYNC generation if NMI is disabled.

OUTPUT PORTS

Address	Usage
0xfd	Turns off NMI generator
0xfe	Turns on NMI generator
0xff	Turns on HSYNC generation
0xfb	Printer

TS1000 .IMG SNAPSHOT FORMAT

The 16410-byte snapshot is broken down as follows:

- 16384 bytes representing the RAM contents from 0x4000 to 0x7fff
- 26 bytes representing the registers in the following (LSB) order:
  AF, BC, DE, HL, IX, IY, SP, PC, AF', BC', DE', HL', I, R.

DISPLAY

ZX81 VIDEO DISPLAY SYSTEM
A tutorial by wilf rigter
last revision 7 Sept 1996

INDEX

1.INTRODUCTION
****

When circumstances combine innovative technical ideas with an economical design and market opportunity some interesting things begin to happen. In 1980, Sinclair was not yet a household word and was perhaps better known for his digital watch and calculator than his ZX80 personal computer.
But Sinclair decided the time had come for an affordable and easy to use mass produced version of the ZX80 with floating point math and a non-flicker display.

The ZX81 was born and as they say "the rest is history".

A key to the economical design of the ZX81 was the video system. Not only was it cheap to manufacture, but the ZX81 video circuit turned out to be versatile with capabilities well beyond the designer's original goals.

2. ZX81 DISPLAY BASICS
****

The standard ZX81 video screen displays 24 rows of 32 characters. Every character has height of 8 scan lines and a width of 8 pixels. The characters to be displayed are located in a block of memory called DFILE. The set of 128 displayable characters includes 64 normal (white on black) uppercase only letters, numbers, symbols and graphics characters and their inverse (black on white). The ZX81 character codes CHR$ 0-63, CHR$118 and CHR$ 128-191, are non-standard (not ASCII). A set of token codes is also used for keywords, functions and commands but these are always expanded to the displayable characters before printing to DFILE. The DFILE is formatted starting with the Sinclair equivalent of a Carriage Return (CHR$ 118) followed by up to 32 CHR$ codes, this repeated 24 times and ending with a CHR$ 118. CHR$ 118 is the opcode for the Z80 HALT instruction for reasons which will be explained later. All other character codes are illegal and if loaded into DFILE will generallly cause a system crash. The collapsed DFILE is used in the 1K and 2K basic ZX81 to minimize screen memory requirements. When empty a collapsed DFILE consists of just 25 CHR$ 118 codes. Each line is expanded when characters are printed to that line. When equipped with 4K or more of memory, DFILE is initialized to the fully expanded format with 24 lines of 32 CHR$ 00 (space) characters and 25 CHR$ 118 line termination characters.
The character codes are not displayed directly but rather are used as address pointers to a ROM video pattern table. The ROM pattern bytes are addressed by a combination of the character code in DFILE and the ZX81 hardware and is loaded into the video shiftregister. Bit 7 of the character code is used by the video hardware to invert the pixels as they are shifted out of the shift register. The display on the screen is generated by the serial bit stream

of pixels a video shift register which turns the TV CRT electron beam on and off as it scans the phosphor coating on the inside face of the picture tube. A fully expanded DFILE with 24 lines of 32 characters per row and 8 pattern bytes per character displays 6144 pattern bytes or 49152 pixels per screen.

3. SLOW MODE VIDEO

In the SLOW mode, the CPU is multitasking between video and program execution. About 80% of the CPU time allocated to video and keyboard service routines and only about 20 % of CPU time is available to execute the application program. In fact, the CPU time is divided in four distinct task blocks per TV frame as shown in TABLE 1. Tasks are switched using a Non Maskable Interupt (NMI) generator to call a NMI service routine which controls task switching from the asynchronous application program to the realtime video routines.

1.	VSYNC, frame count and keyboard	- NMI off
2.	Blank lines/application code	- NMI on
3.	VIDEO DISPLAY routine	- NMI off
4.	Blank lines/application code	- NMI on

TABLE 1 SLOW MODE CPU TASK TABLE

Each task can be described in more detail as follows:

1. During the vertical sync interval, when no video is actually displayed, the CPU executes a fixed length VSYNC routine which increments a FRAME counter, reads 8 rows of keyboard data together with the 50/60Hz mode bit. Any I/O read operation with A0 low (ie FE) addresses the ULA keyboard port. It also causes the ULA to start the vertical sync pulse by clamping the video output to the 0V sync level and simultaneously applies a reset to the ULA 3 bit line counter (LCNTR). After the all the keyboard data is processed (400us later), the CPU executes an OUT FF,A (any OUT will do) which restores the ULA video output to the normal "white level with horizontal sync pulses" and releases the LCNTR reset. At the end of the VSYNC routine, the number of blank lines to the start of the live display are determined from the system variable MARGIN (50/60HZ). Then the NMI generator is turned on and the CPU registers are switched back to the application task.

2. While the CPU executes the application code, the CPU is interrupted every 64 us by the NMI generator at the same time the ULA generates a horizontal sync pulse. The NMI routine increments a blank line counter in A' and returns if there is more time left for application code excecution. When the blank line counter is incremented to zero the NMI routine turns off the NMI generator and switches to the VIDEO DISPLAY routine through a pointer in the IX register.

3. The video display routine sets up the display file pointer, the row and line counters, enables INT and JP(HL) to the start of DFILE + 32K. Each character in the DFILE is interpreted as a NOP instruction except the N/L character which terminates the line. At the end of each line, the INT service routine updates the row and line counters and returns to execute the remaining lines. After 192 lines, the video display routine ends by turning on the NMI generator and CPU switches back to execute application code.

4. As before, during the top blank lines, the NMI routine counts the number of blank lines remaining. At the end of the bottom blank lines, the sequence repeats when the NMI service routine switches back to the VSYNC routine.

4. FAST MODE VIDEO

In the ZX80 compatable FAST mode, the CPU executes either the video routine or any other program but not both which causes the familiar flicker of the display when switching between these tasks. When the application program is running, it is executed using 100% of the available CPU time. Only if the application program is STOPped (in the command mode) or waiting for a keyboard INPUT, or in PAUSE mode is the video is displayed. The video hardware is activated in the same way as the SLOW mode but NMI is always off. In addition, the blank lines at the top and bottom of the screen are also generated in software making the ZX81 ROM fully compatable with the ZX80 hardware.

5. ZX81 VIDEO HARDWARE
****

The ZX81 video hardware consists of the Z80 CPU, ROM, RAM and the
larger part of the ZX81 Sinclair Logic Chip (usually called the ULA)
as shown in FIG 1 with all relevant connections including the
isolation resistors R. For simplicity only the 2K RAM is shown.
The ULA contains a 6.5 MHz crystal oscillator and a frequency divider which
generates horizontal sync pulses at the video output and NMI pulses on
the NMI output. The HSYNC and the NMI outputs can be controlled with the
following I/O operations.

```
1. OUT FD,A - turns off the NMI generator
2. OUT FE,A - turns on  the NMI generator
3. IN A,FE  - turns off the HSYNC generator (only if NMI is off)
4. OUT FF,A - turns on  the HSYNC generator
```

The ULA video output switches between 3 voltage levels. It is normally at the
+5V white level for blank lines. Characters patterns are displayed as black
pixels when the level is +2.5V . The narrow horizontal sync pulses and
wide vertical sync pulses are 0V level as shown in FIG 1 waveform. These
logic levels are reduced with a resistor divider to 1V, 0.5V and 0V (UK/US)
at the input of the TV RF modulator.

```
 white _  _____    _____    _________//__________//____             ____//_____
 black _   |   |__|     |__|   |             |         |<--400us-->|          |
 sync  _   |<-------64us------>|<---64us---->|         |_____//____|          |
               display line         blank        blank   vert sync     blank
```

FIG 1 - VIDEO LEVELS

The HSYNC pulses are 5 usec wide with 64 usec between HSYNC pulses.
The VSYNC is 400 usec wide with 16.6 msec or 20 msec between VSYNC pulses.
VSYNC is used to synchronize the TV vertical oscillator and start the raster
scan at the top of the screen. This occurs when IN A,FE (used for scanning the
keyboard) clamps the video output to the SYNC level. 400us later OUT FF,A
releases SYNC to enable the 64 us HSYNC pulses. The HSYNC pulses continue to
be generated independ of the CPU until the next VSYNC.
The CPU executes the application code during the blank lines at the top and
bottom of the screen while the NMI generator interrupts the CPU every 64 us
and increments a blank line counter to determine if it is time for the VIDEO
DISPLAY of VSYNC routines.

6. ZX81 CHARACTER VIDEO HARDWARE
****

The Sinclair ZX81 character display generator consists of the Z80, ROM, RAM
and the larger part of the ZX81 Sinclair Logic Chip (usually called the ULA)
as shown in FIG 2 with all relevant connections including the isolation
resistors R. For simplicity only the 2K RAM is shown.

```
              ULA               ROM        Z80        2K RAM
        ________________       _____       _____       _____
VIDEO<-| VSHFTREG <-DATA|-----|DATA |-----|DATA |--R--|DATA |
       |  LINECTR ->A0-2|-----|A0-2 |--R--|A0-2 |-----|A0-2 |
       | CHRLATCH ->A3-8|-----|A3-8 |--R--|A3-8 |-----|A3-8 |
       |                |     |A9-12|-----|A9-13|-----|A9-11|
       |           ROMCS|-----|CE   |     |  INT|-----|A6   |
       |                |     |_____|     |     |     |     |
       |           RAMCS|-----------------|-----|-----|CE/OE|
       |             A14|-----------------|A14  |     |_____|
       |             A15|-----------------|A15  |
       |              WR|-----------------|WR   |
       |              RD|-----------------|RD   |
       |              M1|-----------------|M1   |
       |            MREQ|-----------------|MREQ |
       |            IORQ|-----------------|IORQ |
       |             NMI|-----------------|NMI  |
       |            HALT|-----------------|HALT |
       |________________|                 |_____|
```

FIG 2 ZX81 CHARACTER VIDEO DISPLAY CIRCUIT

7. PSEUDO HIRES VIDEO HARDWARE

****

The pseudo hires graphics video display generator consists of the Z80 CPU, ROM, RAM and a large part of the ZX81 Sinclair Logic Chip (usually called the ULA) as shown in FIG 2 with all relevant connections including the isolation resistors R. For simplicity only the 2K RAM is shown.

```
              ULA               ROM        Z80        2K RAM
        ________________      _____       _____       _____
VIDEO<-| VSHFTREG <-DATA|-----|DATA |-----|DATA |--R--|DATA |
       | *LINECTR ->A0-2|-----|A0-2 |--R--|A0-2 |-----|A0-2 |
       | CHRLATCH ->A3-8|-----|A3-8 |--R--|A3-8 |-----|A3-8 |
       |                |     |A9-12|-----|A9-13|-----|A9-11|
       |           ROMCS|-----|CE   |     | *INT|-----|A6   |
       |                |     |_____|     |     |     |     |
       |           RAMCS|-----------------|-----|-----|CE/OE|
       |             A14|-----------------|A14  |     |_____|
       |             A15|-----------------|A15  |
       |              WR|-----------------|WR   |
       |              RD|-----------------|RD   |
       |              M1|-----------------|M1   |
       |            MREQ|-----------------|MREQ |
       |            IORQ|-----------------|IORQ |
       |             NMI|-----------------|NMI  |
       |            HALT|-----------------|HALT |
       |________________|                 |_____|
```

FIG 3 PSEUDO HIRES GRAPHICS DISPLAY CIRCUIT

The only difference between pseudo hiress and Sinclair character hardware is the *ULA LCNTR and the use of the *INT input.
Most pseudo hires core routines do not use INT and the ULA LCNTR is reset to zero every horizontal line. The exception is XTRICATOR which uses INT and makes dual use of the I register in INT mode 2 as a part of the RST vector address when interupted at the end of each horizontal line and at refresh time as a ROM pattern table pointer.

8. TRUE HIRES VIDEO HARDWARE

****

The portion of the ZX81 hardware required for true hires graphics display consists of the Z80 CPU, the RAM, the video shift register and the sync circuit of the ULA as shown in FIG 4 with all relevant connections. Again the 2K SRAM is shown for simplicity but applies to larger SRAM designs as well. If a 16K RAMPACK is used, this must be slightly modified, as will be shown later, to enable the data outputs during RFSH time as required for this hires display method.

```
              ULA               ROM         Z80       2K RAM
        ________________      _____       _____       _____
VIDEO<-| VSHFTREG <-DATA|-----|DATA |-----|DATA |--R--|DATA |
       |                |     |A0-12|-----|A0-15|-----|A0-10|
       |           ROMCS|-----|CE   |     |     |     |     |
       |                |     |_____|     |     |     |     |
       |           RAMCE|-----------------|-----|-----|CE/OE|
       |             A14|-----------------|A14  |     |_____|
       |             A15|-----------------|A15  |
       |              WR|-----------------|WR   |
       |              RD|-----------------|RD   |
       |              M1|-----------------|M1   |
       |            MREQ|-----------------|MREQ |
       |            IORQ|-----------------|IORQ |
       |             NMI|-----------------|NMI  |
       |            HALT|-----------------|HALT |
       |________________|                 |_____|
```

FIG 5 TRUE HIRES GRAPHICS VIDEO DISPLAY CIRCUIT

With the exception of WRX1K which creates a miniature hires screen on a 1K ZX81 all hires programs need a 6K hires graphics file (HFILE).
Suitable RAM for true hires graphics can be implimented by modifying a standard 16K RAMPACK with a couple of diodes and a resistor.

The RAMPACK is modified to enable the data output at RFSH time by cutting the RD and RFSH lines at the edge connector and installing 2 only 1N34A Germanium diodes and a 4.7K pullup resistor. Modify at your own risk!

```
                            +5V
                             |
                           [4.7K]                                  (Figure 5)
                     1N34A   |
connector   RD _______|/|____|_____________ RD OF RAMPACK
                      |\|    |
                             |
connector RFSH _______|/|____|     +5V_____ RFSH of RAMPACK
                      |\|
                     1N34A
```

9. ZX81 CHARACTER DISPLAY TIMING
****

All the Sinclair ZX81 character display hardware shown in FIG 2 is required to generate a standard screen of 24 lines of 32 characters.
The character display starts when the last blank line at the top of the screen has occurred and and the video routine jumps to the DFILE echo above 32K.
The hardware in the ZX81 ULA takes control when any opcode is executed above 32K (A15 high and M1 low) with data bit 6 equal to zero. The video data is loaded in these simplified steps:

1.The ULA loads the character code into a address register in the ULA
2.The ULA forces the data lines low.
3. The CPU interprets the byte as a NOP.
4. The ULA generates part of the ROM pattern table address and the Z80 CPU generates the pattern table pointer with the I register.
5. The pattern byte is loaded into the ULA shift register.

One could say that the Dfile is literally executed with NOPs substituted for each character code. Each NOP executes in 4 CPU clock cycles at 3.25 MHz or 8 pixels at 6.5MHz from the ULA video shift register.

```
           <--------CHARACTER 1-----------><--------CHARACTER 2----------->
   T STATE <--T1--><--T2--><--T3--><--T4--><--T1--><--T2--><--T3--><--T4-->
     (ref)  ___    1___2   3___     ___5    ___     ___     ___     ___
 CPU CLOCK |   |___|   |___|   |___|   |___|   |___|   |___|   |___|   |___|
           _ _______________ _______________ _______________ _______________
    A0-A15 _X_____PC________X___I+CHR+ULA___X______PC_______X___I+CHR+ULA___X
                ________  NOP  __________       ________  NOP  __________
      DATA >---|__CHR___|_____|_ROM DATA_|-----|__CHR___|_____|_ROM DATA_|--
```

FIG 6 ZX81 CHARACTER DISPLAY TIMING

The detailed sequence of operations for each character byte is shown in FIG 6 and described as follows:

1. Each character code (CHR$) byte in DFILE is addressed by the CPU PC, on the rising edge T2 data is loaded from DFILE into the ULA : bits 0-5 into a 6 bit ULA address latch while bit 7 is loaded into 1 bit ULA video invert latch
2. On the falling edge of T2, the ULA forces all CPU data lines to zero.
3. On the rising edge of T3 the low data lines are interpreted by the CPU as a NOP instruction.
4. During T3/4, the CPU executes the Refresh cycle and ROM address lines are generated with I register on A9-A15, the ULA 6 bit character code register on A3-A8, and the ULA modulo 8 line counter on line A0-A2.
5. On the falling edge of T4, pattern data from the ROM is loaded into ULA video shift register and 8 video pixels are shifted out at 6.5MHz
6. If character code bit 7 latch in ULA equals 1, video pixels are inverted.
7. The CPU increments the program counter and fetches the next character code.
8. This repeats until a HALT (Sinclair) is fetched.
9. HALT opcode bit 6 = 1 and is therefore executed (no NOP)
10.The ULA generates a HSYNC pulse independend of the CPU timing and the ULA LCNTR is incremented
11.The halted CPU continues to execute NOPs, incrementing register R and samples the INT input on the rising edge of each T4.
12.When A6, which is hardwired to INT, goes low during refresh time, (bit 6 of the R reg = 0), the Z80 executes the INT routine (below 32K)
13.CPU returns from INT and resumes "excution" of DFILE CHR$ codes.
14.The process repeats 192 times and then INT routine returns to the main

video routine, turns on the NMI generator and switches back to the application code.

10. PSEUDO HIRES DISPLAY TIMING
****

All the ZX81 character display hardware shown in FIG 2 with some exceptions is required to generate a standard screen of 192 lines of 32 pseudo hires patterns. The display starts when the last blank line at the top of the screen has occurred and and the video routine jumps to the 6K DFILE echo above 32K. The hardware in the ZX81 ULA takes control when any opcode is executed above 32K (A15 high and M1 low) with data bit 6 equal to zero.
The video data is loaded in five steps:

1. The ULA loads the character code into an address register
2. The ULA forces the data lines low.
3. The CPU interprets the byte as a NOP.
4. The ULA generates part of the ROM pattern table address. The CPU generates the pattern table MSB address with the I register.
5. The quasi hires pattern byte is loaded into the ULA shift register.

Each NOP executes in 4 CPU clock cycles at 3.25 MHz or 8 pixels at 6.5MHz from the ULA video shift register.

```
           <--------CHARACTER 1-----------><--------CHARACTER 2----------->
   T STATE <--T1--><--T2--><--T3--><--T4--><--T1--><--T2--><--T3--><--T4-->
     (ref)  ___    1___2   3___     ___5    ___     ___     ___     ___
 CPU CLOCK |   |___|   |___|   |___|   |___|   |___|   |___|   |___|   |___|

           _ _______________ _______________ _______________ _______________
    A0-A15 _X_____PC________X___I+CHR+ULA___X______PC_______X___I+CHR+ULA___X
                ________  NOP  __________       ________  NOP  __________
      DATA >---|__CHR___|_____|_ROM DATA_|-----|__CHR___|_____|_ROM DATA_|--
```

FIG 7 SINCLAIR CHARACTER DISPLAY TIMING

The detailed sequence of operations for each character byte is as follows:

1. Each character code (CHR$) byte in DFILE is addressed by the CPU PC, on the rising edge T2 data is loaded from DFILE into the ULA : bits 0-5 into a 6 bit ULA address latch while bit 7 is loaded into 1 bit ULA video invert latch
2. On the falling edge of T2, the ULA forces all CPU data lines to zero.
3. On the rising edge of T3 the low data lines are interpreted by the CPU as a NOP instruction.
4. During T3/4, the CPU executes the Refresh cycle and ROM address lines are generated with I register on A9-A15, the ULA 6 bit character code register on A3-A8, and zero on line A0-A2.
5. On the falling edge of T4, pattern data from the ROM is loaded into ULA video shift register and 8 video pixels are shifted out at 6.5MHz
6. If character code bit 7 latch in ULA equals 1, video pixels are inverted.
7. The CPU increments the program counter and fetches the next character code.
8. This repeats until a RET fetched which returns to the hires routine.
9. RET opcode bit 6 = 1 and is therefore executed (no NOP)
10. The ULA generates a HSYNC pulse independend of the CPU timing and the ULA LCNTR is incremented but the video software resets the LCNTR to zero.
11. CPU returns from the hires routine and resumes "excution" of DFILE CHR$ codes.
12. The process repeats 192 times and then hires routine ends by turning on the NMI generator and by switching back to the application code.

11. TRUE HIRES DISPLAY TIMING
****

Although I will use the WRX hires core routine as an example, the true hires software developed independently by others is very similar. The true hires display starts when the last blank line at the top of the screen has occurred, and NMI jumps via IX to the HR video routine. HR sets up the I and R register pointers to the hires display file (HFILE), then the HR routine jumps to the LBUF routine echo above 32K and loads register R and points to the first NOP opcode. The ULA loads the video shift register when any opcode is executed with A15 high and M1 low and data bit 6 equal to zero.

The hires data is loaded in 3 steps:

1. The CPU executes each of the 4T state NOP instructions.
2. During T3/4 (refresh), the I and R registers appear on the A0-15 lines.

3. The hires byte addressed by I and R is loaded into the ULA shift register.

LBUF consists 32 NOP opcodes, each executing in 4 CPU clock cycles.
During the second part of the opcode excution, the I and R register address the hires byte in HFILE which the ULA loads into the video shift register.
The ULA generated ROM address for the pattern table is not used since ROMCS is not enabled as register I (A8-15) is set up to point to the RAM based HFILE.

```
           <---------HIRESBYTE1-----------><---------HIRESBYTE2----------->
   T STATE <--T1--><--T2--><--T3--><--T4--><--T1--><--T2--><--T3--><--T4-->
            ___     ___     ___     ___     ___     ___     ___     ___
 CPU CLOCK |   |___|   |___|   |___|   |___|   |___|   |___|   |___|   |___|
           _ _______________ _______________ _______________ _______________
    A0-A15 _X______PC_______X______I_+_R____X______PC_______X______I_+_R____X
                ____________    _________       ____________    _________
      DATA >---|_____NOP____|--|__HIRES__|-----|_____NOP____|--|__HIRES__|--
```

FIG 8 TRUE HIRES DISPLAY TIMING

The detailed sequence of operations for each HIRES byte is as follows:

1. The first opcode of the LBUF routine, LD R,A is executed.
2. The following 32 NOPs in LBUF are sequentially executed.
3. On the rising edge of T3 of each instruction fetch, the CPU executes the NOP.
4. During T3/4, the address is generated with the R register on A0-7 and the I register on A8-A15.
5. On the falling edge of T4, a hires data byte from HFILE is loaded into ULA video shift register and 8 video pixels are shifted out at 6.5MHz
6. The CPU increments the program counter and the R register and fetches the next NOP and the next hires byte .
7. This process repeats 32 times.
8. The last opcode of LBUF, JP (IX), is executed to return to the HR routine .
9. The ULA generates a horizontal syncpulse.
10. The HLINE routine increment the I/R register pair by 32 and jumps back to the 32 NOP LBUF routine echo above 32K.
11. The process repeats 192 times.
12. The HIRES video routine may call the Sinclair character routine for the bottom line and restores the registers etc to return to the application code.

Like the other hires routines WRX intercepts the Sinclair video by loading a new video routine vector in the IX register.

12. ZX81 SLOW MODE VIDEO ROUTINES
****

As shown in TABLE 1 the CPU is multitasking between the video routines and the application program in 4 blocks of time.

1. VSYNC INTERVAL ROUTINES

```
0229 DISPLAY-1 DECREMENT FRAME COUNTER
023E DISPLAY-2 CHECK KEYBOARD (START OF VSYNC)
0277 OUT FF,A  ENDS THE VSYNC PULSE
0292 DISPLAY-3 SAVE THE VIDEO POINTER IN IX (0281), RETURN TO APPLICATION
```

2. APPLICATION PROGRAM

```
0066 NMI       COUNTS BLANK LINES, RETURN TO APPLICATION OR VIA JP (IX) TO 0281
```

3. DFILE DISPLAY ROUTINES

```
0281 VIDEO-1   SETUP DISPLAY PARAMETERS, CALL 2B5 (RETURN VIA INT)
02B5 DISPLAY-5 SETUP DISPLAY PARAMETERS, ENABLE INTERRUPT, JP (DFILE)
XXXX (DFILE)   EXECUTES HALTs AND FORCED NOPs IN DFILE
0038 INT       DECREMENTS ROW/LINES COUNTERS RETURN TO DFILE OR TO 028B
0292 DISPLAY-3 SAVE VIDEO POINTER (028F)
```

4. APPLICATION PROGRAM

```
0066 NMI       COUNTS BLANK LINES, RETURN TO APPLICATION OR VIA JP (IX) TO 028F

028F VIDEO-2 JP 229  BACK TO FRAME COUNTER in BLOCK 1
```

The ZX81 video routines follow in fully anotated listings showing more details than IAN LOGAN ZX81 DISASSEMBLY. Needless to say, I used Dr. LOGAN's book extensively during my research into the ZX81 video code.

NOTE: Only code which is relevant to video is shown here.

```
0038            ;INT SERVICE ROUTINE
     DEC C      ;decrement the scan line counter in register C
     JP NZ 0045 ;go SCAN-LINE : repeats 8 times for each DFILE character row
     POP HL     ;point to the start of next DFILE row
     DEC B      ;decrement the ROW counter in register B
     RET Z      ;return to 028B
     SET 3,C    ;load scan line counter in register C with 08 scan lines

0041            ;WAIT-INT
     LD R,A     ;load value DD into register R
     EI         ;enable INT
     JP (HL)    ;execute the NOPs in DFILE

0045            ;SCAN-LINE
     POP DE     ;discard the return address
     RET Z      ;delay (never returns)
     JR 0041    ;got WAIT-INT
                ;-----------------------------------------------------------

0066            ;NMI SERVICE ROUTINE
                ;Interupts application program every 64 usec (HSYNC)
     EX AF,AF'  ;retrieve blank line counter in AF'
     INC A      ;next blank line
     JP M 006D  ;RETURN via 006D if AF' = FF (to NMI-EXIT)
     JR Z 006F  ;JR NMI-CONT if last line
006D EX AF,AF'  ;save blank line counter
     RET        ;return to application or NMI-EXIT

006F            ;NMI-CONT
     EX AF,AF'  ;retrieve main register AF
     PUSH AF    ;now save the application program registers
     PUSH BC
     PUSH DE
     PUSH HL
     LD HL,(DFILE) ;needed only if IX=0281 and
     SET 7,H    ;if DFILE is executed
     HALT       ;1T state synchronization: this HALT is used with special
                ;hardware connected to the CPU WAIT and HALT lines and is
                ;released and synchronized on the falling of the NMI pulse

007A            ;NMI-EXIT
     OUT FD,A   ;turn off NMI generator
     JP (IX)    ;to VIDEO-1 or VIDEO-2
                ;----------------------------------------------------------

0229            ;DISPLAY-1
     LD HL,(FRAMES);get the system variable FRAMES
     DEC HL     ;decrement each frame
     .....
     LD (FRAMES),HL;save the system variable FRAMES

023E            ;DISPLAY-2
     CALL 02BB  ;read the keyboard and load MARGIN with blank lines
     .....      ;also starts the VSYNC pulse
0277 OUT FF,A   ;stops the VSYNC pulse
     LD HL,(DFILE) ;(FAST VIDEO only) - point HL to first HALT for blank lines
     SET 7,H    ;(FAST VIDEO only) - DFILE echo above 32K
027E CALL 0292

0281            ;VIDEO-1
                ;this vector is saved in register IX at 0292
     LD A,R     ;delay
     LD BC,1901 ;set up INT parameters for first HALT at (DFILE)
     LD A,F5    ;set up R register for first HALT at (DFILE)
     CALL 2B5   ;continue setup for DFILE display and return via INT
028B            ;return here from last INT
     DEC HL     ;(FAST VIDEO only) - point HL to last HALT for blank lines
     CALL 292   ;save VIDEO vector in IX, calculate blank lines, POP regs

028F JP 0229    ;VIDEO-2
                ;this vector is saved in register IX at 0292
```

```
0292                ;DISPLAY-3
     POP IX         ;IX=0281 or 028F to vestor to VIDEO-1 or VIDEO-2
     LD C,(IY+56)   ;load number of blank lines from MARGIN (1F in 60 Hz option)
     BIT 7,(IY+59) ;test FAST/SLOW bit
     JR Z,2A9       ;(FAST VIDEO)  branches to generate blank lines
     LD A,C         ;C=(MARGIN)=1F for 60 Hz
     NEG            ;
     INC A          ;
     EX AF,AF'      ;during NMI @ 0066 - AF' is incremented and tested for zero
     OUT (FE),A     ;turn on NMI generator
     POP HL         ;self explanatory
     POP DE
     POP BC
     POP AF
     RET            ;return to application program interupted every HSYNC by NMI
                    ;-----------------------------------------------------------

02B5                ;DISPLAY-5
     LD R,A         ;R increments with each opcode on A0 to A7 during RFSH
                    ;until A6 goes low which generates the INT signal.
     LD A,DD        ;Set the left margin of all other lines, load to R at 0041
     EI             ;Now that R is set up enable INT
     JP (HL)        ;"executes" the DFILE starting with HALT and waits for the
                    ;first INT to come to the rescue.
                    ;-----------------------------------------------------------
```

13. ZX81 FAST MODE VIDEO ROUTINES
****

In order to speed up application program execution time, the FAST mode uses 100% of the CPU time for executing the application program but there are times when the application program is idle. When the program is STOPped or PAUSEd or waiting for keyboard INPUT, the keyboard is checked and if no key is down the video is generated independent of NMI pulses.
In fact, the ZX81 FAST mode video routine was designed to be compatable with the ZX80 hardware so the ROM could be used as a retrofit ROM upgrade.
Since the ZX80 generates the blank lines in software, the ZX81 ROM does the same when in the FAST mode.

The loop of VIDEO routines for FAST video starts with the FRAME/KBD/VSYNC routine at 229:

```
0229 DECREMENT FRAME COUNTER
023D EXIT FAST VIDEO IF FRAMES=0 (END OF PAUSE)
023E CHECK KEYBOARD
0260 EXIT FAST VIDEO IF NEW KEY PRESSED
0292 SAVE THE VIDEO POINTER IN IX (0281)
029B JR Z 02A9 TO BLANK LINE ROUTINE
02A9 GENERATE BLANK LINES
02B3 JP (IX) TO  0281
0281 GENERATE THE DFILE DISPLAY
0292 SAVE VIDEO POINTER (028F)
029B JR Z 02A9 TO BLANK LINE ROUTINE
02A9 GENERATE BLANK LINES
02B3 JP (IX) TO 028F)
028F JP 229  BACK TO FRAME COUNTER
```

Since most of the routines were described in the SLOW MODE VIDEO chapter, only the differences are described here. Compare the way the SLOW mode enters this loop from end of blank line application program execution by saving the main registers of the program and restoring them at the end of 0292. By contrast, the FAST mode does not save any registers and branches out of the 0292 restore main register routine to literally generate the blank lines. This is done at 029B after testing the FAST flag and jumping to a less known routine called DISPLAY-4

```
02A9                ;DISPLAY-4
     LD A,FC        ;first R delay to INT
     LD B,01        ;one row
     CALL 02B5      ;display blank lines
     DEC HL         ;point back to HALT
     EX (SP),HL     ;delay 19T
     EX (SP),HL     ;delay 19T
     JP (IX)        ;IX = 0281 or 028F
```

The routine at 02A9 is called twice each frame to generate the top and bottom blank lines with HL pointing to either the first HALT at the start of DFILE or the last HALT at end of DFILE. Reg C holds the number of blank lines and reg B is set up for 1 row. After VSYNC the 31 top blank lines are generated by calling the diplay routine at 02B5 and excecuting the first HALT at the START of DFILE 31 times. After returning from the display routine HL points to the last HALT+1 and DEC HL is required point HL back to the last HALT of DFILE. After saving the return address in IX, the routine at 029A is reentered with HL pointing to the last HALT and generates the bottom 31 blank lines by excecuting the HALT at the END of DFILE.

14. TRUE HIRES VIDEO SOFTWARE
****

The true hires core routines are distinguished by the use of the I and R register pair as address pointers for the display file. The only other requirement is to execute 32 NOP instructions (or equal) per horizontal line and to update the I and R registers during HSYNC time. More blank lines can be used above and below the display for faster application execution. The listings are compatable source code for the ZXAS assembler both on the ZX81 and under XTender, the ZX81 emulator form CARLO DELHEZ. Check current version of XTender for hires compatability.
These ASCII listings can be used to prepare a formatted 2 REM .l file with the ZXAS.COM program from Jack Raats.

WRX16 - 1984 wilf rigter

This is the hires core used in programs by FRED NACHBAUR and GREG HARDER. It creates a 256x192 high resolution display in a 6144 byte array starting at (HFILE), which can be poked directly from BASIC programs.
START is used to start the hires display and STOP restores the SINCLAIR video. It has a characteristic signature with the I register value greater than 2000 hex.
PART 1 calls LBUF 192 times, displaying 256x192 pixels, calculates blank lines, saves pointer to PART 2 in IX and returns to application code.
PART 2 calls VSYNC etc, calculates blank lines, saves pointer to PART 2 in IX and returns to application code execution.

```
      ;ORIGIN = 16516 (hex 4084)

LBUF  ;Displays one line of 256 pixels
      ;------------------------------

      ;like DFILE, it is called above 32K to activate the ULA video
      ;hardware. The hires bytes may be inverted for special effects
      ;by setting bit 7 of the NOP codes . The hires data is loaded
      ;into the ULA video shift register during the refresh cycles of
      ;the 32 NOP opcodes when the I and R registers sequentially
      ;address 32 bytes of hires data in the 6144 byte HFILE

LBUF
      LD R,A     ;Now load R register
      00 00 00 00;32 bytes of 8 pixels
      00 00 00 00
      00 00 00 00
      00 00 00 00
      00 00 00 00
      00 00 00 00
      00 00 00 00
      00 00 00 00
      JP (IX)    ;Return to HR


HR    ;HIRES DISPLAY ROUTINE PART 1
      ;----------------------------
      LD B,04    ;load delay
HR0   DJNZ HR0   ;delay 56T states to synchronize with HSYNC pulses.
      LD HL,(HFILE);RAMTOP points to the first HFILE byte
      LD B,C0    ;48 horizontal lines
      LD IX,HR1  ;save the return vector in IX (for JP (IX) at end of LBUF)
      JR HR2     ;skip HR1 first time through the loop
HR1   LD DE,20   ;this value for 32 bytes or 256 pixels per line is
      ADD HL,DE  ;added to HL to point to the start of the next HLINE
      DEC B      ;repeat 48 times
      JP Z HR3   ;if this is the last line JP to HR3
HR2   LD A,H     ;the address in HL is then transferred to
```

```
      LD I,A     ;I register
      LD A,L     ;and during LBUF to the R register
      JP C084    ;jump to LBUF @ 4084 + 8000 to start the ULA
HR3   LD IX,HR4  ;save the video vector so that NMI returns to HR4
      JR HR5     ;now get blank lines and return to application code

HR4   ;HIRES DISPLAY ROUTINE PART 2
      ;----------------------------
      CALL 220   ;first PUSH registers then jump to VSYNC, etc
      LD IX,WRX16;save the video vector so that NMI returns to HR
HR5   LD A,(4028);33 or 19 blank lines in system variable MARGIN
      JP 29E     ;save blank lines, start NMI, POP registers and RETURN

      ;--------------------- end of listing ---------------------
```

The hires video is started and stopped by changing the vector address in register IX which is used by NMI to JP (IX) to the video routine. The following routines are synchronized with the display so that the changeover in video mode occurs without display breakup.

```
STOP  ;STOP hires video and return to SINCLAIR video
      ;--------------------------------------------
      LD HL,0281 ;pointer to SINCLAIR video routine
      LD A,1E    ;SINCLAIR ROM pattern table MSB base address (1E00)
      LD I,A     ;pointer to I register
      JR SYNC

START ;Start the hires video
      ;---------------------
      LD HL,HR   ;pointer to the hires video routine

SYNC  ;used by START and STOP to smoothly change video mode
      ;---------------------------------------------------
      PUSH HL
      LD HL,4034 ;FRAMES counter
      LD A,HL    ;get old FRAMES
SYNC1 CP A,(HL)  ;compare to new FRAMES
      JR Z SYNC1 ;exit after a change is detected
      POP IX     ;SINCLAIR video routine

     ;-------------- END OF LISTING ---------------
```

GUUS-FLATER by ENNO BORGESTEEDE (1984)

This hires core uses a ingenious way to intercept the video vector. Instead of changing the value of the IX register, GUUS-FLATER intercepts at the beginning of the DFILE execution by changing the first 4 bytes including the HALT to DI and JP 409F which is the start of the hires routine. At the end of the hires screen the program simply returns to ROM routine at xxxx. It has a characteristic DFILE starting with the DI and JP 409F and the HFILE starts at (4004)

```
(400C)  DI           ;these bytes are loaded into DFILE
        JP 409F      ;to vector to the hires routine

409F    LD B,08      ;delay
40A1    DJNZ 40A1    ;delay
        LD A,R       ;delay
        LD B,C0      ;192 lines
        LD DE,20     ;32 bytes per line
        LD HL,(4004) ;hires file (HFILE) starts at RAMTOP
40AD    LD A,H       ;MSB address of HFILE
        LD I,A       ;load MSB of HFILE pointer
        LD A,L       ;LSB address of HFILE
40B3    JP C0B6      ;JUMP above 32K
40B6    LD R,A       ;load LSB of HFILE pointer
40B8    "COPYRIGHT 1984 ENNO BORGESTEEDE " ; same as 32 NOPs
40D8    JP 40DB      ;JUMP below 32K
40DB    ADD HL,DE    ;next hires line
40DC    DJNZ 40AD    ;next line repeats 192 times
40DE    LD A,1E      ;restore ROM pattern table pointer
40E0    LD I,A       ;load pointer
40E2    RET          ;join the SINCLAIR video in progress
```

HRG7 - in progress

WRX16K - 1996 wilf rigter <rigter@cafe.net>

The original WRX was written in 1984 but recent renewed interest has yielded newer more efficient versions. WRX16K 1996 is the most compact version of the WRX yet and will display a true bit mapped 256x192 hires screen. It was designed to work with the modified 16K RAMPACK or 16K SRAM and you must first lower RAMTOP with POKE 16389,96 then NEW before loading.
The hires mode can be started and stopped with the same routines shown in the WRX16 listing above. The simple START is used for starting the hires mode by changing video vector address in the IX register.
Hires is stopped with the inline code segment called "BREAK" which returns synchronously to the Sinclair video mode when the space key is pressed.
The HFILE is a 6K linear array starting at (4004) but is easily relocated.
Note that HFILE must start on a 32 byte boundary (2000,2020, etc).

```
ORG               ;16516 (hex 4084)

START LD IX,HR    ;simple start of the hres mode
      RET

LBUF LD R,A      ;load HFILE address LSB
      0 0 0 0     ;32 NOPs = 256 pixels
      0 0 0 0
      0 0 0 0
      0 0 0 0
      0 0 0 0
      0 0 0 0
      0 0 0 0
      0 0 0 0
      RET NZ      ;always returns

HR
      LD B,7      ;delay
HR0   DJNZ HR0    ;delay
      DEC B       ;reset Z flag
      LD HL,(4004);HFILE starts at RAMTOP or HSCRN (note below)
      LD DE,20    ;32 bytes per line
      LD B,C0     ;192 lines per hires screen
HR1   LD A,H      ;get HFILE address MSB
      LD I,A      ;load MSB into I register
      LD A,L      ;get HFILE address LSB
      CALL C089   ;CALL LBUF + 8000
      ADD HL,DE   ;next line
      DEC B       ;dec line counter
      JP NZ HR1   ;last line
HR2
      CALL 292    ;return to application program
      CALL 220    ;extra register PUSH and VSYNC
BREAK             ;this code segment is optional
      CALL F46    ;check break key
      LD A,1E     ;restore pattern table pointer
      LD I,A
      JR NC STOP  ;skip the HR vector load if BREAK
      LD IX,HR    ;load the HR vector
STOP  JP 2A4      ;return to application program

HSCRN 2000        ;this is used with SRAM at 8K - 16K
```

Note:

HFILE can be relocated to use SRAM between 8 to 16K by changing
LD HL,(4004) to LD HL,(HSCRN).

------------------ end of listing ------------------

WRX1K -1996 wilf rigter

This little true hires program is special because it runs on a 1K/2K ZX81.
It creates a miniature 64x48 high resolution display in a 384 byte array starting at (RAMTOP), which can be poked directly from BASIC programs.
When START is called, it collapses the SINCLAIR DFILE and expands the hires file above RAMTOP in order to efficiently utilize the 1K memory.
When STOP is called, it recovers the space above RAMTOP to make more RAM

avialable for DFILE.

```
      ;ORIGIN = 16516 (hex 4084)

LBUF  ;Dummy display file
      ;------------------

      ;like DFILE, it is called above 32K to activate the ULA video
      ;hardware but only bit 7 character code data is used. The hires
      ;data is loaded into the ULA video shift register during the refresh
      ;cycles of the 8 NOP opcodes when the I and R registers
      ;sequentially address 8 bytes of hires data in the 384 byte HFILE
      ;NOTE: in this special case of short (8 byte) video lines the delay
      ;opcodes (E3 and 40) have bit 6 high to suppress the video display
      ;at the start and end of each horizontal line

      E3 E3 E3 E3;Delay 76T states
      LD R,A     ;Now load R register
LBYTE 00 00 00 00;8 bytes of 8 pixels
      00 00 00 00;
      40 40 40 40;Delay 20T states
      40         ;Delay 4T
      JP (IX)    ;Return to HR

START ;Makes room above RAMTOP and starts the hires routine
      ;----------------------------------------------------
      LD IX,HR    ;This is the start of the new video routine
      LD BC,180   ;384 bytes are required for a 64X48 display
      CALL EC5    ;is there enough room to lower ramtop?
      LD HL,(4004);get the old RAMTOP (1K/2K)
      CCF         ;calculate the new RAMTOP value by
      SBC HL,BC   ;Subtracting the HFILE length

STACK ;used by START and STOP to change RAMTOP without NEW

      OUT FD,A    ;Turn off the NMI generator during STACK move
      LD (4004),HL;Save RAMTOP value in the sytem variable
      DEC HL      ;point to the first byte below RAMTOP
      LD (HL),3E  ;and mark it with 3E
      DEC HL
      LD SP,HL    ;Load the STACK POINTER
      DEC HL
      DEC HL
      LD (4002),HL;Load the ERROR STACK POINTER
      OUT FE,A    ;Turn on NMI
*      JP 676      ;resume BASIC program execution at NEXT LINE



HR    ;PART 1 calls LBUF 48 times, displaying 64x48 pixels, calculates blank
      ;lines, saves pointer to PART 2 in IX,and returns to application code
      ;PART 2 calls VSYNC etc, checks the BREAK key, calculates blank lines,
      ;saves pointer to PART 1 in IX and returns to application code.

      LD B,04    ;load delay
HR0   DJNZ HR0   ;delay 56T states to synchronize with HSYNC pulses.
      LD HL,(4004);RAMTOP points to the first HFILE byte
      LD B,30    ;48 horizontal lines
      LD IX,HR1  ;save the return vector in IX (for JP (IX) at end of LBUF)
      JR HR2     ;skip HR1 first time through the loop
HR1   LD DE,08   ;this value for 8 bytes or 64 pixels per line is
      ADD HL,DE  ;added to HL to point to the start of the next HLINE
      DEC B      ;repeat 48 times
      JP Z HR3   ;if this is the last line JP to HR3
HR2   LD A,H     ;the address in HL is then transferred to
      LD I,A     ;I register
      LD A,L     ;and during LBUF to the R register
      JP C084    ;jump to LBUF (4084 + 8000) to start the video
HR3   LD IX,HR4  ;save the video vector so that NMI returns to PART 2
      JR HR5     ;now get blank lines and return to application code

HR4   ;PART 2

      CALL 220   ;first PUSH registers then jump to VSYNC, etc
      CALL F46   ;test the BREAK key
      JR NC STOP ;and exit HR if key is down
      LD IX,HR   ;save the video vector so that NMI returns to PART 1
```

```
HR5   LD A,(4028);33 or 19 blank lines in system variable MARGIN
      ADD A,47   ;71 more blank lines for fast application code execution
      JP 29E     ;save blank lines, start NMI, POP registers and RETURN
STOP             ;EXIT hires video to restore RAMTOP and SINCLAIR video
      LD A,1E    ;SINCLAIR ROM pattern table MSB base address (1E00)
      LD I,A     ;pointer to I register
      LD HL,(4004);load the current RAMTOP
      LD DE,180  ;HFILE length
      ADD HL,DE  ;is added to the current RAMTOP
      LD IX,0281 ;SINCLAIR video routine
      JR STACK   ;exit HR via STACK to change RAMTOP


     ;------------------END OF LISTING----------------
```

15. PSEUDO HIRES CORE ROUTINES
****

Pseudo hires software uses CHR$ code + register I to address 1 of 64 pattern bytes in the ROM. The CHR$ codes are limited 0 to 63 and their inverse (128 to 191) and the value of I is choses to point to a block of pattern bytes in ROM which has the greatest randomness and least duplication of values. This method is called "pseudo" hires because fewer than 50% of the 256 patterns required for true hires are available for display. This results in incomplete or missing pixel patterns but for many application (games etc) this is not a problem. The advantage of this method is the fact that it runs on systems the standard 16K RAMPACK and is emulated by Xtender. It has a characteristic 6336 byte DFILE consisting of 192 RET opcodes spaced at 33 byte intervals.

ROCK CRUSH by Steve McDonald

In progress

3DHIRES author unknown

In this example core routine the expanded hires "DFILE" starts at 6700 hex

```
START   LD A,04   ; select an "interesting" pattern table
        LD I,A    ; load into ROM pattern table pointer
        LD IX,42C4; load pseudo hires vector
        RET

42C4    LD HL,E6DF; HL is used as the hires "DFILE" pointer
        LD DE,0021; 32 CHR$ + RET = 33 bytes per line
        DI        ; INT not used
        LD C,FE   ; IO address to reset the ULA row counter
        LD B,16   ; delay
42CF    DJNZ 42CF ; delay
        LD B,C0   ; 192 lines of 33 bytes
42D3    IN A,(C)  ; apply reset to row counter
        OUT FF,A  ; release reset from row counter
        ADD HL,DE ; next "DFILE" line (E700,E721, etc)
        CALL 42EC ; "execute" the "DFILE" via JP (HL) at 42EC
        DEC B     ; decrement line counter
        JP NZ 42D3; last line? repeat 192 times.
        CALL 0292 ; restore main registers, return to application
        CALL 0220 ; extra register PUSH then VSYNC
        LD IX,42C4; hires routine vector
        JP 02A4   ; restore main registers, return to application
        JP (HL)   ; jump to the hires "DFILE" echo above 32K
```

XTRICATOR by Software Farm

This unusual pseudo hires core routine intercepts the video by setting INT mode 2 in which the interrupting device supplies part of the INT vector address. Since the idle data bus is FF and the I register is set to 40, the INT vector is 4000 when the A6 line interupts at the end of the horizontal line.

```
4083    LD HL,40A5 ;return vector
4086    PUSH HL    ;save vector
4087    LD HL,E500 ;hires file
```

```
408A    PUSH HL    ;save vector
408B    LD B,07    ;delay
408D    DJNZ 408D  ;delay
408F    LD A,1E    ;Sinclair ROM patterns
4091    LD I,A     ;load pattern table pointer
4093    LD DE,C201 ;D = 194 lines
4096    DEC E      ;set Z FLAG
4097    JP Z 409A  ;delay
409A    POP HL     ;HL = E500
409B    DEC D      ;decrement the line counter
409C    RET Z      ;after last line only : return to 40A5
409D    SET 0,E    ;E = 01
409F    JP 40A2    ;delay
40A2    LD A,00    ;delay
40A4    JP (HL)    ;jump to 6500 above 32K

40A5    LD A,04    ;"special" ROM pattern
40A7    LD I,A     ;load pattern table pointer
40A9    RET        ;return to xxxx

40AA    LD A,40    ;INT mode 2 MSB address
40AC    LD I,A     ;load the INT mode 2 vector
40AE    IM2        ;start INT mode 2
40B0    RET

40B1    LD A,1E    ;Sinclair ROM patterns
40B3    LD I,A     ;load pattern table pointer
40B5    IM1        ;restore INT mode 1
40B7    RET

40B8    LD HL,6500 ;load a call instruction
40BB    LD B,C1    ;the start of HFILE (E500)
408D    LD (HL),CD ;CALL 4096
408F    INC HL
40C0    LD (HL),96
40C2    INC HL
40C3    LD (HL),40
40C5    INC HL
40C6    LD A,20     ;generate a 6144 spaces starting
40C8    LD (HL),00  ;at 6503
40CA    INC HL
40CB    DEC A
40CC    JR NZ 40C8
40CE    DJNZ 40C6
40D0    RET
```

KEYBOARD MATRIX

	b0	b1	b2	b3	b4
NOT b0	**Shift**	**Z**	**X**	**C**	**V**
NOT b1	**A**	**S**	**D**	**F**	**G**
NOT b2	**Q**	**W**	**E**	**R**	**T**
NOT b3	**1**	**2**	**3**	**4**	**5**
NOT b4	**0**	**9**	**8**	**7**	**6**
NOT b5	**P**	**O**	**I**	**U**	**Y**
NOT b6	**ENTER**	**L**	**K**	**J**	**H**
NOT b7	**SPACE**	**Period**	**M**	**N**	**B**

NB: Don't know whether high of low indicates keypress. Also relevence of top 3 bits unknown.

KEYBOARD POLLING ROUTINE

```
    ld     hl,$ffff        ; 0002BB 21 FF FF
    ld     bc,$fefe        ; 0002BE 01 FE FE
    in     a,(c)           ; 0002C1 ED 78
    or     $01             ; 0002C3 F6 01
    or     $e0             ; 0002C5 F6 E0
    ld     d,a             ; 0002C7 57
    cpl                    ; 0002C8 2F
    cp     $01             ; 0002C9 FE 01
    sbc    a,a             ; 0002CB 9F
    or     b               ; 0002CC B0
    and    l               ; 0002CD A5
    ld     l,a             ; 0002CE 6F
    ld     a,h             ; 0002CF 7C
    and    d               ; 0002D0 A2
    ld     h,a             ; 0002D1 67
    rlc    b               ; 0002D2 CB 00
    in     a,(c)           ; 0002D4 ED 78
    jr     c,$02c5         ; 0002D6 38 ED
    rra                    ; 0002D8 1F
    rl     h               ; 0002D9 CB 14
    rla                    ; 0002DB 17
    rla                    ; 0002DC 17
    rla                    ; 0002DD 17
    sbc    a,a             ; 0002DE 9F
    and    $18             ; 0002DF E6 18
    add    a,$1f           ; 0002E1 C6 1F
    ld     ($4028),a       ; 0002E3 32 28 40
    ret                    ; 0002E6 C9
```

KNOWN SYSTEM VARIABLES

Address		Description
0x400c	DFILE	Pointer to display file.
0x4010	VARS	Points to variable storage (straight after DFILE)
0x4025	LASTK	Used in key reading routines.
0x4028	MARGIN	Number of blank lines above/below page.
0x403b	FAST(?)	Bit 7 - FAST mode on/off.