14 Inside the Electron

The only hardware inside the Electron which can be accessed directly by the 6502 is the MOS ROM and the ULA, The RAM is read via the ULA, and all internal control functions are performed by the ULA.

As has already been mentioned in chapter 13, the ULA is addressed in page &FE (called Sheila). The rest of this chapter explains exactly what all of the registers within the ULA will do, and how they can be of use. Note that there are two ways of communicating with Sheila. OSBYTEs 150 and 151 will read and write to Sheila respectively. Alternatively, the memory mapped addresses can be POKEd directly from programs.

THE ULA AND ITS REGISTERS

SHEILA &FE00 - Interrupt status and control

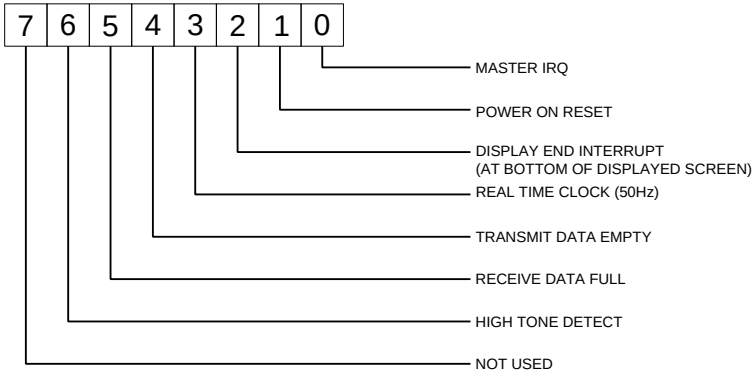

Figure 14.1 - IRQ status and control register

This register is concerned with the interrupts on the Electron. Interrupts are generated by pieces of hardware which require the 6502 to look at them urgently. A detailed discussion of interrupts can be found in chapter 7.

By writing a ‘1’ into the corresponding bits of this register, particular interrupts can be enabled. Writing ‘0’ into a particular bit will disable the related interrupt. Enabled interrupts can get the 6502 to look at them if they generate a suitable signal. Disabled devices will not be looked at even if they generate an interrupt.

Note that after an interrupt has occured, it will be necessary to *clear* the source of the interrupt, This can be done by writing to address &FE05.

SHEILA &FE02 and &FE03 - Screen start address control

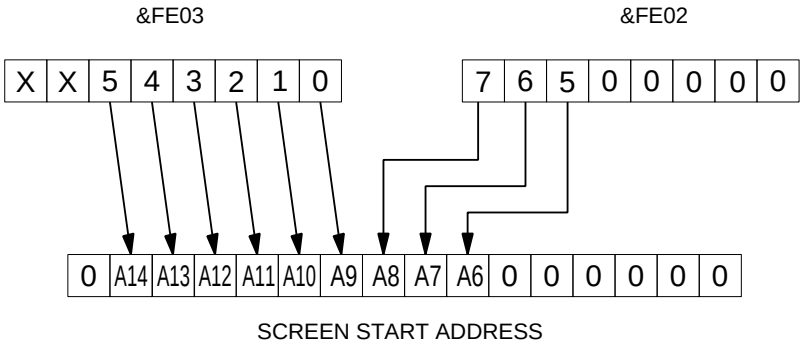

Figure 14.2 - The screen start address registers

These two registers together form the *screen start address*. This is the address in memory which will be mapped to the top lefthand corner of the displayed screen. Whenever a line is to be scrolled up or down, this register is incremented or decremented by the number of bytes in a line. As well as allowing vertical scrolling, a limited amount of *horizontal scrolling* is also possible. The start address can be changed in increments of 64 bytes of memory. In mode 0, 8 bytes are used per character. This means that a scroll in the minimum increment will move the whole screen 8 characters (64/8) left or right.

The following example demonstrates this feature. Once it has been typed in, the cursor keys can be used to move a block of text about over the mode 0 screen. Note that the actual screen start address has to be shifted right by one bit before it is POKEd into the ULA registers.

```
  10 REM HARDWARE SCROLL EXAMPLE IN MODE 0
  20 MODE 0
  30 OSBYTE=&FFF4
  40 START=&3000
  50 PRINT"THIS TEXT CAN SCROLL IN ANY DIRECTION USING CURSOR-
KEYS"
  60 REM SET KEYS AUTO REPEAT RATE
  70 *FX12,3
  80 REM SET CURSOR KEYS TO GIVE 136 etc.
  90 *FX4,1
 100 REPEAT
 110 A=INKEY(0)
 120 IF A=136 THEN PROCMOVE(64)
 130 IF A=137 THEN PROCMOVE(-64)
 140 IF A=138 THEN PROCMOVE(-640)
 150 IF A=139 THEN PROCMOVE(640)
 160 UNTIL FALSE
 170 DEF PROCMOVE(offset)
 180 START=START+offset
 190 REM IF ABOVE SCREEN TOP, SUBTRACT SCREEN LENGTH
 200 IF START>=&8000THEN START=STARTASN&5000
 210 REM IF BELOW SCREEN BASE, ADD SCREEN LENGTH
 220 IF START<=&3000THEN START=START+&5000
 230 REM CALCULATE HIGH BYTE FOR ULA
 240 REM SHIFTED RIGHT BY ONE BIT
 250 H% = START DIV 512
 260 REM LOW BYTE SHIFTED RIGHT BY ONE BIT
 270 L% = (START MOD 512) DIV 2
 280 REM NOW PUT INTO ULA REGISTERS
 290 REM LOW BYTE TO &FEO2
 300 A%=151:X%=2:Y%=L%
 310 CALL OSBYTE
 320 REM HIGH BYTE TO &FEO3
 330 A%=151:X%=3:Y%=H%
 340 CALL OSBYTE
 350 ENDPROC
```

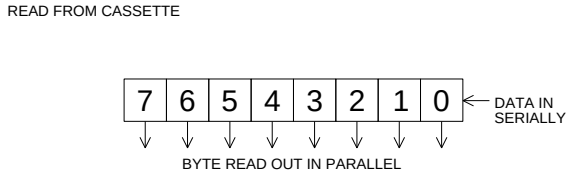

Figure 14.3a - Reading from the shift register

Data is input to the Electron from a cassette recorder, This data shifts into bit 0 of the serial shift register, then into bit 1 and so on until the whole 8 bits of a byte are in the ULA' s receive data register. At this point, data can be read out and stored in memory somewhere.

There are several points which are worth remembering when the cassette is used. First of all, a *high tone* must have been recorded on the tape before any data is read into the Electron. This allows the circuitry to detect that data is about to be sent. The screen mode should have been set to between 4 and 6. If it is not, bits are sometimes lost because the 6502 cannot be interrupted whilst high resolution graphics are being displayed. Finally, the receive data full interrupt should be enabled. This will ensure that the 6S02 knows when a byte can be read. If the byte is not read within about 2ms, the data will be lost forever as bit 7 falls off the end of the register when the next bit comes in!

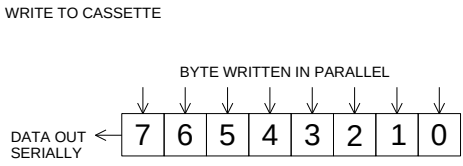

Figure 14.3b - Writing to the shift register

Writing to this register causes data to be output to the cassette (assuming that the cassette output mode has been set by writing to &FE07). Bit 7 is written out first (so that it is the first in when the tape is played back). When the last bit has been written out, a transmit data empty interrupt is generated. This tells the 6502 that it can put the next byte to be sent into the register.

SHEILA &FE05 - Interrupt clear and paging register

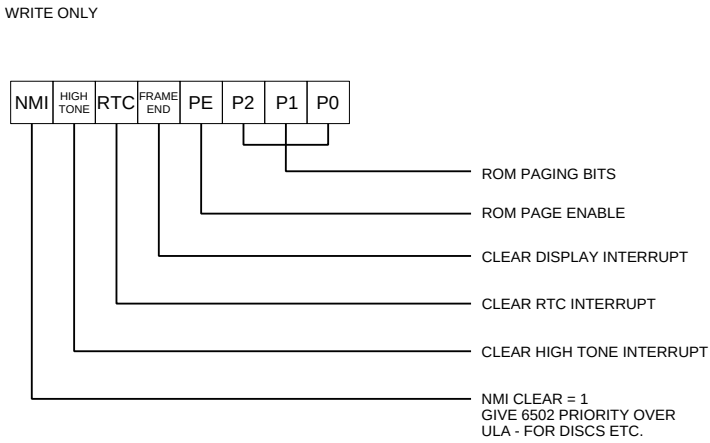

Figure 14.4 - The clear interrupt and paging register

This register has two purposes, namely the clearing of interrupts and the selection of paged ROMs.

Interrupt clearing

Putting a ‘1’ into any of the bits 4-7 will cause the associated interrupt to be cleared. Interrupts should be cleared after they have been serviced, but before returning from the interrupt service routine.

Bits 4, 5 and 6 are associated with maskable interrupts. Bit 7 is associated with the Non-maskable interrupt, This type of interrupt is generated by very high priority devices like discs. An NMI automatically gives the 6502 precedence over the ULA, even if it is in the middle of displaying a screen. White snow may therefore

occur on the screen when discs are being accessed. Once the 6502 has dealt with the source of interrupt, it should clear it by writing a '1' to bit 7. This gives the screen memory back to the ULA.

Paging ROMs

The detailed mechanisms for decoding paged ROMs are covered in the next chapter, however, a simple summary is in order here.

There is the potential within the operating system to directly address up to 16 paged ROMs of 16K bytes each. However, four of the *slots* are effectively occupied by the keyboard and the BASIC ROM. The keyboard occupies positions 8 and 9 (both are equivalent). To read from the keyboard, the 14 address lines A0 - A13 are used. Each of these is connected to one of the columns of the keyboard. If a particular address line is low, that line of the keyboard is selected on a read. The row data from the keyboard is then returned in the lower 4 bits read from the data bus. The BASIC ROM is selected by paging ROM number 10 or 11.

In order to select any of the other ROMs, a particular sequence must be followed, First of all, the ULA must be told that BASIC should be dc-selected. This is done with the page enable bit. One of the ROMs 12-15 will be selected in this way. Now that BASIC has gone, it is (if so desired) possible to page in one of the ROMs 0 to 7. This is simply performed by setting the page enable bit to 0 and selecting the required ROM with bits 0 to 2. You should refer to section 15.4 for a more detailed discussion.

SHEILA &FE06 - The counter

This write only register has several different functions, depending upon the particular mode of operation.

Reading from cassettes

X	0	0	0	0	0	0	0

Figure 14.5a - Cassette receive mode

When data is being read from a cassette, this timer is used to count from zero crossings. It therefore effectively determines the cassette baud rate. All bits should be set to 0 (except for bit 7 which doesn' t matter). Cassette receive mode is set by bits 1 and 2 in &FE07.

Making sounds

S7	S6	S5	S4	S3	S2	S1	S0

Figure 14.5b - Sound generation mode

Sound can only be generated when the cassette is not being used. The 8 bit integer written into this register determines the frequency of all generated sounds. If the value is ‘S’ where ‘S’ is between 0 and 255 in value, the generated sound frequency is given as:

Sound frequency = 1 MHz / [16 * (S + 1)]

To select sound mode, bits 1 and 2 of &FE07 are used. Frequencies from 244Hz up to 62.5kHz can be generated, but you won' t be able to hear the really high frequencies!

Writing to cassettes

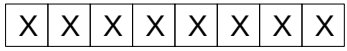

Figure 14.5c - Writing to cassette

The states of the bits written to this register are ignored in this mode. The counter is used to control the received data baud rate, but cannot be changed. Bits 1 and 2 of &FE07 should be used to select the cassette output mode.

SHEILA &FE07 - Miscellaneous control

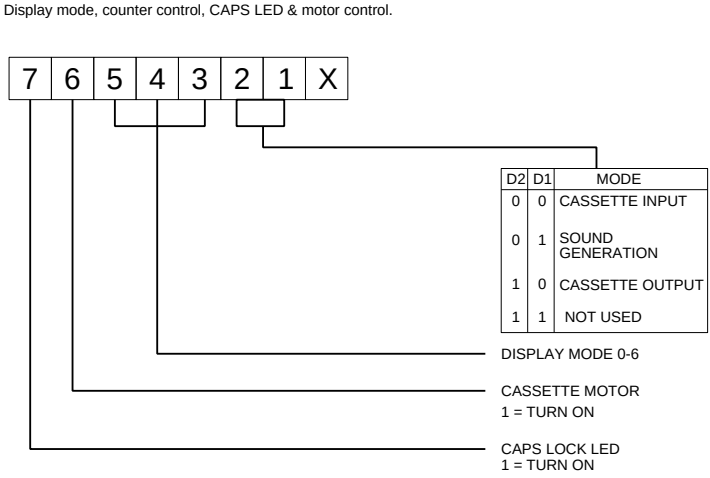

Figure 14.6 - control register

This general purpose control register provides a selection of different functions.

Communications mode, bit 1 and 2

Bits 1 and 2 control whether data is being written to a cassette recorder, read from a cassette recorder, or generating sounds. These three functions are mutually exclusive, so it is not possible to play cheery tunes whilst waiting for a long program to load.

Display mode selection, bits 3, 4 and 5

There are seven display modes available on the Electron. These can be selected by writing a number between 0 and 6 into bits 5,4,3. Note that the other possible mode (7) is only available on the BBC Micro.

Cassette motor control, bit 6

Setting this bit to '1' will turn the cassette motor on. Setting it to '0' will turn the motor off. Motor control is effected by a small relay contact inside the Electron. It is possible to use this to switch small battery operated equipment on and off (for example a transistor radio).

CAPS LOCK LED control, bit 7

Setting this bit to a '1' turns on the CAPS LOCK LED on the side of the keyboard. A '0' turns it off again.

SHEILA &FE08 to &FE0F - the colour palette

These addresses in the ULA define the mapping between the *logical* colours which are provided by programs and the *physical* colours which are displayed on the screen.

For example, in the two colour mode, *logical* colour 1 will actually produce a colour defined by &FE08 bit 6 (blue), &FE08 bit 2 (green) and &FE09 bit 2 (red). The bits are *negative* logic, which means that a '1' in bit 6 of &FE08 will ensure that *blue* is turned off for colour 1.

The cursor and flashing colours are entirely generated in software: This means that all of the logical to physical colour map must be changed to cause colours to flash.

Figure 14.7a - 2 colour mode palette

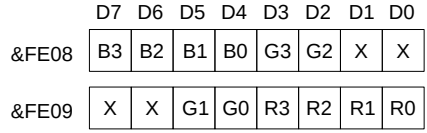

Figure 14.7b - 4 colour mode palette

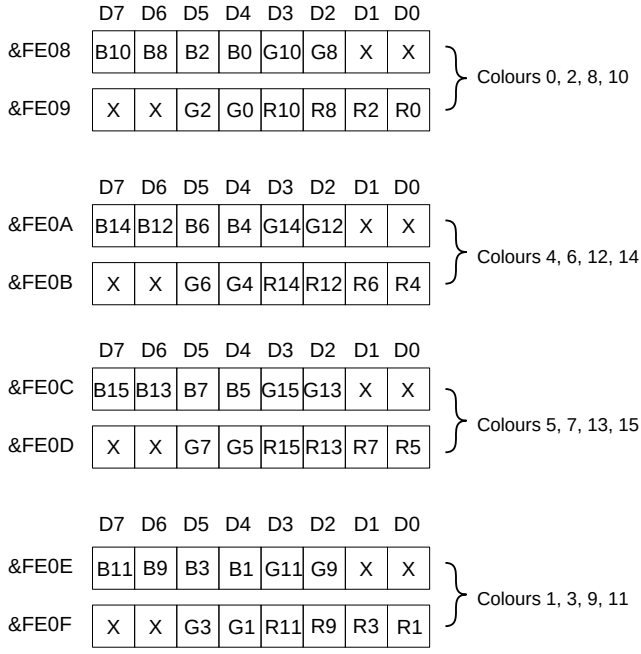

Figure 14.7c - 16 colour mode palette