Chapter 20: LYNX HARDWARE

AN OVERVIEW

The Z80

The Lynx uses a Z80 microprocessor which was originally designed by Zilog Inc., but today it is 'second sourced' by Mostek, Sharp, SGS ATES, and others. It is very popular chip, both for home and business computers, and for industrial control applications because it has a powerful instruction set which includes all the instructions offered by a popular earlier microprocessor, the Intel 8080, and so can run its software, which includes CP/M.

The Lynx makes provision for you to experiment with machine code, both from the familiar world of Basic and through the Machine Code Monitor. But remember that when you program in Basic the operating system protects you from crashing the computer -- by detecting errors and displaying warnings. When you program in machine code the Z80 will obey your instructions without question; if program crashes you will probably have to start again -- by resetting with the [BREAK] key (see Chapter 19) or by switching the Lynx off and on again.

But a crash is not as terrible as it sounds: microprocessors do not know how to read a keyboard, put characters on a screen, and so on, they can only do so by following a precise sequence of instructions. If you interfere with the flow of these instructions, the computer will not be able to carry out normal operations and will seem to crash, but inside, the hardware will still be executing your instructions.

If you want to experiment with machine code, you will need to know some of the principles of Lynx hardware -- so read the rest of this chapter -- and you will need a Z80 programming manual (details of two popular books are given in Chapter 16). Then you can refer to the sections describing CODE statements in Basic programs and the Machine Code Monitor (see Chapter 17).

The 6845 Cathode Ray Tube Controller

The 6845 was originally designed by Motorola, and is now second sourced by Hitachi, Synertek and others. Its function is much simpler than the Z80, since it is essentially a collection of counters. But if, for example, you want to adjust the format of the screen, or alter scrolling, you need to know how it works.

The 6845 can be described as a Programmable Address Sequencer. It 'counts' its way through video memory, from top to bottom, selecting each byte of data in turn and sending it to the TV or monitor (both Cathode Ray Tubes devices) where the Cathode Ray (electron beam) sweeps cross the screen at high speed depositing the data from the video memory onto the screen. The 6845 and the electron beam are synchronised together, with the 6845 in control. For more about the 6845, see later in this Chapter.

High Resolution Video

The Lynx has a high resolution bit mapped colour display. Bit mapped simply means that each point on the screen corresponds to a bit in video memory; more precisely, each point has three primary colours associated with it, red, blue and green, and each of these is either set on or off according to the value of a bit in video memory. Some computers do not have bit mapped screens because many users are satisfied with a text display. Moreover, high resolution takes up a lot of memory. If the computer stores its picture of the screen as ASCII values -- which character number is displayed in which character position -- and generates the picture of the character from special hardware, the screen requires much less memory. But unfortunately, it is very difficult to draw a straight line, a circle, or to redifine characters on this sort of screen. The Lynx is able to display not only text but also sophisticated graphics.

To see how the Lynx handles its screen, let's look at the hardware and software processes involved in displaying a single dot.

The screen memory is arranged with each byte of video memory appearing as a horizontal row of eight bits on the screen. Writing one dot must not change the other seven pixels (bits) in the screen byte, so we need a read-modify-write operation; and this must be done for up to 3 colours. A further complication is that the Z80 can't access the the screen memory whilst the 6845 is accessing it without interfering with the display, so it must be synchronised with the line and frame blanking periods when the electron beam is flying back to start a new line or new frame and normal scanning is suspended.

The procedure for writing a character is equivalent to writing several dots. Accessing the screen via machine code or ROM routines is covered in detail later in this chapter. (See also Chapter 14 for a description of the Basic commands available for controlling the screen).

Bank Switching

The Lynx can be expanded to 192K or 256K of RAM, together with up to 40K of ROM including external ROM. But the Z80 can only address a total of 64K of memory. 'Bank witching' is a technique which allows it to address more -- theoretically, as much as you need. It requires special hardware built around the Z80 and software to drive it. In this section we'll explore the concepts involved in bank switching -- we'll look at the details later!

Normally, in a single bank system, the Z80 is kept busy scanning the keyboard, writing to the screen and executing programs. This also happens in the Lynx, except when you want to access the screen. The screen memory is not in this "normal" bank -- the Lynx screen can use up to 128K, so it has to have 2 dedicated banks.

The video banks are a bit like a parallel world -- you can't peek and poke for them, you have to find the switch that let's you pass through into them. The Z80 can throw the switch by writing to a port. But remember that the Z80 is continually moving through memory in an orderly manner, executing instructions one after another. If you switch banks, all the addresses it manipulates now refer to a different block of physical memory (the parallel world). It is not directly aware of the bank switch, but it must still recieve the correct sequence of instructions to maintain control of the computer. So program flow must carry on across the boundary: before you switch banks you must place executable code into the target bank plus a mechanism for returning to the normal bank again. You need to be careful!

But once you have this critical code, you can move data between banks and because each bank can be seperately read and write enabled you can be fetching instructions from one bank whilst you are writing to another. You can pass a large block of code to a bank, then execute it by transferring control to that bank when you're ready.

One complication is that is that the ROM bank (bank 0) can be simultaneously read enabled with another bank -- this is ordinarily not possible since the two banks would conflict -- and takes priority over the other bank for all addresses from

0000H to 5FFFH

and also

C000H to DFFFH if XROMI (external ROM) is present

and

E000H to FFFFH if XROMI is present.

This means that Basic need not concern itself with bank switching except when it is accessing the screen. A bank of ROM enabled alone is of little use since even for the smallest programs you usually need some RAM.

Input/Output and System expansion

The Lynx allows for "general purpose" expansion via the forty-way connector at the back. You can attach peripherals such as parallel printer, joystick and disk drives using special expansion packs. These can be ganged together, to a maximum of 3 packs -- limited by the Lynx power supply which is not rated to drive more.

There is provision for up to 16K of external ROM and also a fourth memory bank of up to 64K which would allow the Lynx to run CP/M+ -- an enhanced version of CP/M.

The forty-way connector gives you access to all the main internal bus signals (data, address and control) so you can attach home made hardware -- the electrical details are given later in the chapter,

A number of dedicated I/O signals are provided via the DIN sockets -- full electrical details are given later.

The cassette socket allows you record and replay Basic programs, blocks of memory and blocks of data store. It also provides a high level analogue output signal.

The RGB socket lets you use a colour monitor display (nd on French models provides the signals for a Peritel display).

The serial socket provides serial data channels, both in and out, together with handshake in both directions. It can be used with many different baud rates and transmitting formats, allowing you to communicate with the majority of 'RS232' type devices and other Lynxes (see Chapter 13).

The light pen socket offers 3 useful signals: light pen; a composite video signal, to drive black and white monitor; and an analogue to digital input.

Finally there is a UHF output suitable for running domestic colour or black and white TV. In some countries this may be VHF output.

The Lynx offers several different memory options. The following table summarises their different specifications:

LYNX	USER MEMORY	VIDEO MEMORY	VIDEO RESOLUTION	CP/M CAPABILITY
48K	16K	32K	256x256	NO
64K	32K	32K	256x256	NO
96K	64K	32K	256x256	NO
128K	64K	64K	512x256	YES
192K CP/M+	128K	64K	512x256	YES
192K with HI-RES	64K	128K	512x512	YES
256K	128K	128K	512x512	YES

All graphics commands in Basic are compatible with the different screen resolutions.

HARDWARE FEATURES IN DETAIL

WAIT states

WAIT states slow down the operation of the machine. The Z80 is continually executing one machine cycle after another; normally each machine cycle involves a transaction (read or write) between the microprocessor and either memory or an I/O device. Each device (memory or I/O) external to the Z80 has a specified access time. If this time is not met the device may not operate correctly.

WAIT states are inserted by the hardware to lengthen certain machine cycles so that the timings are met. Each machine cycle is measured in 't states' --

there may be 3, 4, 5 or 6 depending on the type of instruction. One t state lasts on sixth of microsecond. A WAIT state is an extra t state added to the normal number.

If you know where t states have been added, and how many, you can work out the length of each machine cycle and time a program precisely.

The following table summarises the rules governing WAIT states. The example shows how to predict where extra t states will occur.

Description		Effect
XROM READ	M1 CYCLES	TWO extra t states
ALL OTHER	M1 CYCLES	ONE extra t state
BANKØ ENABLED MEMORY READ	M2,3,4,5 CYCLES	ONE extra t state
ANY	I/O CYCLES	ONE extra t state
ANY OTHER	CYCLE	NO extra t state

An XROM read cycle is a memory read from bank 0 external ROM -- addresses C000H to FFFFH.

An M1 cycle is any machine cycle where the machine brings the $\overline{\text{M1}}$ signal (Z80 pin 27) low -- enables it. This comprises op-code fetches (note that some instructions have a two byte op-code, each byte requiring an M1 cycle) and the interrupt acknowledge cycles (maskable and non-maskable).

An I/O cycle is any machine cycle that reads from or writes to a port. Note that the WAIT STATE added here is additional to the one the Z80 always inserts into I/O operations.

WAIT STATE example

Assume that the bank switch contains 00H nd so both bank 0 ROM and bank 1 RAM are accessible, also suppose that there is external ROM present from E000H to FFFFH.

```
E8FE     LD(HL),A
         M1 fetch opcode              CASE 1
         M2 send A to (HL)            CASE 5

E900     JP (7530)                    ;jump out of XROM into RAM
         M1 fetch opcode              CASE 1
```

```
         M2 fetch first address byte   CASE 3
         M3 fetch second address byte  CASE 3

7530     LD A,
         M1 fetch opcode               CASE 2
         M2 fetch operand              CASE 3

7532     OUT(82),A                     ;switch out bank 0
         M1 fetch opcode               CASE 2
         M2 fetch port address         CASE 3
         M3 output byte to port        CASE 4

7535     RLC(HL)
         M1 fetch first opcode byte    CASE 2
         M1 fetch second opcode byte   CASE 2
         M2 fetch (HL)                 CASE 5
         M3 send (HL)                  CASE 5
```

INTERRUPTS

A Z80 program can be interrupted so that it can respond quickly to unpredictable or infrequent events. There are two types of interrupt -- maskable and non-maskable. Two seperate pins on the Z80 recieve the different signals: $\overline{\text{INT}}$ (pin 16)recieves the maskable and $\overline{\text{NMI}}$ (pin 17) the non-maskable. These are brought out of the forty-way expansion connector pins 23 (maskable) and 22 (non-maskable). Both interrupts are used by the Lynx itself, you can still use them yourself.

The maskable interrupt pin of the microprocessor can be activated either by the [BREAK] key (for as long as it's pressed) or by the CURSOR signal from the 6845 (pin 19). The CURSOR output is a programmable signal normally intended to generate a visible cursor on a monitor screen, using extra hardware which brightens or inverts the video signal at the appropriate point. But the Lynx uses the signal to provide a simple means of synchronising screen access to the frame blanking period.

When you switch the machine on, the 6845 is initialised to generate the desired picture format and the CURSOR signal is programmed to activate briefly (0.66 us) at the very bottom right hand corner of the screen, just before the beginning of the frame blanking period. This signal is "stretched" so it's easier to detect (40 us +- 8 us) and can be used to divert the microprocessor to updating the screen at a time when the picture will not be corrupted (no snow!).

The computer needs to detect the difference between a CURSOR interrupt and a [BREAK] interrupt. This is possible because the CURSOR signal is brief, whilst the [BREAK] key is held down for a long time (in computer terms).

You can use the maskable interrupt with external equipment since the CURSOR signal can be turned off by reprogramming the 6845 (see the following section). The [BREAK] key cannot be disabled.

NORMAL 6845 REGISTER VALUES		
RØ	Horizontal total	5F hex
R1	Horizontal displayed	4Ø hex
R2	Horizontal sync position	4C hex
R3	Horizontal sync width	37 hex
R4	Vertical total	46 hex
R5	Vertical total adjust	1C hex
R6	Vertical displayed	3F hex
R7	Vertical sync position	44 hex
R8	Interlace & skew	ØØ hex
R9	Character raster count	Ø3 hex
R10	Cursor start raster address	Ø3 hex
R11	Cursor end raster address	Ø3 hex
R12,13	Start address	ØØ,4Ø hex
R14,15	Cursor address	ØF,FF hex
R16,17	Light pen address	—

The non-maskable interrupt pin of the Z80 has only one internal use on the Lynx -- single stepping. The single step feature of the Lynx monitor (see Chapter 17) allows you to execute a program one instruction at a time. The hardware, being activated by an input from port 84 causes a non-maskable interrupt an exact number of instructions later, returning control to the monitor, which displays the new register values. When it's not being used for single stepping, the non-maskable interrupt can be used via the forty-way expansion port.

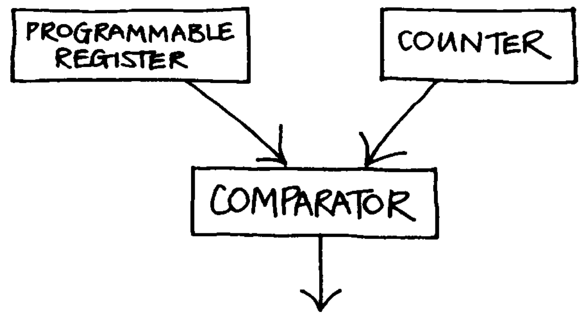
If you want to experiment and substitute your own interrupt routines, it's easy to trace the ordinary program flow to the system interrupt handlers, and redirect it to your own routines, since they're both vectored through RAM calls. Note that you should select only maskable interrupt mode 1, the other two modes are not supported by the hardware.

Programming the Lynx's 6845

The 6845 has seventeen internal registers. When you switch the Lynx on, one of the first things the Operating System Software does is initialise the 6845. If this is not done correctly the picture will not be stable.

The essence of TV or monitor display is timing. The picture is redrawn exactly 50 times a second, each line on the screen is scanned in 64us, each individual pixel is present for one twelfth of a micro second. The display device (TV or monitor) needs information in a video signal to distinguish successive lines of picture data (a line SYNC), and to indicate the end of the whole frame (a frame SYNC). These signals trigger the flyback of the electron beam to the start of a new line or frame. The 6845 provides the majority of these timing signals.

Let's look at some of the internal structures of the IC. For example, Register 1 is the "horizontal displayed" register and works like this: it contains the number of bytes to be displayed along one scan line. Associated with it is a counter register which starts with a value of zero at the beginning of a line, then increments each time a byte is sent to the screen. (This register is 'invisible' to the programmer). The two registers have their contents combined by a device called a comparator which detects when the registers have the same value -- when 32 bytes have been displayed -- and turns the display off.



The other parts of the IC use similar mechanisms: the 6845 is really nothing more than a series of counters. We'll now look at each of its registers in turn -- read the section as a whole because important points are covered where appropriate. Remember that the 68445 is byte-oriented - it is not aware of individual pixels, which are clocked out by special hardware.

R0 Horizontal Total

R0 stores the total number of byte periods in one scan.

To work out the value, take the number of bytes displayed, and add the number that could be displayed up to the end of the next line, while the electron beam is turned off, then subtract one from the total. One byte is displayed every 2/3 us; TVs and monitors are designed to have a line scan time of 64 us, so R0 is initialised to 42 decimal or 2A hex. Changing this value is not recommended because it would interfere with correct TV or monitor operation.

R1 Horizontal Displayed

R1 stores the number of displayed byte periods in one horizontal line.

The 128K Lynx normally displays 64 bytes across the screen, so this register is initialised with 64 decimal or 40 hex. Changing the value will affect the width of the screen displayed and disrupt the normal picture, causing a skewing of successive character lines. This skewing happens because -- supposing you reduce R1 to 63, for example -- the character which was at the end of one line now appears at the beginning of the next line, and the effect mounts up as it moves down the screen. Moreover, the skew is based on the 6845s character blocks, which are 4 pixels high (see below).

R2 Horizontal Sync position

The horizontal sync is used by the TV or monitor to trigger the flyback of the electron beam to the start of the next line. This sync is usually placed roughly central in the line blanking period -- its position will affect the left-right positioning of the picture. Taking the first displayed byte of the line as number zero, the register holds the number of the byte where the sync is located. The 128K Lynx initialises this to 4C hex.

R3 Horizontal Sync Width

The horizontal sync width can be adjusted in terms of byte periods, to match the specification of the TV or monitor involved, and the duration of the byte period. The 128K lynx initialises this register with 37 hex and unless you're using a non-standard monitor you shouldn't need to change it. A value of zero would remove the sync pulse altogether.

R4 Character Row Total

This register is associated with the total number of horizontal scans in one frame, including those which are not displyed. A frame is displayed every 20 ms, and a scan line takes 64 us, so there are roughly 312 scans. But the 6845 is oriented towards character based displays and in this case its characters are set to 4 scan lines high -- the register is initialised with 46 hex -- giving total row count of about 78. This is done because R4 is only 7 bit -- not large enough to store 312 -- so the characters have to be more than 2 pixels high. The value chosen had to be a power of 2 so that the video memory

could be adressed in one continuous block.

A value of (M-1) in the register will give total of M character lines per frame. But you shouldn't need to alter the initilisation value.

R5 Vertical Total Adjust

This register is needed because R4 is too coarse a way of defining the frame duration -- it must be as close to 50Hz as possible. The adjustment is done in scan line units and put into R4. It is initialised at 10 hex, and adjusting it may result in a poor picture.

R6 Character Row Displayed

This register holds the total number of displayed 6845 character rows minus one; the display can only be a multiple of 6845 character rows. The initialisation value of R6 is 3F hex.

R7 Vertical Sync Position

The vertical sync is used by the TV or monitor to trigger the flyback of the electron beam to the top of the screen. It is programmed in multiples of 6845 character rows (the required number minus one) and is initialised to 44 hex which positions the picture centrally on the screen.

R8 Interlace Mode

This is a 2 bit register which selects mode operation and on the 128K Lynx it is initialised to select a non-interlaced display.

An interlaced display is a means of producing a high resolution picture without losing speed, using up too much memory or needing a very expensive monitor. To produce a display with 512 horizontal lines, alternate frames display alternate sets of 256 lines -- in a particular frame only every other line of the picture is displayed.

R9 Maximum Raster Address

The 6845 uses characters 4 pixels high but this has no effect on the height of the characters actually displayed since they are under software control. R9 sets the number of scan lines in a character row -- the pixel height of the characters. It is initialised to 3 (one less than the number of scan lines needed. Any adjustments to this register really need to be supported by hardware changes to achieve anything useful.

R10 Cursor Start Raster and R11 Cursor End Raster

Thė 6845 provides the signal to generate a "hardware cursor" on a display. The Lynx's flashing block cursor is generated by software; and the hardware cursor is used to interrupt the Z80 at the very end of the visible screen, and indicate that frame blanking is about to begin. R10 and R11 select which rows, or bytes, of the 6845 character the cursor is covering generate the cursor signal. The Lynx needs only one interrupt per screen, at the very base of the screen, so both R10 and R11 are set to 3. The upper 3 bits of R10 control cursor blinking, but do not apply, so bit 6 is set to zero. If bit 5 is set, the cursor signal will disappear altogether; so you can use 23H to disable cursor interrupts.

R12 and R13 Start Address

This 14 bit register determines the memory address from which the first byte on the screen is fetched. Scrolling can be controlled via this register; but there is no control over the raster start address, and verticl scrolling is by four pixel increments.

Strictly speaking, this register holds a character address (we'll look at the actual memory address a character occupies later, when we consider the memory map of the video system). The 6845 -- with its "characters" -- was originally designed for use in low resolution displays using a character ROM rather than high resolution display.

Character addresses increase from left to right, and from top to bottom, in much the same way as the screen is scanned, except that a character row contains 4 scan lines. Adding or subtracting 64 from the start address causes a one character line scroll (4 pixels). You can obtain an imperfect form of horizontal scroll by using other values.

R14 and R15 Cursor Position

The cursor position register holds the address of 6845 character on the screen (see R10 and R11 for the special use made of the cursor). The cursor is placed at the bottom right hand corner of the screen on initialisation; R14 is loaded with 0F hex and R15 with FF hex.

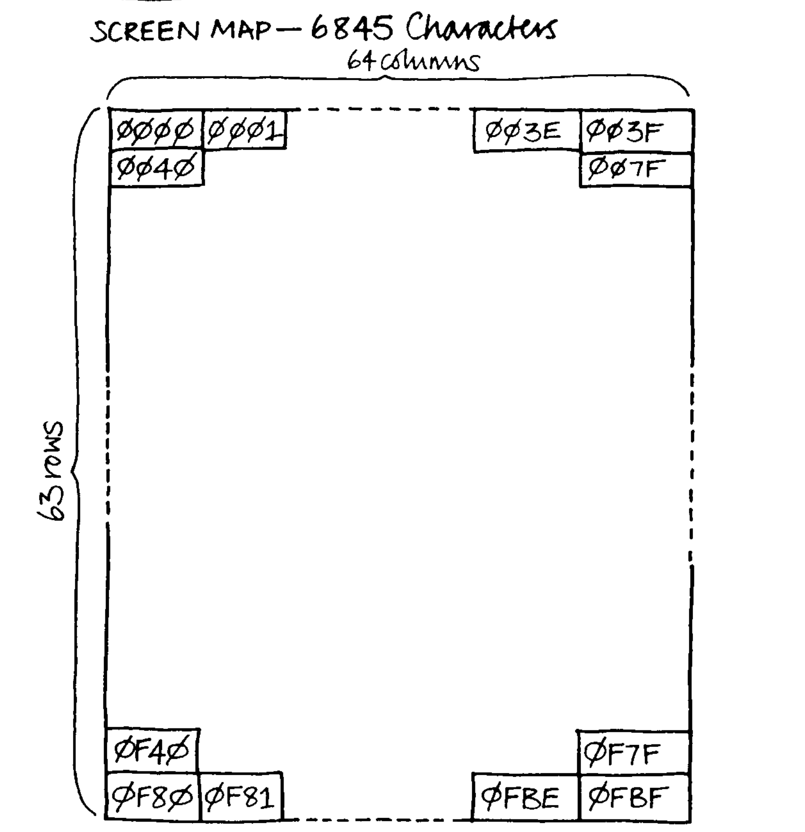
R16 and R17 Light Pen Position

The light pen position register is a read only register which will contain the character address of a light pen, providing the necessary electrical connections have been made. Note that this limits the accuracy of a light pen to a rectangle 8 pixels x 4 pixels, although this can be improved with clever software!

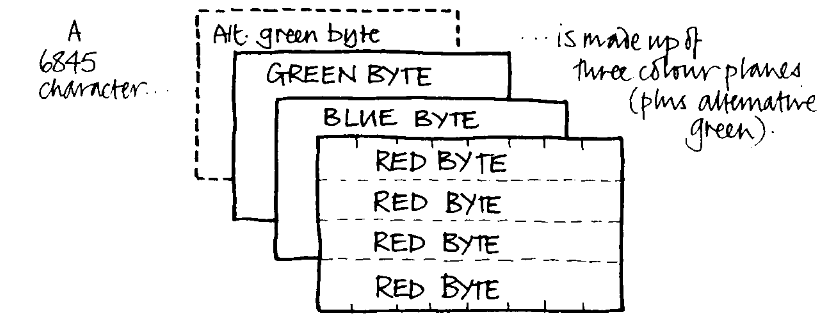
Driving a Bit Mapped Screen

The 128K Lynx has 64K bytes of memory for the display and 64K for User RAM. These occupy two seperate banks and you need to control the bank switch to access video memory. In this section we'll look at the memory map and some examples of how to access the screen using machine code.

In the previous section we saw that the 6845 has characters which are 8 pixels wide and 4 pixels deep. The following diagram illustrates how these chracters cover the screen in 63 rows and 64 columns, and how each of the characters has three colour planes, blue, red and green (together with alternative green, which can be displayed instead of ordinary green -- for more details see Chapter 19.



Note that the start address can be altered, in which case all the values above will be displaced by the same amount.



The screen is further subdivided into individual byte addresses, each of the four colour blocks (red, blue, green and alternative green) taking up 16K each. The following diagram shows how these addresses are mapped to the physical screen. Note that if you change the 6845 start address, this will change.

Read or write accesses to the screen should be synchronised to either the line blanking or the frame blanking period. Line blanking lasts for about 22 us at the end of each visible line and can be used for single byte transfers; frame blanking lasts much longer -- about 3.4 ms -- but occurs only once every 20 ms, so is suited to larger transfers of data. Remember that if you don't synchronise screen accesses you'll get brief 'blackouts' -- tiny black lines -- on the part of the picture currently being drawn.

The following examples show how you can read and write bytes during interline blanking. The first example executes code entirely in user RAM. The interline access bit of port 80 is used to make the Z80 wait until the next frame blanking period; this is invisible to the programmer, and is achieved by "bus requesting" the CPU, to suspend operation. The CPU access bit of port 80 is used to give access to the video memory.

WRITING A BYTE TO THE SCREEN

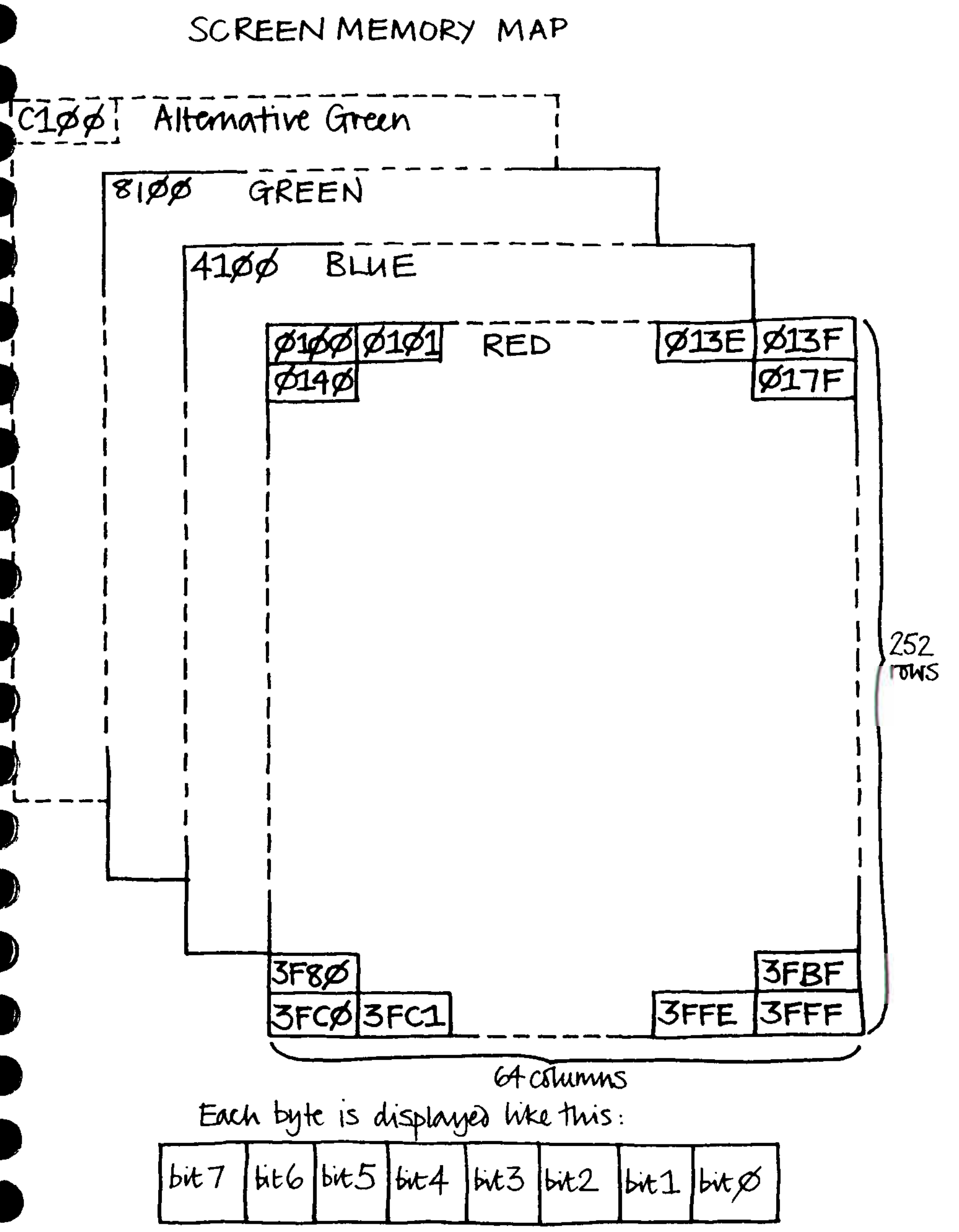
```
8000      21 AB 16  LD HL,16ABH        ;HL points into video RAM
8003      16 7B     LD D,7BH           ;D holds byte to be written
8005      3E C0     LD A,11000000B
8007      D3 82     OUT (82H),A        ;enable write to bank 2, disable bank 1
8009      3E 40     LD A,11000000B     ;A holds mask to set interline bit
800B      D3 80     OUT  (80H),A       ;CPU waits here until interline blanking
800D      3E 20     LD A,00100000B     ;A holds mask to set CPU access bit
800F      D3 80     OUT (80H),A        ;CPU has access to video
8010      72        LD (HL),D          ;write byte to video
8011      AF        XOR A              ;clear A register
8012      D3 80     OUT (80H),A        ;disable CPU access
8014      D3 82     OUT (82H),A        ;reset bank latch
8016      C9        RET
```

The second example is more complex because to read a value from the screen you also have to be reading op-codes (only one bank can be read enabled at a time).

The first part copies 5 bytes of code across into the video RAM using the write routine above. Interline synchronisation and CPU access is achieved as before, but the bank switch is now done after these operations because it's only after you've achieved synchronistion and access that you can fetch op-codes from bank 2 video RAM.

READING A BYTE FROM THE SCREEN
(uses write byte routine)

```
8030      21 4E 80  LD HL,804EH        ;HL points to code to be moved to
                                        video RAM
8033      06 05     LD B,5H            ;B is loop counter
8035      56        LD D,(HL)          ;D holds byte of code
```



```
8036    CD 05 80  CALL (8005H)       ;call the write byte routine
8039    23        INC HL
803A    10 F9     DJNZ F9H           ;loop to copy five bytes
803C    01 60 80  LD BC,8060H        ;BC is destination address in user
                                      RAM
803F    21 4E 80  LD HL,804EH        ;HL is source address in video
8042    3E 40     LD A,01010000B
8044    D3 80     OUT (80H),A        ;set CPU access bit
8046    3F 20     LD A,00100000B
8048    D3 80     OUT (80H),A        ;set CPU access bit
804A    3F 0E     LD A,00001110B
804C    D3 82     OUT (82H),A        ;enable read from video, write to
                                      user, now fetching opcodes from
                                      video
804E    7E        LD A,(HL)          ;read byte from video
804F    02        LD (BC),A          ;write byte to user RAM
8050    AF        XOR A              ;clear A
8051    D3 82     OUT (82H),A        ;restore bank latch; now fetching
                                      opcodes from user RAM
8053    D3 80     OUT (80H),A        ;reset CPU access
8055    C9        RET                ;return
```

BANK ARCHITECTURE AND CONTROL

The 128K Lynx's bank architecture is shown in the diagram.

Bank 0 is read only and contains the Basic ROMs. Bank 1 is roughly divided as shown for Basic, or else can be turned over completely to CP/M. Note that banks 3 and 4 are available for expansion of video and user RAM respectively.

128K LYNX MEMORY MAP

	ØØØØ –1FFF	2ØØØ –3FFF	4ØØØ –5FFF	6ØØØ –7FFF	8ØØØ –9FFF	AØØØ –BFFF	CØØØ –DFFF	EØØØ –FFFF
BANK Ø	BASIC ROMS			NOT AVAILABLE			EXTERNAL ROM I	EXTERNAL ROM II
BANK 1	STORE			WORKSPACE RAM				
BANK 2	RED		BLUE		GREEN		Alternative green	
BANK 3	AVAILABLE FOR VIDEO EXPANSION							
BANK 4	AVAILABLE FOR USER RAM EXPANSION							

128K LYNX INTERNAL I/O

FUNCTION	DATA								ADDRESS																	
	D7	D6	D5	D4	D3	D2	D1	DØ	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ø	RD/LR	Conventional address
CRTC WRITE	← REGISTER DATA →																	Ø				1	1	1	OUT	87
	← REGISTER SELECT →																	Ø				1	1	Ø	OUT	86
D/A LATCH			B5 MSB	B4	B3	B2	B1	BØ LSB										Ø				1	Ø		OUT	84
BANK LATCH	$\overline{\text{WREN1}}$	WREN2	OLSEL	WREN4	$\overline{\text{RDENØ}}$	$\overline{\text{RDEN1}}$	RDEN2	RDEN4										Ø				Ø	1		OUT	82
VIDEO LATCH	SERIAL DATA OUT	INTERLINE CONTROL ACTIVE HIGH	CPU ACCESS ACTIVE HIGH	GRN/ ALT. GRN 1=ALT. GRN	CASSETTE MOTOR ACTIVE HIGH	CASSETTE ENABLE ACTIVE HIGH	SERIAL H/S OUT	SPEAKER ACTIVE HIGH										Ø				Ø	Ø		OUT	80
CRTC READ	← DATA FROM READABLE REGISTER DEFINED BY PREVIOUS REGISTER SELECT →																	Ø				1	1		IN	87
SINGLE STEP																		Ø				1	Ø		IN	84
SERIAL BUFFER	CLOCK					CASS. ANALOGUE IN	SERIAL DATA IN	SERIAL H/S IN										Ø				Ø	1		IN	82
KEYBOARD	SHIFT	ESCAPE	↓	↑	SHIFT LOCK			1					Ø	Ø	Ø	Ø		Ø				Ø	Ø		IN	~~Ø~~Ø80
			C	D	X	E	4	3					Ø	Ø	Ø	1		Ø				Ø	Ø		IN	Ø180
		CONTROL	A	S	Z	W	Q	2					Ø	Ø	1	Ø		Ø				Ø	Ø		IN	Ø28Ø
			F	G	V	T	R	5					Ø	Ø	1	1		Ø				Ø	Ø		IN	Ø38Ø
			B	N	SPACE	H	Y	6					Ø	1	Ø	Ø		Ø				Ø	Ø		IN	Ø48Ø
			J		M	U	8	7					Ø	1	Ø	1		Ø				Ø	Ø		IN	Ø58Ø
			K		,	O	I	9					Ø	1	1	Ø		Ø				Ø	Ø		IN	Ø68Ø
			;		.	L	P	Ø					Ø	1	1	1		Ø				Ø	Ø		IN	Ø78Ø
			:		/	[	@	–					1	Ø	Ø	Ø		Ø				Ø	Ø		IN	Ø88Ø
			→		RETURN	←	7	DELETE					1	Ø	Ø	1		Ø				Ø	Ø		IN	Ø98Ø

A[illegible] DA[illegible] ADDRESS[illegible] C[illegible].

LYNX EXTERNAL I/O MAP

DESCRIPTION	DATA								I/O ADDRESS								DATA DIRECTION
	D7	D6	D5	D4	D3	D2	D1	DØ	A7	A6	A5	A4	A3	A2	A1	AØ	
DISK	1793 RØ–R3								✳	1	Ø	1	Ø	Ø	ØØ to 11		READ
	1793 RØ–R3								✳	1	Ø	1	Ø	1	ØØ to 11		WRITE
	1=125ns Ø=250ns	1=No PRECOMP	NC	EPROM ENABLE	MOTOR	SIDE	SEL 1–4		✳	1	Ø	1	1	Ø	Ø	Ø	WRITE
JOYSTICK	SPARE	SPARE	FIRE	SPARE	RIGHT	LEFT	DOWN	UP	✳	1	1	1	1	Ø	1	Ø	READ
	SPARE	SPARE	FIRE	SPARE	RIGHT	LEFT	DOWN	UP	✳	1	1	1	1	Ø	1	1	READ
CENTRONICS PRINTER INTERFACE	PRINTER STATUS WORD					SEL	PE	BUSY	✳	1	1	1	1	1	Ø	Ø	READ
	PRINTER INITIALISE SIGNAL								✳	1	1	1	1	1	Ø	1	R/W
	DATA 8	DATA 7	DATA 6	DATA 5	DATA 4	DATA 3	DATA 2	DATA 1	✳	1	1	1	1	1	1	Ø	WRITE

✳ = "Don't Care"

NOTE When a port is specified as write, then a read from that port address will also effect a write, and vice versa, a write to a read port results in data being put onto the bus.