Chapter One
The Hardware

If you have used a BBC Micro for any length of time you will not have missed the fact that it is a very versatile machine. This versatility comes in part from its remarkably clever hardware design and in part from the extensive and well-designed resident software in the form of BBC BASIC, the assembler and the MOS (Machine Operating System). Even if your main interest lies in software, knowing something about the hardware that makes your BBC Micro 'tick' will help you to get the best from it. This knowledge needn't be at all detailed. It's not important to know what every chip does, only what the different general areas do and how they affect each other, and you can acquire this sort of knowledge even if electronics is not your subject.

In this chapter we take an overview of the BBC Micro's hardware by building up a block diagram. Some of the sections of the block diagram will be explained in more detail in other chapters where they are discussed in connection with the other side of the story - the software. Others are examined at length here. If you find any of the detail tough going then don't despair - simply skip the section or turn to a more software-oriented chapter and wait until you need to know about the particular subject that you found difficult before turning back to this chapter. There are many things inside the BBC Micro that don't make very much sense until you need to know about them!

The elements of a computer

There is nothing startling about the design of the BBC Micro. You won't find any powerful new microprocessor inside and its total memory capacity is limited to a fairly standard 64K bytes. What makes

the design special is not any single component or feature but the way a range of things have been brought together with a great deal of skill and forethought. The BBC Micro' sdesign is intricate rather than revolutionary.

If you look at Figure 1.1 you will see the parts that every computer, including the BBC Micro, has to have in one form or another to work. Although the BBC Micro adopts this traditional computer pattern there is something special to say about each part.

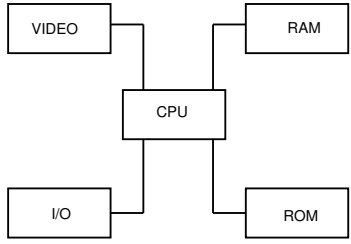

Fig. 1.1. Simplified diagram of the BBC Micro.

The CPU and memory

The CPU (Central Processing Unit) is a 6502 microprocessor of the same type that you will find in many older machines, such as the PET and the APPLE. However, in the case of the BBC Micro it is a double speed device - in other words it operates at 2 MHz. This is a great advantage that isn' twasted by surrounding the fast 6502 with slower memory - everything in the BBC Micro is built for speed! However, it is sometimes necessary for the 6502 to talk to slower devices and to make this possible it can run at half its usual speed - i.e. at I MHz.

The RAM is special for the same reason - it is fast. Traditional RAM will allow roughly one memory access every microsecond but the 4816 dynamic RAMs inside the BBC Micro are accessed four times every microsecond. As the 6502 - fast as it is - can only manage to use two of these memory accesses, you might be puzzled why the memory needs to be so fast. For the answer to this mystery we will have to wait for a discussion of the video section. Apart from speed, the 4816 dynamic RAMs are fairly standard 16K bit chips. So, for the Model A you need eight to make 16K bytes and for the Model B you need sixteen chips to make 32K bytes and this is the maximum amount

of RAM that can be installed. However, it's far from the maximum amount of memory.

The standard BBC Micro has 32K of ROM in the form of two 16K chips. This is a very large amount of ROM storage by current standards. One of the 16K chips holds a large program known as the MOS (Machine Operating System) which is responsible for co-ordinating the machine's functions and making some of its features easier to use. We will take a more detailed look at the MOS in Chapter Three. Even though the MOS is a large program, it only uses ASK of the 16K. Very little of the remaining 1K of the ROM is used. In fact, most of its address space is given over to memory mapped I/O devices, but this will become a little clearer when we look at the memory map in detail later on in this chapter. The second 16K ROM is most definitely all used as it stores the BBC BASIC interpreter and the 6502 assembler, which are both the subjects of later chapters. What is interesting about this ROM is that it can be replaced by any of three alternative ROMs under software control.

Inside the BBC Micro there are five 16K ROM sockets. One is used for the MOS ROM already discussed and the other four can hold various ROMs. However, only one of these four ROMs can be in use at any one time. For example, suppose one of the sockets held the BBC BASIC ROM (as is the case with nearly all machines) and another held a ROM for another language such as Pascal. Then, by using only software - in other words, without having to dive inside the machine each time - you can select which ROM is to be active and have either BASIC or Pascal. This idea of a number of ROMs sharing the same address space is known as paging. By using paging the BBC Micro can have as much as 5 X 16K (including the MOS) of ROM installed. The range of things that you could put in the four ROMs isn't limited to languages. You could install applications ROMs containing, for example, word processors, accounting packages etc. As well as being able to take 16K ROMs, the four paged sockets can also be used with four 4K ROMs to fill the full 16K or with two pairs of 8K ROMs providing only two alternative fillings of the 16K. However, these situations are not common.

Switching between any of the four ROMs involves the use of a register at FE30. This register is in fact a write only device, i.e. you can use it to change ROMs but not to find out which ROM is selected. If you open your BBC Micro then you will see a row of ROM sockets on the right-hand side of the edge nearest the keyboard. As mentioned

earlier, most systems will have only two of the sockets filled. The one on the far left contains the MOS and the one next to it contains the BBC BASIC. If you number the ROM sockets starting with the one that holds BBC BASIC as ROM 0 then you can select the ROM of your choice by writing its number to the register. So, if you want to select BBC BASIC, write zero to the register, If you want to select the ROM in the socket to the immediate right of the BASIC ROM, then you have to write a I to the register. This all sounds easy enough but beware that the ROM that you are deselecting isn't necessary for your program to run. In other words, don't deselect the BASIC ROM from a BASIC program!

The video section

The video section of the BBC Micro is perhaps the most interesting of all. Its workings will be examined in some detail in Chapter Four but it is worth sketching out its overall configuration here. The information to be displayed on the TV screen is stored in the machine's RAM. In display modes 0 - 6 the dot pattern for the entire display is stored in RAM. In other words every dot displayed on the screen is stored in the RAM. However, in mode 7 only, the ASCII code for each character displayed on the screen is stored in memory. The actual dot patterns that make up the shapes of the characters on the screen are stored in an additional ROM that is inaccessible to the 6502 and is used only in mode 7. Most computers produce their video display in the same way that the BBC Micro produces its mode 7 display.

There are two main parts to the video section - a standard 6845 video generator, and a very special custom-made chip called the video processor. The video generator chip shares access to the RAM with the 6502 processor; it uses the two memory accesses per microsecond that the 6502 cannot (see the previous section on the CPU). Thus, in every microsecond the memory access sequence is:

6502, video generator, 6502, video generator

The video generator has access to the RAM for the simple reason that this is where the information to be displayed is stored. However, the video generator doesn't handle or process any of this information - this task is reserved for the video processor! What it actually does is to

generate the correct sequence of addresses at the correct time, to ensure that the RAM gives up its information in the correct order and at the correct time. It has other jobs to do as well but they are all connected in some way with timing and organisation. For example, the video generator produces the regular part of the video signal in the form of line and frame sync pulses. The data that the RAM produces as a result of being addressed by the video generator is not at all suitable for direct conversion into a video signal. For one thing, it is produced eight bits at a time whereas a video signal needs one bit at a time. For another, the video signal needs colour information and this is coded within the eight bits. As already mentioned, the chip that takes the outputs from RAM and turns it into a video signal is the video processor chip. This not only converts the eight bits coming from memory into a serial stream but also decodes the colour information to produce a standard RGB (Red, Green, Blue) colour output, However, in mode 7 the video processor does very little. The ASCII codes stored in memory are fed directly to the character generator ROM which produces a standard RGB output all on its own!

The video section is a little complicated so it is worth summarising all the information in the form of a block diagram (see Figures 1.2 and 1.3). The thumbnail sketch of the way the graphics section of the BBC Micro works will be extended considerably in Chapter Four so don't worry too much if you find yourself wondering about the details. All will be revealed later!

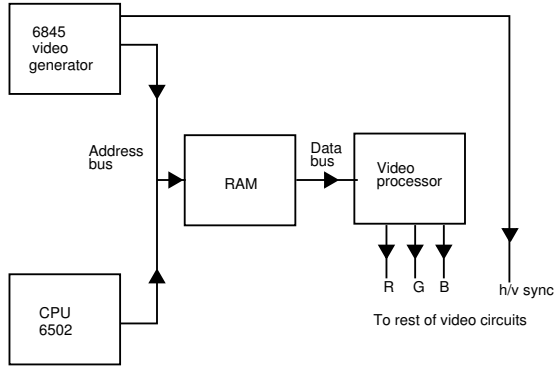

Fig. 1.2. Simplified block diagram of video section in modes 0-6.

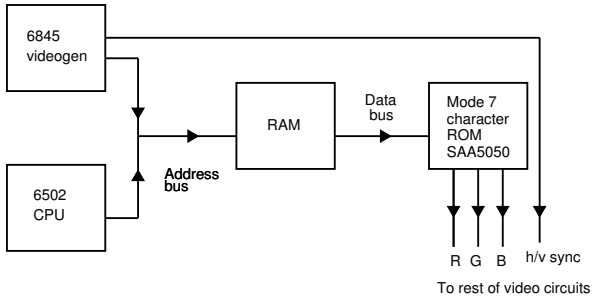

Fig. 1.3. Simplified block diagram of video sections in mode 7.

The interfaces

This is in fact a heading under which to gather together a wide range of different circuits! Some of these, such as the sound generator, the user port and the A to D convertor, for example, are dealt with at length in other chapters but it is worth producing a summary of all the interfaces circuits inside a standard BBC Micro. One thing that all the interface circuits have in common is that they use the 1K of address space not used by the MOS ROM.

The interfacing circuits within a standard Model A BBC Micro are:

1. Cassette and RS423 serial.
2. VIA(Versatile Interface Adaptor) - A. This is a parallel interface looking after internal devices such as the keyboard and the sound generator.
3. The 1 MHz extension bus.
4. The tube.

In the standard Model B machine we have to add:

5. VIA-B - a parallel interface that looks after two external ports, the centronics printer and the user port.
6. An A to D convertor (a mPD7002) that can be used as a general purpose measuring device or with joysticks.

There are other interface circuits that can be added to the BBC Micro beyond even these six, such as the disc interface, but these are of less

general interest and will be discussed in Chapter Nine. We will now take a look at each of the above interfaces, apart from the A to D convertor which is dealt with in detail in Chapter Six and is so separate that it adds little to our understanding of the overall machine.

The cassette and RS423 interface

Every BBC Micro comes equipped with a cassette interface. The interface also doubles as a general purpose serial interface. It is true that owners of the Model A cannot use this serial interface but this is only because the two buffer chips that provide the power to drive the serial output to the RS423 standard are missing. (The RS423 standard is simply an improved version of the older and better known RS232 or V24 standards. For our purpose it may be considered entirely compatible with both.)

The cassette interface on the BBC Micro relies on two major components. The first is a 6850 ACIA (AsynChronous Interface Adaptor) which is responsible for changing data from a parallel to a serial format and vice versa. This is all that is necessary for the RS423 interface (apart from the aforementioned buffering). However, the cassette interface works by recording two audio tones corresponding to the binary zeros and ones in the serial bit stream

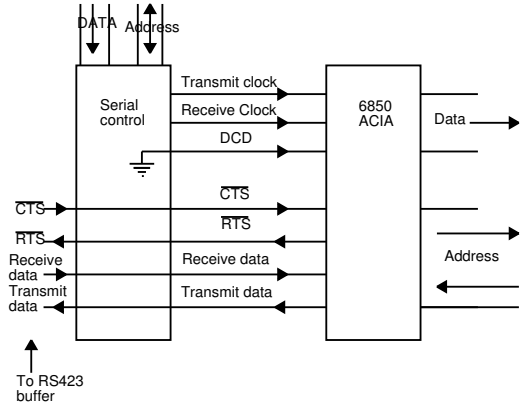

Fig. 1.4. Serial Interface set-up for RS423

produced by the 6850 ACIA. It is the second of the major chips in the cassette interface that is responsible for changing audio tones to bits. This is the second custom-made chip (the first being the video processor) in the BBC Micro. As well as changing bits to tones and tones to bits, it is responsible for providing the clock signals that determine how fast the ACIA receives and transmits (i.e. it sets the baud rate) and it selects where the ACIA should take its inputs and output from - the cassette or the RS423 buffers.

The simplest configuration is when the interface is set up to drive the RS423 serial port. In this case the only thing that the custom serial control chip does is to generate the transmit and receive clocks, and pass on all the input to and output from the 6850 ACIA. You can see this in Figure 1.4. To help anyone interested in using the RS423 interface the following table describes the function of each of the ACIA signals:

	ACIA *signal*	*Function*
$\overline{\text{DCD}}$	(No carrier)	Not used on the BBC Micro's RS423 interface.
$\overline{\text{CTS}}$	(Not clear to send)	A signal to the BBC Micro indicating when it is OK for it to send data.
$\overline{\text{RTS}}$	(Not ready to send)	A signal from the BBC Micro indicating that it is ready to receive data.

Connecting a printer, or anything else, to the RS423 port is very often a matter of getting the receive and transmit rates correct and deciding what, if anything, to connect to the control signals. The BBC Micro's side of the control signals is simple enough. To drive a printer the only two signals required are transmit data and clear to send (CTS). However, depending on which of the many available printers is used, the printer may need rather more signals than this to actually print anything! This is governed by the details of the printer's hardware, which should be explained in its documentation,

The situation is a little more complicated when the interface is set up to drive the cassette. In this case most of the ACIA's signals are intercepted by the serial controller. This can be seen in Figure 1.5. When recording data the serial controller switches on the cassette motor and then synthesises sine waves of the correct frequencies as the

serial data is fed to it from the ACIA. The only ACIA control line used in recording is the ready to send (RTS) line which enables (i.e.

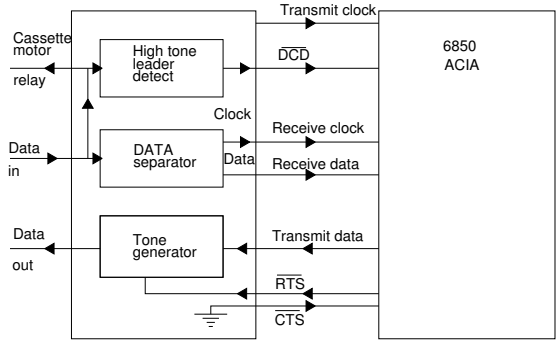

Fig. 1.5. Serial interface set-up for cassette.

switches on and off) the tone generator. The recording rate is set by the frequency of the transmit clock. On playback, things are just a little more complicated. First, the high continuous tone (2400 Hz for 5 seconds) is detected by a special circuit, the high tone leader detect. When this is detected, the data carrier detect (DCD) line is set to enable the ACIA to receive data. That is, before the high tone is detected the ACIA will not accept any data that might be coming from the tape and hence through the serial controller, The main part of the playback circuit, however, is the data separator. This accepts the recorded tones from the tape recorder and changes them into a stream of zeros and ones suitable for the ACIA to convert to a parallel form. As well as detecting whether the input tone is high or low, the data separator generates a clock signal that is used as the receive clock to the ACIA. The advantage of this is that it takes into account any changes in tape speed that might have occurred between recording and playback.

This description of how the serial interface works is all very well but how do you select which of the cassette or the RS423 interface is to be active? How do you select the transmit and receive rates? The answers to these questions depend on knowing about certain control registers in the interface area of memory. The serial interface has three control registers - two belong to the ACIA and one belongs to the serial controller. The addresses where these and other registers can be found are given in the section below on the BBC Micro' smemory map but for the sake of completeness the addresses and functions of each register are given below:

Address (in Hex)	*Function*
FE08	ACIA control and status.
FE09	ACIA receive and transmit;
FE10	Serial controller's register.

The two ACIA registers are rather strange in that their function depends on whether you are reading or writing them. For example, the register at FE08 is a control register when you write to it and a status register when you read it! If you think about it this makes a great deal of sense - why should you want to read a control register and write a status register? The function of the bits in the ACIA's control register can be seen in Figure 1.6. For normal use, bits 7,6,5,1 and 0 should be

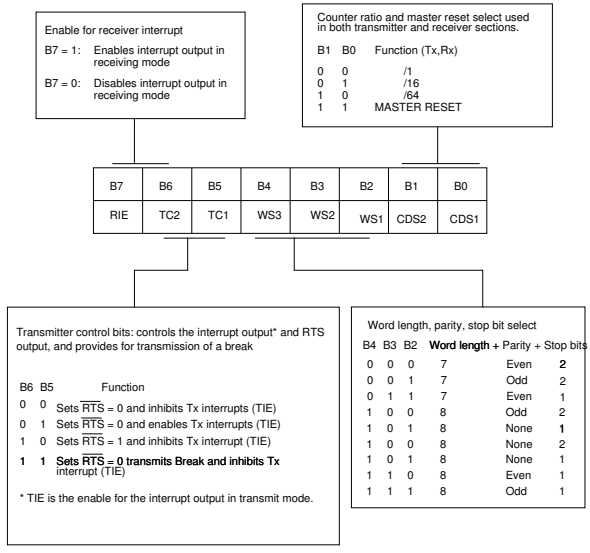

Fig. 1.6. ACIA Control Register format.

left as the BBC Micro sets them, i.e. use the appropriate FX commands to set the baud rates and generally initialise the register. The only bits that might need altering are bits 4,3 and 2 to set the required serial word format (for more information, see the VDU example in Chapter Eight). The meaning of the bits in the ACIA' sstatus register can be seen in Figure 1.7. The data transmit and receive registers can be read

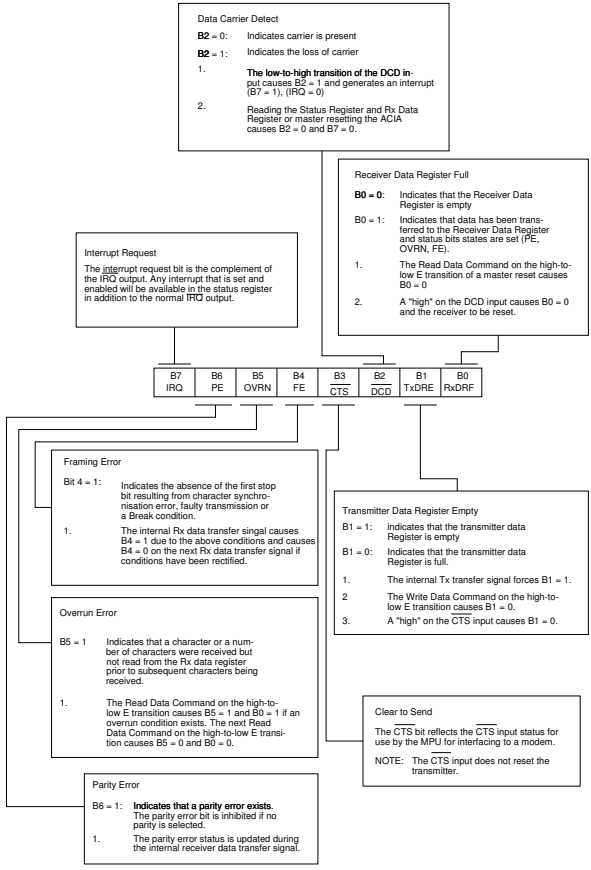

Fig. 1.7. The ACIA Status Register format.

and written as required. If bit 0 of the status register is 1 this means that the receive register contains a character which should be read before the next character is received and hence overwrites it. If bit 1 of the status register is I a character may be placed in the transmit register to be transmitted bit by bit. If bit 1 is 0 then you shouldn' twrite anything to the transmit register because this would overwrite the character currently being transmitted. The other bits in the status register are concerned either with telling the user about.errors that have occurred or with the status of the external device.

The final register of interest is the serial control register and its format can be seen in Figure 1.8. Bit 7 controls the cassette motor

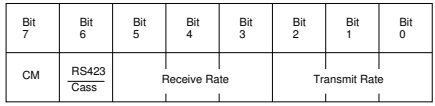

Fig. 1.8 The serial control register

relay. If it is 1 then the relay is closed and the motor is on. To see this, try?&FE10=k80 which will switch the motor on and?&FE10=0 which will switch it off again. Bit 6 controls which of the cassette or the RS423 interface is selected - 0 selects the cassette and a I selects the RS423. The final six bits work in two groups of three to control the baud rate for receive and transmit. These work as indicated in Table I. I below.

Table 1.1. Baud rates produced by bit patterns in the serial control register.

Baud rate	*Receive bit* 3 4 5	*Transmit bit* 0 1 2
75	1 1 1	1 1 1
150	1 1 0	1 1 0
300	1 0 1	1 0 1
1200	1 0 0	1 0 0
2400	0 1 1	0 1 1
4800	0 1 0	0 1 0
9600	0 0 1	0 0 1
19200	0 0 0	0 0 0

Notice that the order of the bits is the reverse of what you would normally expect.

A lot of information has been introduced in this section without an example of how to use it. This is because direct access to the serial interface is best done via the 6502 assembler and this is not discussed until Chapters Seven and Eight. If you cannot wait that long, turn to Chapter Eight where you will find a program that turns the BBC Micro into a VDU making use of much of the information in this section.

The VIAs

A fully expanded machine contains two VIAs (Versatile Interface Adaptors). VIA-A is used for internal tasks and VIA-B is dedicated to user-defined tasks. A VIA is fundamentally a parallel interface but it has many other functions and capabilities. For example, it contains two independent timers and a serial shift register. In fact it rivals the 6502 itself in its complexity! Because it is such a complicated and versatile device a large part of Chapter Six is devoted to it. As Chapter Six is mostly concerned with VIA-B, the user's VIA, it is worth taking a brief look at what VIA-A is doing. Without knowledge of how a VIA works this is necessarily an incomplete description but the information in Chapter Six rounds off the picture.

VIA-A is used by the BBC Micro to interface the keyboard, the sound generator, the A to D convertor, and the optional light pen, and VIA-A's timer is used by BASIC to provide the variable TIME. It is also used to handle hardware scrolling, a topic which is dealt with in Chapter Four. There are two ten-bit ports in every VIA, usually referred to as port A and port B. Eight bits of each port can be individually selected to be either inputs or outputs. The remaining two bits are a little more restricted in their use and are normally kept for special purposes. The A side of the VIA handles the keyboard and the B side handles the 'odds and ends',

The keyboard is connected to the eight data bits on port A. One of the two special bits - CA2 - is also used to detect when a key is pressed. The keyboard has two modes of operation - free-running and program-controlled. When the keyboard is free-running, each key is repeatedly scanned in turn until a key is pressed when a signal (an interrupt) is sent by CA2 to the 6502. This causes the 6502 to stop whatever it is doing and 'payattention' to the keyboard. When this

happens, the keyboard is switched to its programmed mode of operation and the keys are scanned one by one, using the eight A side data lines until the key that is pressed is found. To understand this a little more fully you have to know the layout of the keyboard, which is presented in Figure 1.9. Being arranged into a matrix consisting of 10

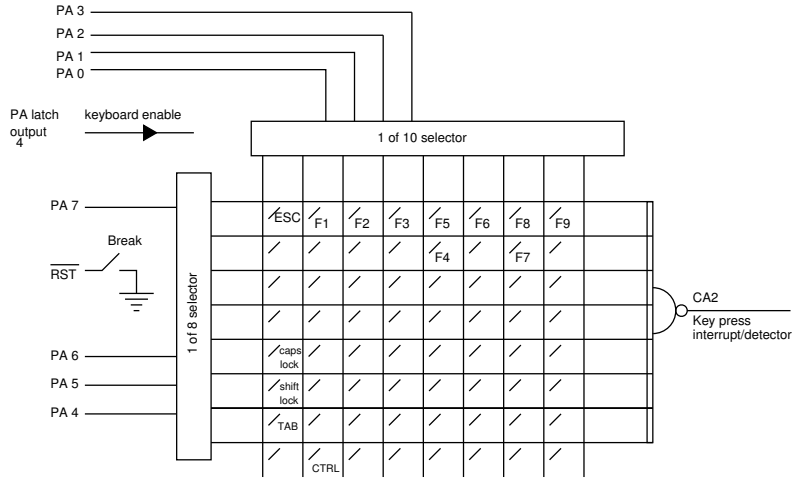

Fig. 1.9. The keyboard in program mode.

columns by 8 rows means that any column can be selected using four bits and any row can be selected using three bits. The column number is specified using bits 0-3 and the row number is specified using bits 4 - 6 (see Figure 1.9). In addition, bit 7 is used to control access to the keyboard. The second special bit on the A side ' of the VIA is connected to the vertical sync signal for timing purposes. The important thing to notice from this description of the keyboard is that, as the keyboard causes an interrupt every time a key is pressed, keyboard service can occur at any time. In particular the ESCAPE key is just like any other key, i.e. it doesn' tuse any special *reset* function - in this sense any key could be used to interrupt program operation. However, the BREAK key is connected to the 6502' sreset line and is different from the other keys, causing a hardware reset when it is pressed. It is possible, however, for the machine to tell the difference between a reset that occurs when the machine is first switched on - a *cold reset* - and a reset caused by pressing the BREAK key - a *warm reset*. This distinction allows the machine to decide whether it is worth taking any notice of the memory contents, in particular user-definable

key definitions. Even though it is different from the other nine function keys, this means that the BREAK key can be associated with a *KEY definition. The BREAK key always causes a reset but, following a warm reset, the machine obeys any instructions assigned to *KEY 10, i.e. the BREAK key.

The B side of the VIA has a number of jobs to do. The two special bits CB I and CB2 are used to detect the end of conversion of the A to D convertor and a signal from the optional light pen respectively. Bits 0 to 3 are fed into a 74LS259 addressable latch which provides eight different outputs. The first three bits, i.e. 0 to 2, determine which output from the latch is affected. Thus, 000 alters the first output, 001 alters the second ouput, and so on. Bit 3 determines the state that the selected output takes. That is, if bit 3 is I the selected output changes to 1. To describe what the latch does involves describing the effect of each output. The first output enables the 75489 sound generator. The second and third are connected to the optional speech synthesiser. The fourth is used to enable the keyboard, i.e. to load column and row numbers. The fifth and sixth are used in the hardware scrolling and are set according to the display mode that the BBC Micro is in. The seventh and eighth drive the caps lock and shift lock LEDs. What is interesting is the way the latch is used to select any combination of a number of devices. The sound generator chip and the optional speech synthesiser are both connected to the A side data bits 0 to 7. You can think of the A data side as a *slow data bus* communicating with whichever device is selected by the addressable latch. This means that only one of keyboard, sound generator and speech synthesiser can be activated at any one time. The final four bits of side B are used a little more simply. Bits 4 and 5 are used as ' fire button' inputs from the analog connector. Bits 6 and 7 are used to control the optional speech synthesiser.

This just about finishes the description of VIA-A (for a summary see Figure 1.10) - except to remind the reader that it also provides the timing function for the BASIC variable TIME, (In fact it uses timer 1 but this will be explained in Chapter Six).

The 1 MHz bus

The 1 MHz bus is not so much an interface, but is more a way of connecting other interfaces to the BBC Micro. The reason why it is

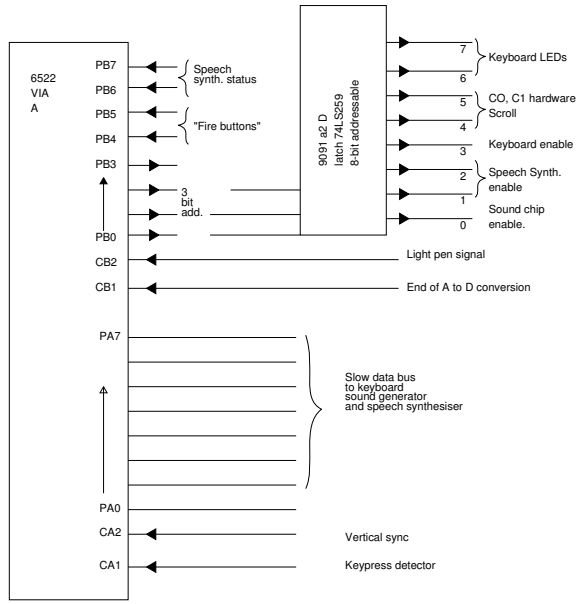

Fig. 1.10. The use of VIA-A

called the '1MHz' bus is that the speed at which the 6502 normally works, i.e. 2 MHz, is too fast for most standard components so the clock which governs the speed of access is slowed to I MHz when the 6502 is using addresses that correspond to the 1 MHz bus.

The bus itself isn't a collection of all the address and control lines that the BBC Micro uses internally, as is the case with the expansion buses of most other micros. Instead it only includes the full data bus, the eight low address lines A0-A7, and a few control lines. The presence of A0 to A7 means that the bus can specify any one of ¼K of address locations - but which address locations? If you are looking only at the low eight bits of the address then 0045, say, looks exactly the same as 3545, because the low eight address bits are the same in both cases. To solve this problem, the 1 MHz bus includes two

extra addressing lines - NPGFC and NPGFD. These are normally high, at logic 1, but if an address beginning with FC is used NPGFC goes low and if an address beginning with FD is used NPGFD goes low. You might be able to see now why these two lines have such long names - NPGFC stands for 'not page FC' and NPGFD stands for 'not page FD' Obviously, if you connect anything to the 1 MHz bus and use a combination of NPGFC and A0 - A7 to enable it, it has an address in the range FC00 to FCFF. If you use NPGFD and A0-A7 then it has an address in the region FD00 to FDFF. What this means is that you can connect external devices, running at 1 MHz to the BBC Micro, using either of the above range of addresses. However, Acorn have already suggested uses for most of this address space in their Application Note No. 1 - The 1 MHz Bus which can be obtained from Acorn. Only addresses between FCCO to FCFE are marked for 'user application' but this should be enough for most purposes.

This short sketch of the 1 MHz bus in insufficient if you are interested in connecting your own equipment to the BBC Micro, In particular be warned that there is a slight timing problem to be solved before any equipment will work reliably. Details of this and other features of the 1 MHz bus can be found in the Application Note mentioned above which should be obtained by anyone considering interfacing to the 1 MHz bus.

The tube

The tube interface is superficially like the 1 MHz bus in that it supplies a subset of the address lines - AO to A6 and an extra address line called the tube. There are two differences, however. First, the tube works at the full 2 MHz and secondly it is claimed for exclusive use by Acorn products. There is no real reason why the tube couldn't be used as a fast version of the 1 MHz bus but it is likely to be of much more use as originally intended. For one thing, Acorn have produced a custom chip that can be used to pass data between the 6502 in the BBC Micro and other processors very quickly and in a standard format. Using only seven address lines means that the tube can only be used to address 128 distinct locations. The tube address control, when used in combination with AO-A6, places the tube at FEE0 to FEFF.

The whole machine - a block diagram and the memory map

After studying the various parts of the BBC Micro you should be able to make something of Figure 1.11 which shows the areas of the machine in roughly the same position that they are placed on the printed circuit board. Some of the extra devices, such as the disc controller and the speech synthesiser, have been left out for clarity.

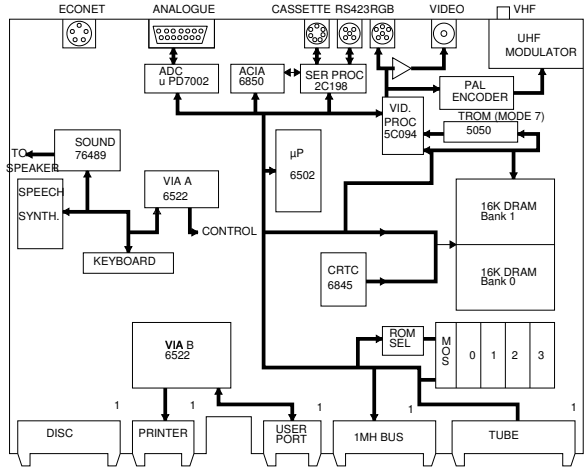

Fig. 1.11. Complete block diagram.

It is also worth drawing together all the information on the position of various things in memory. A complete memory map can be seen in Figure 1.12. The RAM area (16K or 32K depending on model) is used for general system storage, BASIC programs, and the screen displays. More details will be given of RAM usage in the relevant chapters. Notice the four paged ROMs starting at B000; remember that only one is actually present in the memory at any one time and that which one it is is controlled by the ROM select latch. The area of greatest interest is the 1K I/O area at the top of the memory. This can be seen in greater detail in Figure 1.13. The lower ½K of this area is used by devices connected to the 1 MHz expansion bus, as was

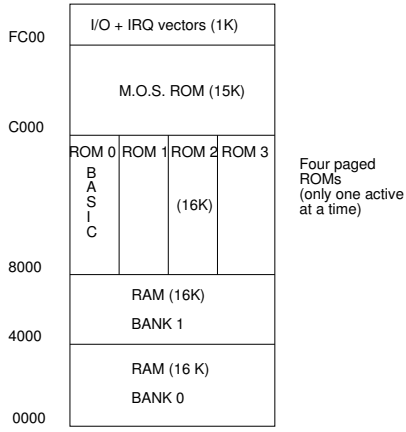

Fig 1.12. Complete memory map.

described earlier. The 128 bytes starting at FE00 are used by internal I / 0 devices, many of which have already been discussed. The addresses given on the left hand side are the start addresses of any registers .that the device might have. The details of the control and status registers of each device will be given in the chapters where they are discussed more fully. The exceptions are the FDC - Floppy Disc Controller - and the ADLC - Advanced Data Link Controller - which are not part of the standard BBC machine. Notice that the details of the serial controller and the ROM select have been given in this chapter.

Knowing about hardware

If you know the details of the hardware of a particular computer then the temptation is to make use of it! In other words, once you know about hardware, easy ways of doing things and even new things to do often occur to you. Now with most machines this is a very acceptable way to proceed but with the BBC Micro there has to be a note of caution. The BBC Micro is intended to be the start of a very advanced

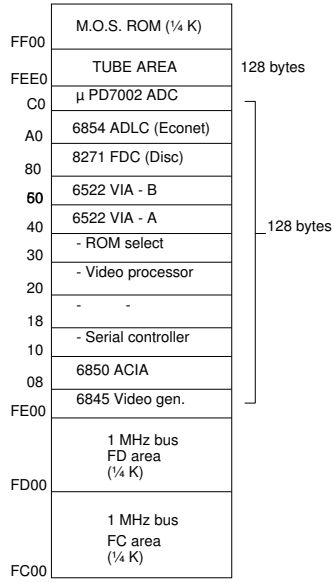

Fig. 1.13. I/O area of memory

system. In particular the tube can be used to connect other processors. If you have written a program that runs on a standard BBC Micro by directly ' fiddling' with the hardware, then this program is hardware-dependent. If the design of the BBC Micro changes, even slightly, then the chances are your program will not work. In particular, your program will not work on a second processor connected over the tube. This may, however, not worry you too much. After all, if the program works on your BBC Micro, why worry? However, it is something to keep in mind when producing programs that you hope will be useful to other people.

To help with this problem, the BBC Micro' sMOS provides a number of standard routines to enable you to modify memory and I/O locations. In addition there are also routines that provide standard ways of dealing with the internal devices. Acorn suggest that if these routines are used instead of direct access to the device or location then your programs will run on a modified BBC Micro, even on the second

processor! You will find an example of using the MOS to deal with I/O devices in Chapter Six.

A second point to be aware of, and perhaps even beware of, is that the BBC Micro makes extensive use of interrupts in its operation. This makes it a much faster and more flexible machine but it can make it more difficult to program at the machine code level if you need to make use of interrupts yourself.

Conclusion

This chapter has presented an overview of the BBC Micro's hardware and looked in detail at some of the features that are not covered extensively in later chapters. Don't worry if you have found parts of this chapter hard to assimilate. Looked at in isolation, hardware is difficult to understand unless you are well-versed in electronics. In some ways, this chapter is meant to be treated as a reference section and you'll find the information it contains will become much easier to understand when you encounter a situation that actually requires it. So if you carry on exploring aspects of the BBC Micro for yourself you'll find yourself returning to this chapter time and again.