Chapter 1

Overview Of A Micro Computer

The typical computer consists of the following parts:

1. The Central Processing Unit or CPU, which does the actual processing, arithmetic, and decision-making of the computer, and also controls the operation of the rest of the computer.
2. The memory, which stores programs being performed, as well as data and results.
3. Input and output equipment (also called I/O devices), which includes things like printers, keyboards, and the like. Furthermore, there are circuits called *I/O interfaces* which connect the I/O devices to the CPU and memory. (This definition leaves the status of things like disk drives and tape drives a bit hazy - some people would include these in the memory category, and some in the I/O category. We will use the latter.)

With this breakdown in mind, we can look at the types of computers. Historically, as well as in size order, computers can be broken down into four types:

a. Mainframe computers are the very large ones, which often occupy an entire room (perhaps even a very *large* room). Originally, of course, all computers were this large; these days, computers like these are used by the Internal Revenue Service, large corporations, or universities for commercial or research applications. In a typical mainframe computer, the CPU might be in one floor-standing cabinet, the memory in another, I/O interfaces in another, and the actual I/O devices in a few more.
b. Minicomputers are smaller and newer, usually occupying one or more cabinets standing on the floor, but small enough to fit into an ordinary office or laboratory, along with other equipment. It too could be used for business or research applications, but is not as powerful as the very

large mainframes. In a typical minicomputer, The CPU might be one or more printed circuit boards, the memory might be a few more, and each I/O interface might be one or more pc boards as well. But all of these might be mounted in the same cabinet.

c. Micro computers are generally quite small, usually in desk-top cabinets, and less powerful than either of the above. But, given enough memory, these too can be used for both business and research applications. In most cases, the CPU is just one or two integrated circuits (*IC*s) occupying just part of a printed circuit board, which might even contain some memory or other circuitry as well.

d. Microcomputers (notice that we are now spelling it as one word, not two) are the smallest of the lot. In a typical microcomputer, a single integrated circuit contains the CPU, some memory, and even some of the I/O interfaces as well. In fact, the entire IC might be called a microcomputer. They are almost used strictly for control applications - for example, you might find one inside a camera or inside a traffic light.

The 68000 Microprocessor (also called a Micro Processing Unit, or MPU) would be the CPU in a Micro Computer. Actually, some of the 68000's faster cousins are powerful enough that they can do jobs often reserved for minicomputers in the past. And so the distinction between the minicomputer and the micro computer is becoming blurred as time goes on. In fact, some people believe that the minicomputer may even cease to exist, as micro computers take over many of their jobs.

The SK68K Computer Trainer

As shown in the photograph of Fig. 1-1, the SK68K microcomputer trainer is all contained on one printed circuit board. It contains the following:

a. The 68000 microprocessor is the large integrated circuit near the bottom.

b. The main memory consists of the 32 small integrated circuits in the bottom left corner plus the four medium size IC's near the center. The 32 ICs in the corner provide 1 megabyte of dynamic RAM (the main random-access memory); the four IC's in the center provide 4K (4 kilobytes) of static RAM with battery back up (the two slightly smaller ICs), and 32K of ROM (read-only-memory) with space for more (in the two slightly larger ICs).

c. The I/O interfaces are located mostly along the top edge and the top right of the board, and include 4 serial ports to connect to terminals, modems, printers, etc., two parallel ports for printers or other I/O devices, a floppy disk interface for up to four drives, a sound interface for a speaker, a clock/calendar chip, an interface for a PC or XT clone keyboard, and six interface connectors for additional clone-compatible I/O boards. (The clones we are talking about are the Japanese, Taiwanese or Korean computer components often known as IBM-compatible because they are interchangeable with those of IBM PC- and XT-style computers.)

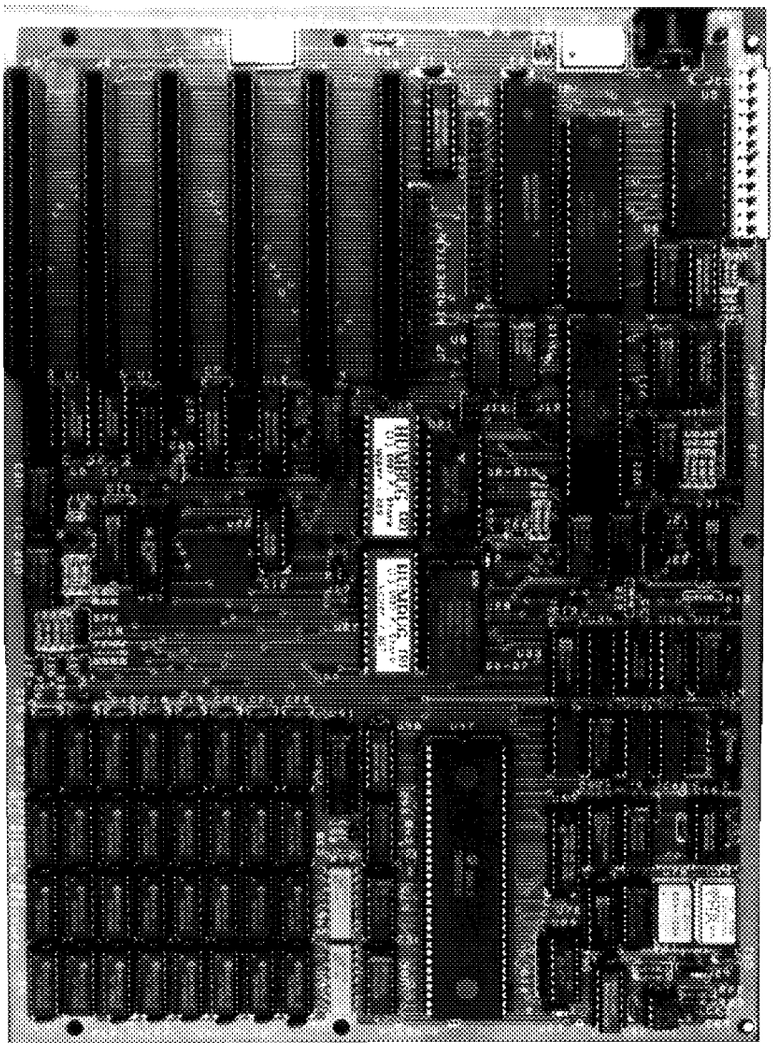

Fig. 1-1. The SK68K Printed Circuit Board.

d. Finally, the rest of the board contains various smaller ICs used for timing and control, and to interconnect the remaining parts together. In fact, these remaining ICs are often called *glue chips* for that reason.

The fact that the SK68K has connectors to accept XT-compatible clone components such as keyboards, video boards, and hard disk controllers, makes the SK68K somewhat unique. It allows us to expand the Trainer and give it additional capabilities, and do so at a very low cost by taking advantage of the fact that such clone components are produced in very large quantities and are therefore very inexpensive.

The Block Diagram

The best way to get an overall view of the SK68K trainer and how it works is to start with the block diagram in Fig. 1-2. In general terms, this diagram describes any microcomputer, not just the SK68K.

The heart of the diagram is the microprocessor, a Motorola 68000 in our case. It is driven by a clock, which is nothing more than a high frequency oscillator which generates a square wave. In the SK68K, this clock will most likely be an 8 MHz signal, though it could go as high as 16 MHz. The clock synchronizes everything occurring in the system so that it occurs at a fixed speed.

The right side of the diagram contains three essential parts: ROM, RAM, and I/O interfaces. ROM and RAM are both part of the computer's memory, but ROM can only be read (hence, its name read-only-memory or ROM), meaning that the data and programs in the ROM were placed there once the factory and can now be used by the trainer but not changed. RAM, on the other hand, is read-write- memory (somewhat misnamed as RAM rather than RWM - but then, have you ever tried to *pronounce* RWM?) The Trainer can write (store) data or programs in RAM, can later read them back, and can also change them at any time. Furthermore, ROM is permanent even when the power is turned off, whereas RAM is erased when power disappears (unless special precautions are taken, such as providing a small battery to keep the RAM powered up even when the rest of the system is shut off. Such memory is called *battery backed-up*.)

The ROM

The ROM in the SK68K system consists of two 28-pin ICs called EPROMs or Erasable Programmable ROMs. If you purchased these from an electronic distributor, they would be empty or erased. But the two EPROMs which come with the SK68K kit have been programmed with a copy of the HUMBUG debugging program and with a Basic translator. The

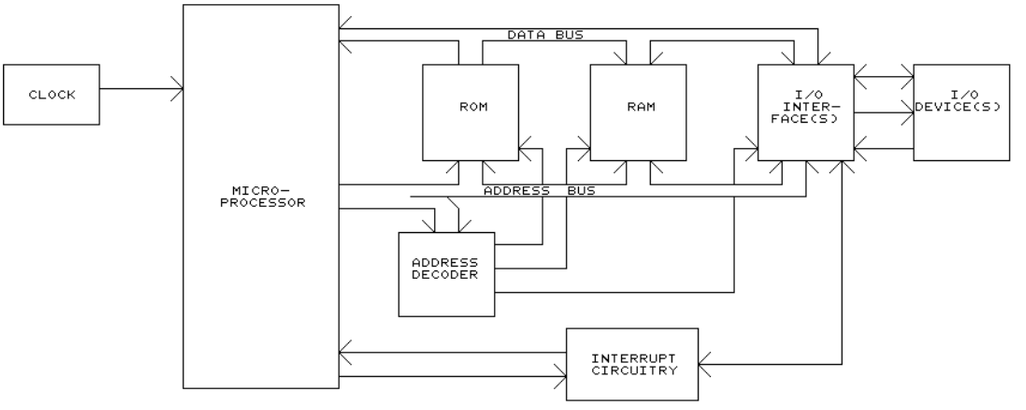

Fig. 1-2. Typical computer block diagram.

computer can read and use these programs, but cannot erase or change them.

The RAM

The SK68K computer's RAM consists of two parts - static RAM, and dynamic RAM.

Most computers would generally have either static RAM (abbreviated SRAM) or dynamic RAM (DRAM), but not both. We use both because they each have their advantages and disadvantages. For large memories, dynamic RAM is cheaper and smaller - without it, it would not be practical to provide 1 megabyte of static RAM at a reasonable cost. On the other hand, for small memories static RAM is the right choice because it is much simpler - and easier to debug in case of problems! Hence the SK68K computer will first be built with a small amount of static RAM using just two integrated circuits. Since the static RAM circuitry is so simple, it will probably work immediately without any problems, giving us the ability to run Basic and HUMBUG. Once the static RAM is all working, then we can add the dynamic RAM, consisting of thirty-two 256K dynamic RAM ICs plus a batch of support ICs. If there is a problem, we can use HUMBUG to debug the dynamic RAM. This kind of bootstrapping makes the building of a large system like the SK68K from scratch a lot more practical.

There is actually an ulterior motive to providing static RAM - to provide a clock/calendar we need only unplug one of the RAM ICs and substitute a clock/calendar IC which is totally pin compatible. We will use the MK48T02 which provides not only a clock and calendar, but also some static RAM of its own - and a built-in battery to keep the clock and RAM going while the computer is turned off.

I/O Interfaces

Although Fig. 1-2. shows just a single box labelled I/O Interfaces, the SK68K computer's I/O is actually quite complex. It consists of two MC68681 DUARTs to provide four serial interfaces, one 68230 parallel interface/timer, a 1772 floppy disk controller, keyboard interface, speaker interface, a number of extra support ICs, all of the circuitry needed to interface to the six PC-compatible interface connectors, plus the interrupt circuitry shown at the bottom, which allows I/O devices to interrupt the 68000 when they need it; the latter is absolutely essential for using a clone keyboard.

Some microcomputers often also provide DMA or Direct Memory Access circuits. This is a feature which is often used when the CPU has difficulty keeping up with fast I/O devices such as disk drives. Since the 68000 does not have any problems keeping up (and DMA really complicates the computer), we chose not to use it in the SK68K computer.

The Data Bus

As Fig. 1-2 shows, the two main sets of connections between the microprocessor and the ROM, RAM, and I/O interfaces, are the data bus and the

address bus. The term bus is used to signify that a number of parallel wires are used to carry data simultaneously.

The data bus is used to move numbers (which could be numeric data, instructions, or text) between the microprocessor, memory, and I/O devices. If you look at the arrowheads at the ends of the data bus in Fig. 1-2, you will see why the data bus is said to be bidirectional. Data may go in either direction - left or right.

In our case, the data bus consists of 16 wires, each of which carries one bit or binary digit. (See Appendix B if you are not that familiar with binary numbers and bits.) Thus the 68000 can transfer a 16-bit number to or from the microprocessor all at once. As we will see, however, the 68000 handles numbers in chunks of 8 bits (called a byte), 16 bits (two bytes, also called a word), or 32 bits (four bytes, also called a long word.) When transferring a byte, the 68000 uses only half of the data bus; when transferring a long word, it uses the data bus twice, transferring 16 bits at a time.

The number of bits on a data bus - also called the width of the bus - obviously has a bearing on the speed - the wider the bus, the more bits can be moved at a time, so the faster the computer runs. But there is more to the story - the size of numbers that can be handled internally in the microprocessor is also important.

The very first general-purpose microprocessors - the 8080, 6800, 6502, and Z-80 - had an 8-bit data bus and also handled 8-bit numbers internally. For this reason these were called 8-bit microprocessors.

The next generation of chips, such as the 6809 and 8088, still had 8-bit data buses, but could now handle 16-bits internally. This gave them extra power, but they were still bogged down by the slow speed at which they could transfer data to and from memory and I/O devices.

The next step up included the 8086, 80186, and 80286, processors which could handle 16-bit numbers both internally and externally, and which are called 16-bit processors.

The 68000 is one step higher yet - it still only has a 16-bit data bus, but can handle 32-bit numbers internally.

Finally, at the top of the current pyramid are the 80386 and 68020, both of which handle 32-bit numbers both internally and on the data bus. These are true 32-bit processors.

But even this is not the entire story - there are still other factors which affect computer speed. Though there are processors which have a wider data bus than the 68000, a bus that's twice as wide doesn't necessarily mean a computer that's twice as fast unless you consistently run programs that make full use of that width. What does make the 68020 and 80386 faster than the 68000 or 8086 is their more extensive use of a cache. This is an area of memory within the processor that holds instructions or data that are read out of memory before they are needed. Whereas older processors would only read data out of memory at the instant it is needed - and then have to wait for it - newer processors may spend their spare time pre-reading a few bytes ahead of themselves, and store the bytes read just in case they should be needed in the next few instructions. Alternatively, they may store instructions that have been recently used in case they are needed again soon. In this way they avoid the need to wait for data or instructions to come in

from memory. The 68000 does have a small cache, but it is too small to provide a significant saving.

The Address Bus

The other major bus, the address bus, carries addresses. That is, in order to save data into memory, or read data from memory, the processor must specify exactly where in memory that data is located. This is done with a numeric address, sent out on the address bus. As you can see from the arrowheads in Fig. 1-2, the address bus carries data from left to right; that is, it is unidirectional. (There is an important exception - when a computer uses DMA, then addresses may come out of an I/O interface and travel to the left.)

The width of the address bus determines exactly how much memory a computer can have. If the bus had only three lines, for example, then each address would consist of just three bits. Since each bit can only be either a 0 or 1, there would be just eight possible addresses - 000, 001, 010, 011, 100, 101, 110, or 111 - since there is no other three-bit number that can be made out of ones and zeroes. Hence the maximum number of addresses - or put another way, the maximum possible number of locations in the memory of this computer - would be eight, which also happens to be equal to two to the third power. That is, $2^3 = 8$.

In general, the maximum number of addresses is 2 to the same power as the number of address lines. For example, most 8-bit computers have 16 address lines in their address bus, so they have a maximum of $2^{16} = 65536$ addresses.

Since a K in computer terms is 1024 (not 1000 as in ordinary electronics), 65536 works out to be exactly 64K locations.

Newer microprocessors have more address lines than their predecessors:

microprocessor	address bus width (bits)	maximum memory size
8080, 6800, etc.	16	64K
8088, 68008, etc.	20	1 megabyte
68000	24	16 megabytes
68020, 80386	32	4 billion bytes

As you might expect, there is more to the story than just the width of the address bus. Consider the 20-bit bus of the 8088 and 68008, for example. Both of these processors can address up to a megabyte of memory, but the 68008 (the smaller cousin of the 68000) can do so in one continuous piece, whereas the 8088 must split that memory into 64K segments. Handling the segmenting greatly complicates a program - that's why many programs written for the 8088 (such as Microsoft's BASIC or BASICA) can only use 64K of memory at a time, whereas a Basic on the 68008 has no such limitation.

Thus the 68000 can easily handle programs and data that use up the entire 16 megabytes of memory ... almost. There is a difference between the

way that Intel and Motorola processors handle I/O. In a computer using a Motorola processor like the 6800 or 68000, the I/O interface connects to the processor in exactly the same way as the memory, with the result that memory and I/O use the same addressing scheme. Thus if the 68000 were to use 1 megabyte to address I/O, then there would only be 15 megabytes left for memory. Intel processors do not have that limitation - they use the entire normal address range for memory, and have a separate set of addresses (usually much smaller) just for I/O. Although some people point this out as a weakness in the Motorola approach, in practice it makes very little difference since I/O seldom requires more than just a few dozen (or perhaps a few hundred) addresses. There are still plenty of addresses left for memory. In most cases, a 68000 or 68020 has so many possible addresses that we can afford to waste thousands - maybe even millions - of addresses on I/O without feeling the pinch.

A list of addresses in a computer and what they are used for is called a memory map. Table 1-1 shows a simplified memory map of the SK68K computer. 1 As you can see, there is still plenty of memory left for expansion, probably a lot more than most of us would care to pay for.

Table 1-1. Simplified SK68K Computer memory Map

Memory Range (hex)	Description
000000 - 0FFFFF	Dynamic RAM (1 megabyte)
100000 - BFFFFF	Empty - for expansion (11 megabytes)
C00000 - DFFFFF	Addresses for PC expansion slots (2 megabytes)
E00000 - F7FFFF	Unused (1.5 megabytes)
F80000 - F9FFFF	ROM (128K)
FA0000 - FBFFFF	Addresses for PC expansion slots (128K)
FC0000 - FDFFFF	Unused (128K)
FE0000 - FE3FFF	I/O Interfaces (16K)
FE4000 - FEFFFF	Unused (48K)
FF0000 - FF7FFF	Static RAM (32K)
FF8000 - FFFFFF	Unused (32K)

The Address Decoder

As Fig. 1-2 shows, the address bus coming out of the microprocessor is split into two parts - part goes into the address decoder, while part goes to the ROM, RAM, and I/O interfaces.

The job of the address decoder is to look at the address on the bus and decide whom it's intended for. For example, as Table 1-1 shows, the dy-

1 Table 1-1 shows memory assignments, but some of these may not be used. For example, 32K is assigned to static RAM, but only 4K is actually installed.

namic RAM occupies addresses 000000 through 0FFFFF. Whenever the address decoder sees any address beginning with the hexadecimal digit 0, it recognizes it as a RAM address, and sends a signal to the RAM that effectively says “Hey, you! This address is meant for you ... go to work." This signal is called an enable or select signal. If it goes directly to an IC, then it is called a chip enable or chip select, often abbreviated CE or CS.

Fig. 1-2 shows just one address decoder, connected to the ROM, RAM, and I/O interfaces. In practice, though, most computers split that address decoder into two or more smaller decoders, each of which services just one part of the computer. Part of the reason is that it is easier to build that way, but there is a second reason as well - not all the decoders look at the same part of the address bus.

In the case of dynamic RAM, the address decoder need only look at the leftmost hex digit of the address; that is, it looks at the four leftmost bits, which must equal 0000 (a hex 0) for the RAM to go to work (see Appendix B for a discussion of binary and hexadecimal digits if you need to brush up.)

The decoder for the ROM, on the other hand, must look at seven bits. As Table 1-1 shows, the ROM occupies addresses F80000 through F9FFFF. Since there are other parts of the computer whose addresses also begin with the hexadecimal digit F, the ROM’s address decoder must look at more than just the first digit F - it must also check that the second digit is either an 8 or a 9. This is done by looking at individual bits of the address.

When written in binary, the lowest ROM address - F80000 - begins with the bits 1111100 and then continues with 17 zeroes, like this:

F80000 = 1111 1000 0000 0000 0000 0000.

The address F9FFFF also begins with 1111100 but then continues with 17 ones, like this:

F9FFFF = 1111 1001 1111 1111 1111 1111.

All other ROM addresses also begin with the bits 1111100, but have different combinations of 17 zeroes and ones at the end. Thus any address which starts with the bits 1111100 applies to the ROM; the ROM’s address decoder therefore looks for a 1111100 bit pattern in the first seven bits of the address, and sends an enable signal to the ROM as soon as it sees it.

Hence different parts of the address decoder look at different bits of the address bus. Some parts may only look at one or two bits, other parts may look at four or six, and some parts of a typical computer’s address decoder may look at 16, 32, or even more bits in some computers.

Conclusion

We conclude this chapter by just reviewing that the typical computer consists of a CPU (called a microprocessor in a micro computer), some memory (both ROM and RAM), I/O devices and their I/O interfaces, and various other circuitry made up of glue chips. And let us not forget the subject of the next chapter - the power supply.