300-76-0001- 1
P/N 000015
AUGUST 1976

SPHERE 300
CORPORATION

OPERATOR AND REFERENCE MANUAL

Revised October 1976

This document replaces:

SPHERE 1 COMPUTER SYSTEM OPERATOR AND REFERENCE MANUAL SET

Prepared by: Ernest L Dixon
E. Dixon

Approved by: Douglas D. Hancey
D. Hancey

ABSTRACT:

This document gives a brief overview of computing in general and gives detailed examples and guidelines to the assembly and use of the SPHERE 300 series computers and those components manufactured by SPHERE CORPORATION.

940 North 400 East • North Salt Lake, Utah 84054
(801) 292-8466

REVISION STATUS

Page	Issue
*Title	October 1976
*ii	October 1976
iii thru v	Original
*vi	October 1976
vii thru viii	Original
1-1	Original
*1-2	October 1976
1-3 thru 1-5	Original
2-1 thru 2-19	Original
3-1 thru 3-7	Original
*3-8	October 1976
3-9 thru 3-11	Original
4-1 thru 4-3	Original
5-1 thru 5-3	Original
6-1 thru 6-55	Original
7-1 thru 7-4	Original
8-1 thru 8-7	Original
9-1 thru 9-11	Original
*9-12	October 1976
9-13 thru 9-21	Original
10-1 thru 10-8	Original
*10-9 thru 10-10	October 1976
*10-11 thru 10-12	Added October 1976
A-1 thru A-2	Original
B-1 thru B-3	Original
C-1 thru C-2	Original
*C-3 thru C-6	October 1976
C-7 thru C-14	Original
D-1 thru D-216	Original
E	Newsletters to be filed as necessary
F-1 thru F-7	Original
G-1 thru G-18	Original
H-1 thru H-14	Original
I	Schematics to be filed as necessary
J-1 thru J-5	Original

Pages changed or added during minor revisions are marked by * above for quick reference. Information affected by a revision is indicated by a vertical line in the outter margin of the page.

SPHERE 1 COMPUTER SYSTEM OPERATOR AND REFERENCE MANUAL SET

TABLE OF CONTENTS

Table of Contents cont'd.

1 INTRODUCTION

The purpose of this manual set is to provide the user with information that will aid him in solving problems. Because of its great cost reduction the SPHERE computer system has introduced computing to many new applications. These new applications are often thought up by first time computer users, therefore, a new approach to the manual organization has been taken. There are sections which should be of interest to the hobbyist, other sections will satisfy the needs of the advanced user, and others will be of interest to both advanced and novice users. It would be suggested that you glance through the information that is found in this manual so that you may effectively use it for future reference.

1.1 SYSTEM DESCRIPTION

The SPHERE system contains a central processor, memory, input and output devices just as other digital computers on the market today. The system is a fully capable general purpose computer. Because of its size one might be deceived into believing that it has limited capability. The SPHERE system is far more capable than the UNIVAC 1 which cost $1,000,000 to insurance and other firms in early commercial computing. Yet one UNIVAC 1 was used to handle the maintenance, billing, and claims of a million different insurance policies. A million dollars for a UNIVAC 1 was an honest cost justified expense which saved many millions of dollars. The advent of the microprocessor has reduced the size and price of the computer. Computing ability and computing speed continue to increase. These capability increases mean substantial changes in the way we think about computers. When the computer cost $1,000,000, a million dollars to program the computer to run "efficiently" was justifiable. Today, a million dollar programming effort on a $1,000 or less computer is ridiculous. People and their time are valuable. COMPUTING EFFICIENCY AT THE EXPENSE OF A VALUABLE PERSON'S TIME is a waste of a valuable resource. The SPHERE system is the first designed around people and their needs.

1.1.1 SOFTWARE

The Program Development System (PDS) includes an EDITOR, MINI-ASSEMBLER, and a debugging package. It also may include a CRT, floppy disk and audio cassette software drivers. Although most computer processing occurs at the character (8 BIT) level, it is sometimes desirable to use 16 bit arithmetic so we have provided an extended 16 bit instruction set in the PDS system. This package rounds out the "SYSTEM" concept for our smallest systems. There are proponents of various computer languages everywhere. Each language is suited more or less to a specific group of applications. Although the advent of the microprocessor really

dictates some new philosophies in computing language, the BASIC language seems to come closest to this philosophy. Because of its widespread use we have selected it to be our first computer language. A SPHERE operating system is supplied to all users of SPHERE equipment. The software is set up to the configuration required for that system's hardware. This system includes a comprehensive 300 page operator's manual. The software supplied to make the SPHERE system a useful "SYSTEM" is attractive; however, the real contribution that SPHERE offers is one of commitment. The SPHERE "SYSTEM" concept demonstrates only the surface of the real technological advances that are possible when true design innovation is combined with foresight and state-of-the-art technology. The SPHERE "SYSTEM" concept is the commitment. The fundamentals of software are discussed in section 2.3. Details on SPHERE software are contained in section 9.

1.1.2 HARDWARE

The SPHERE computer system was designed to provide an uncompromising computer system at minimal cost. The keyword to the design is the word "SYSTEM". Every phase of the design has been influenced by the "SYSTEM" philosophy. To justify the system title, a "COMPUTER" must perform an application acceptably. Recently, the cost of peripherals and software have substantially exceeded the cost of the computer, but without them, a computer cannot perform much of anything acceptably. With the onset of the microprocessor, real design innovations have been possible, but without the system philosophy, a microprocessor can only reduce the processor cost. Peripherals, memory, and software continue to be expensive. The SPHERE computer is uniquely cost effective because it utilizes real design innovations to reduce the amount of circuitry required throughout the system. The SPHERE add-on memory board will support 4, 8, 12, or 16K of dynamic random access memory. Our power supply has been placed in a separate chassis to eliminate a common source of heat. This allows the system to run cooler and eliminates the need for an expensive fan. The system uses a standard TV monitor for a 512 character display. The use of the TV and other common components has reduced the cost and allowed more machine versatility. Further cost reductions have been achieved by replacing the front console (lights and switches) with the TV terminal, keyboard, and a program in Read Only Memory (ROM) that performs the same function, only better. The CPU card is packaged to provide all of the basic functions required by a useful system, thereby eliminating

Hardware cont'd.

The system uses a standard TV for a 512 character display. The use of the TV and other common components has reduced the cost and allowed more machine versatility. Further cost reductions have been achieved by replacing the front console (lights and switches) with the TV terminal and a program in Read Only Memory (ROM) that performs the same function, only better. The CPU card is packaged to provide all of the basic functions required by a useful system, thereby eliminating unnecessary extra PC BOARDS. In order to insure a full offering of high quality peripherals from the onset, we have selected manufacturers who already have peripherals which interface to our product. This philosophy has allowed us, in the case of our disk, to select already running software (namely a disk operating system) which we may offer to our users immediately. Other peripherals that are available with our system include a low cost line printer and a paper tape reader/perforator. These devices are interfaced to the system via a single interface module which also serves as a programmable digital Input/Output port. The SPHERE system also supports its own set of terminals, the lowest cost terminals available today.

1.2 DIGITAL COMPUTER HISTORY

The digital computer is an outgrowth of thought and effort spanning three centuries, directed at reducing the time and tedium of human calculation. Plaise Pascal, the French mathematician and philosopher, developed in 1642 the first mechanical calculating machine. Proposals for new, and more powerful, computing devices appear in nineteenth century literature. In 1812 the Englishman, Charles Babbage, constructed in mechanical form the recognized prototype of the computer. His "analytical engine" had a "store" (the equivalent of internal memory) and a "mill" (the equivalent of an arithmetic unit), and it was designed to accept input data in the form of cards with holes in them, an idea that Babbage borrowed from Jacquard's loom. In 1889, Herman Hollerith, a statistician employed by the United States government, developed and patented the first line of primitive punched-card equipment consisting of a punch, a sorter, and a tabulator, to aid in the compilation of the 1890 census data. The era of the modern computer dates from the late 1930's. The years 1939 to 1944 mark the successful development, at Harvard University, of the first automatic, general-purpose digital computer (electro-mechanical). The Automatic Sequence Controlled Calculator, as the machine was called, was developed by Professor Howard Aiken with financial and technical assistance from the IBM Corporation. At about the same time, pioneering work in this area was also being done by George Stibitz of Bell Laboratories, who developed an electro-mechanical automatic digital computer. Concepts of modern digital computers were defined in detail between 1943 and 1946 by John von Neumann, a Hungarian mathematician working at Princeton University, and by the team of J. P. Eckert and J. W. Mauchly of the University of Pennsylvania, who perfected and built ENIAC (Electronic Numerical Integrator and Computer), the first electronic digital computer, in 1946.

Since then, the development of computer technology has accelerated in an explosive fashion. Never before has a technological product been developed and improved to such an extent over such a brief period. The modern computer is a far cry from its calculating precursors indeed. It must be pointed out that the connotation of the term "computer" is now, unfortunately, somewhat misleading as it evokes the image of "computing", i.e., some arithmetic operation with its manipulation of numerical data. The notion that the computer is a strictly numerical device is quite incorrect. A computer is basically a symbol-manipulating device and numerical symbols are merely one of the symbol classes that the computer can manipulate. The computer is a machine that duplicates and amplifies certain powers of the human mind. It provides an extension to man's intellect. It has been called the universal machine and man's ultimate machine. In the latter sense, it is seen as the supreme technological achievement, because the computer deals not in raw power but in the sublime, abstract processes of mental work. In this connection, we should realize that before the advent of computers, practically every other tool or machine ever invented and built served as an extension of man's legs (locomotion), back muscles (materials displacement), or arms and hands (manufacturing). Today's most advanced machines such as the bulldozer, the combine, the automatic machine tool, the jet plane, and the rocket-powered space vehicle all belong to this category. The computer, on the other hand, is in a category all by itself. It does not, alone, achieve any physical feats whatever. It can plan and control physical action by other machines (and men), but its own outputs is always symbolic and therefore abstract. The symbols that a computer has the ability to manipulate are numerical digits, letters of the alphabet, special characters, and subcharacters or bits which represent information in the form of data. A computer processes information by means of receiving, storing (remembering), operating on, and producing (output) data. These information-handling operations are directed by a program of instructions which itself is stored in the computer's memory. This principle of utilizing internally stored, alterable instructions to control the action of the machine is what provides the computer with a versatility, a logical flexibility, and an open-endedness that are not matched by anything short of a living organism. In addition to duplicating certain intelectual processes, a computer is also capable of performing clerical tasks that can be viewed as routines performed by rote, such as the retrieval of records, posting and filing, i.e., record keeping, as well as transmitting, regenerating, and display of information over distance. Although the computer is relatively limited, as compared to the human mind in it's range of capability and although it uses rather crude methods of internal processing, it has the advantages of speed, total recall, and complete accuracy. Speed is the dominant characteristic, as electronic computer circuitry typically handles signals at rates of millions per second.

The computer is also called an electronic data-processing system because it is actually composed of several connected, interacting, and mutually dependant machines and devices operating in harmony under central control. Another dimension of the computer concept is the relationship between machinery and stored program (hardware and software). They are inseparable in that both, in combination, determine the total power of the computer, which has some of its capabilities designed into its physical mechanism (wired-in capabilities) and others supplied through stored instructions (programmed capabilities). Thus hardware and software are but two different aspects of the same thing, the machine called the computer. In discussing it on a conceptual level, one more point remains to be made, i.e., that the computer, in addition to all the attributes mentioned above, is characterized by a universality of its use. Unlike other machines, all of which have a relatively limited, special purpose, the computers is truly a general-purpose machine in its sphere of duplicating mental processes. Its application potential is universal, in the sense that it can potentially be employed wherever the human mind is at work. As a result of the tremendous advancements in micro-miniturization, through large scale integrated circuits (complete computers on a chip), we are much better at designing and manufacturing computers than we are at using them anywhere near their potential. It has been estimated that, if further development of computer technology stopped right now, several generations of the computer's users would be kept busy discovering its many unexploited capabilities and generally learning how to use computers more fully, and therefore more successfully. To help close this gap between technology and utilization of the computers as a tool in many more applications not possible before LSI, is our purpose in developing the inexpensive SPHERE computer system. We hope that this information processing tool of immense power and tremendous potential will serve you well. We will be constantly applying our efforts towards aiding you, through continuous development and communications interchange. Good luck.

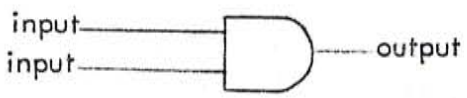
SECTION 2

2.0 STARTING AT THE BOTTOM

Imagine multiplying the numbers MCMV x XIII. Obviously the mechanics of the problem over shadow the learning of the concept. In the same way, a lack of experience in a specific field can obscure the fields beyond. Some find themselves at the bottom of the heap when it comes to computers or some phase of computing. The following sections are useful in learning the fundamentals. They are not intended to be a complete text they will however aid tremendously an understanding of the function and form of computers. Starting at the bottom then . . .

2.1 COMPUTER LOGIC FUNDAMENTALS

Computers are designed using electronic circuits just as radios and TV's are. However, computers are generally far more complex than TV's and radios. Digital computers have a common feature that eases their design and construction. The term"digital" gives a clue, "digital" implies a fixed amount as opposed to an approximate amount or analog. Electricity is inherently analog as we do not have instruments that can measure exact voltage. (We may be able to approximate a voltage using many decimal places, i.e. 1.6924682 volts, but it continues many decimal places beyond that). Digital electronics assign a certain voltage range that attribute on or one, and another voltage range the attribute off or zero. Although these voltage ranges are theoretically arbitrary, characteristics of certain circuits commonly used in digital electronics have developed standards which bare a "family" name. These names include DTL (Diode Transistor Logic), RTL (Resistor Transistor Logic). TTL (Transistor Transistor Logic), and many more. Currently the most commonly used logic is TTL (sometimes referred to as T^2L). Generally this logic family uses a zero to .8 volt signal to indicate a zero or off state and a signal of .8 to 5.0 volts to indicate an on or one state. A one state is usually measured at about 3 volts. A further simplification of digital logic is that a relatively small number of prescribed functions can be combined in various ways to perform even the most complex functions. These simple functions are referred to as gates. They have one or more input wires and one or more output wires. An example of a gate would be the "and" gate. It has 2 inputs and one output. If the first input is on (one) AND the second input is on (one) the output will be on (one). If either one or both of the inputs are off the output of the AND gate will go off (zero). Gates are graphically displayed as a symbol with input and output lines drawn from specific points on the symbol. The "and" symbol appears as:

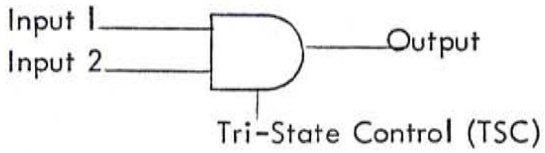


In addition to the graphic representation of a gate, a functional representation of a gate would be listed in a logic table or logic diagram. Here inputs and their states (one or zero) are listed in a tabular form so that matching a desired row and column will indicate the correct output state. An "and" gate logic diagram appears as:

		INPUT 1	
		0	1
INPUT 2	0	0	0
	1	0	1

OUTPUTS

The output of a standard TTL gate can usually supply enough current to wire to the inputs of 10 other standard TTL gates (10 standard loads). Other forms of non-standard TTL logic exist but some design consideration must be made when using these devices. Some types are low power, high speed, schottky, low power schottky, CMOS, and other "compatible" MOS. Each non-standard TTL or compatible device should be checked for power compatibility with respect to the number of loads each will drive with respect to another. Two types of TTL logic not mentioned here-to-fore are Tri-state ™ and Open Collector. Tri-State (sometimes called high impedience state) does not mean a third set of voltages which indicate a third state. It simply means that the gate has been disconnected from the line. This is useful when many different gates would like to use the same wire to communicate at mutually independent times (party line). Computer systems use this philosophy extensively. An "and" gate that has Tri-State capability would be displayed as:

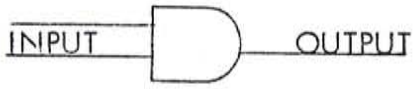


It's logic diagram:

Inputs			Output	
1	2	TSC		
0	0	1	0	
0	1	1	0	
1	0	1	0	
1	1	1	1	* high impedience state
*	*	0	*	

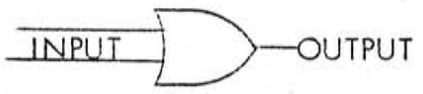
If an "*" appears in a logic diagram it indicates a "don't care" state. In other words, that input is not used to calculate the output if other conditions of the logic diagram are met. Normally only one output is used to connect to inputs to other gates as needed (within load limits). When this rule is violated, circuit damage will result. This is not true of party line (tri-state or open collector) circuits. Tri-State places only one output on the line at a time. Open collector on the other hand simply has protective circuitry to eliminate possible damage due to multiple outputs on the same line.. If any output is on, the combined output will be on, otherwise it will be off. Following is a list of common gates:

The AND gate takes several inputs and produces a 1 output if all inputs are 1.



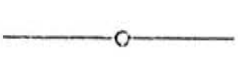
		Ø	1	Line 1 Input
LINE 2	Ø	Ø	Ø	
	1	Ø	1	
		output		

An OR gate takes several inputs and produces a 1 output if any input is a 1.



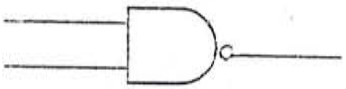
		Line 1	
		Ø	1
LINE 2	Ø	Ø	1
	1	1	1

A NOT gate inverts the current to produce the opposite output.



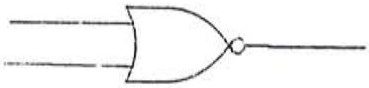
Line 2 \ Line 1 Input	0	1
1	1	0

A NOT gate put in front of an AND gate produces a NAND gate.



Line 2 \ Line 1 Input	Ø	1
Ø	1	1
1	1	Ø

An OR gate put in front of an OR gate produces a NOR gate.



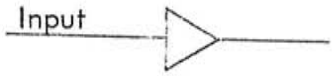
	Line 1 Input	
	Ø	1
Ø	1	Ø
1	Ø	Ø

A Buffer gate allows a single load line to drive 10 output lead lines.

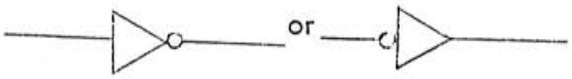
When a NOT gate is added to a delay it becomes an inverter.

DELAY

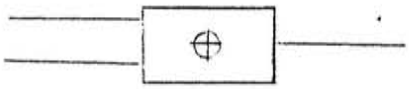
Input



INVERTER

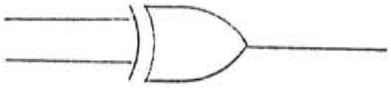


An exclusive or (XOR) gate produces a 1 output if there is only one input ON.



	Line 1 Input	
	Ø	1
Ø	Ø	1
1	1	Ø

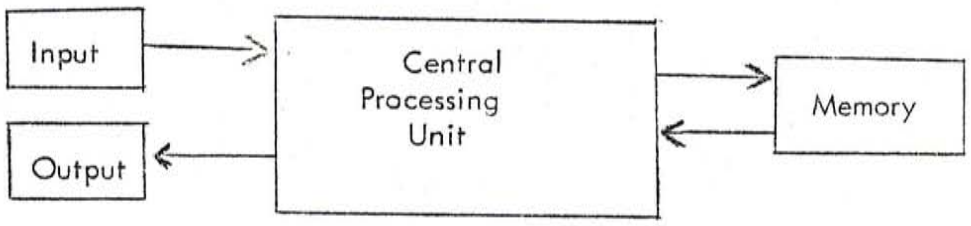
also drawn



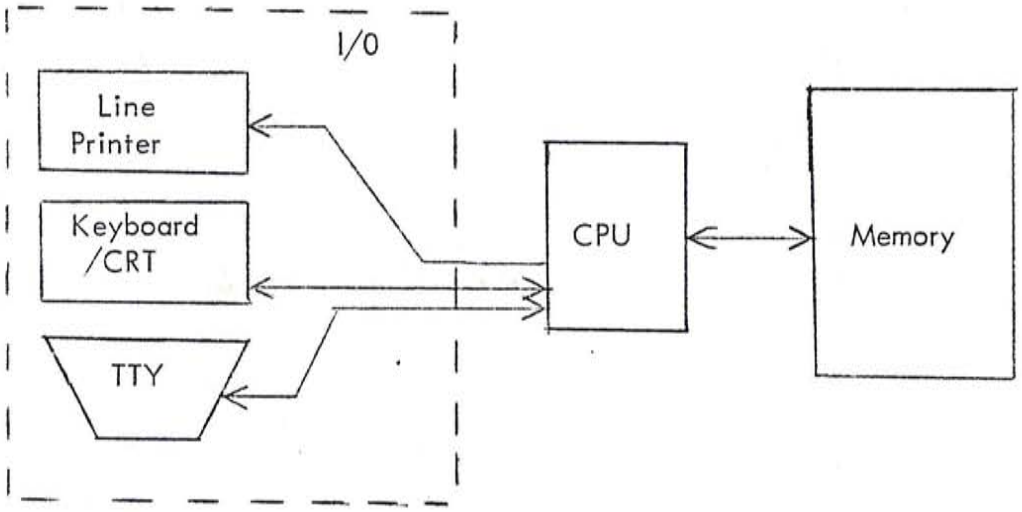
Of times, a complex set of gates will simply be expressed as a box with inputs and outputs identified. A description and logic diagram will accompany such descriptions. If the circuit is repetitive in function such as an adder or shift register, a sample circuit and/or logic diagram will be documented. Through the use of new technology, logic gates have successively grown smaller. Now thousands of these gate circuits can fit in a package the size of a postage stamp. These circuits are called integrated circuits (IC's). They are combined in various fashions to create simple memories, complex memories, complete computer processors (CPU's) and a whole array of other circuits. In commercial applications, they are usually packaged in dual in-line packages (DIPS).. These are some times referred to as "Bugs" as they appear to have a body and many legs. The legs are spaced at .10 inches and are designed to be inserted into printed circuit boards for subsequent soldering. Appendix D includes descriptions of circuits used in the SPHERE Computing systems. It includes reference information for maintainance and theory. Consult IC manufactures literature for more detailed information.

2.2 COMPUTER ARCHITECTURE

Computers and Buildings are designed by architects to perform various functions. In many ways their design is similar, they each have basic components that are required, then materials are selected and arranged in a suitable form. Computer architecture refers to what units are to be in a computer system and their characteristics and interconnections. Some units, such as the arithmetic unit, are composed solely of logic circuits designed to perform a particular function. Other units may be mechanical in nature such as card readers or teleprinters. A computer is basically a data manipulation machine. It transforms one set of data into another set according to a fixed list of instructions called a program. To accomplish this, a computer contains several units to more and change data. The basic unit is a Central Processing Unit (CPU), which interprets the program commands and does the requested data manipulation. Another basic unit is the memory, which stores the data used by the system. Also needed are units to perform input and output on the stored data. A general configuration would be similar to the one shown in fig 2.1



Computers are fast accurate and inflexable while the humans that use them are slow, clumsey, and flexible. The interface between the two has always been a problem. In the early days of computing, the most widely used devices for input and output were the 80 column card and a printer running at 150 to 600 lines a minute. This was very wasteful for both man and machine, as the machine would usually spend the majority of its time waiting for I/O to be completed while the man would spend up to two or three days waiting to get his program key-punched, run on the computer and the results back. To improve ease of use, time-sharing and multiprogramming systems were developed in the early 60's. Multiprogramming allowed several programs to be in the machine at a time with one running while the others performed I/O, thus greatly improving machine usage. Time-sharing allowed several users multipul access to a computer at the same time. The user would use a teleprinter or a CRT (cathode ray tube) display to enter information directly into the machine. These developments allowed better use of very expensive machines. With the progress in electronics, the price of computer systems fell during the 70's to the point where a small computer was less than the cost of an earlier terminal. A typical system of this type is the Sphere I, which allows I/O to be performed with the keyboard/CRT, printer, teletype, or digital I/O lines. A typical computer would then look as in Fig 2.2.



Once the data is entered into the computer, it must be stored for access by the CPU as needed. There are several types of memory available with different costs and speed. Memory devices fall into two main catagories, sequential devices and random access devices. Sequential memories are those where each unit of memory is ordered one after the other so that the data is accessed in a specific order. Typical are magnetic tape units or cassette tapes. Sequential units offer the lowest cost per unit of information but have the longest access time to a specific information unit. It is useful in applications where there is a lot of information to be processed in a specific unvarying order. Randome access memories are those where the time to retrieve any random unit is the same as for any other data unit. This type of memory offers the fastest unit access but has the smallest storage capacity. This is the memory accessed directly by the CPU when it is executing a program. High speed memory has traditionally been made out of ferrite "cores", so called because each resembles a doughnut in shape with read, write and sense wires run throughout the hole or "core" of the doughnut. Random access memorys can also be made up of flip-flops, which is an electrical circuit capable of storing one binary digit. The flip-flop was originally used in designing circuits for the purpose of storing the output of other circuits so that circuits would have access to previous results. With the advent of metallic oxide semiconductor large scale integration(MOS LSI) circuits, it has become possible to put several thousand flip-flop circuits on a single chip. The Sphere I uses such chips with 4096 (4K) bits (binary digits) of memory on each chip. Even though the memory is now semiconductor, it is often still refered to as "core" memory to denote that it is the memory used by the CPU for storage of executing programs and data. Disk and drum memories are devices that are both sequential and random access in nature, offering moderate size at moderate cost. They consist of a disk on drum that is coated with a magnetic material rotating continuously past a set of read or write heads. There is one 'track' of data under each read or write head. The devices are random access in that any given data is accessable within an average period of time. This is called the 'access time' and averages one half the time it takes the disk to rotate once. On most disk units, there is only one read or write head which is moveable between tracks. There is then a track latency or access time. The disk used on the Sphere I is a "floppy" disk with 64 tracks having 4096 bytes of data per track. The type of memory used is thus determined by tradeoffs of access time, capacity and cost. With access time increasing with storage capacity and decreasing with cost. Devices other than core memory are usually referenced through a control unit designed to interface that device to the CPU and core memory. A typical system would appear as in Fig 2.3

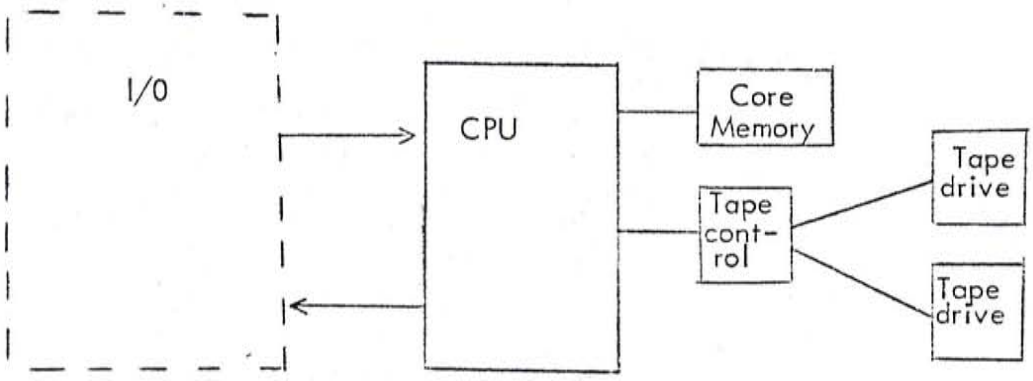
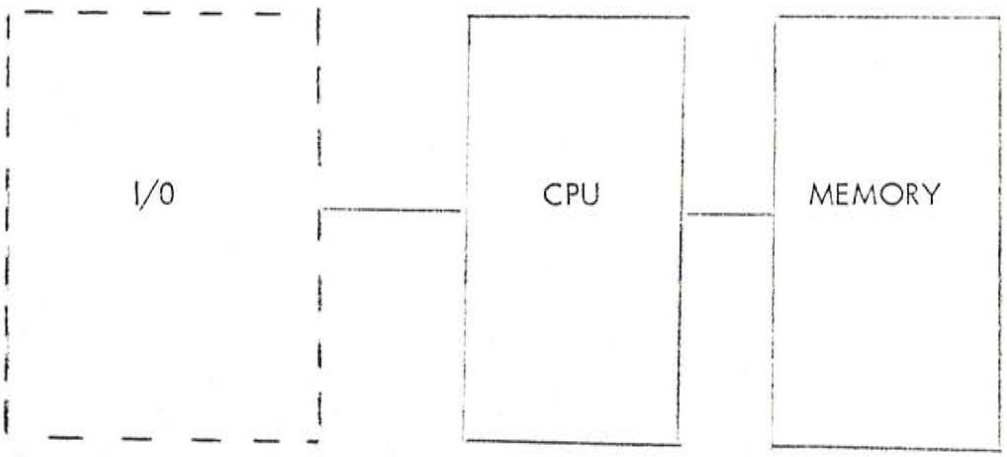


Fig 2.3

The CPU carries out the instructions of the program for manipulating the data. It is composed of three main subunits, the arithmetic/logic unit (ALU); the register unit, and the control unit. The arithmetic/logic unit does the actual data minipulation performing such arithmetic functions as addition and subtraction, and such logical functions as comparing two units of data, shifting them and performing "and" "or" "XOR"s and other functions of the individual bits of data. The registers act as storage for specific units of data. The registers are accumulators, index registers, the program counter and the stack pointer. An accumulator is a fixed storage location used by the ALU. An instruction such as ADD A would take the contents of the accumulator and the contents of the memory location A and add them together and put the result in the accumulator. The index register is used as a pointer into main memory. It is used on a randome access basis for accessing data. Its contents can be added to a fixed starting address for randome accessing arrays of data. The stack pointer is used for referencing the stack. A stack is an area of main memory reserved for sequential (zero address) accessing. It is referenced on a data-that-is-last-in-is-first-out (LIFO) basis. It is used to store temporary data and subroutine linkage return addresses during program jumps. While most data has fixed locations (addresses) in main memory during execution of a program, the stack data varies dynamically during execution of a program. A stack reference is called zero addressing because all references to data on the stack are from the top-of-stack (TOS) which is implicit in any stack operation. The stack pointer is sometimes referred to as the TOS pointer register. The index register can be used to reference data a given distance from the TOS. This allows programs to be dynamically assigned blocks of data as needed. The program counter is pointer into main memory, pointing to the location of the next program instruction to be executed.

It is changed either automatically with each instruction execution or modified by the program to change the instruction flow. The control unit of the CPU takes the program instructions, determines what is to be done, then activates the various gates, registers, adders, etc. to perform the requested operations. It also tests the results of instructions to determine what location the next instruction is to be taken from. In a computer, while all data consists of binary digits, these bits are grouped into bigger units of data for manipulation by the computer. The basic unit is the byte, which consists of 8 bits. A byte can thus contain a value between 0 and 255 decimal.. It is used most frequently for representing a written character such as 'A', 'B', or '+'. Because it can't contain a very big number, bytes are put into larger groupings called words, which usually contain 4 bytes (32 bits) on large computers and 2 bytes (16 bits) on small computers. The basic unit used on the Sphere I is the byte, thus making it ideal for character and text manipulation. A detailed diagram of a computer system would thus appear as in Fig 2.4



2.3 SOFTWARE

While a computer is a very sophisticated piece of circuitry, it is absolutely useless without being told what to do. It must be given instructions on how to handle the data. A computer program is basically a set of such instructions for handling data. Software consists of all the programs and data needed by a computer system to produce useful results. An analogy of a computer program could be made with a country post office that could only hire a moron as postmaster. Unfortunately, while being a conscientious worker, he can't remember how to do his job. So someone must continually tell him what to do. Because no one else is around for that job, the post office management decided to leave instructions for him to follow on what to do to get the job of sorting the mail done. In order to help the moron keep track of the large number of instructions he had to follow, they were put in unused slots in the sorting bin, one instruction per slot. The instructions had to be very simple for the moron to follow. Typical instructions would be like "Pick up a letter from the mail pile", or "If the letter lacks a stamp, take your next instruction from slot #57", On coming to work in the morning, the moron would take the first instruction from slot #1, then another from box #2 and so forth unless the instruction specifically said to go to another box for the next instruction. By following each instruction, the moron could then perform any task the Post Office could think of. A computer is very similar. The computers central processing unit (CPU) is the moron, slavishly following the instructions that it takes from memory. Memory consists of a set of numbered locations, with each location or memory cell containing an instruction or an item of data. Having a computer "moron" do useful work for you thus becomes a task of producing an explicit set of instructions for it to follow. The set of instructions is called a "program" and the task of producing then is called programming. The task of programming can be broken down into several steps. The first step is to analyze the problem. What does the problem consist of? What information is used and what is the desired output. Defining the boundaries to the problem area is one of the most critical in producing a workable solution. Once the problem has been defined, it can be broken up into modules for ease of handling since most problems are too large to tackle in one piece. These are then further subdivided into smaller modules. This subdividing continues until the modules are small enough to work with. A diagram of such a top-down modularized struction would appear as in Fig 2.3.1

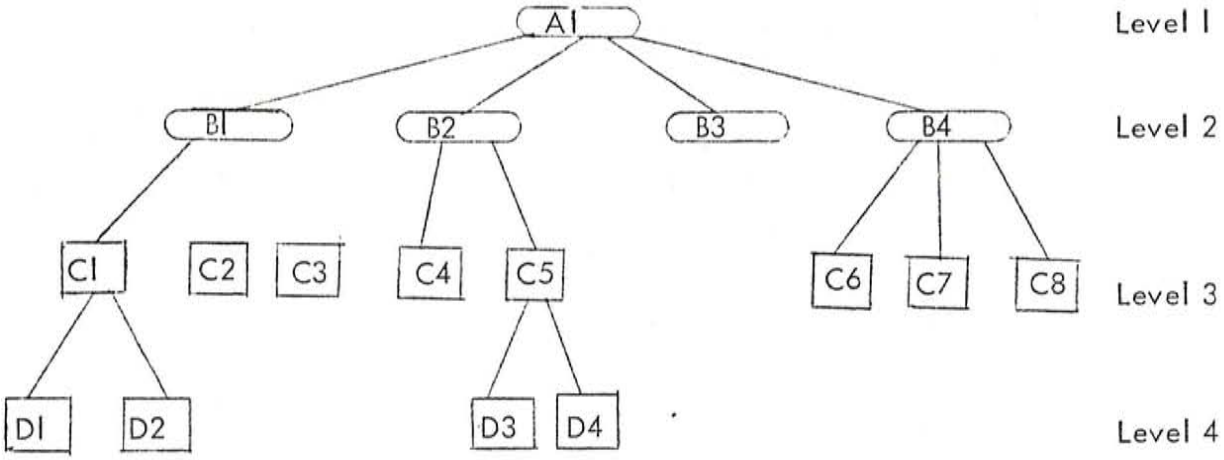
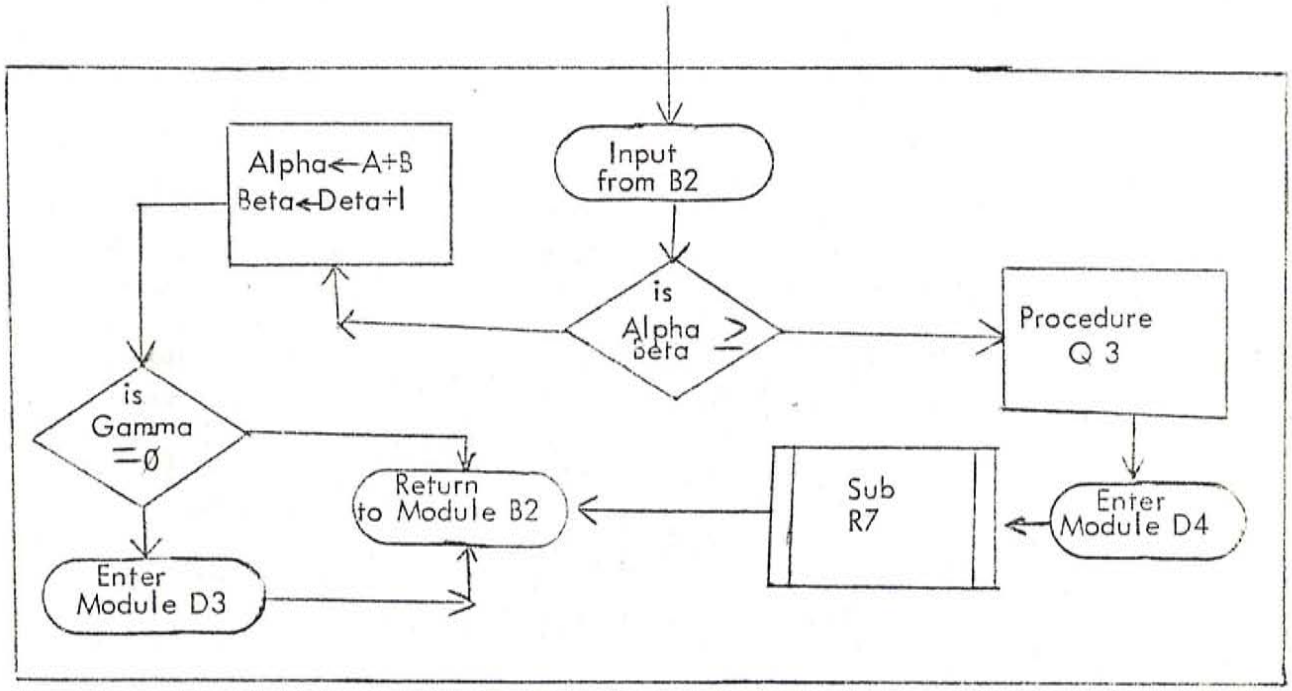


Fig 2.3.1

Each module on any given level is independent of any other module on that level, but are connected by modules on higher levels. This enables one part of the program to be changed with minimal effects on other parts of the program. After the overall problem has been modularized, a detailed diagram description called a flowchart can be produced describing each step in a given module. For instance, module C5 might appear as:



For clarity, a complex function such as procedure Q3 can be described on a separate flow chart. By using this process of zooming in for more detailed descriptions, flowcharts can be kept simple and understandable. After a detailed description of the problem and the procedures used in producing the result have been written, the flowcharts can be used to translate or 'code' the processes into a computer language most suitable for that application, such as BASIC, RPG or FORTRAN. When the program has been coded, it is run on the computer and the process of debugging begins. Debugging is the process of getting a program to work correctly. Bugs are errors in the program. There are two types of bugs: syntatic and semantic. Syntatic errors are those that do something not allowed by the programming language, such as jumping to a nonexistant statement or something like A← B/0. These bugs should be recognized by the language and an error message given. Semantic bugs are where the programmer says one thing but means another, such as subtracting a number where it should have been added. These errors in meaning must be found by testing each module with preselected input data for which the result is known. After the bugs are found, the program must be corrected and debugged again, as correcting one bug may result in another bug. Remember Murphey's third law: "If anything can go wrong, it will, but only after it appears to be correct". After the program is debugged and running, the question must be asked "is the information the computer is producing what I really needed in the first place?" Errors can be made in the original problem analysis and not show up until the prospective user tells you what you can do with your program. In order to help the user develop his programs, computers come supplied with a set of programs called system software, which is designed to make the computer easier to use. The basic system software program is the executive or operating system. The executive schedule and runs all of the users programs and schedules the input/output operations. It does all of the file handling, including device assignment and logical name/physical device translation. In a multiprogramming or time sharing system, it determines which program is to run at any given time. The executive can be thought of as the chief decision maker of the computer system. Most useful of all systems as far as the programmer is concerned, are the computer languages available to write programs with. A computer language is nothing more than a restricted set of instructions understandable by both computer and human. The most basic computer language is machine language, which consists of the actual binary numbers executed by the CPU. Because strings of binary numbers are hard for humans to use, a form of it called an Assembly Language has been developed. Assembly Language allows symbolic names to be used instead of binary numbers. For instance, the nmonic ABA (add accumulator B to accumulator A) would stand for the binary number 00011011.

Because this is still a very constrained format, high level languages such as BASIC, RPG II, TRAC or COBOL have been developed. A high level language would allow a statement such as A⇐B+C - (D/5) instead of the equivalent 10 or 15 assembly instructions. Each high level language is aimed at a specific users such as COBOL(Common Business Oriented Language) for business use, ALGOL(ALGOrithmetic Language) for scientific uses, BASIC (Beginners All purpose Symbolic Interpretive Computer) for novice users, and TRAC (Text Reconing and Compiling) for text and character manipulation. High level languages fall into two major devisions, compilers and interpreters. A compiler translates from the high level language directly into machine language, which is then the program that the computer executes. The high level language program is called the source code or source language and the machine language translation is known as object code or object program. An interpreter doesn't translate the program into machine language but interprets each statement as the program is executed. For short programs, an interpreter will run fast as it doesn't have any 'compile time' overhead, that is it doesn't spend any time in translating to machine language. However, a compiled program will execute seven to ten times faster than an interpreted version. Another segment of system software is the utility programs. These programs consist of such goodies as text editors, used for creating and changing source language programs; file utilities, used for changing or manipulating files; linkers to hook different programs together and other useful programs. No matter what high level language is used, it will contain certain features for the implimentation of programs. These features may vary widely from language to language in how they are inplimented, but their functions are similar as to what they do. The most basic construct is the arithmetic statement. This statement allows a number to be calculated from an arithmetic expression of variables and constants, and to be assigned to another variable. A variable is a number that has a name and that can change in value during execution of the program. Typical arithmetic statements would be:

ALPHA⇐ A x B + 5.1 - (3.2÷C);

LET ALPHA = A*B+5.1 - (3.2/C)

ALPHA EQUALS A TIMES B PLUS 5.1 MINUS 3.2
DIVIDED BY C.

#(DS, ALPHA, #(SB, #(AD, #(ML, A, B), 5.1), #(DV, 3.2, C)))

Software cont'd.

An arithmetic statement also makes use of built in functions such as sine, cosine and square root as well as operators like add and subtract. There must, of course, be input/output statements, or else what the computer does is not very useful. These statements move data to and from the executing program and the executing program and another data source, such as a file, a teleprinter, paper tape, or another program. These statements can also be used to format the data so it appears in a desired manner. A logical instruction is a statement that compares two values and returns a value of 1 or 0 depending on whether the comparison was true or false. The logical expression is used with the control statement to provide run time control of which statements are executed. The simplest control command, the branch, is of the form GO TO X which causes the computer to branch and start executing at the statement labeled X. A conditional branch would use a logical expression for a statement like 'IF ALPHA $\geq$ 1 THEN GO TO X'. Individual statements can be executed in statements such as 'IF GAMMA = 0 THEN ALPHA$\leftarrow$ A+B ELSE ALPHA$\leftarrow$0. Another form of control statement is the loop. The loop causes repeated execution of a block of code until a specific logical condition is met. A typical format would be 'WHILE A$\geq$ 1 DO XXX is a series of statements. Usually, each time the block of code is executed a value can change by a specific amount such as 'WHILE A $\geq$ 1 STEP A BY 2 DO XXX'; which would increase the value of A by 2 each time through the loop. As programmers, like most other people, are somewhat lazy and don't like to do more work than they have to the subroutine was invented. A subroutine is a segment of code that can be used several times in different places in the program. This makes programs much easier to write, as certain functions are made use of repeatedly. A subroutine is defined once and then explicitely called later in the program for greater flexability, each time a subroutine is called, different numbers or data can be passed to the subroutine from the main program. This allows it to act similar to a built in function such as SINE. By the use of software, the computer can become a very useful tool. Without it, it is nothing more than a blinking black box of lights.

2.4 APPLICATIONS

A computer is a general purpose tool for problem solving. A piece of hardware called a "general purpose computer" can perform absolutely nothing by itself because there is no such thing as a general purpose problem. A piece of software designed specifically to "apply" some general purpose computing power to solve a specific problem is called an "application" program. Not all software (programs) are written to solve a specific problem. Some are written specifically to aid application programs performance. These "SYSTEM" programs make the computer more useful to a greater number of application programs. All programs are not application programs. Almost any problem that takes mental effort can be solved by a computer with an appropriate application program. A computer then is a problem solving tool which, if used properly, can provide the user the ability to solve his problems. There are as many applications as there are imaginations. There are, however, general application subjects that we would like to illustrate so that you can understand how a typical non-computer activity is transformed into an application on a computer. We will illustrate an accounting problem.

> Late in December the last month of the taxable year, Arthur Morton and his wife Lois, engage you to prepare their income tax return and advise them concerning other matters that may affect the amount of their tax liability. They are entitled to a total of five exemptions for themselves and their dependent children. They have been using the cash method of determining taxible income and expect to file a joint return.
>
> Mr. Morton is employed at a salary of $24,000 for the current year and has been notified that he will receive an increase of $300.00 per month, effective in January. He has no deductable expenses connected with his employment. Income from dividends for the current year will amount to $900.00 for Mr. Morton and $400.00 for Mrs. Morton.
>
> During the current year, Mr. Morton sold A company stock at a loss of $1000.00. He proposes to sell his holdings with "B" company stock before the end of the year. These were acquired three years ago at a cost of $2000, and the market price which has been relatively stable is currently $3000. He plans to use the funds to pay amounts pledged to his church, $500 for operating budget and $2500 for building fund both due by the following March 31st.

He also suggests that it might be adventageous to use the $1000 loss on the sale of "A" company stock as an effort to gain on "B" company stock.

Miscellaneous payments for the current year to date that would qualify as deductions from gross income total $400. In addition, real estate taxes of $350 for the first half of the year have been paid. Unpaid real estate taxes of $350 for the second half of the year are due on December 31st, but may be paid as late as January 10 without penalty.

This illustration is a painful but real life situation. By attempting to do this income tax problem, you would have a lot of headaches in researching the tax exemptions. In addition, computing the appropriate amount of exemption to determine the amount of tax you need to pay would be difficult. At some point in time many other individuals have had to encounter each of the problems Mr. Morton of our example had to research. A Tax Consultant might write a computer program which considers these possibilities in detail. He would include all information necessary to advise of wise future tax decisions and current tax liabilities. All you would need to do with this computer program is input the information listed in the above problem and the computer determine your legal tax exemptions. Then it would determine the amount which you could deduct so as to save as much of your income from taxes as possible. The computer could also inform you of the status of your total asset holding, liabilities and cash balances through the use of its many intelligence. Applications such as this are virtually unlimited and each can be efficiently handled by a computer. Following is a list of common applications:

BUSINESS

cash receipts
cash disbursement
notes receivable and interest income
retail method of inventory costing
perpetual inventory control
depreciation based on averages
intangible asset amortization
journal recording and posting
depletion schedules
mortgage amortization schedules
payroll deduction computation
checking account balance
taxes payable
and many, many more

Applications cont'd.

Education

test checking and Scoring
student records
teacher evaluation
book index
class scheduling
periodical referencing
audio-visual cataloging
current subject reference
rare document storage
great talks and speeches
teacher planning
thesis research
and many, many more

Real Estate

Prospective customer index
Property catagorizing
Multiple listings index
Residential development program
Mortgage and interest compiler
Proposed commercial development format
Monthly payment breakdown
and more

Science

Physics experiment
Chemical analysis
Motion study
Sound and Wave investigation
Zoological reference and documentation
Biological reference and documentation
Testing and experimentation
Cross reference index
Geological survey studies
Geothermal research
Solar energy analysis
and the list goes on

Applications cont'd.

Banking

Accounts Rec.
Savings deposits
Certificates of deposit
Checking account balance
Notes rec.
Customer credit information
Federal reserve deposits
Trust fund accts.
Interest rate computations
Auto installment loans
Mortgage handling
Statistical computations
and lots more

Personal Computer Use

Checking account balance
Savings, stocks and bonds, etc.
Household budgets
Dates to remember file
Christmas card list
Menus and shopping list
Security system and crime preventor
Vital document record
As many uses as you can think of

Games

Button, Button, Who's got the button
Create a random maze
3D tic tac toe
14 Civil War battles for 2
Star trader
Try to beat the taxman
Baseball
Basketball
Black Jack
Computer draws a playboy bunny
Pro football
Monopoly
Checkers
Chess
and hundreds more

Applications cont'd.

A computer is more than a practical tool for compiling data and solving problems. A tool or machine that can teach you and help you to broaden your scope of thought is indeed helpful to personal development. Through the implementation of computer games, mental stimulation can occur. Games Encourage Imaginative and Constructive Responses. The result of games are unimportant---we take risks, tolerate uncertainty, and proceed with less-than complete understandings the ability to react creatively to new and unexpected situations are constantly changing. We develop our decision making capabilities and problem solving skills, games are fun. The computer waits patiently while you experiment with different lines of thought. It's available to play any game you want, and as many times as you'd like. A simulation is a model of a real life situation. With your computer, you can reconstruct and analyze real life situations, if you want to be creative. Your computer can do complicated bookkeeping.---You can create the initial conditions, establish the limits, and then analyze the results. With your own computer, a simulated situation can be repeated as many times as you want. Many people ask, "Why play games at all?" A game involves competition as a relationship among players. On a computer we can transform a competitive 2 player game into a teamwork relationship with many more players. Instead of always playing one side, through the use of your computer, you can alternate positions throughout the game. Computer games can introduce new ideas into your built in computer. Games are open ended, multi-purposed, and can be suited to your particular whims. Games can be the tool which could re-open your mind to a vast new frontier of ideas.

SECTION 3

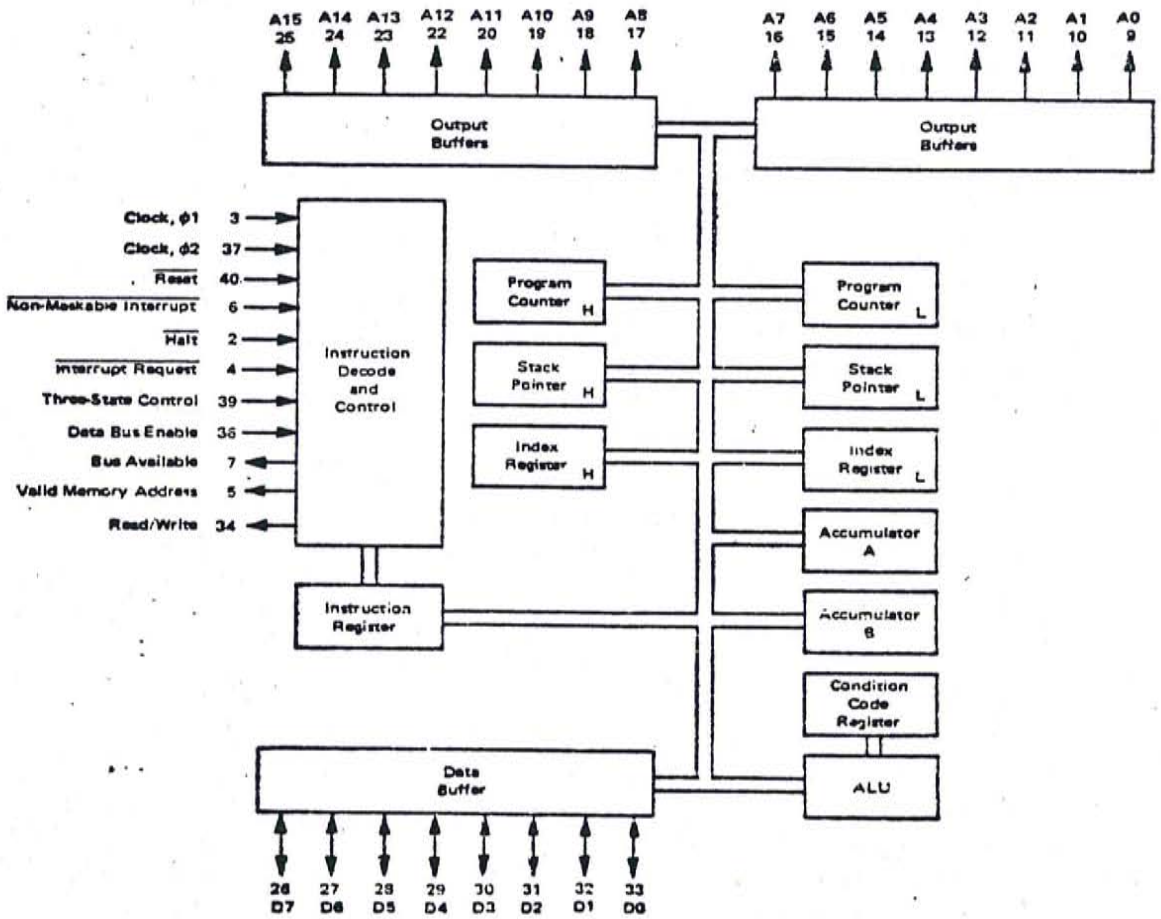
3.0 SYSTEM ORGANIZATION

In Section I, a general description of the System was given. In this chapter, a more detailed description of the device making up the system will be given. The Motorola MC 6800 is an 8-bit microprocessor forming the central control function for the System. As in all computer systems, its main function is to:

1. Control the sequence and execute the instructions received from a program stored in memory to accomplish a given task (prepare a payroll, etc.)
2. Perform data transfers from one point in the system to another.
3. Perform logical and arithmetic operations on data during transfers as needed.
4. Monitor and respond to external influences that alter program sequences.

Figure 3.1 shows in block form the internal structure of Sphere CPU.

CPU
INTERNAL STRUCTURE
Figure 3.1



System Organization cont'd.

Figure 2 shows the relationship of the MPU to the other devices on the external BUS, which make up the system. Note that all devices are paralleled on the bus; i.e., the address, data and control lines are 'common' to all devices. Each device has a unique address and responds only when its address and required control signals are presented on the bus. All address decoding is done by the devices themselves. This bus structure eliminates the need for seperate I/O channels for each device and simplifies the interface requirements. In addition, because each device contains its own status and data registers, they look just like memory locations to the MPU. Each device is therefore addressed, no special I/O instructions are needed. In fact, any instruction that can access memory or operate on data in memory can also manipulate the I/O device registers. This provides much greater programming flexibility than would be possible with a system requiring special I/O instructions. This structure also simplifies and standardizes the interfore requirements for peripherals.

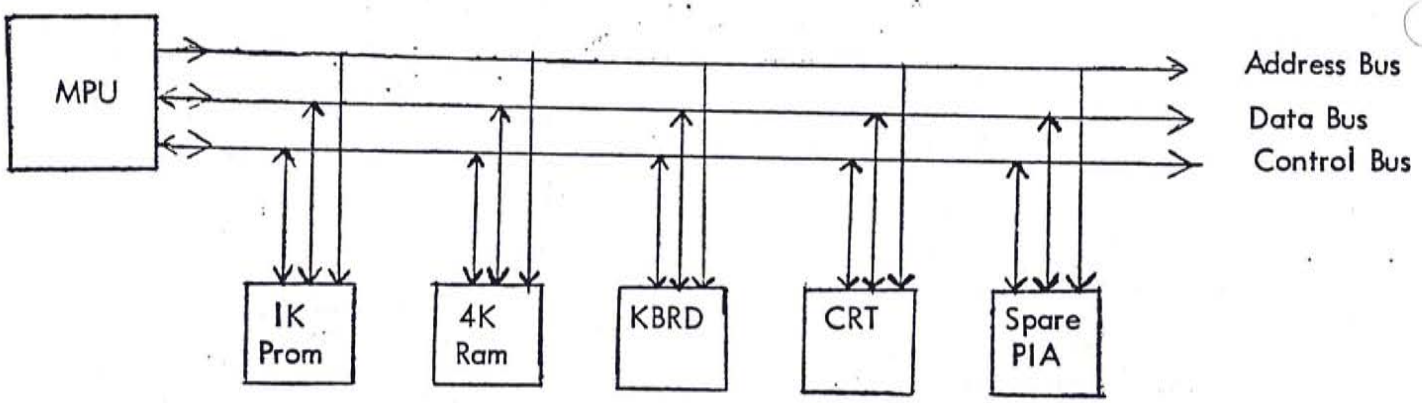
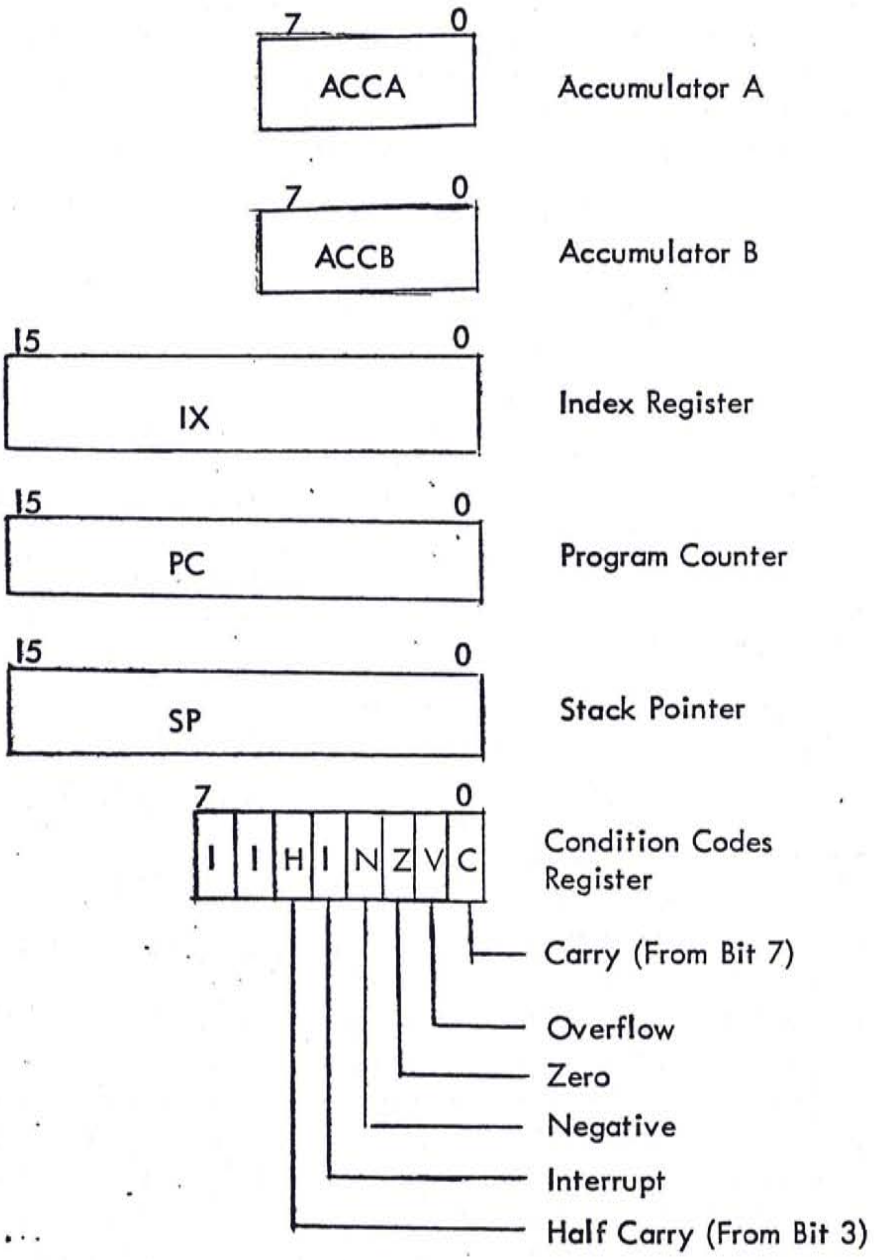


Figure 2

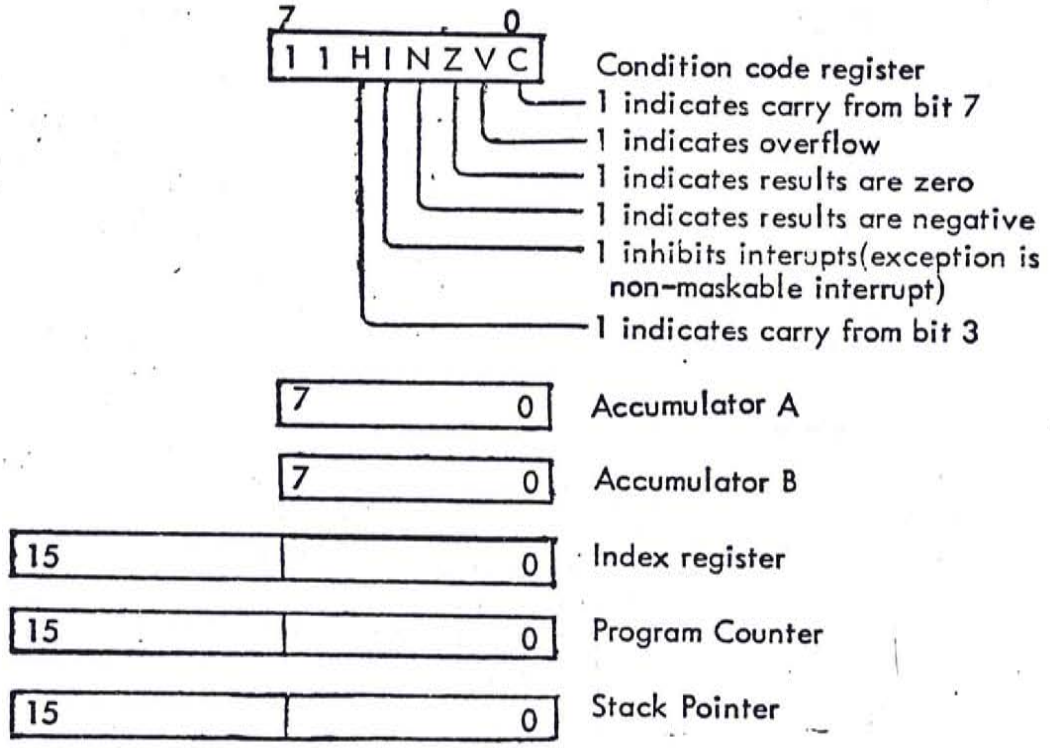
Fig. 3.1 shows a programming model of the MPU. This shows the registers that can be directly accessed by the programmer. The bit size of each register is also shown. The following is a general description of each module, which combined make up the basic SPHERE system.

Figure 3.1



Several of the components of the processor are not directly accessable to the user and are explained here for information purposes only. The processor contains a control unit which provides the necessary timing to control the instruction decoding and execution sequences. The arithmetic and logic (ALU) performs the actual arithmetic operations, such as addition, subtraction, etc., and all boolean type operations, such as AND, OR, etc. Each accumulator receives and transmits data to and from the bus and temporarily stores results from the ALU. The input and output buffers provide the access to the address and bi-directional data lines for the MPU. The instruction register holds the instruction read from memory. Its output is decoded by the control unit. Also shown are the main MPU and peripheral control lines. The remainder of the registers are accessable to the programmer. The working environment for programming consists of 72 instructions, seven addressing modes, six registers and memory (including the stack). Figure 3.2 gives a summary of the six registers.

Figure 3.2 Registers



A stack consists of a number of contiguous memory locations for temporary storage. The information stored on the stack may consist of status during interrupts, subroutine linkage return addresses, and data. The stack pointer is a 16 bit register containing the address of the next position in which to store information. As information is stored, the stack pointer is decremented to the next free position. As data is recalled from the stack, the stack pointer is incremented to the previous position, and then the data is retreived. It is the programmers responsibility to initialize the stack pointer register.

3.1 THE ONE CARD COMPUTER

The logical approach to the solution of any problem is to determine the minimum requirement for a satisfactory result. In the case of a small computer, an absolute minimum would include a reasonable amount of memory, a capable CPU, Input/Output capability, a real-time clock, read only memory program which if used in conjunction with a terminal can replace the computer's switch panel and display lights, and finally it ought to have a convenient Bus structure to easily attach other modules. The SPHERE ONE CARD COMPUTER includes all of the above features plus a few that also deserve attention. Below each of the principal features are listed:

3.1.1 THE CPU

The MOTOROLA 6800 microprocessor is the most advanced microprocessor available today. It reduces the necessity for support components and includes features not found on computers of many times the size. The IBM 370 for example will not store all of its registers automatically upon receipt of an interrupt as the 6800 system resembles the architecture of the Digital PDP-11 in many ways. These include instructions that "PUSH" data onto a STACK temporary storage is no longer required the data may be "POPed" off of the stack. The 6800 doesn't have as many registers as the 8080; however the 6800 has several addressing modes which in most cases completely outweigh its lack of registers. These modes are particularly advantageous when tables are processed. However, when three of four counters are being incremented or decremented the 8080 is faster, but in either case the 6800 is easier to program.

3.1.2 MEMORY

This system uses the 2107 A type 4K by 1 dynamic random access memory. This memory was used because it is the least expensive memory available and would lower the cost of the system. All refresh circuitry for the system is included on the CPU board.

3.1.3 I/O

If this board is used in a stand alone situation a CPU must communicate to the outside world. Therefore, the system is supplied with 16 programmable I/O lines as an option. Four additional lines which may be used as programmed interrupts are also supplied on the board.

3.1.4 REAL - TIME CLOCK

A stand alone process control system and many other systems require the capability of monitoring the progress of an activity. The SPHERE system has a real-time clock which will interrrupt the system at a set interval. This interval is a function of the refresh clock which is set at 1 ms. The interrupt may occur at 1x, 2x, 4x, 8x, or 16x the refresh rate. The interval may also be set externally. The rate is determined by a wire strap.

3.1.5 EPROM

The Erasable Programmable Read Only Memory used by the system is the 1702 A. Programmers for this EPROM are commonly available so that users may find programming the system for a stand alone application a reasonable task. When delivered with a SPHERE system the EPROM contains a Program Development Systems (PDS) which is described under "SOFTWARE".

3.1.6 BUS STRUCTURE

The BUS is driven by tri-state TTL buffers which are capable of driving 35 standard TTL devices. The BUS is connected to this board via three 14 pin dual-in-line connectors which will transmit and receive information over 3 14 conductor flat ribbon cables. Eight data, 16 address, BUS and control lines are transmitted bidirectionally to and from the CPU, memory, and perephials. I/O devices, buffer, and status registers are addressed as memory locations at the top (HIGH ORDER LOCATIONS) of memory in much the same as the digital PDP-11. This means that about 35,000 devices could be attached to the system (theoretically). It also means that any machine instructions may operate on device buffer and status registers as they would to memory. This limits the maximum memory on SPHERE systems to 56K instead of the theoretical 64K because the high order 8K is reserved for device status and buffer registers.

3.1.7 POWER-ON RESET

When power is applied to this board, circuitry forces a reset to the processor until the system power has had time to stabilize. The system will immediately thereafter jump to a specific location in the read only memory (EPROM) to begin meaningful processing.

3.1.7 POWER-ON RESET

When power is applied to this board, circuitry forces a reset to the processor until the system power has had time to stabilize. The system will immediately thereafter jump to a specific location in the read only memory (EPROM) to begin meaningful processing.

3.2 SYSTEM 310

This computer system is capable of satisfying the needs of the user who wishes to program, develop, and debug programs for light process control, experimenting, and some educational purposes. As with all SPHERE "SYSTEMS", the computer was designed to perform a useful function. It was not intended to be a useless computer with a lot of money spent on front console. All SPHERE systems are shipped with software and a commitment that software developed in the future by SPHERE or one of its users will be available at minimal cost. The PDS SYSTEM is included in the read only memory of this system. It and other software which is available is described in Section 9. Expandability has been considered from the onset. Some of these considerations include additional memory to 56K inter-computer communications, a full line of peripherals, home and industry utility, and lowering cost while increasing performance in the future. Below are listed the modules contained in this system:

3.2.1 CPU2

This module contains all of the features listed under "THE ONE CARD COMPUTER".

3.2.2 KBD2

This module includes a standard typewriter style alpha-numeric keyboard layout.

3.2.3 CRT 1

This module contains the necessary electronics to display 512 characters on a video monitor. The 64 character ASCII character set is displayed in a matrix of 32 characters by 16 lines. Each character is displayed in a matrix of dots, 5 dots wide and 7 dots high. To display a character a computer program simply moves the desired character into a memory position which is also the display refresh buffer. The refresh buffer is located in the high-order 8K of memory. It

consists of 512 bytes of static RAM that is organized to be accessed by the CPU and CRT simultaneously without degrading the access time to the CPU. Output from this module to the video monitor appears as a composite video signal. Etches for RF modulator (adjustable from channels 1-3) have been left on the PC board, and schematics have been provided; however, components have not been supplied because this type of circuit requires FCC testing and approval. Instructions for TV modifications are generally available.

3.2.4 PWR/2

The power supply has been designed expressly for the SPHERE system. It produces 5 volts at 3.5 amps, 12 volts at .7 amps, -5 volts at .2 amps, and -12 volts at .5 amps.

3.2.5 BCB 1

Each of the system modules is connected via a system bus. The bus consists of three flat ribbon cables containing 14 conductors each. Each cable is connected to each board via a 14 pin dual-in-line (DIP) connector. Each board has three standard 14 pin IC sockets where each of the three bus cables attach.

3.2.6 PCB 1

Power is bussed to each of the boards of the system via a separate 14 conductor ribbon cable. This cable is attached to each board via a 14 pin dual-in-line connector.

3.2.7 OPR 1

The operator/reference manual set is designed to introduce the SPHERE system to the new computer user. It describes in detail how each instruction works. It also describes in detail interrupts, stack operations, Input/Output, peripheral device characteristics, memory organization, projected device reserved locations and limited characteristics, and execution timing. Programming examples are included to illustrate various hardware features and a section is included to introduce programming concepts to the first time computer user. Appendices are included to aid program development. Although this manual set is comprehensive, some users may require further information so references are amply provided. The manual set is loose bound to receive updates and includes sections where Global Newsletters, kit assembly instructions, manuals, and maintenance manuals may be kept. Kit assembly

3.2 SYS 1 HOBBIEST

This computer system is capable of satisfying the needs of the user who wishes to program, develop, and debug programs for light process control, experimenting, and some educational purposes. As with all SPHERE "SYSTEMS", the computer was designed to perform a useful function. It was not intended to be a useless computer with a lot of money spent on front console. All SPHERE systems are shipped with software and a commitment that software developed in the future by SPHERE or one of its users will be available at minimal cost. The PDS SYSTEM is included in the read only memory of this system. It and other software which is available is described under the heading "SOFTWARE". Expandability has been considered from the onset. Some of these considerations include additional memory to 56K inter-computer communications, a full line of peripherals, home and industry utility, and lowering cost while increasing performance in the future. Below are listed the modules contained in the system:

3.2.1 CPU2 — This module contains all of the features listed under "THE ONE CARD COMPUTER ".

3.2.2 KBD2 — This module includes a standard typewriter style alpha-numeric keyboard layout. The keyboard module 2-key utilizes rollover.

3.2.3 CRT 1 — This module contains the necessary electronics to display 512 characters on a television or video monitor. The 64 character ASCII character set is displayed in a matrix of 32 characters by 16 lines. Each character is displayed in a matrix of dots, 5 dots wide and 7 dots high. To display a character a computer program simply moves the desired character into a memory position which is also the display refresh buffer. The refresh buffer is located in the high-order 8K of memory. It consists of 512 bytes of static RAM that is organized to be accessed by the CPU and CRT simultaneously without degrading the access time to either CPU or CRT (dual port memory). Output from this module to the video monitor appears as a composite video signal or separate horizontal, vertical, and video signals. Etches for RF modulator (adjustable from channels 1-3) have been left on the PC board, and schematics have been provided; However, components have not been supplied because this type of circuit requires FCC testing and approval. Instructions for TV modifications are included with purchase.

3.2.4 PWR 1

The power supply has been designed expressly for the SPHERE 1 system. It produces 5 volts at 5 amps, 12 volts at 3 amps, -5 volts at 100 mA, and -12 volts at 400 mA.

3.2.5 BCB 1

Each of the system modules is connected via a system bus. The bus consists of 3 flat ribbon cables containing 14 conductors each. Every other conductor is grounded to eliminate cross talk (electrical noise). Each cable is connected to each board via a 14 pin dual-in-line (DIP) connector. Each board has 3 standard 14 pin IC sockets where each of the three bus cables attach.

3.2.6 PCB 1

Power is bussed to each of the boards of the system via a separate 14 conductor ribbon cable. This cable is aitached to each board via a 14 pin dual-in-line connector.

3.2.7 OPR 1

The operator/reference manual set is designed to introduce the SPHERE 1 system to the new computer user. It describes in detail how each instruction works. It also describes in detail, interrupts, stack operations, Input/Output, peripheral device characteristics, memory organization, projected device reserved locations and limited characteristics, and execution timing. Programming examples are included to illustrate various hardware features and a section is included to introduce programming concepts to the first time computer user. Appendixes are included to aid program development. Although this manual set is comprehensive, some users may require further information so references are amply provided. The manual set is loose bound to receive updates and includes sections where SWAP newsletters, kit assembly instructions, manuals, and maintenance manuals may be kept. Kit assembly instruction manuals are a part of the package; however, each module in kit form contains an associated kit assembly manual which may be kept in this binder. SPHERE has introduced its user group to promote interchange of ideas, useful circuits, comments, gripes, software (from games to statistical packages), announcements (i.e. user has 10 Amp 5 volt power supply for $15.00 type!). The SWAP newsletter will not be governed by the marketing arm of the company. Hopefully the users will completely govern this group in the future. SWAP membership is included with any "SYSTEM" purchase or with the purchase of the OPR1 manual set. Future membership fees will be determined by users.

3.3 SYS 2 INTELLIGENT

This system was specifically designed to solve the needs of two different users.

1. The user who wishes to communicate to other devices over serial lines such as a telephone.
2. The user who wishes to utilize this device as a stand alone computer, and use the communications facility to save and restore programs and data using a standard audio cassette.

The communications facility is implemented as a single module (PC board) which contains a standard asyncronous communications interface and a modem. Serial communications to other devices such as a teletype or other computer may take place without the use of a modem. This system includes all of the features found in SYS 1/KIT plus the following:

3.3.1 SIM Serial Interface Module

This module provides the facility to communicate data in several serial forms. The module has two independent serial I/O ports. One can only be connected to a Kansas City standard cassette interface. The other can either be connected to a second cassette interface or any one of the following EIA RS-232-C interface, 20 mA current loop, direct TTL interface, or industry standard low-speed modem enabling communications over standard telephone lines. The ACIA's can accept data in 8 bit parallel format from the CPU and transmit it serially with a start bit and 1-2 stop bits. Seven or eight data bits may be transmitted with optional even or odd parity. At the same time data in same format may be received serially, presented to the CPU in 8 bit parallel format. The data will be checked for proper parity (if desired) and false start bits will be rejected. Communications may occur at several standard strap-selectable rates, which are 110,150,300,600, 1200,2400,4800, and 9600 Baud. Baudot Code can be supported with minor modification if 20% speed degredation is acceptable. X-on and X-off functions are provided by an on board relay. This module optionally contains a complete ORIGINATE/ANSWER modem. The modem will operate at a maximum speed of 600 Baud. The SH, RING, +V, DH, DA, DR, and GND signals are provided for the CBT type of DAA (Direct Access Arrangment). A speaker and a microphone are all that are required to complete the acoustic coupler. No cableing is provided with this unit.

3.4 SYS 3 BASIC

This system was designed for the user who wishes to have total stand alone program development. The features found in the earlier mentioned systems along with the additional 16K of memory included with SYS 3 supplies the user enough memory for major program design.

3.4.1 MEM 1

This module contains 16 K of memory which provides ample space for the Basic interpreter and user programs. The memory board is designed with four rows of memory chips. Each row has eight chips with 4096, 8-bit bytes per row. Total number of bytes per board is 16,384. Memory addresses are selectable by a wire jumper on the PC Board. The selective jumper is a 4-bit, full-adder chip, programmed to subtract out all addresses below the desired starting point. With this you can select the address area you want for memory (within 4K boundaries) storage without having to rewrite your program.

3.4.2 BASIC SOFTWARE

The Basic Software package included is a fully extended language which includes matrix operators, string operators and file functions, plus the capability of calling the assembler subroutines. The Basic package includes the following utility commands: APPEND, DELETE, GET, LENGTH, LIST, RENUMBER, RUN, SAVE, and SCRATCH. The operators are: = AND, OR, NOT, MAX, MIN. The statements are: CHAIN, COMMON, DATA, DIM, END, FOR ... NEXT, GO TO, GO TO... OF, GO SUB, IF ... THEN, IMAGE, INPUT, LET NEXT, PRINT, PRINT USING, READ, REM, RETURN, STOP. The functions are: DEF, ABS, EXP, INT, LOG, RNO, SQR, SIN, COS, TAN, ATN, LEN, SGN, TAB. Matrix operations are: COM, MAT IDN, MAT ZER, MAT CON, MAT INPUT, MAT PRINT, MAT +, MAT -, MAT *, MAT=, MAT TRN, MAT INV. File processing statements are: OPEN, PRINT #, READ #, END #. Full string processing is supported.

3.5 SYS 4 "CLASSIC"

Total computer capabilities are included in this system. SYS 4 has been designed for the user who needs large amount of memory and peripheral capabilities. Included is a 65 line per min., 80-column printer; also included are two IBM compatible floppy disk modules complete with FDOS disk operating system.

SYS 4 "CLASSIC" cont'd.

The disk operating system handles file maintainance, and provides an editor, assembler, and debugging facility with file handling extentions for thorough disk utilization. This system includes everything previously listed plus the following, less SIM board:

3.5.1 LPT 1

The SPHERE 1 Line Printer produces 80 columns of 5x7 dot matrix characters at 110 characters per second, or 65 lines per minute. The impact head prints bi-directionally on 8 1/2 inch roll paper using a conventional teletype ribbon. This line printer has been designed with high reliability and extremely low cost required by small scale data handling systems. This system features the ability to print double wide characters for heading and other applications. An adjustable width tractor feed mechanism is available for use with fan-fold forms. Up to four highly legible copies may be produced.

3.5.2 DSK 1

The disk interfaces to the SPHERE 1 via the peripheral interface module and disk cable assembly and is fully supported by the Flexible Disk Operating System (FDOS) package. The flexible disk used is media and format compatible to the IBM 3540 and 3740 with a maximum data storage capacity of 256,256 bytes per diskette. A single controller handles up to four drive units which may be individually write-protected. Hardware track seek and seek verification as well as CRC generation and verification insure data validity.

3.6 PIM (Peripheral Interface Module)

Both the Disk and the Line Printer are interfaced through this module which provides up to 64 digital I/O lines and 16 control lines (4 PIA's). The module also includes some circuitry dedicated to the Line Printer interface. The LPT 1 interface requires one PIA and Disk interface requires 1 1/2 PIA's. Strapping is provided to address up to 8 PIM boards on a system.

OPERATING NOTES FOR SYSTEM 1

P.D.S. (V3A)

V3A for the Sphere System 1 consists of a curser based editor, debugger, assembler and a set of utility routines.

When power is turned on in the system, the screen should blank and the curser start blinking on the second line of the screen. If the power-up did not trigger a hardware reset (the screen has a set of random characters on it, the same as when the EPROMs or the CPU are not attached), hit reset (~~CONTROL, SHIFT, RESET~~). (2 unmarked keys)

The system is now in EXEC mode. The three commands accepted are ctrl E, ctrl D and ctrl A. If the control key is not pressed with the other key the curser will move to the next line down. A command should be given before the curser reaches the bottom line as power-up may have left the editor in the wrong mode.

Ctrl E will set the editor to the on mode, clear the screen and home the curser (upper left hand corner). The computer is now ready to accept text. As many lines can be entered as desired keeping in mind that there is no checking for buffer overflow. When the screen is cleared, it is filled with hex 6Ø, which is displayed as a blank, and is used as a carriage return indicator by the editor. Because of this all edited text uses hex 6Ø instead of hex ØD for carriage return. (the PUTCHR routine accepts either 6Ø or ØD as a CR). When a line is moved off the top of the screen it is stored in the low area of the buffer. When it is moved off the bottom of the screen it is stored in the Hi area of the buffer (from BUFEND down). When scrolling down and the bottom of the buffer is hit, the curser is set to the bottom of the screen to indicate that the first line of text has been reached. A carriage return is also inserted before the first line. As the first line is now a blank line, the curser can be homed and the screen scrolled down one line so a line of text can be inserted before the previous first line.

To exit the editor, scroll up until all the text that is to go into the buffer has moved off the top of the screen, then push the escape key. The computer will then re-enter the EXEC mode. BUFFLO will point to the highest byte stored in the buffer memory upon exit.

The only restriction on the editor is that each line must have a carriage return. It is OK to insert or delete anytime and the first line may consist of no characters. Remember that the space bar must be used to insert spaces over the CRs on the screen.

If the editor was used to write a source program, the next command after scrolling the source program off the top of the screen will be ctrl A to assemble the program. The assembler should assemble and then exit to EXEC mode.

The assembled program's object code will start at BUFFLO unless an equate was made (a space in position Ø, followed by an equal sign, with the starting address starting in position 7 on). The first line or any other line can not consist of only a carriage return. If such a line is encountered all sorts of wierd things will probably happen. If a line is to be used from comments only, space over (with the space bar) to position 9 before starting the comment.

After the program has been assembled, type in ctrl D to enter the debugger. The starting location can then be opened. To open a location, type in 0 (alpha) followed by the number. The number (of a varying length of characters) must be terminated by an escape character. As the unit is in screen edit mode during number input, curser controls are in effect. Typing a G command after opening the start of the program will start program execution. Typing in an invalid command will cause some instruction to be executed, as instructions are determined by range instead of by character.

To re-edit a source program, open location FC68 and jump to it. This will enter the editor in re-edit mode. The existing text can then be scrolled down onto the screen. To enter the exec from DEBUG, open & jump to location FC18. A reset should be performed after ESC from re-edit. Carriage returns should follow only after col. 7 in assembly source programs. If a CR comes before col. 7, it is possible to pick up data from the next line.

The cursor is formed by setting bit 7 of any CRT buffer location to a 1.

The terminating character for the editor is permanently set to escape. Both BUFEND and ENDMEM are set to 4095. The assembler terminates when a character other than a blank or an ' = ' is found in col.1, so if 'END' is typed on the last line of the program the assembler will stop.

The SWI hardware interrupt vector at location FFFA (on SYS 1) points to the debugger at location FE4F instead of low memory Ø1ØØ as shown in the memory map. Also, DSTASM (28) is replaced by BUFFLO (2Ø).

The following program, 'CONVRT', is an example of a typical program making use of the mini-assembler and the utility routines. It accepts a number in octal, decimal, or hexadecimal and converts it to a number in any base between base 2 and base 42. The number is typed in in the same manner as in Debug, followed by the base (on the next line) in octal, decimal or in hexadecimal. The converted number is then printed out. The number the user types in is terminated by an escape.

PROGRAM NAME CONVRT VERSION 1
PROGRAMMED BY ERIC JAMESON
DATE OCT 1,1975
PAGE 1 OF 1

	MACHINE				SYMBOLIC				
LABEL / EQUATE	OP CODE	OP TYPE	OPERAND HEX,OCT,DEC		LABEL	OP CODE	OP TYPE	OPERAND	COMMENTS
Ø 1	3 4	6	7 8						
=			4Ø Ø						PROGRAM ORIGEN
S	BDE	F	EF4		START	JSR	E	IMPNUM	READ IN A # FROM THE CRT
	D7	D	CØ			STAB	D	TMPB	} SAVES BINARY # IN TMP LOC.
	97	D	C1			STAA	D	TMPA	}
	BDE	F	D16			JSR	E	CRLF	CARRIAGE RETURN - LINE FEED
	BDE	F	EF4			JSR	E	IMPNUM	INPUTS BASE FROM CRT FOR CONV.
	97	D	Ø5			STAA	D	AR2	}
	4F					CLRA			} SETS BASE FOR CONVERSION
	97	D	Ø4			STAA	D	AR3	}
	BDE	F	D16			JSR	E	CRLF	NEXT LINE
	D6	D	CØ			LDAB	D	TMPB	} RESTORES # FOR CONVERSION
	96	D	C1			LDAA	D	TMPA	}
	DE	D	1C			LDX	D	CSRPTR	SETS OUTPUT POINTER
	BDE	F	F64			JSR	E	BINASC	CONVERTS BINARY # TO ASCII
	BDE	F	D16			JSR	E	CRLF	NEXT LINE
	2Ø	R	@S			BRA		S	RETURNS FOR NEXT NUMBER
END									

↑ SPACE ↑

9.1 PROGRAM DEVELOPMENT SYSTEM

PDS represents a unique approach to the software of a low cost computer system. By the addition of software routines located in a 1 K PROM even the smallest system is capable of doing useful work as well as performing all of the functions of a switch panel and display lights. The software, consisting of a debugger, assembler, editor, 16 bit arithmetic, and ASCII conversion routines, allows for development of user software in much the same way as large disk based systems. The SPHERE Debugging Aid (SDA) is designed to aid in program development, as well as replace the usual switch panel and display lights function. It allows the user to easily view and alter the contents of memory or CPU registers from the keyboard-CRT display. The Mini-Assembler allows the user the ability to input source assembly language programs and output unrelocatable binary object code. It can handle up to 62 symbolic addresses, different operand sizes and octal, decimal and hexadecimal operands. The operation codes are entered in hexadecimal (i.e. ADDA immediate is "8B"). Included is a set of routines for 16 bit arithmetic manipulation. The routines include 16 bit multiply and devide as well as ASCII-to-BINARY and BINARY-to-ASCII conversion routines operating on 16 bit binary numbers. It also includes software subrutines to perform input-output compatable with the SCOS on audio cassettes, modems or RS232 type serial or current loop serial input-output thru ACIA's. The system includes a built in CRT based editor allowing scrolling and text insertion and deletion based on a cursor, allowing easy text manipulation.

9.1.1 PDS SYSTEM COMMANDS

The SPHERE Program Development System is contained wholly Programable Read Only Memory. When power is turned on, or when the Control Shift and Reset keys are simultaneously pushed on the keyboard the system initializes its parameters and goes into command mode. Entry into Command Mode causes the display to blank and enables input from the keyboard. All valid commands are a single character. If a Non-command character is entered it will have the same effect as a Carriage Return. The acceptable Command Characters are listed below with their associated function:

"Ctrl A"	ASSEMBLE
"Ctrl D"	DEBUG
"Ctrl E"	EDIT
"Ctrl R"	Re-edit (on V3D, V3N & SYS2N only)
"Ctrl I"	Initialize cassette (on SYS2N only)
"Ctrl L"	Load (on SYS2N only)
"Ctrl S"	Save (on SYS2N only)
"Ctrl G"	Go (on SYS2N only)

These functions are described in detail in succeeding sections of this manual.

9.1.1.1 MINI-ASSEMBLER

This module is entered via the Ctrl A Command from the keyboard. The Mini-Assembler assembles source code from a fixed position in memory. This position is the location where the editor deposits source code. This assembler requires that only one statement reside on a line at a time. Each line to be assembled must appear in fixed format. Below is a description of the assembler statement format.

Example	B	_	_	I	F	_	E	*	2	3	4
Position On Input Line	Ø	1	2	3	4	5	6	7	8	9	10

LOCATION LABEL: Position Ø

This position contains any one of the Sixty three ASCII characters or a blank. Placing a non-blank character in this position allows the user to access the location in other places by a label rather than an address.

EQUATE: Position 1

This position may contain an "=" or a blank. If it is equal to "=", then the location label (position) will reference the location specified by the operand positions (7 +).

OP-CODE: Positions 3 & 4

These positions may contain spaces or a two digit, hexadecimal equivalent of an instruction code. If the field is blank no allocation of memory will be made for the instruction code. Otherwise the supplied two hexidecimal digits will be converted to a single byte and be deposited to current location.

OPERAND TYPE: Position 6

This field may contain one of the three letters R, D, E, or a blank. If the operand type is blank no operand will be evaluated and no allocation of memory will take place. An "R" specifies that the operand in positions 7+ will create one byte of data relative to the current label assignment plus 2. This operand type is effective for branches a "D" specifies that the operand in positions 7+ will create one byte of data. An "E" specifies that the operand in positions 7+ will create two bytes of data.

OPERAND VALUE: Positions 7 +

These positions must be terminated by a space or a carriage return. The four forms of operand values are described below:

(1) @ - The at sign "@" followed by a single character indicates that data shall be referenced at the location specified by the letter following the "@". The reference is to the last definition encountered before it is used. The definition can be made on either the first or second pass.

9.1.1.2

(2) XXXX - where XXXX is a hexidecimal number value. If a number overflows the 16 bit BA register, only the low order 16 bits are saved.

(3) .YYY - where YYY is a decimal number. The period indicates that the number is in decimal digits.

(4) *ZZZ - where ZZZ is an octal number. Any leading ASCII character from hex 00 to 2D indicates that the digits following are octal digits.

The assembler returns to command mode on completion. The start of executable code will be at the end of the source text or at a location specified by an equate using the PC (a blank followed by the " = ").

To end a program, type 'END' on the last line.

An origen can be made by the statement " = 123", allowing the user to specify placement of the object code by the number in col. 7.

The following sample program is an infinate loop back to itself. Upon exit from the assembler, D (examined with the T command in the debugger) will contain 4ØØ and location 4ØØ will contain an Ø1.

```
   =                 4  Ø  Ø  1     set origen to 4ØØ
D                    1              no-op
   Ø  1              @  D  0        branches back to D
E  N  D                             exits the assembler
Ø  1  2  3  4  5  6  7  8  9
```

If the assembler did not put the code at the proper origen, it is probably because the cursor was moved off the top of the screen (this puts a blank line at the start of the text) and the line thusly added was not filled with blanks past column 7. Always put in spaces to column 7.

9.1.3 SPHERE DEBUGGING AID (SDA)

The SPHERE Debugging aid is designed to Aid the user in debugging his programs. It allows the user to perform functions equally done through the front console, such as modify the contents of a location or start program execution, as well as debug programs. It is entered from command mode by typing a Ctrl D. Initially the "> " will appear indicating that a debug command will be accepted. The following are the acceptable debug commands:

(SYS2N, V3N, V3D)	(V3A)	
" ↵ "	"↵"	LINE when a carriage return is typed, any location being referenced will no longer be available for examination without typing the address again.
" ␣ "	"␣"	CHANGE after the contents of a location have been displayed and the user wishes to change the contents of the location, he may type a space followed by an octal, decimal or hexadecimal number following the format at 9.1.1.2 to insert new data.
+	+	OPNNXT when the plus ("+") is typed the location following the last displayed location will be displayed and available to other commands.
-	-	OPNPRE when the minus ("-") is typed the location prior to the last displayed location will be displayed and available to other commands.
↑B	B	BRKSET this feature causes program operation to cease at the current location whenever it is encountered. This breakpoint instruction is accomplished by placing a SWI instruction in the last examined location and saving its prior value in a location in low memory. This feature is requested by typing a "B". Breakpoints are used in place of a front console single step switch.
↑C	C	CLRSET The breakpoint is cleared when a "C" is typed. The location that was replaced by a SWI instruction by breakset is returned to its original value.
↑E	E	EXIT an "E" is typed when the user desires to resume operation after a breakpoint has been encountered. A "C" must be typed prior to use of this routine.
↑G	G	GO when a "G" is typed Control is transferred to the program being debugged at the location last examined.

NOTE: an ↑ refers to the control key.

↑ O O OPNLOC opens the location whose address is typed in and displays the byte contents in hexadecimal format. The address is typed in immediately after the O and can be a octal, decimal or hexadecimal number following the ASCII number conversion formats (9.1.1.2). The address is then printed out on the next line as a hexadecimal number followed by a space. The format is:

```
> O X X X X

> Y Y Y Y    Z Z
```

where XXXX is the address in octal, decimal or hexadecimal terminated by an escape character (to exit from the editor). YYYY is the hexadecimal address; and ZZ is the hexadecimal contents of the opened byte.

↑ R R OPNREG or OPNTOS This instruction opens the Top-Of-Stack. This is used for stepping through the contents of the stack. When DEBUG is entered by a breakpoint, this instruction is used for examining and modifying the CPU registers. When a breakpoint is encountered, the register contents are put on the stack in the following manner;

Ø. Condition Code
1. Accumulator B
2. Accumulator A
3. Index Register HIGH
4. Index Register LOW
5. Return Address HIGH
6. Return Address LOW

By stepping through the stack with the + command, the registers can be examined and changed.

NOTE: the return address for the breakpoint is calculated in the Exit instruction in the V3A prom set while in the V3D, V3N and SYS2N it is calculated when the breakpoint passes control to the debugger.

↑S	S	SETSTK Set stack command. This instruction sets the stack pointer to point to the currently opened location. This is used for changing the stack location from its initial reset position of 512_D.
↑T	T	OPNTBL Opens the symbol table. The location opened is the address of the first character symbol following the T command. The symbol is any one of the symbols used by the Assembler, including the space symbol (program counter).

To exit from the debugger to a user program, use the E or G commands. To exit back to the executive command mode interpreter, open location EXEC (FC18) followed by a G command. (V3A only)

* * *

↑J	(not on V3A)	JUMP Jumps to a user subroutine. This instruction uses a JSR to exit from the Debugger while the G (go) command uses a JMP.
↑X	(not on V3A)	EXEC This command exits from the Debugger and returns to the executive command interpreter.

9.1.4 EDITOR

The editor is a routine that does input to a buffer from the keyboard/CRT. It allows for modification of text by cursor manipulation. It is entered by the character ctrl E typed in the command mode. The editor allows for scrolling in order to allow editing of a text segment larger than can be displayed on the screen. If the cursor is moved off either end of the screen, it causes text to be scrolled up or down. The editor commands are as follows:

KBD/2	KBD/1	
Cntl L or Line Feed	HOME	sets cursor to home position.
Cntl X or CLEAR	CLEAR	clears the screen from the cursor position to the bottom of the screen.
Cntl R	→	moves the cursor one position to the right.
Cntl T	←	moves the cursor one position to the left.
Cntl Q	↑	moves cursor up one line.
Cntl S	↓	moves cursor down one line.
Cntl DELETE	KBD ← Cntl	moves cursor to left of screen.
	↵	carriage return - puts cursor to left of screen on the next line.
	Cntl D	deletes the bottom line on the CRT. (top line on V3N)
	Cntl I	insert new line at last line on the CRT (top line on V3N)
	ESC	exit from the editor.

WARNING

BUFFER OVERFLOW MAY CAUSE LOSS OF DATA IN THE BUFFER.

NOTE: Each line must have a carriage return for scrolling.

NOTE: Don't scroll off screen in EXEC until after editor has been run.

NOTE: For keyboards other than the KBD/1 that do not contain the optional cursor control keypad, the equivalent control characters must be used for cursor movement.

9.1.4.1 RE-EDIT

The re-edit command is called from the executive and allows source text to be revised. When entered, the previously edited text is in the low part of the edit buffer. This text can now be scrolled back down onto the screen from the edit buffer by moving the cursor off the top of the screen. Note that if the cursor is moved up past the top line, the editor will move the cursor to the bottom line and insert a blank line (a line with a carriage return only) at the beginning of the text. This command is on the V3D, V3N, and SYS2 prom sets only.

The V3N and V3d prom set also have changes on the debugger. Two new commands have been added ;

↑ J Jump - performs a JSR to a users subroutine

↑ X Exit to Exec - exits the debugger and goes back to the executive

All the debugger letter commands are now control characters, i.e. the R command is now Ctrl R.

Exec has a new command "Ctrl R", (Re-edit) which allows the editor to be entered with the previous text intact. With a ↑ R the old text can be scrolled back onto the screen.

Insert and Delete commands in the editor are now at the top of the screen instead of the bottom of the screen.

The utility routines are a set of subroutines residing in PROM and used by other PDS routines. They are called by:

JSR E, SUBNAME or

BSR SUBNAME

and made use of reserved locations in low memory as described in the MEMORY MAP. All parameters are passed through low memory locations or registers.

	PDS V3A addresses	PDS V3N (V3D) addresses
BINARY TO ASCII	JSR E,FF64	FF64

The BINASC routine converts a 16 bit number in BA (CPU register B contains the HI byte, register A contains the LO byte) into a string of ASCII digits starting at the address in the X register. Conversion can be any base from base 3 to base 41. For bases greater than 10, the ASCII alphabet A-Z and the following characters are used for representing digits, as is the case with hexadecimal. The base is specified in ARB. On output, the index register X points to the last digit in the number plus one.

ASCII TO BINARY	JSR E,FF22 or	FF22
	JSR E,FF37 with base in ARØ(base 2 thru 16)	FF37

This routine converts from a ASCII number string pointed to by X to an unsigned binary number in BA. Conversion is from Octal, Decimal, or Hexadecimal. The base is specified by the first digit of the string. Octal has an '*' for the first digit. Decimal has a '.' for the first digit. Any other digit will default to hexadecimal. This routine terminates when any nonhexadecimal digit is encountered.

MULTIPLY	JSR E,FF93	FF93

The MULT routine multiplies 2 unsigned 16 bit numbers to give a 16 bit result.
BA ← BA x ARA

DIVIDE	JSR E,FFAF	FFAF

This reentrant subroutine divides the 16 bit number in BA by the 16 bit number in ARA in low memory.

$$\frac{BA}{ARA} \rightarrow \begin{matrix} \text{BA-quotient} \\ \text{ARA-remainder} \end{matrix}$$

CRLF	FD16	prints carriage return - line feed.	FD14
SUB 32	FCBE	moves curser up one line.	FCCB
ADD32	FCC8	moves curser down one line, CSRPTR is passed to ADD32 in X.	FCD5
LFTJST	FD0F	moves curser to the left of the screen	FCFD
HOME	FC32	homes curser	FC37

PDS V3N (V3D) addresses		PDS V3A addresses
FC4A	GET A CHARACTER	FC45

The GETCHR subroutine inputs one single character from the keyboard while blinking the cursor at the screen position pointed to by CSRPTR. The character value is returned in accumulator A.

FCBC	DISPLAY A CHARACTER	FCAD

The PUTCHR subroutine takes the character in accumulator A and displays it on the CRT at the position pointed to by CSRPTR. The cursor is then advanced by one. If the character printed to is a carriage return, then the cursor is put to the first position on the next line. Also scrolls.

FC67	INPUT A STRING	FC60

The IMPSTR routine (the editor) allows a string of characters to be read in from the keyboard, echoed on the CRT and stored in buffer memory. IMPSTR contains the editor described above. It starts inserting characters at the location specified in BUFADR and continues until BUFEND is hit or the character in ENDCHR is typed. At restart time, BUFADR is set to 512_D, BUFEND to the end of memory and ENDCHR to the ESC character.

FD8E	OUTPUT A STRING	FD8D

The OUTSTR routine displays the character string from OUTBUF to OUTEND on the CRT starting at the current cursor location.

FB91	INPUT FROM CASSETTE	Not Applicable

The INCASS routine reads in a block of data into BUFADR. It is also called from command mode by the L command.

FB2D	OUTPUT TO CASSETTE	(Initialize Cassette with FBØØ before use)

The OUTCAS routine dumps the memory data between BUFADR and BUFEND onto cassette tape. It is called from command mode by the S command.

FE71	IMPCHR	does getchr and putchr.	FE5C
FEE4	IMPNUM	inputs a number from the current curser location; BA←number	FEF4
FE64	DEBUG	JMP E,FE4F debugging routine	
FC6F	REEDIT	FC68 editor entry for re-editing text	
FC73	EDITRD	FC6C Re-edit without clearing the screen	
FC75	EDITIN	FC6E edit on screen without storing in buffer (SCNPTR set to start of text)	
FFØ2	PNTBYT	FECC Prints acc. A in hex at current curser position.	
FC3D	CLEAR	FC38 Clears screen from curser	

DIVIDE This subroutine divides the 16 bit number in accumulators BA by the 16 bit number in ARA. This routine modifies the index register.

$$\frac{BA}{ARA} \rightarrow \begin{matrix} \text{BA-quotient} \\ \text{ARA-remainder} \end{matrix}$$

EDITIN The EDITIN routine provides an input routine that can read a screen of text using cursor controls and carriage returns and returns to the caller when an ESC is pushed. Low memory location SCNPTR points to the start of the text.

EDITOR The Editor routine allows a string of characters to be read in from the keyboard, echoed on the CRT and stored in buffer memory. Editor contains the editor described above. It starts inserting characters at the location specified in BUFADR and continues until BUFEND is hit or the character in ENDCHR is typed. At restart time, BUFADR is set to 200_X, BUFADR to FFF_X. ENDCHR is permanently set to the ESC character. On input accumulator A should be non-zero.

EDITRD This entry point to the REEDIT subroutine skips the blanking of the CRT. Like the EDITOR and REEDIT subroutines, accumulator A should be non-zero.

GETCHR The GETCHR subroutine inputs one single character from the keyboard while blinking the cursor at the screen position pointed to by CSRPTR. The character value is returned in accumulator A, the cursor pointer is in the index register as well as CSRPTR.

HOME homes cursor. Both CSRPTR and the index register are reset to the first location of CRT display.

INPCHR does GETCHR and PUTCHR.

INPNUM Inputs a string of characters terminated by an (ESC) character then calls ASCBIN to convert the string to a binary value in accumulators B,A.

LFTJST moves cursor to the left of the screen. Uses accumulator B.

MULT The MULT routine multiplies 2 unsigned 16 bit numbers to give a 16 bit result. Accumulators BA X ARA → BA. The index register is used in this routine.

```
^C/E                    as system starts up in EXECUTIVE mode, enter Editor
=     400CR            origin this assembly to 400x
S  BD EFEE4CR          define label S to this instruction, S JSR INPNUM
   D7 IC0CR                    STAB  TMPB   save input number
   97 DC1CR                    STAA  TMPA   to be converted
   BD EFD14CR                  JSR   CRLF   output carriage return - line feed
   BD EFEE4CR                  JSR   INPNUM input number for base
   97 D05CR                    STAA  AR2    set up number for base
   4F  CR                      CLRA         of conversion
   97 D04CR                    STAA  AR3
   BD EFD14CR                  JSR   CRLF
   D6 DC0CR                    LDAB  TMPB   restore number to convert
   96 DC1CR                    LDAA  TMPA
   DE D10CR                    LDX   CSRPTR pick up address to output to
   BD EFF64CR                  JSR   BINASC convert and output the number
   BD EFD14CR                  JSR   CRLF   output a carriage return - line feed
   20 R@SCR                    BRA   S      do it again
END^C                          END          last statement of assembly is END
seventeen CR 's         scroll all text off screen with CR's
ESC                     exit the Editor
^C/A                    Assemble the Editor's output
^C/D                    execute Debug
.C/O 407ESC             open a location and set a breakpoint to show
0407 BDC/B              how it works, not necessary for proper execution.
.C/O 400ESC             open first location of program and begin
0400 BDC/A12ESC         execution.  Enter number to convert
.C/B                    breakpoint encountered, Debug entered automatically
01F9 xx+                open top of stack, see condition code register,
01FA 00+                     accumulator B,
01FB 12+                     accumulator A,
01FC xx+                     index register HIGH,
01FD xx+                     index register LOW,
01FE 04+                     program counter HIGH,
01FF 07C/C                   program counter LOW, clear breakpoint
0407 BDC/E              automatically displays restored value. Exit breakpoint
^2ESC                   enter base
1010                    12x converted to base 2=10010
^.80ESC                 enter number to convert, 80A
^10ESC                  enter base, 10x=16A
50                      output converted number, 50x
^*527ESC                enter number, 5278
^.40ESC                 enter base, 40A
8N                      output converted number, 8N40
```

PDS OPERATION

FIGURE 9.2

SECTION 10

10.1 SPHERE BASIC SYSTEM

The SPHERE basic computer system consists of:

1. MOTOROLA MC6800 MICROPROCESSING UNIT with control logic.

2. 1,024 8 bit words of Reprogrammable Read only Memory Preprogrammed with the Micro Editor, Assembler, debugger, load & dump routines and the Mini-Executive

3. 4,096 8 bit words of Dynamic Random Access Memory

4. ASCII Keyboard for inputting data

5. TV (or CRT) Driver logic for output display of data (TV not supplied)

6. Peripheral Interface Adapter to allow user to interface his own 1/0 device (s).

7. Power supply, which supplies power to all components of system.

10.1.1 BUS STRUCTURE

The System Bus provides the address, control, and bi-directional data lines between the CPU and the peripherals via three separate flat ribbon cables and 14 pin dual-in-line connectors. These cables are carried from one device to another in a common bus fashion. All peripheral devices are tied onto the bus in parrallel and constantly monitor the address lines. If a device detects its own unique address, it responds as required, depending upon whether the data transfer is a read or write function. Each signal is driven from the CPU through Tri-State buffers. This technique of a common bus 1/0 structure makes it possible to connect many devices to the bus and makes interfacing requirements simpler. The signals that make up the System Bus will be described under the CPU description.

10.1.2 GENERAL THEORY OF OPERATION

The only thing any computer can do, from the largest to the smallest, is transfer data from one place to another. During transfer, however, the data can be altered or manipulated in various ways by the CPU, such as adding two pieces of data together and transfering the results to a device such as memory. This function of transfering of data is done under program control to accomplish a given TASK. In addition, the CPU is capable of making logical decisions based on results of tests performed on data and altering the program sequence or flow accordingly. The computer cannot think for itself - it performs sequences of "Instructions" that it understands and can act upon which have been stored in memory in the form of a functional program. The program counter is the register the CPU uses to determine which instructions to perform - it "points" (or provides the address) to the instruction to be executed in memory.

A typical sequence to perform an Addition of two number is as follows:

LDAA

ADDA

STAA

1. The CPU sends an address on the address bus, which corresponds to the location of an instruction to load the accumulator with one of the operands. The memory responds by sending the 8 bit binary equivelant of the instruction to the CPU which then stores the Instruction in the "Instruction Register" (This sequence is called "Fetching" the instruction.) The CPU control logic decodes the instruction and determines that it is a "Load Accumulators" instruction. It then computes the address of the first operand, puts this address on the bus and receives the operand from that address in memory. This operand is received and stored by the CPU in the accumulator. The program counter (PC) is now incremented by one. The CPU then puts the PC address on the bus and fetches the next instruction from memory. The instruction decoder determines that this is an instruction to add the contents of another memory location to what was previously stored in the accumulator. The CPU then accesses memory again to obtain the second operand stored there. When it receives the operand the CPU supplies both operands to the Arithmetic and logic Unit (ALU) which does the actual addition.

The result of the addition is temporarily stored in the accumulator. The CPU then accesses memory to fetch the next Instruction, which tells the CPU to store the contents of the accumulator in some location in memory. Note that the only thing the CPU did was to transfer two operands from memory, add them together and then store the result back in memory. If the result of the addition was zero, or a negative number, or the result of the addition was a number to large to fit in the 16 bit accumulator (overflow), then, if desired, the program could have tested the condition codes register for these conditions and altered the program sequence accordingly. So the computer can only do what it is told, in a sequential fashion, but does it at such high speeds compared to a human, that it becomes a tremendously valuable tool.

10.1.3 CRT THEORY OF OPERATION

The CRT control module stores data received from the CPU in a 512 word static RAM, arranged as a 16 line by 32 word buffer. The CRT buffer memory can be accessed from the CPU or by the address generators E6, 10, 11, 22 on the CRT board when the screen is being updated. E3, 4, 5 determine whether the CPU address lines or the CRT generated addresses will be supplied to the buffer. This is done via the CPU address decode logic and the resultant "CPU SELECTED" signal. (i.e., the buffer is a dual-port memory; either the CPU is accessing it or the CRT update circuitry.) Since the buffer looks like memory to the CPU, it can be written into or read out of. The data transfer direction is determined by the R/W control line.

The Video display format is comprised of 16 lines containing 32 characters each. Each character is made up of 35 dots in a 5 x 7 arrangement with 5 horizontal and 7 vertical dots. A character generator (2513) decodes the ASCII data provided into the correct dot pattern for the character being displayed. Horizontal spacing of characters is accomplished by displaying a blank dot column between each character. Vertical spacing is accomplished by sweeping three blank video lines between each character. The first line A B C D Ø is generated by having the row counter E22 feed zero bits into the row select of the character generator. Then as the row counter counts off Rows 1 thru 7, Rows 1 thru 7 of the character are decoded and processed but when E22 pin 11 sees the 8 and 9 counts of the row counter through E27 pin 3 its output goes low thus enabling the video blanking circuitry which places all zeros on the row select of the character generator creating the other two blank lines.

Along with video information a vertical and horizontal Sync must be supplied.

Oscillator E29 initiates the horizontal Sync pulse which is fed thru inverter E16 to E24 pin 5 and 10 where a 4-us horizontal Sync pulse is generated. The pulse goes to E12 pin 5 where it is wire or'ed with the vertical Sync pulse.

The falling edge of the Sync pulse on E24 pin 8 triggers E31 a one shot, which puts out a positive pulse on pin13that is adjustable by R19 from 4 to 20us. A delay pulse causes a lag between the beginning of a video sweep and the keyboards generation of characters, giving an adjustable left hand margin. $\overline{Q}$ of E31 pin 4 inhibits dot oscillator E31 pin 12 through and - or - invert gate E28. E31 pin 13 resets E6 and E11, the 16 - bit counters that keep track of the selected horizontal character. Since we are starting a new line, we must clear the counter to prepare it for in coming data. At the end of a negative going pulse on pin 4 of E31 the row counter, E22, is incremented and in the case of a ripple carry, E10, the line counter is incremented as well.

Keeping track of the 10 rows of horizontal lines forming each character is the job of E22 a decade counter which has a unique BCD output for each seven rows of dots and three blank rows used for painting a character and supplying vertical spacing.

A 4 bit counter E10 keeps track of the 16 sets of ten row lines. Together E22 and E10 provide a distinct BCD code for each of the 10 x 16 video scan lines. A complete video frame contains around 262 lines. Since the scan line counter E22 and E10 can only count to 160 we allow it to run for another cycle. E23 has been in the high state during the last 160 scan lines and is now toggled via and gate E27 pin 8 and NAND gates E24 pin 11 and E18 pin 8. The video blanking circuit for the character generator is activated when E23 pin 9 (Q) is toggled low and forces the generation of blanks from the character generator.

This continues until the line counter gets to 40. Lines 40 through 50 are then used to generate the vertical Sync. NAND gate E18 along with inverters E16 pins 8,10 and 12 perform the actual line number decoding. The out put of the horizontal oscillator is NAND'ed as well in E18 pin 6 with the line counter data, this chops the vertical Sync signal required by the video monitor. The output at E18 pin 6 is fed to E12 pin 4 where it is joined with the horizontal Sync to form the composit Sync signal at the output of ANDgate E12 pin 6. At line 50 the vertical Sync generation is stopped and the line and row counters continue their count to 104 which is decoded by E18 at pin 8. The $\overline{Q}$ output of E23 is NAND'ed by the decoder E18, since the 102 count is not significant when in the "display dot video" mode.

The output of E18 at pin 8 in turn generates a positive clock pulse to E23 via AND gate E23 making the Q output of E27 high again as it was in the beginning. The same signal from the output of E18 pin 8 resets row counter E22 and line counter E10 to Ø thus completing 262 lines required for the Raster. Again briefly the 264 lines required for a full video frame are generated as follows. 160 lines of video, 40 lines of blanking, 10 lines of vertical Sync and 54 more lines of blanking.

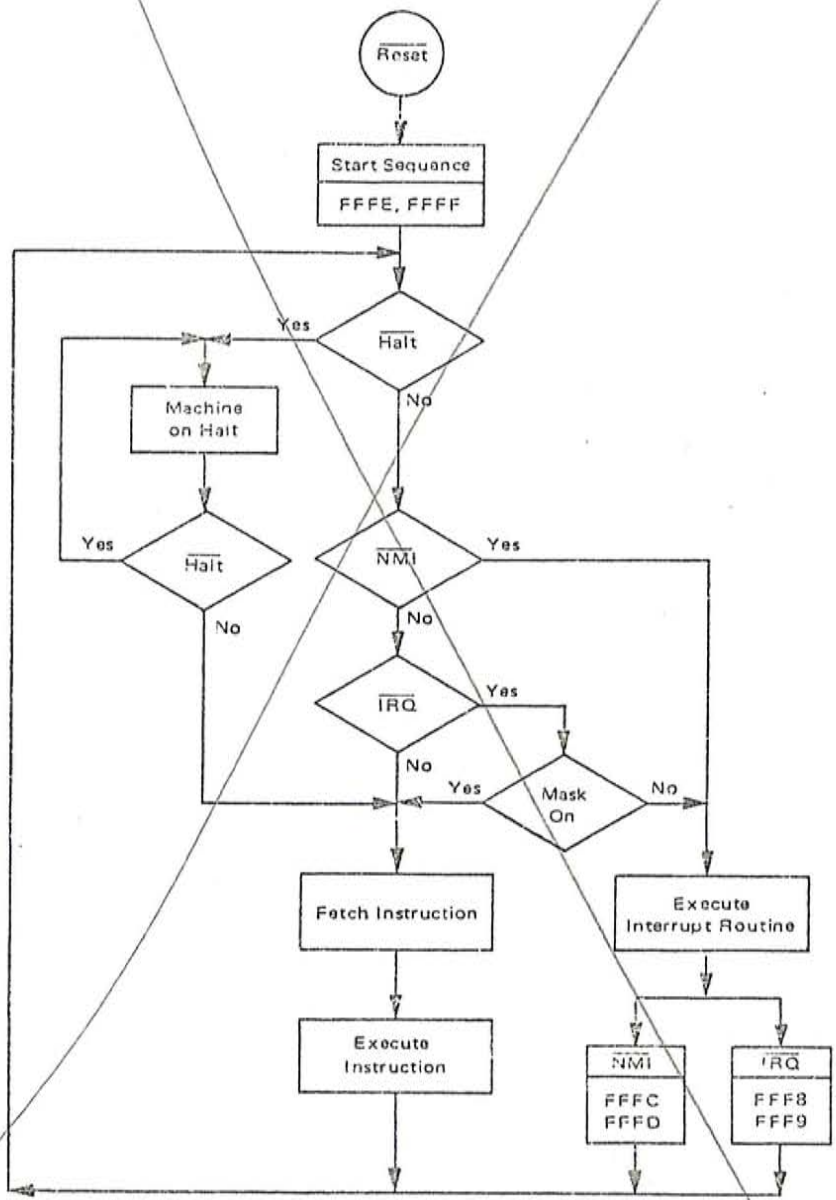
Resistor R20 dictates the cycle time for oscillator E31 on pin 12 between 150 and 300 us which in turn determine the width of the characters. The "Dot Clock" is not the output of E31 pin 12 however but rather the out put of and - or - invert gate E28. Its output is normally high but goes low for about 30us each time E31 pin 12 resets. The 30us pulse time is determined by the propagation time of E28 and E31. The "Dot Clock" is used to toggle shift registers E9 and E21.

The horizontal data for each character is made up of five dots and one blank for horizontal spacing on each video line. The video data for the horizontal portion of each character is parallel loaded from the character generator into 4-bit shift registers E9 and E21 with zero, bit 1, bit 2 and bit 3 going into E9. Bit 4, bit 5, zero and a one going into E21.

The serial output of E21 is tied high to load one's into the shift register to replace the character data as it is shifted out of the register bit by bit. E15 monitors the parallel output of the dot register and goes low when six bits have been clocked out it detects the ones that have been loaded into the register. A low transistion on the output of E15 is inverted by E17 pin 4 and changes the dot register from a shift up to a parallel load data mode. This same pulse also increments the character counters E6 and E11. Each time pin 6 goes high and the dot register is set up to parallel load, new data and E11 is incremented thus keeping track of which of the 32 horizontal character positions we are working with the dot data is shifted out bit by bit, on pin 10 of E9 at a rate set by the dot clock, it then goes to E12 pin 12 and 13 where it is mixed with the horizontal and vertical Sync pulses to form the composite video signal which is buffered by Emitter follower Q1 and fed to the video monitor.

Figure 1 is a flowchart showing the general instruction flow within the CPU. On power up, the CPU is automatically vectored to address FFFE and FFFF, which the prom programmer preloads with the starting address of his program. Note that the locations FFFE and FFFF are contained in the PROM in the SPHERE system and automatically transfers control to the Executive upon power up.

FIGURE 1 CPU FLOWCHART



10.1.4 KEYBOARD MODULE

The keyboard converts key depressions to valid ASCII equivelent codes for inputting to the CPU. The keys are arranged in a matrix array and are scanned until a key is depressed. As long as a key is not depressed, the output of E15 is a low, which keeps the Interrupt Flip-Flop E10 cleared and enables E12. E1 is clear so Ø2 is able via E12, to clock keyboard scanner (up-counter) E13, 16. Assuming counters are cleared initially, the output of E2, (one of 8 decoder-6 outputs used) selects 'COL Ø' line because input pins 13, 14 and 15 are all lows. As the counter progresses through its count, it selects a column, counts thru 16 possible combinations on that column, increments to the next column because of overflow from E16, and continues counting through another 7 possible keys. When the counter has reached 7F, all keys have been scanned and sequence begins over again. Assume a '6' key has been pressed. The counter counts until the count of 36 (hex), at which time COL 3 line and pin 2 input of E15 are enabled. Since E11 has open collector gates, a ground, or logic "0" is supplied to pin 2 of E15. Therefore pin 10 output of E15 goes high, or a logic 1. This enables E10, disables count gate E12 so that counting stops and causes E1 to trigger. After approximately 1.6 msec timeout, E10 sets, providing the interrupt, CA1 and CA2 to PIA. Since the counters have stopped counting, the address contained was frozen and equals the ASCII equivelent of a '6' (0110110 = 36 hex). This code is then supplied to the data inputs of the PIA. If interrupts have been enabled, the PIA exerts IRQA, which becomes IRQ on the bus and causes a CPU interrupt. The data line output of the PIA, when addressed by the CPU in response to a program request for keyboard data or an interrupt request, equals the ASCII code for '6'. The data lines are activated only when the PIA is actually addressed to do so. Refer to programming chapter. E1 is a debounce circuit, providing a 1.6 msec delay to allow key bounce to stabilize before providing an interrupt to the PIA. A CNTR, shift, and Reset keying combination will cause a total system reset. CS0, CS1, CS2 and gates feeding these PIA select signals decode the PIA address from the bus. RS0 and RS1 select a specific register within the PIA. Refer to programming chapter. Balance of logic inhibits shift characters on all lower case Alphabetic characters and allows case selection on special keys.

10.1.4.1 KEYBOARD 8 DIGITAL I/O LINES

Connector bus X4 contains 10 lines from the keyboard PIA (6820) chip. These are the PB0 to PB7, CB1, and CB2 Digital lines from the PIA and can be used for discrete controls. See the Motorola MC6800 Application Manual for suggestions.

10.1.5 CPU/1 MODULE - REFER TO CPU LOGIC SCHEMATICS

There are four classes of control signals which control the execution of the MC 6800 CPU. The first pair of control signals are the two phase clocks, Ø1 and Ø2 which provide the basic timing to the system. The second pair of signals, HALT and BUS AVAILABLE (BA) are used to stop program execution and free up the address and data bus for other uses. The interrupt signals make the CPU responsive to outside control and are listed in decreasing order of interrupt priority: RESET, NON-MASKABLE INTERRUPT (NMI), and MASKABLE INTERRUPT (IRQ). The three state control (TSC) and DATA BUS ENABLE (DBE) control lines provide a way to momentarily remove the CPU from the busses. The TSC line is not used in the basic SPHERE system. The SPHERE system uses INTEL 2107 A-8 or equivalent dynamic RAM's (Random Access Memory) for main memory. These are 4096 X 1 bit N-channel MOS chips. 8 of these chips are arranged into a 4K x 8 bit main memory. Extended memory is done by implementing additional 4K blocks of RAM chips. Refer to appendix B for a complete description of this device. The 1702A Programmable ROM (PROM) - read only memory - are 256 x 8 bit chips with 4 chips making up the total 1024 x 8 bit memory. These chips contain the standard system software programmed into them but can be erased and reprogrammed. Complete specifications and reprogramming information is contained in Appendix B for these devices. Two 9602 retriggerable one-shots (E43) and the MPQ6842 clock buffer generate the basic clocks, Ø1 and Ø2 to the system. The clocks are free running and the RC time constants (R9, R10, C18, C19) were chosen to provide a basic cycle time of 1.34 usec. Waveforms are shown in Figure 1. The two clocks are non-overlapping, meaning that Ø1 is low before Ø2 goes high and conversely, Ø2 goes low before Ø1 goes high again. Additional circuitry (D11, R12) on Ø2 one-shot is used during writing of the dynamic RAM's to stretch or lengthen Ø2 from .84 usec to 1.0 usec. This insures that adequate time is provided during Ø2 for memories to sense the written data. Due to the capacitor storage equivelent design of the dynamic RAM's, they must be refreshed every 2 msec or data will be lost. E8 is a free running clock providing the timing for the refresh cycle (2 msec). When E29 sets, this generates REFRESH which enables the address counters E22, 27 and address line drivers E21 and halts the CPU. Bus available goes high after completing the current instruction, disabling the CPU address buffers, E42, 44, 45 and enables Ø1 clock to refresh address counters (generator). On the next Ø1 after the CPU has halted, Ø1 begins clocking the refresh address generator, which is a simple up-counter.

As it counts, the addresses are supplied on the bus, which are the RAM memory 'Row' addresses necessary for refreshing. On count 64, the counter resets REFRESH, HALT and Address counters. The CPU then proceeds to perform the next instruction in sequence after it halted. The circuitry comprising E35, E36, and Gate E23 insure that the RAM's get refreshed even if the CPU should be halted or should hang up on a bad instruction, etc. E40, 41 are Bi-directional data line buffers and the direction is determined by the (Read/write) line. If the CPU is doing a LOAD instruction for example, this is a READ from some peripheral to the CPU. So the R/W line is a high. This configures the buffers for a data transfer to the CPU. A WRITE, such as a STORE instruction does the opposite. Any instruction that addresses the dynamic RAM (ØØØØ thru ØFFF) is detected by address decode gate E24 and generates CHIP SELECT (CS) and CHIP ENABLE (CE), via E28 and transistor Q1. CE is also generated by REFRESH. E36 insures during a Write cycle to memory that there is at least a 200 nsec delay between CHIP ENABLE and WRITE ENABLE (WE). Chips E2,5,11,17,19,26,32, and 34 constitute the 4,096 X 8 bits per word RAM. E1, 31 are tri-state buffers and are active when reading data from memory. They are enabled by CS. When writing to memory, these buffers are disabled (put in high-impedance or disconnect mode) and data is transferred directly to memory. Gates E4,10,16,18 and select 1 of 4-256 x 8 EPROM chips. The addresses begin at FCØØ and extend to FFFF. The address bits AØ to A7 then select 1 of the 256 words to be read. These are read-only device's and contain the system software (Assembler, Editor, etc.) The EPROM's can be reprogrammed but only with special equipment. Refer to EPROM data sheets in Appendix B. E18 also supplies chip selects to the spare PIA (Peripheral Interface Adapter.) The total decoding is FØ4Ø to FØ43. The spare PIA can be configured by user for interfacing his own peripheral devices to the system. Since the I/O peripherals are addressed the same as memory, a certain amount of address space has been reserved for I/O devices. Figure 2 shows the address mapping or how the 65,535 available addresses (maximum that can be addressed with 16 bit address lines) are allocated on the SPHERE system.

When the CPU/2 board is used as a single card card computer, the three signals on the right side of the board provide a 20 ma current loop to connect directly to a teletype common, Transmit, and Receive wires. In this mode the 512 Bytes of PROM perform the teletype I/O routine as well as a Debug mode. The jumper on PB0 and PB7 may be omitted and all 20 digital I/O lines will be available for dedicated use by the customer's own PROMs (up to 1K Bytes.) When using the teletype I/O routines the CPU clock must be set up at 1.50 usec $\pm$ 2% in order to interface with the required 110 BAND rate of the standard teletypes.

10.1.5.1 REAL TIME CLOCK

E7 counts refresh pulses to establish a series of real time interrupts. The P.C. board has dotted lines showing the various frequencies that may be selected. See Schematic. A reset can be initiated by putting FØ44 (HEX) on the address lines. Also a system reset will reset the clock.

10.1.6 CPU/2 MODULE - REFER TO CPU LOGIC SCHEMATICS

There are four classes of control signals which control the execution of the MC 6800 CPU. The first pair of control signals are the two phase clocks, Ø1 and Ø2 which provide the basic timing to the system. The second pair of signals, HALT and BUS AVAILABLE (BA) are used to stop program execution and free up the address and data bus for other uses. The interrupt signals make the CPU responsive to outside control and are listed in decreasing order of interrupt priority: RESET, NON-MASKABLE INTERRUPT (NMI), and MASKABLE INTERRUPT (IRQ). The three state control (TSC) and DATA BUS ENABLE (DBE) control lines provide a way to momentarily remove the CPU from the busses. The TSC line is not used in the basic SPHERE system. The SPHERE system uses INTEL 2107 A-8 or equivalent dynamic RAM's (Random Access Memory) for main memory. These are 4096 X 1 bit N-channel MOS chips. 8 of these chips are arranged into a 4K x 8 bit main memory. Extended memory is done by implementing additional 4K blocks of RAM chips. Refer to appendix B for a complete description of this device. The 1702A Programmable ROM (PROM) - read only memory - are 256 x 8 bit chips with 4 chips making up the total 1024 x 8 bit memory. These chips contain the standard system software programmed into them but can be erased and reprogrammed. Complete specifications and reprogramming information is contained in Appendix B for these devices. Two 9602 retriggerable one-shots (E45) and the MPQ6842 (E51) clock buffer generate the basic clocks, Ø1 and Ø2 to the system. The clocks are free running and the RC time constants (R11, R12, C22, C23) were chosen to provide a basic cycle time of 1.50 usec. The two clocks are non-overlapping, meaning that Ø1 is low before Ø2 goes high and conversely, Ø2 goes low before Ø1 goes high again. This insures that adequate time is provided during Ø2 for memories to sense the written data. Due to the capacitor storage equivelent design of the dynamic RAM's, they must be refreshed every 2 msec or data will be lost. E8 is a free running clock providing the timing for the refresh cycle (2 msec). When E9 sets, this generates REFRESH which enables the address counters E22, 27 and address line drivers E21 and halts the CPU. Bus available goes high after completing the current instruction, disabling the CPU address buffers, E44, 47, 48 and enables Ø1 clock to refresh address counters (generator). On the next Ø1 after the CPU has halted, Ø1 begins clocking the refresh address generator, which is a simple up-counter.

As it counts, the addresses are supplied on the bus, which are the RAM memory 'Row' addresses necessary for refreshing. On count 64, the counter resets REFRESH, HALT and Address counters. The CPU then proceeds to perform the next instruction in sequence after it halted. The circuitry comprising E37, E38, and Gate E23 insure that the RAM's get refreshed even if the CPU should be halted or should hang up on a bad instruction, etc. E42, 43 are Bi-directional data line buffers and the direction is determined by the (Read/write) line. If the CPU is doing a LOAD instruction for example, this is a READ from some peripheral to the CPU. So the R/W line is a high. This configures the buffers for a data transfer to the CPU. A WRITE, such as a STORE instruction does the opposite. Any instruction that addresses the dynamic RAM (ØØØØ thru ØFFF) is detected by address decode gate E24 and generates CHIP SELECT (CS) and CHIP ENABLE (CE), via E28 and transistor Q1. CE is also generated by REFRESH. E37 insures during a Write cycle to memory that there is at least a 200 nsec delay between CHIP ENABLE and WRITE ENABLE (WE). Chips E2,5,11,17,19,26,34, and 36 constitute the 4,096 X 8 bits per word RAM. E1,33 are tri-state buffers and are active when reading data from memory. They are enabled by CS. When writing to memory, these buffers are disabled (put in high-impedance or disconnect mode) and data is transferred directly to memory. Gates E4,10,16,32 and E13 select 1 of 4-256 x 8 EPROM chips. The addresses begin at FCØØ and extend to FFFF. The address bits A0 to A7 then select 1 of the 256 words to be read. These are read-only device's and contain the system software (Assembler, Editor, etc.) The EPROM's can be reprogrammed but only with special equipment. Refer to EPROM data sheets in Appendix B. E18 also supplies chip selects to the spare PIA (Peripheral Interface Adapter.) The Total decoding is FØ4Ø to FØ43. The spare PIA can be configured by user for interfacing his own peripheral devices to the system. Since the I/O peripherals are addressed the same as memory, a certain amount of address space has been reserved for I/O devices. Figure 2 shows the address mapping or how the 65,535 available addresses (maximum that can be addressed with 16 bit address lines) are allocated on the SPHERE system.

When the CPU/2 board is used as a single card computer, the three signals on the right side of the board provide a 2Ø ma current loop to connect directly to a teletype common, Transmit, and Receive wires. In this mode the 512 Bytes of PROM perform the teletype I/O routine as well as a Debug mode. The jumper on PBØ and PB7 may be omitted and 19 digital I/O lines will be available for dedicated use by the customer's own PROMs (up to 1K Bytes.) When using the teletype I/O routines the CPU clock must be set up at 1.50 usec ± 2% in order to interface with the required 110 BAUD rate of the standard teletypes.

10.1.6.1 REAL TIME CLOCK ON CPU/2

E7 counts refresh pulses to establish a series of real time interrupts. The P.C. board has dotted lines showing the various frequencies that may be selected. See Schematic. A reset can be initiated by putting FØ44 (HEX) on the address lines. Also a system reset will reset the clock. The clock interupts enter the spare PIA on CA1 (pin 4Ø) and can be used through that PIA device as a CPU real time interupt. The PIA may be programmed to interupt or ignore the clock. Reset may also be done through the PIA by reading from the data register A.

interrupt divide clock reset done by 1) master reset
2) R/W to FØ44
3) read keyboard input!
so must not read keyin data except when interrupt goes off !!

Figure 2

Address Space
Allocation

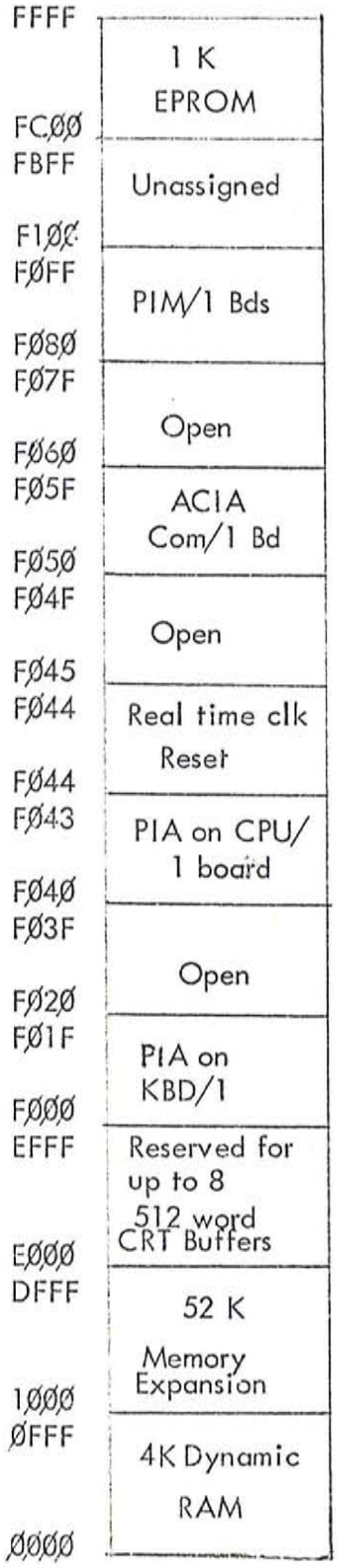
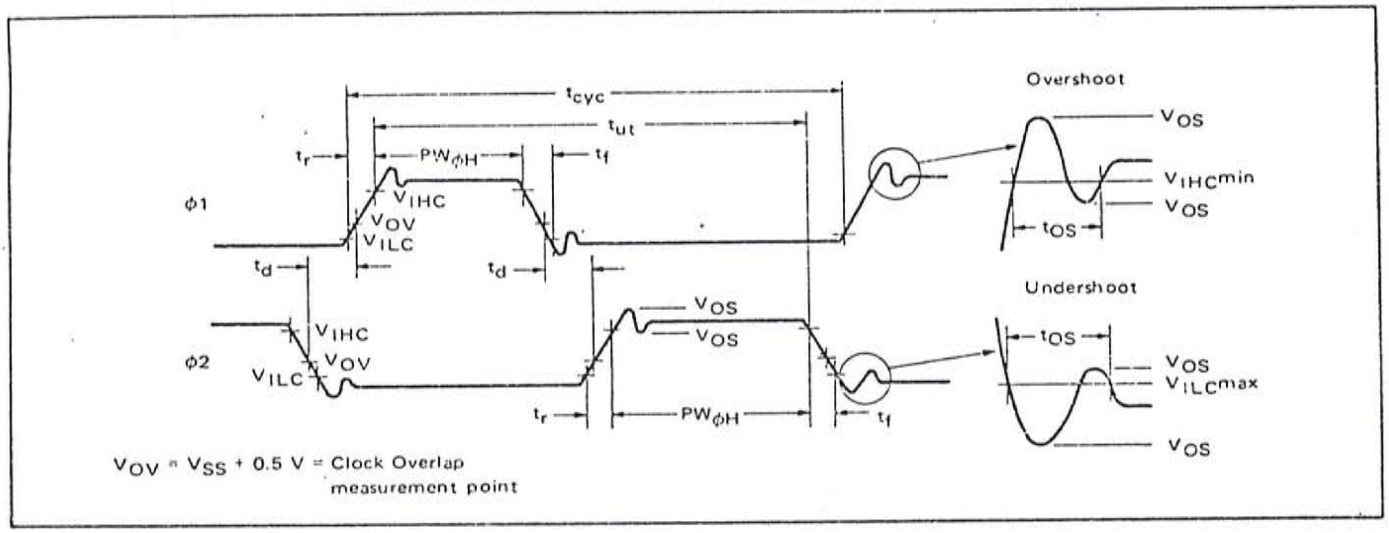


Figure 3

CPU Clock Waveforms

FIGURE 1 – CLOCK TIMING WAVEFORM



When the CPU/2 board is used as the One Card Computer, the three signals on the right side of the board provide a 2Ø am current loop to connect directly to a teletype common, Transmit, and Receive wires. Also available are RS232c interfaces. In this mode the 512 Bytes of ROM perform the teletype I/O routine as well as a Debug mode. The jumper on PBØ and PB7 may be omitted and 19 digital I/O lines will be available for dedicated use by the customer's own ROM's (up to 1K Bytes.) When using the teletype I/O routines the CPU clock must be set up at 1.50 usec ± 2% in order to interface with the required 110 BAUD rate of the standard teletypes.

The real time clock is also on the CPU/2 board. E7 counts refresh pulses to establish a series of real time interrupts. The P.C. board has dotted lines showing the various frequencies that may be selected The real time clock counter can be reset by putting FØ44 (HEX) on the address lines. Also a system reset will reset the clock. The clock interupts enter the E3 PIA (F041) on CA-1 (pin 4Ø)and can be used through that PIA device as a CPU real time interupt. The PIA may be programmed to interupt or ignore the clock.

10.2.4 ROM/1

The Read Only Memory Board consists of two sections of logic, the first provides those elements common to the second, address buffering, data gating and power control. The second set of logic is repeated four times and contains the necessary address select logic for the four 1702's in its bank.

Address lines A0-A7 are buffered and isolated from the address bus by E3 and E4, while A8 and A9 are buffered by two of the four bus driver gates of E7 that are not used for data gating. A10 through A15 are inverted and buffered by two 7404 gates each in E1 and E2 to provide the necessary signals for address selection. The bank selects from the four banks are OR'd together in E5a and AND'd together with R/W, VMA, and Ø2 to produce the board select line which enables the 8097 tri-state bus drivers E6 and 2/6 E7. Diodes D1-D4 drop the -12v from X6-5,12 to -9v.

The address select for the bank is provided by the jumpers at J1-J4. The combined A10-A15 true and inverted signals are AND'd by E13,E19, E25, E31 and provide the BANK SELECT A through D signals. The BANK SELECT also feeds the C and D inputs of E8, E14, E20, E26 which along with A8 and A9 (A and B inputs) the binary to

BCD 7442 provides outputs to select one 1702 in E9, E10, E11, or E12 (or E15-18, E21-24 or E27-30). Address lines A0-A7 are bused to all 1702's from E3 and E4. And D0-D7 is bused from all 1702's to E6 and E7.

Resistors R1 through R24 provide pullup and logic conversion between TTL and CMOS. Once a bank of 1702's is selected the selection of which 1702 is from right to left (away from the address select logic to near the address select logic). The address select logic for each bank of 1702's consists of 3 row of holes; the one nearest E1 is 7 holes and is Address true. The next row, Address inverted, is 12 holes, three extra on the left and two on the right (DO NOT USE THESE HOLES) the address bits for the bank select logic is the third row, 6 holes. Bit A15 is the left most hole, bit A10 is the right most hole of the address select row. In some banks an extra feed-through may have been required, DO not mistake this for an address hole.

10.2.5 SIM/1

The serial Interface module is built around two ACIA (Asynchronous communications Interface Adaptor) chips. The board is segregated into 9 main areas: Board decoding, cassette interface number 1, cassette interface number2, ROM, Baud rate generators, Teletype interface, RS232 interface, TTL direct interface, and modem. The basic board just decodes the address lines to allow selection of the PROM, ACIA number1, or ACIA number2.

BOARD DECODING
Three strapable addresses are provided on the board. As initially set ACIA port number 1 is located at F050 & F051. The second port is located at F060/1 (hex). If a second SIM/1 board is added to the system, A1 on the second board would be cut and a jumper placed along the dotted lines by the A1 etch. The addresses on this board will then be F052/3 and F062/3. The PROM will always be located at FB00 to FBFF on all boards. Hence do not put a PROM in the socket on any added boards after the first one. A third board can be added by changing the jumper at A2 instead of A1. This will give another set of unique ACIA ports for control. Table 10.1 shows the configuration of the 8 boards that could be independantly addressed. The A0 address line determines which of the two registers in the ACIA is being selected. When A0 is high the transmit/receive data registor is selected.

A1	A2	A3	ACIA No.1 Address Logical	Physical	ACIA No.2 Address Logical	Physical
-	-	-	Ø	FØ5Ø/1	1	FØ6Ø/1
X	-	-	2	FØ52/3	3	FØ62/3
-	X	-	4	FØ54/5	5	FØ64/5
X	X	-	6	FØ56/7	7	FØ66/7
-	-	X	8	FØ58/9	9	FØ68/9
X	-	X	A	FØ5A/B	B	FØ6A/B
-	X	X	C	FØ5C/D	D	FØ6C/D
X	X	X	E	FØ5E/F	F	FØ6E/F

X= Jumper added and etch cut.

TABLE 1Ø.1

CASSETTE INTERFACES

There are two identical cassette interface circuits in the center section of the SIM/1 board. One is controlled through ACIA 1 and the other through ACIA No.2 (FØ6Ø). The theory of operation has been fully explained in such publications as BYTE (see the Bit Boffer P.3Ø March,76). In summary: the audio cassette input sine wave is squared by the 311 comparators. If it is a long pulse, a one shot has time to fire providing an extra 2 clock pulses and indicating a long pulse received. A short pulse (1/2 of a long pulse) one shot reset and each edge provides a clock pulse. Hence a simulated clock frequency is generated and the 1's and Ø's are are recaptured. The trimpot allows setting the trigger level to the 4ØØ1 IC to compare the long to short ratio actually present, thus audio cassette player speed variations can be tuned out. The output data is fed to the 4Ø18 (counter) through two clocked gates. The resulting outputs are summed through different resistors to provide a symetrical step function looking like a sine wave. Further RC networks smooth out the signal and an output transistor provides ample drive to the cassette recorder. The long and short digital pulses now look like two frequencies of about 12ØØ Hz and 24ØØ Hz. E38 is a free running multivibrator that provides the 3ØØ baud rate for both cassette clocks. This frequency is fed into the ACIA clock input when used as a cassette port. The ACIA should be set for 16X rate. Each cassette has a relay (K1 and K2) provided to allow the cassette drive to be turned on or off when the CPU calls the ACIA port Ø for example, K1 will be energized by $\overline{RTS}$ going low and the 74Ø7 buffer pulling current through the coil of K1. The normally open contact now closes. If the relay contacts have been plugged into the remote terminal on a cassette recorder, the cassette drive is turned on. The software in the PROM will not send valid data until after a long enough delay for the cassette drive to get up to speed. In the event the cassette recorder does NOT have a remote on/off plug, a 12 volt coil relay may be used to switch the 11ØVAC line. A seperate output through R13 for cassette 1 and R14 for cassette 2 ,(optional current limiting resistors) will be tied to the top of the coil in the new relay and the bottom of the coil will be tied to the normally open line of K1 (K2). The common will be jumpered to ground (or -12 if a 24 volt coil is used). Hence the SIM/1 board provides the coil power to drive a larger relay for direct switching of 11ØVAC lines. Since these relays always respond when the respective ACIA is called, they can be used to turn on/off teletypes or any other relay or device desired.

PROM
Whenever the 16 address lines from the CPU board contain FE xx in Hex, the ROM will be addressed. This read only memory chip contains logic to enable reading and writing to the cassette tapes. See listing of the contents of the ROM for details of the software operations.

BAUD RATE GENERATORS
The SIM/1 Board has a series of straps for selection of desired baud rates. It is prewired to 3ØØ baud but this strap maybe cut and standard frequency from 15Ø to 96ØØ may be used. A trimming resistor (R74) may be added to compensate for the 96Ø1 free running one shot being off frequency. If the 11Ø baud rate is desired for a teletype, the 15Ø line must be selected AND C47 (a 33 pf cap.) must be strapped in the place labeled TTY. This will slow the main clock down to within a range that R74 can be selected to yield a precise 11Ø baud rate. Usually R74 is about 12ØK Ω. External frequencies may be added through X1 pins 8 & 9 when the 625 KHz (15Ø baud rate multiplies) and the 455KHz (11Ø baud rate) jumpers on the lower left sector of the board are added.

TELETYPE INTERFACE
With the input clock to the No. 1 ACIA set at 11Ø baud rate (see baud rate generator), the output data will be correctly formatted by the ACIA if the 64x mode is selected by software. Two 4N33 photo couplers are used to isolate the teletype ± 12 volt, 2Ø ma current loops from the + 5 volt TTL logic. When using the teletype interface, the RS232 and TTL jumpers must be put in and the 2Ø ma jumper left out. Also set up the baud rate generator and ACIA jumpers per the assembly instructions.

RS232 INTERFACE
Any desired input baud rate can be selected. The TTL and 2Ø ma jumpers must be put in place. E5 and E13 are standard RS232 driver/receivers. Except for the baud rate selection the operation is like the teletype. An RS232 line has more drive capability than either TTL or 2Ø ma current loops. The voltage swings are ± 12 volts.

TTL DIRECT
This interface allows high speed data transfers over short direct wire connections. The ACIA can be initilized to 1x, 16x, or 64x. The baud rate can be any selected value. The 2Ø ma and RS232 jumpers must be put in. The voltage swings will be +5 volts compatible with any other TTL/DTL input. The driver is a 7404 with 1Ø TTL loads maximum.

MODEM
The modem section is built around the 6860 modem chip and the 6850 ACIA. See the M6800 microprocessor applications manual for details. In addition two filter sections are included making full single wire communications possible. Jumpers are provided on the board to select single wire full duplex. See assembly instructions.

ASC11 CHARACTER SET (7 BIT CODE)

HEX	DEC	OCT	BINARY		CHAR	DESCRIPTION
00	Ø	000	00000000		NUL	Null
0I	I	00I	0000000I		SOH	Start of Heading
02	2	002	00000I0		STX	Start of Text
03	3	003	00000II		ETX	End of Text
04	4	004	00000I00		EOT	End of Transmission
05	5	005	00000I0I		ENQ	Enquiry
06	6	006	00000II0		ACK	Acknowledge
07	7	007	00000III		BEL	Bell
08	8	0IØ	0000I000	*	BS	Back Space
09	9	0II	0000I00I	*	HT	Horizontal Tab CTRL-I
0A	IØ	012	0000I0I0	*	LF	Line Feed
0B	II	013	0000I0II		VT	Vertical Tab
0C	I2	014	0000II00		FF	Form Feed
0D	I3	015	0000II0I	*	CR	Carriage Return C - M
0E	I4	016	0000III0		SO	Shift Out C - N
0F	I5	017	0000IIII		SI	Shift In CTRL - O
IØ	I6	02Ø	0001 0000		DLE	Data Link Escape CTRL - P
II	I7	02I	0001000I		DC1	Device Control 1 ↑ CTRL - Q
I2	I8	022	00010010		DC2	Device Control 2 → C - R
I3	I9	023	00010011		DC3	Device Control 3 ↓ C - S
I4	2Ø	024	00010100		DC4	Device Control 4 ← S - T
I5	2I	025	00010I0I		NAK	Negative Acknowledge
I6	22	026	00010II0		SYN	Synchronize
I7	23	027	00010III		ETB	End of Transmission Block
I8	24	03Ø	00011000	*	CAN	Cancel
I9	25	03I	00011001		EM	End of Media
IA	26	032	00011010		SUB	Substitute
IB	27	033	00011011	*	ESC	Escape
IC	28	034	00011100		FS <	File Separator
ID	29	035	0011101		GS =	Group Separator
IE	30	036	0011110		RS >	Record Separator
IF	3I	037	0011111		VS	CTRL - ←
20	32	040	0100000	*	SP	Space ⏎
2I	33	04!	0100001		!	S 1 Exclamation
22	34	042	0100010		"	S 2 Double Quote
23	35	043	0100011		#	S 3 number or pound
24	36	044	0100100		$	S 4 dollar sign
25	37	045	0100101		%	S 5 Percentage
26	38	046	0100110		&	S 6 Ampersand
27	39	047	0100111		'	S 7 Apostrophe or single quote
28	40	050	0101000		(	S 8 Parentheses
29	4I	05I	0101001		)	S 9 Parentheses
2A	42	052	0101010		*	S : Astrick
2B	43	053	0101011		+	S ; plus
2C	44	054	0101100	*	,	comma

CHAR GEN

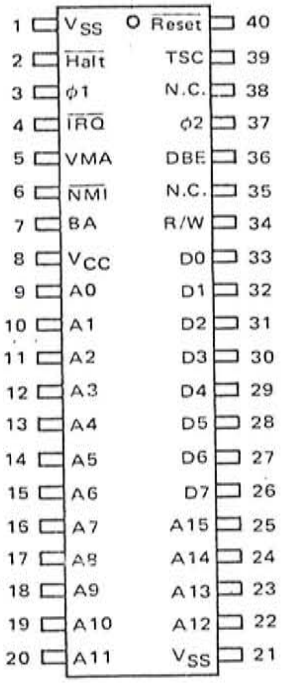
HEX	DEC	OCT	BINARY	CHAR	DESCRIPTION
2D	45	055	0101101	-	minus
2E	46	056	0101110	.	period
2F	47	057	0101111	/	slash
30	48	060	0110000	0	zero
31	49	061	0110001	1	one
32	50	062	0110010	2	two
33	51	063	0110011	3	three
34	52	064	0110100	4	four
35	53	065	0110101	5	five
36	54	066	0110110	6	six
37	55	067	0110111	7	seven
38	56	070	0111000	8	eight
39	57	071	0111001	9	nine
3A	58	072	0111010	:	colon
3B	59	073	0111011	;	semi-colon
3C	60	074	0111100	<	less than
3D	61	075	0111101	=	equal
3E	62	076	0111110	>	greater than
3F	63	077	0111111	?	question
40	64	100	1000000	@	at sign
41	65	101	1000001	A	Letter A
42	66	102	1000010	B	letter B
43	67	103	1000011	C	letter C
44	68	104	1000100	D	letter D
45	69	105	1000101	E	letter E
46	70	106	1000110	F	letter F
47	71	107	1000111	G	letter G
48	72	110	1001000	H	letter H
49	73	111	1001001	I	letter I
4A	74	112	1001010	J	letter J
4B	75	113	1001011	K	letter K
4C	76	114	1001100	L	letter L
4D	77	115	1001101	M	letter M
4E	78	116	1001110	N	letter N
4F	79	117	1001111	O	letter O
50	80	120	1010000	P	letter P
51	81	121	1010001	Q	letter Q
52	82	122	1010010	R	letter R
53	83	123	1010011	S	letter S
54	84	124	1010100	T	letter T
55	85	125	1010101	U	letter U
56	86	126	1010110	V	letter V
57	87	127	1010111	W	letter W
58	88	130	1011000	X	letter X

HEX	DEC	OCT	BINARY	CHAR	DESCRIPTION
59	89	131	1011001	Y	letter Y
5A	90	132	1011010	Z	letter Z
5B	91	133	1011011	[	left bracket
5C	92	134	1011100	\	back slash
5D	93	135	1011101	]	right bracket
5E	94	136	1011110	↑	up arrow
5F	95	137	1011111	←	back arrow
60	96	140	1100000	`	back quote or accent mark
61	97	141	1100001	a	small letter a
62	98	142	1100010	b	small letter b
63	99	143	1100011	c	small letter c
64	100	144	1100100	d	small letter d
65	101	145	1100101	e	small letter e
66	102	146	1100110	f	small letter f
67	103	147	1100111	g	small letter g
68	104	150	1101000	h	small letter h
69	105	151	1101001	i	small letter i
6A	106	152	1101010	j	small letter j
6B	107	153	1101011	k	small letter k
6C	108	154	1101100	l	small letter l
6D	109	155	1101101	m	small letter m
6E	110	156	1101110	n	small letter n
6F	111	157	1101111	o	small letter o
70	112	160	1110000	p	small letter p
71	113	161	1110001	q	small letter q
72	114	162	1110010	r	small letter r
73	115	163	1110011	s	small letter s
74	116	164	1110100	t	small letter t
75	117	165	1110101	u	small letter u
76	118	166	1110110	v	small letter v
77	119	167	1110111	w	small letter w
78	120	170	1111000	x	small letter x
79	121	171	1111001	y	small letter y
7A	122	172	1111010	z	small letter z
7B	123	173	1111011	{	left brace
7C	124	174	1111100	|	vertical bar
7D	125	175	1111101	}	right brace
7E	126	176	1111110	~	approximate or tilde
7F	127	177	1111111	DEL	Delete

APPENDIX C

PIN ASSIGNMENT

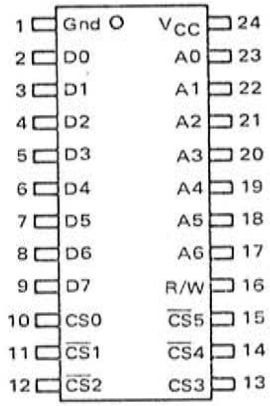
MC6800 CPU



MC 6810L

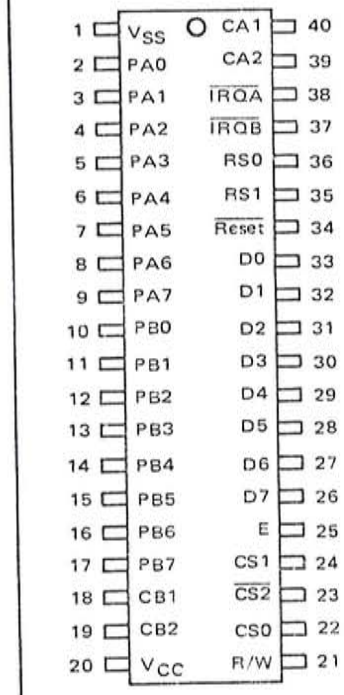
128 X 8-BIT STATIC RANDOM ACCESS MEMORY

PIN ASSIGNMENT



PIN ASSIGNMENT

MC6820 PIA



Vss = Ø v.

Vcc = +5 v.

DBE = Data Bus Enable - input is the three-state control signal for the MPU that enables the data bus drivers when in the high state.

C S = Chip Select

APPENDIX C

INTERFACING SIGNALS

The Sphere system interface is chiefly TTL compatible. It consists of 4 main bus cables and provisions for peripheral adaptors. See Table 1.

THE DATA BUS is a 14 strand ribbon cable connecting to socket X3 on all boards. This Bus contains the 8 data lines (D0 - D7), The Bus Available (BA), and the $\overline{\text{REFRESH}}$ signals. Eight pins are used for the data bus. It is bi-directional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving 35 standard TTL loads. A high on BA indicates the CPU is NOT using the 8 data lines and they are in a TRI-STATE (high impedance) mode. In this mode the Data Bus is available for other functions. BA can drive up to 35 TTL loads. The $\overline{\text{REFRESH}}$ signal will go low each time a dynamic memory bank is to be refreshed or updated. During these times the address lines will be determining the set of memories that are to be refreshed and the CPU will be in a halted mode. $\overline{\text{REFRESH}}$ can drive up to 8 TTL loads.

THE ADDRESS BUS consists of two 14 conductor flat ribbon cables which join X1 and X2 on all boards. Table 1 gives the pin numbers. The signals with their definitions are as follows:

CLOCK PHASE TWO (Ø2) - 670 KHZ signal, High .95u.s and Low .54u.s, capable of driving 35 TTL loads.

ADDRESS BUS (A0-A15) - Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving 35 standard TTL loads. When the output is turned off, it is essentially an open circuit. This permits the CPU to be used in DMA applications.

$\overline{\text{HALT}}$ - When this input is in the low state, all activity in the machine will be halted. This input is level sensitive. In the halt mode, the machine will stop at the end of an instruction, Bus Available will be at a high level and valid Memory Address will be at a low level.

Transition of the $\overline{\text{Halt}}$ line must not occur during the last 250 ns of phase one. To insure single instruction operation, the $\overline{\text{Halt}}$ line must go high for one Phase One Clock cycle.

THREE-STATE CONTROL (TSC) - In order to use this line a jumper must be added on the CPU board. This input causes all of the address lines and the Read/Write line to go into the off or high impedance state. This state will occur 500 ns after TSC = 2.4V. The Valid Memory Address and Bus Available signals will be forced low. The data bus is not affected by TSC and has its own enable (Data Bus Enable). In DMA applications, the Three-State Control line should be brought high on the leading edge of the Phase One Clock. The Ø1 clock must be held in the high state and the Ø2 in the low state for this function to operate properly. Since the MPU is a dynamic device, it can be held in this state for only 5.0 us or destruction of data will occur in the MPU.

READ/WRITE (R/W) - This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read (high) or Write (low) state. The normal standby state of this signal is Read (high). Three-State Control going high will turn Read/Write to the high state. Also, when the processor is halted, it will be in the high state. This output is capable of driving 35 standard TTL loads.

VALID MEMORY ADDRESS (VMA) - This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not Tri-State. 35 standard TTL loads may be directly driven by this active high signal.

$\overline{\text{INTERRUPT REQUEST (IRQ)}}$ - This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The $\overline{\text{Halt}}$ line must be in the high state for interrupts to be recognized.

The $\overline{\text{IRQ}}$ has a high impedance pullup device internal to the chip; however a 3.3K ohm external resistor to V_{CC} has been used for wire-OR and optimum control of interrupts.

2 ms

TABLE 1 CONNECTOR STANDARD PIN ASSIGNMENTS

DATA BUS

Pin No.	X1	X2	X3	PIA - A side X4,X7,X9,X11	PIA B side X5,X8,X10,X12	CASSETTE Sim Board X24
1	R/W → *	A2 → *	D4 ↔ ≠	RESET	RESET	CAS IN - 1 (rec)
2	A9 → *	A4 → *	D7 ↔ ≠	PA 0	PB 0	Relay - 2
3	VMA → * **	A3 → *	BA → **	PA 1	PB 1	Relay 2
4	A14 → *	A5 → *	OPEN	PA 3	PB 3	Relay 1
5	A15 → *	A6 → *	D3 ↔ ≠	PA 5	PB 5	Relay 1
6	HLT ←	A7 → *	D2 ↔ ≠	PA 6	PB 6	TX - 20ma (1)
7	NMI ←	A8 → *	D0 ↔ ≠	OPEN	OPEN	GROUND
8	OPEN	A0 → *	D1 ↔ ≠	GROUND	GROUND	RX - 20ma (1)
9	OPEN	A1 → * (halt high, 3K to Vcc)	REFRESH	PA 7	PB 7	-12V 20ma com (1)
10	A13 → *	IRQ (intrpt) ←	OPEN	PA 4	PB 4	ext relay 1 (+12)
11	A12 → *	OPEN	OPEN	PA 2	PB 2	CASIN - 2 (rec)
12	A11 → *	RESET ←	OPEN	CA 2 - STROBE	CB 2	CASOUT - 2 (write)
13	A10 → *	TSC ←	D5 ↔ ≠	CA 1	CB 1	CAS OUT - 1 (write)
14	Ø2 →	OPEN	D6 ↔ ≠	+5V DC	+5V DC	ext relay 2 +12

** - Low when TSC is high
* - disable when TSC is high
≠ - disable when DBE is low

C.1 INTERFACING SIGNALS cont'd

$\overline{\text{RESET}}$ - This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial start up of the processor. This is an Open-Collector line with the pull up resistor on the CPU board. Lower this signal will cause the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced high. For the restart, the last two (FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{\text{IRQ}}$.

Figure C-1 shows the initialization of the microprocessor after restart. Reset must be held low for at least eight clock periods after V_{CC} reaches 4.75 volts. If Reset goes high prior to the leading edge of Ø2, on the next Ø1 the first restart memory vector address (FFFE) will appear on the address lines. This location should contain the higher order eight bits to be stored into the program counter. Following, the next address FFFF should contain the lower order eight bits to be stored into the program counter.

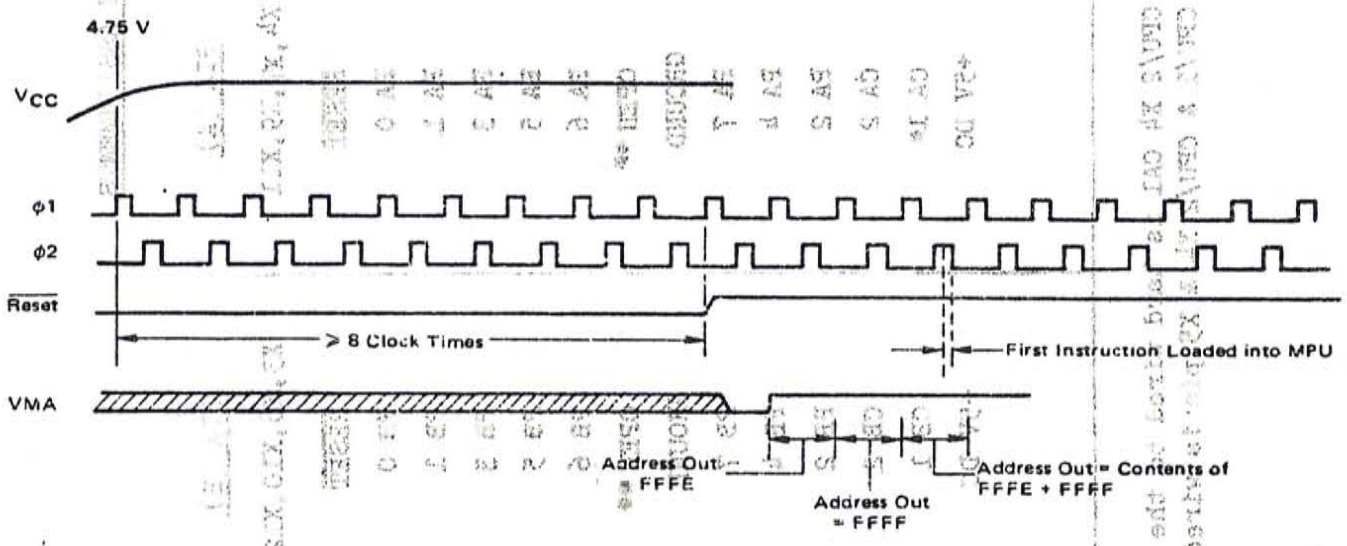


FIGURE C.1
INITIALIZATION OF MPU AFTER RESTART

TABLE C-1

CONNECTOR STANDARD PIN ASSIGNMENTS

Pin No.	Address Bus		Data Bus	PIA (A)	PIA (B)	KBD/2
	X1	X2	X3	X4,X7,X9,X11	X5,X8,X10,X12	X20
1	R/W	A2	D4	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$
2	A9	A4	D7	PA 0	PB 0	D 0
3	VMA	A3	BA	PA 1	PB 1	D 1
4	A14	A5	OPEN	PA 3	PB 3	D 3
5	A15	A6	D3	PA 5	PB 5	D 5
6	$\overline{\text{HLT}}$	A7	D2	PA 6	PB 6	D 6
7	$\overline{\text{NMI}}$	A8	D0	OPEN **	OPEN **	OPEN
8	OPEN	A0	D1	GROUND	GROUND	GROUND
9	OPEN	A1	$\overline{\text{REFRESH}}$	PA 7	PB 7	Repeat/D 7
10	A13	$\overline{\text{IRQ}}$(intrpt)	OPEN	PA 4	PB 4	D 4
11	A12	OPEN	OPEN	PA 2	PB 2	D 2
12	A11	$\overline{\text{RESET}}$	OPEN	CA 2	CB 2	Character Strobe
13	A10	TSC	D5	CA 1*	CB 1	unused
14	Ø2	OPEN	D6	+5V DC	+5V DC	+5V DC

* On CPU/2 X4 CA1 is hard wired to the Real Time Clock

** On CPU/1 & CPU/2 X4 & X5 pin is wired to +5VDC

TABLE C-1 cont'd

CONNECTOR STANDARD PIN ASSIGNMENTS

Pin No.	Power Bus	CRT/1	Line Printer			SIM/1	
	X6	$\overline{X5}$	X13	X14	X15	X24	X25
1	GND		D5	$\overline{RUN}$	$\overline{SWI}$	C1 EAR	MODEM OUT
2	GND		D4	$\overline{STROBE}$	$\overline{HOME}$	C2 REM	RI
3	+12 VDC		D3	$\overline{LAMP\ 1}$	$\overline{EMPTY}$	C2 REM	RS232 R_x
4	-5 VDC		D2	$\overline{FEED}$	SW 2	C1 REM	OH
5	-12 VDC	For use with light pen	D1	$\overline{CLEAR}$	JAM	C1 REM	RS232 T_x
6	+5 VDC		D0	$\overline{LAMP\ 2}$	P0	20maTx	TV
7	+5 VDC		+5 VDC	OPEN	OPEN	GND	GND
8	GND		GND	GND	GND	20maR_x	MODEM IN
9	GND		GND	GND	GND	20maCOM(-12)	-12
10	+12 VDC		GND	GND	GND	C1 +12(rem)	TTL R_x
11	-5 VDC		GND	GND	GND	C2 EAR	SH
12	-12 VDC		GND	D6	GND	C2 MIC	DA
13	+5 VDC		GND	GND	GND	C1 MIC	TTL T_x
14	+5 VDC		D7	GND	GND	C2 +12(rem)	GND

C.1 INTERFACING SIGNALS cont'd

NON-MASKABLE INTERRUPT ($\overline{NMI}$) - A low going edge on this input requests that a non-maskable interrupt sequence be generated within the processor. As with the Interrupt Request signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{NMI}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{NMI}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC andFFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ has a high impedance pullup resistor internal to the chip; however a 3.3K ohm external resistor to V_{CC} has been used for wire-OR and optimum control of interrupts.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupt lines that are sampled during Ø2 and will start the interrupt routine on the Ø1 following the completion of an instruction.

Figure C-2 is a flow chart describing the major decision paths and interrupt vectors of the microprocessor. Table C-2 gives the memory map for interrupt vectors.

Vector MS	Vector LS	Description
FFFE	FFFF	Restart
FFFC	FFFD	Non-maskable Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interrupt Request

TABLE C-2
MEMORY MAP FOR INTERRUPT VECTORS

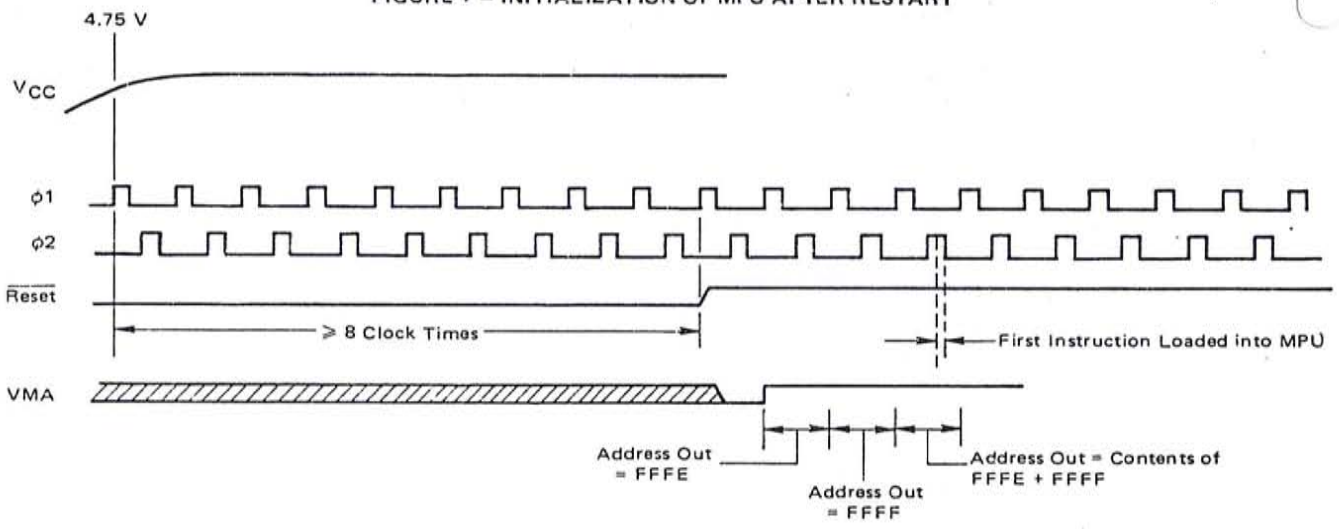
TABLE 1 cont'd. CONNECTOR STANDARD PIN ASSIGNMENTS

Pin No.	X6 (POWER)	X5	X13	X14	X15	Sim Board X25
1 (8)	GND	For Future Use with Light Pen	D5	RUN	SWI	(XMIT-coupler)
2 (9)	GND		D4	STROBE	HOME	(RING)
3 (10)	+ 12VDC		D3	LAMP 1	EMPTY	RX-RS232 ①
4 (11)	- 5VDC		D2	FEED	SW 2	(OH)
5 (12)	- 12 VDC		D1	CLEAR	JAM	TX-RS232 ①
6 (13)	+ 5VDC		D0	LAMP 2	PO	(+V)
7 (14)	+ 5 VDC		+5VDC	OPEN	OPEN	- GND ①
8 (1 -)	GND		GND	GND	GND	(RECV-COUPLER)
9 (2 -)	GND		GND	GND	GND	
10 (3 -)	+ 12VDC		GND	GND	GND	RX-TTL ①
11 (4 -)	- 5VDC		GND	GND	GND	(SH)
12 (5 -)	-12VDC		GND	D6	GND	(DA)
13 (6 -)	+ 5 VDC		GND	GND	GND	TX-TTL ①
14 (7 -)	+ 5 VDC		D7	GND	GND	-GND

$\overline{\text{RESET}}$ - This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial start-up of the processor. If a positive edge is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced high. For the restart, the last two (FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{\text{IRQ}}$.

Figure 1 shows the initialization of the microprocessor after restart. $\overline{\text{Reset}}$ must be held low for at least eight clock periods after V_{CC} reaches 4.75 volts. If Reset goes high prior to the leading edge of Ø2, on the next Ø1 the first restart memory vector address (FFFE) will appear on the address lines. This location should contain the higher order eight bits to be stored into the program counter. Following, the next address FFFF should contain the lower order eight bits to be stored into the program counter

FIGURE 1 – INITIALIZATION OF MPU AFTER RESTART



NON-MASKABLE INTERRUPT ($\overline{NMI}$) - A low going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the Interrupt Request signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{NMI}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{NMI}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ has a high impedance pullup resistor internal to the chip; however a 3.3k ohm external resistor to V_{CC} has been used for wire-OR and optimum control of interrupts.

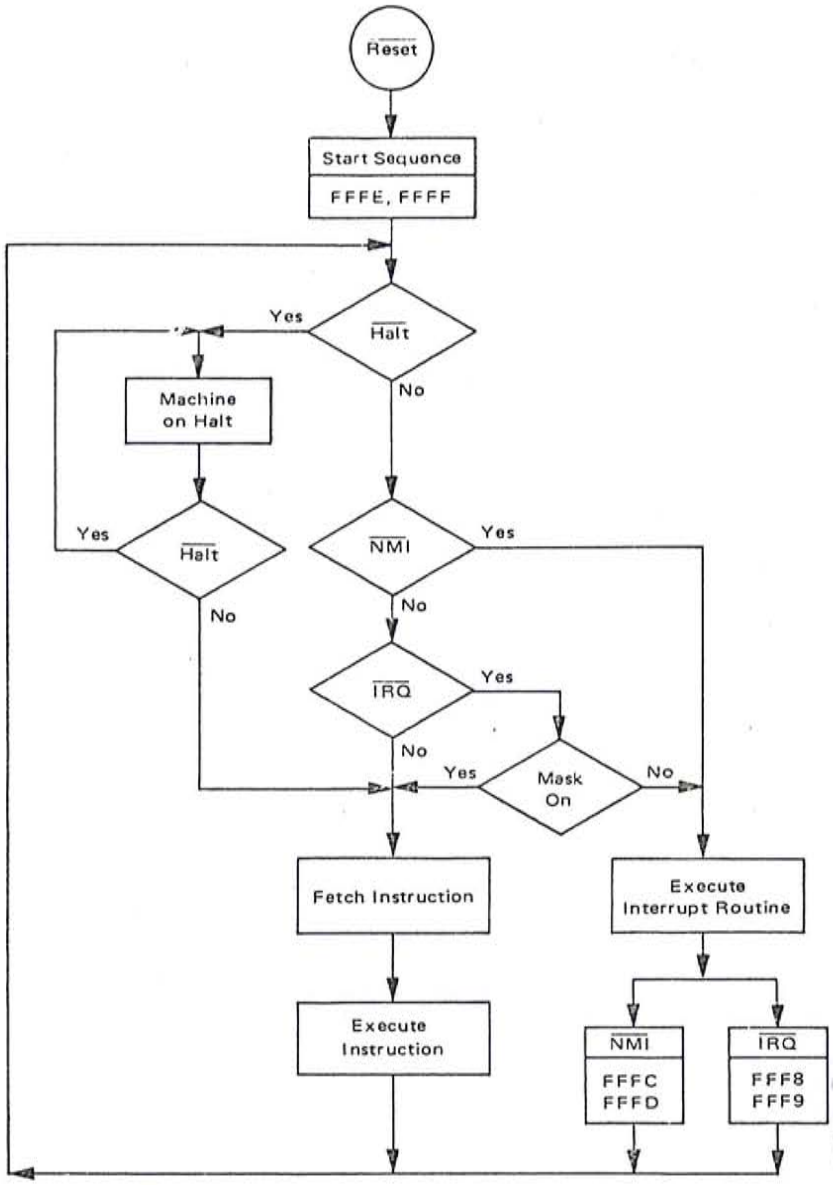
Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupt lines that are sampled during $\emptyset 2$ and will start the interrupt routine on the $\emptyset 1$ following the completion of an instruction.

Figure 2 is a flow chart describing the major decision paths and interrupt vectors of the microprocessor. Table 2 gives the memory map for interrupt vectors.

TABLE2 – MEMORY MAP FOR INTERRUPT VECTORS

Vector MS	Vector LS	Description
FFFE	FFFF	Restart
FFFC	FFFD	Non-maskable Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interrupt Request

FIGURE 2 – MPU FLOW CHART



THE PIA BUS originates on different boards.

The Peripheral Interface Adaptor on the CPU/1 P.C. Board is connected to sockets X4 and X5. On the Peripheral Interface Module (PIM/1) P.C. card there are 4 PIA chips with 8 sockets numbered X4-X5 and X7-X12. All interconnections remain common for ease of use and standardization. See Table 1 for pin numbers.

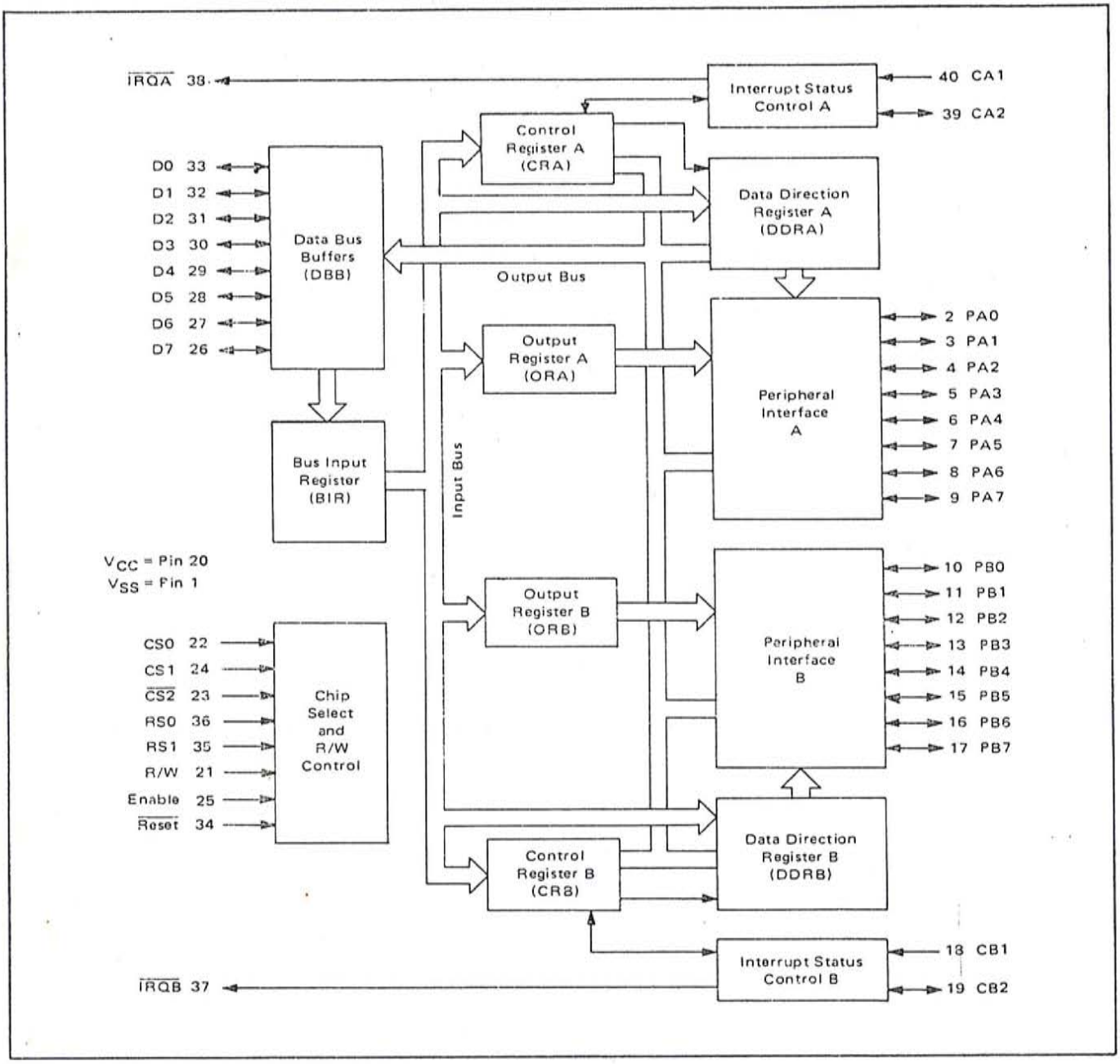
PIA INTERNAL ORGANIZATION

An expanded Block Diagram of the PIA is shown in Figure 3. Internally, the PIA is divided into two symmetrical independent register configurations. Each half has three main features: an Output Register, a Control Register, and a Data Direction Register. It is these registers that the CPU treats as memory locations, i.e., they can be either read from or written into. The Output and Data Direction Registers on each side represent a single memory location to the CPU. Selection between them is internal to the PIA and is determined by a bit in their Control Register.

The Data Direction Registers (DDR) are used to establish each individual peripheral bus line as either an input or an output. This is accomplished by having the CPU write "ones" or "zeros" into the eight bit positions of the DDR. Zeros or ones cause the corresponding peripheral data lines to function as inputs or outputs, respectively.

The Output Registers, ORA and ORB, when addressed, store the data present on the CPU Data Bus during and CPU write operation. As used here, an "CPU Write" operation refers to the execution of the "Store" instruction, i.e., writing into Output Register A is equivalent to execution of STAA PIA .ORA by the CPU. Similarly, an "CPU Read" operation is equivalent to execution of the "Load" instruction: LDAA PIAORA. This data will also appear on those peripheral lines that have been programmed as outputs. If a peripheral line has been programmed as an input, the corresponding bit position of the Output Register can still be written into by the C PU, however, the data will be influenced by the external signal applied on that peripheral data line.

During an .CPU Read operation, the data present on peripheral lines programmed as inputs is transferred directly to the system Data Bus. Due to differing circuitry, the results of reading positions programmed as outputs differ slightly between sides A and B of the PIA. On the B side, there is three-state buffering between Output Register B and the peripheral lines such that the CPU will read the current contents of ORB for those bit positions programmed as outputs. During an CPU Read of the A side, the data present on the Peripheral lines will effect the CPU Data Bus regardless of whether the lines are programmed as outputs or inputs. The bit positions in ORA designated as outputs will be read correctly only if the external loading on the Peripheral lines is within the specification for one TTL load. That is, a logic one level could be read as a logic zero if excessive loading reduced the voltage below 2.0 volts.



MC6820 PIA – Block Diagram

Figure 3

PIA Internal Organization cont'd.

The two Control Registers, CRA and CRB, allow the CPU to establish and control the operating modes of the peripheral control lines, CA1, CA2, CB1, and CB2. It is by means of these four lines that control information is passed back and forth between the CPU and peripheral devices. The control word format and a summary of its features is shown in Figure 4.

The Data Direction Register access bit (b_2 = DDR Access) is used in conjunction with the register select lines to select between internal registers. For a given register select combination, the status of the DDR bit determines whether the Data Direction Register (b_2 of DDR = 0) or the Output Register (b_2 of DDR = 1) is addressed by the CPU.

Each Control Register has two interrupt request flags, b_7 = IRQA(B) 1 and b_6 = IRQA(B) 2; they are set by transitions of the CA1 (CB1) and CA2 (CB2) control lines and can be read by an CPU read Control Register operation. The status of the interrupt flags cannot be altered by an CPU write instruction, that is, IRQA(B) 1 and IRQA(B) 2 are Read Only with respect to the CPU. They are indirectly reset to zero each time the CPU reads the corresponding Output Register or can be cleared with the hardware $\overline{\text{Reset}}$.

Bits b_0 and b_1 of the Control Registers determine the CA1 (CB1) operating mode. A "one" written into b_1 by the CPU will cause subsequent positive going transitions of the CA1 (CB1) input to set IRQA(B)1; if b_1 = 0, negative going transitions on CA1 (CB1) cause IRQA(B)1 to set. If b_0 = 1 when the IRQA(B)1 flag goes high, the PIA's external interrupt request line, IRQA(B), immediately goes low, providing a hardware interrupt signal to the CPU. The external interrupt is disabled if b_0 = 0 when the internal interrupt is set by CA1 (CB1). If b_0 is later set by an CPU Write Control Register operation, the disable is immediately released and a pending external interrupt request will occur.

When b_5 = 0, b_3 and b_4 of the Control Register perform similarly to b_0 and b_1, controlling the $\overline{\text{IRQA(B)2}}$ interrupt via the CA2 (CB2) input. The IRQA(B) interrupt terminal, when enabled, responds to either IRQA(B)1 or IRQA(B)2.

If b_5 = 1, CA2 (CB2) acts as an output and will function in one of three modes. If b_4 is also equal to one, CA2 (CB2) serves as a program controlled set/reset output to the peripheral and follows b_3 as it is changed by CPU Write Control Register operations. If b_4 = 0 when b_5 = 1, CA2 (CB2) can be used in either a pulse strobed or handshake mode. Operation of the two sections differ slightly for these two operating modes. In the handshake mode (b_3 = 0) CA2 is taken low by the negative transition of the CPU Enable Pulse following an CPU Read Output Register operation and returns high when IRQA1 is next set by CA1.

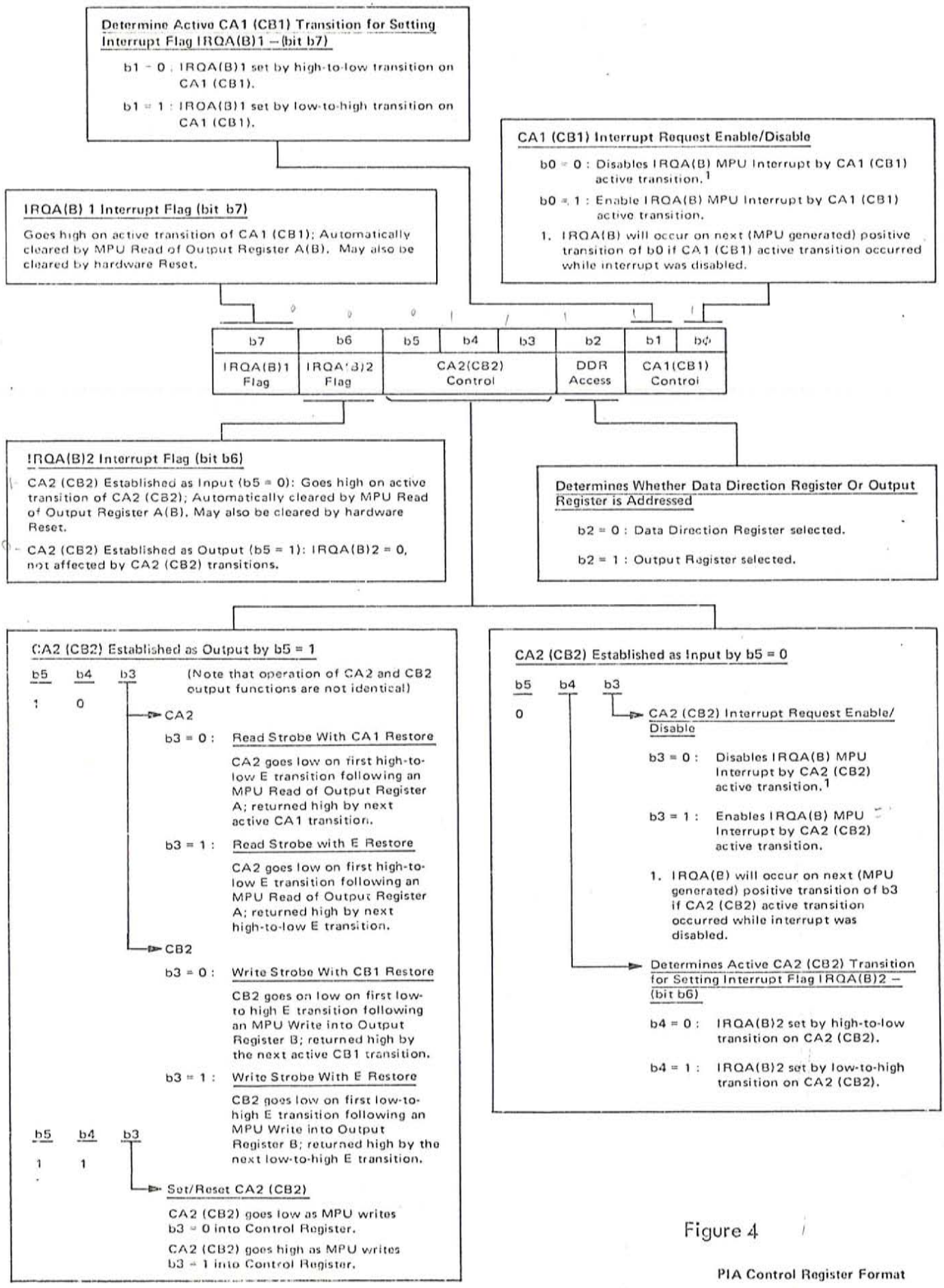


Figure 4

PIA Control Register Format

PIA Internal Organization cont'd.

This, in effect, tells the peripheral it has been read and allows it to acknowledge via CA1. The "B" Side operation is similar except that CB2 is taken low following an MPU Write Output Register operation and returned high by the next CB1 transition; this tells the peripheral it has been written into and allows it to respond via CB1.

In the pulse strobed mode ($b_3 = 1$), CA2 is again set low by a Read Output Register command, but is now returned high by the negative transition of the next MPU originated Enable Pulse. CB2 operation is similar except that an MPU Write Operation initiates the pulse. The use of A side for Read and B side for Write in those figures is not meant to imply that A and B sides must be used only for peripheral data in and out, respectively. However, the strobe modes are implemented only as shown, i.e., a strobe is not generated by an A side Write or a B side Read. Strobes can be generated for these cases by including "dummy" instructions in the program. For example, an A side Write instruction can be followed immediately by an A side dummy Read to generate the strobe. Similarly, a B side Read can be followed by a dummy Write.

LINE PRINTER

For interfacing signals to the Line Printer see the PIM/1 schematic and Table 1 for pin numbers. Socket X15 receives signals from the Line Printer. They include HOME, EMPTY, SW1, SW2, JAM, and P0. When HOME is low, it indicates the print head is at either side of the printer and ready for data. When EMPTY is low, it indicates no data has been entered into the storage buffer. SW1 will go low when the RED switch (left) is depressed. This signal will generate a clear signal for the printer. SW2 will go high when the YELLOW switch (right) is depressed. This signal places the printer on-line when first pressed (Light comes on) and Off-Line when pressed again (Light goes out).

JAM will go high if the paper is jammed or some other printer malfunction occurs. The RED Lamp (left) will come on Once the jam condition is corrected, the clear switch (RED) can be depressed to reinitialize the printer (RED Light will go out). P0 is the out of paper signal and acts the same as JAM. Holding the RED clear switch down will allow automatic paper feeding.

Sockets X13 and X14 provide signals to the printer. D0 to D5 contain the Data codes. D6 provides for lowercase letters and D7 provides for double width printing. They are true when high. RUN going low will cause the print head to start. FEED going low will cause a line feed. STROBE going low will cause the data on D0-D7 to be loaded into the buffer register. After the first strobe pulse, the EMPTY signal will go high. CLEAR going low resets the printer from a JAM or P0 condition.

Line Printer cont'd.

A system reset also causes a $\overline{\text{CLEAR}}$ signal. $\overline{\text{LAMP}}$ 1 going low causes the RED (left) lamp to light. $\overline{\text{LAMP}}$ 2 going low causes the YELLOW (right) lamp to light.

P. C. BOARD CONNECTIONS

On the CRT/1 board there are two types of signals. The "COMP VIDEO" and "GND" are used to connect via 2 wires directly to the composite video input of a video monitor. The two long paths are used for a TV antenna "Proximity" connection. See the assembly instructions for the CRT board. These wires may be soldered directly into the P.C. board hole as labeled. If the Video monitor is over 10 feet distant from the CRT board, coaxial cable should be used.

On the Communications board (COM/1 or CAS/1) there are a large number of solder holes or "posts" for direct wiring to various interface I/O peripherals. As explained in the assembly instructions, these include cassettes, modems, teletypes, RS232 lines, 20 ma current loop, and TTL compatible devices. Wires may be taken from the board edge nearest the back panel to "JACKS" along the terminal chassis or directly out to the devices. An ON/OFF relay with one normally open 10 watt contact (DC only) is also provided on this board. This relay can be used for computer control of the cassette, TTY, or any other device within the 10 watt limit.

On the PIM board the top two PIA chips with their sockets may be used to control a floppy disc (IBM Compatible types). The third PIA is prewired out to a large "breadboard" area for the customer to interface any devices he can dream up. The IC sockets have power provided to pins 14 or 16 and Ground on pins 7 or 8. A place is present for resistors and capacitors. The 4th PIA is used for the printer interface circuitry. The 20 I/O lines may be used by removing the printer interface circuitry. Then X11 and X12 will provide the connections via 14 conductor flat ribbon cables. Similiarly the first 3 PIA's can be used via their 14 pin sockets with flat ribbon cables out through the bottom or back of the terminal. This makes 80 I/O lines with +5 VDC, RESET, and Ground available as Digital TTL compatible (1 load each) interface signals.

On the CPU/2 board at the right hand side are 3 solder points for interface to a teletype or other 20 ma current loop device. In this mode add the two jumpers PB0 and PB7 and use the appropriate software. Note that PB0 and PB7 on the X5 connector socket should not be used as I/O lines. The PIA (MC6820) is required with this teletype option.

APPENDIX D

INTEGRATED CIRCUIT DESCRIPTIONS

DEVICE OPERATION

At the bus interface, the ACIA appears as two addressable memory locations. Internally, there are four registers: two read-only and two write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and three peripheral/modem control lines.

POWER ON/MASTER RESET

The master reset (CR0, CR1) should be set during system initialization to insure the reset condition and prepare for programming the ACIA functional configuration when the communications channel is required. Control bits CR5 and CR6 should also be programmed to define the state of $\overline{RTS}$ whenever master reset is utilized. The ACIA also contains internal power-on reset logic to detect the power line turn-on transition and hold the chip in a reset state to prevent erroneous output transitions prior to initialization. This circuitry depends on clean power turn-on transitions. The power-on reset is released by means of the bus-programmed master reset which must be applied prior to operating the ACIA. After master resetting the ACIA, the programmable Control Register can be set for a number of options such as variable clock divider ratios, variable word length, one or two stop bits, parity (even, odd, or none), etc.

TRANSMIT

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

RECEIVE

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of the leading mark-to-space transition of the start bit. False start bit deletion capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for an 8-bit word (7 bits plus parity), the receiver strips the parity bit (D7 = 0) so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read again to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the shift register. The above sequence continues until all characters have been received.

INPUT/OUTPUT FUNCTIONS

ACIA INTERFACE SIGNALS FOR MPU

The ACIA interfaces to the MC6800 MPU with an 8-bit bi-directional data bus, three chip select lines, a register select line, an interrupt request line, read/write line, and enable line. These signals, in conjunction with the MC6800 VMA output, permit the MPU to have complete control over the ACIA.

ACIA Bi-Directional Data (D0-D7) – The bi-directional data lines (D0-D7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs an ACIA read operation.

ACIA Enable (E) – The Enable signal, E, is a high impedance TTL compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the MC6800 $\phi 2$ Clock.

Read/Write (R/W) – The Read/Write line is a high impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When Read/Write is high (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is low, the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the Read/Write signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS0, CS1, $\overline{CS2}$) – These three high impedance TTL compatible input lines are used to address the ACIA. The ACIA is selected when CS0 and CS1 are high and $\overline{CS2}$ is low. Transfers of data to and from the ACIA are then performed under the control of the Enable signal, Read/Write, and Register Select.

Register Select (RS) – The Register Select line is a high impedance input that is TTL compatible. A high level is used to select the Transmit/Receive Data Registers and a low level the Control/Status Registers. The Read/Write signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{IRQ}$) – Interrupt Request is a TTL compatible, open-drain (no internal pullup), active low output that is used to interrupt the MPU. The Interrupt Request remains low as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set.

CLOCK INPUTS

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16 or 64 times the data rate may be selected.

Transmit Clock (Tx Clk) – The Transmit Clock input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx Clk) – The Receive Clock input is used for synchronization of received data. (In the ÷ 1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transiton of the clock.

SERIAL INPUT/OUTPUT LINES

Receive Data (Rx Data) – The Receive Data line is a high impedance TTL compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used. Data rates are in the range of 0 to 500 kbps when external synchronization is utilized.

Transmit Data (Tx Data) – The Transmit Data output line transfers serial data to a modem or other peripheral. Data rates are in the range of 0 to 500 kbps when external synchronization is utilized.

PERIPHERAL/MODEM CONTROL

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are Clear-to-Send, Request-to-Send and Data Carrier Detect.

Clear-to-Send ($\overline{CTS}$) – This high impedance TTL compatible input provides automatic control of the transmitting end of a communications link via the modem Clear-to-Send active low output by inhibiting the Transmit Data Register Empty (TDRE) status bit.

Request-to-Send ($\overline{RTS}$) – The Request-to-Send output enables the MPU to control a peripheral or modem via the data bus. The $\overline{RTS}$ output corresponds to the state of the Control Register bits CR5 and CR6. When CR6 = 0 or both CR5 and CR6 = 1, the $\overline{RTS}$ output is low (the active state). This output can also be used for Data Terminal Ready ($\overline{DTR}$).

Data Carrier Detect ($\overline{DCD}$) – This high impedance TTL compatible input provides automatic control, such as in the receiving end of a communications link by means of a modem Data Carrier Detect output. The $\overline{DCD}$ input inhibits and initializes the receiver section of the ACIA when high. A low to high transition of the Data Carrier Detect initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier when the Receive Interrupt Enable bit is set.

ACIA REGISTERS

The expanded block diagram for the ACIA indicates the internal registers on the chip that are used for the status, control, receiving, and transmitting of data. The content of each of the registers is summarized in Table 1.

TRANSMIT DATA REGISTER (TDR) | write

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed and RS • $\overline{R/W}$ is selected. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go low. Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within one bit time of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

RECEIVE DATA REGISTER (RDR) | read

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) in the status buffer to go high (full). Data may then be read through the bus by addressing the ACIA and selecting the Receive Data Register with RS and R/W high when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

CONTROL REGISTER 0 write

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/W are low. This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1) – The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver sections of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set high to reset the ACIA. After reseting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

CR1	CR0	Function
0	0	÷ 1
0	1	÷ 16
1	0	÷ 64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4) – The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even Parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6) – Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send output, and the transmission of a Break level (space). The following encoding format is used:

CR6	CR5	Function
0	0	$\overline{RTS}$ = low, Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = low, Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = high, Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = low, Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7) – Interrupts will be enabled by a high level in bit position 7 of the Control Register (CR7). Interrupts from the receiver section, Receive Data Register Full being high or by a low to high transistion on the Data Carrier Detect signal line, are enabled or disabled by the Receive Interrupt Enable Bit.

STATUS REGISTER

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is low and R/W is high. Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0 – Receive Data Register Full indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect being high also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1 – The Transmit Data Register Empty bit being set high indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The low state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect (DCD), Bit 2 – The Data Carrier Detect bit will be high when the $\overline{DCD}$ input from a modem has gone high to indicate that a carrier is not present. This bit going high causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains high after the $\overline{DCD}$ input is returned low until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ input remains high after read status and read data or master reset have occurred, the $\overline{DCD}$ status bit remains high and will follow the $\overline{DCD}$ input.

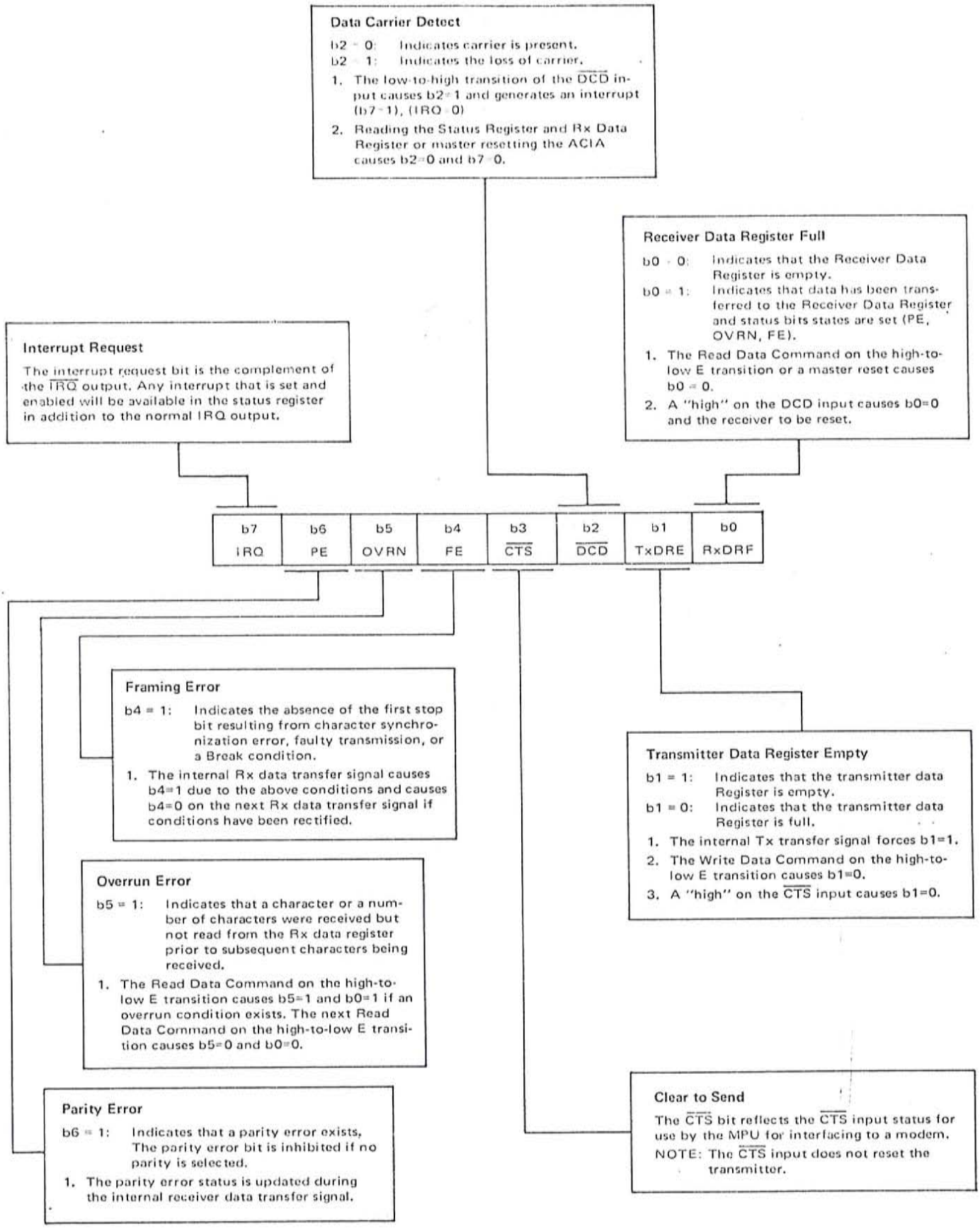
Clear-to-Send ($\overline{CTS}$), Bit 3 – The Clear-to-Send bit indicates the state of the Clear-to-Send input from a modem. A low $\overline{CTS}$ indicates that there is a Clear-to-Send from the modem. In the high state, the Transmit Data Register Empty bit is inhibited and the Clear-to-Send status bit will be high. Master reset does not affect the Clear-to-Send Status bit.

Framing Error (FE), Bit 4 – Framing error indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the 1st stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The framing error flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5 – Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The Overrun does not occur in the Status Register until the valid character prior to Overrun has been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register. Overrun is also reset by the Master Reset.

Parity Error (PE), Bit 6 – The parity error flag indicates that the number of highs (ones) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver parity check results are inhibited.

Interrupt Request (IRQ), Bit 7 – The IRQ bit indicates the state of the $\overline{IRQ}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{IRQ}$ output is low the IRQ bit will be high to indicate the interrupt or service request status.



ACIA Status Register Format

Enable for Receiver Interrupt

b7 = 1: Enables Interrupt Output in Receiving Mode

b7 = 0: Disables Interrupt Output in Receiving Mode

Counter ratio and Master reset select used in both transmitters and receiver sections

b1	b0	Function (Tx, Rx)
0	0	÷1
0	1	÷16
1	0	÷64
1	1	MASTER RESET

b7	b6	b5	b4	b3	b2	b1	b0
RIE	TC2	TC1	WS3	WS2	WS1	CDS2	CDS1

Transmitter Control Bits: Controls the Interrupt Output* and RTS Output, and provides for Transmission of a Break

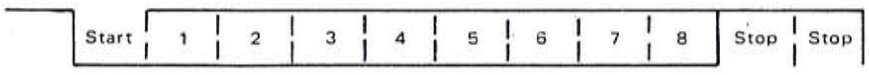
b6	b5	Function
0	0	Sets RTS = 0 and inhibits Tx interrupt (TIE)
0	1	Sets RTS = 0 and enables Tx interrupt (TIE)
1	0	Sets RTS = 1 and inhibits Tx interrupt (TIE)
1	1	Sets RTS = 0, Transmits Break and inhibits Tx interrupt (TIE)

*TIE is the enable for the interrupt output in transmit mode.

Word Length, Parity, and Stop Bit Select

b4	b3	b2	Word Length	+ Parity	+ Stop Bits
0	0	0	7	Even	2
0	0	1	7	Odd	2
0	1	0	7	Even	1
0	1	1	7	Odd	1
1	0	0	8	None	2
1	0	1	8	None	1
1	1	0	8	Even	1
1	1	1	8	Odd	1

ACIA Control Register Format

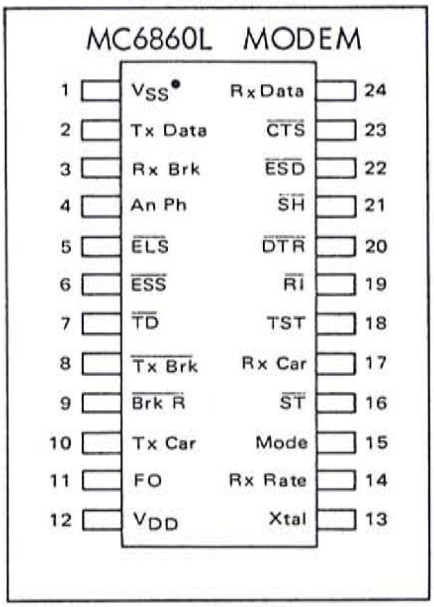


Start Bit – "Space" – Logic Zero
"Mark" – Logic One

FORMAT OF CASSETTE DATA

MC6860L

PIN ASSIGNMENT

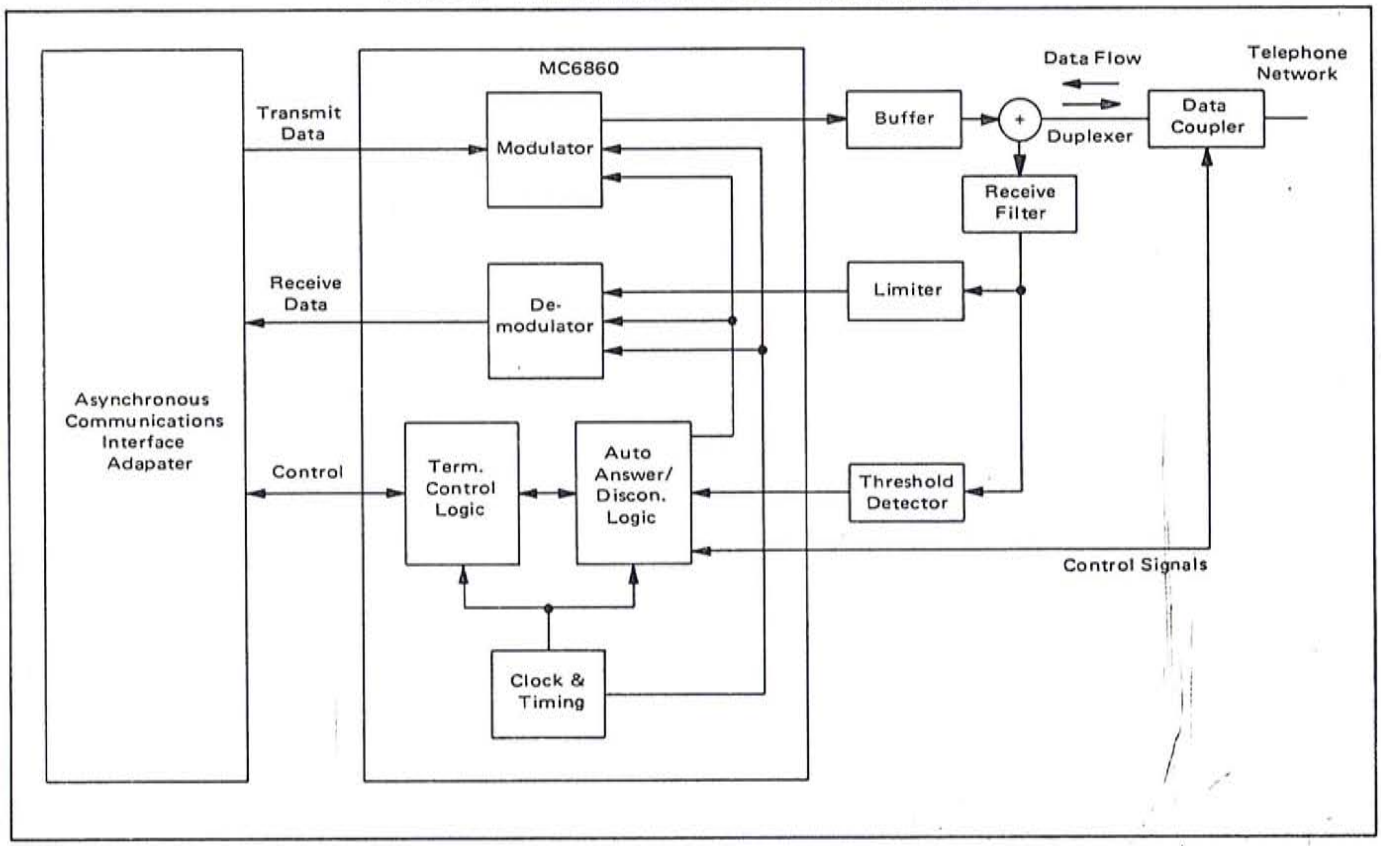


0-600 bps DIGITAL MODEM

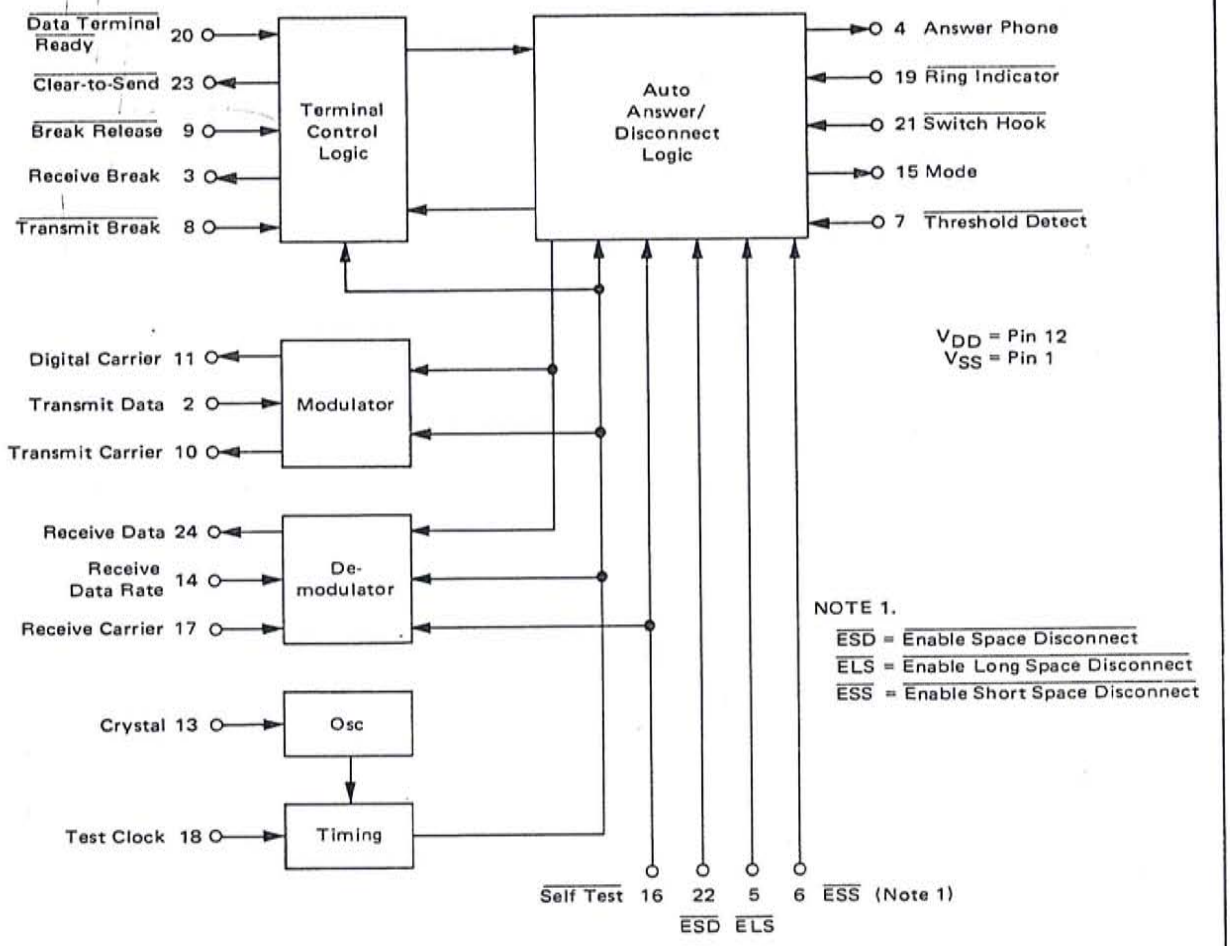
The MC6860 is a MOS subsystem designed to be integrated into a wide range of equipment utilizing serial data communications.

The modem provides the necessary modulation, demodulation and supervisory control functions to implement a serial data communications link, over a voice grade channel, utilizing frequency shift keying (FSK) at bit rates up to 600 bps. The MC6860 can be implemented into a wide range of data handling systems, including stand alone modems, data storage devices, remote data communication terminals and I/O interfaces for minicomputers.

FIGURE 2 – TYPICAL MC6860 SYSTEM CONFIGURATION



BLOCK DIAGRAM



DEVICE OPERATION

GENERAL

Figure 2 shows the modem and its interconnections. The data to be transmitted is presented in serial format to the modulator for conversion to FSK signals for transmission on the telephone line. The modulator output is buffered before driving the line.

The FSK signal from the remote modem is received via the telephone line and filtered to remove extraneous signals such as the local Transmit Carrier. This filtering can be either a bandpass which passes only the desired band of frequencies or a notch which rejects the known interfering signal. The desired signal is then limited to preserve the axis crossings and fed to the demodulator where the data is recovered from the received FSK carrier.

The Supervisory Control provides the necessary commands and responses for handshaking with the remote modem, along with the interface signals to the data coupler and communication terminal. If the modem is a built-in unit, all input-output (I/O) logic need not be RS-232 compatible. However, if the modem is a stand-alone unit the computer-modem I/O interface must conform to the EIA specification. The use of MC1488 and MC1489A line drivers and receivers will provide the required interface.

Answer Mode

Automatic answering is first initiated by a receipt of a Ring Indicator ($\overline{RI}$) signal. This can be either a low level for at least 51 ms as would come from a CBS data coupler, or at least 20 cycles of a 20-47 Hz ringing signal (low level $\geq$ 50% of the duty cycle) as would come from a CBT data coupler. The presence of the Ring Indicator signal places the modem in the Answer Mode; if the Data Terminal Ready line is low, indicating the communication terminal is ready to send or receive data, the Answer Phone output goes high. This output is designed to drive a transistor switch which will activate the Off Hook (OH) and

Data Transmission (DA) relays in the data coupler. Upon answering the phone the 2225-Hz Transmit Carrier is turned on.

The originate modem at the other end detects this 2225-Hz signal and after a 450 ms delay (used to disable any echo suppressors in the telephone network) transmits a 1270-Hz signal which the local answering modem detects, provided the amplitude and frequency requirements are met. The amplitude threshold is set external to the modem chip. If the signal level is sufficient the $\overline{TD}$ input should be low for 20 μs at least once every 32 ms. The absence of a threshold indication for a period greater than 51 ms denotes the loss of Receive Carrier and the modem begins hang-up procedures. Hang-up will occur 17 s after $\overline{RI}$ has been released provided the handshaking routine is not re-established. The frequency tolerance during handshaking is ±100 Hz from the Mark frequency.

After the 1270-Hz signal has been received for 150 ms, the Receive Data is unclamped from a Mark condition and data can be received. The Clear-to-Send output goes low 450 ms after the receipt of carrier and data presented to the answer modem is transmitted.

Automatic Disconnect

Upon receipt of a space of 150 ms or greater duration, the modem clamps the Receive Break high. This condition exists until a Break Release command is issued at the receiving station. Upon receipt of a 0.3 s space, with Enable Short Space Disconnect at the most negative voltage (low), the modem automatically hangs up. If Enable Long Space Disconnect is low, the modem requires 1.5 s of continuous space to hang up.

Originate Mode

Upon receipt of a Switch Hook ($\overline{SH}$) command the modem function is placed in the Originate Mode. If the Data Terminal Ready input is enabled (low) the modem will provide a logic high output at Answer Phone. The modem is now ready to receive the 2225-Hz signal from the remote answering modem. It will continue to look for this signal until 17 s after $\overline{SH}$ has been released. Disconnect occurs if the handshaking routine is not established.

Upon receiving 2225 ±100 Hz for 150 ms at an acceptable amplitude, the Receive Data output is unclamped from a Mark condition and data reception can be accomplished. 450 ms after receiving a 2225-Hz signal, a 1270-Hz signal is transmitted to the remote modem. 750 ms after receiving the 2225-Hz signal, the Clear-to-Send output is taken low and data can now be transmitted as well as received.

Initiate Disconnect

In order to command the remote modem to automatically hang up, a disconnect signal is sent by the local modem. This is accomplished by pulsing the normally low Data Terminal Ready into a high state for greater than 34 ms. The local modem then sends a 3 s continuous space and hangs up provided the Enable Space Disconnect is low. If the remote modem hangs up before 3 s, loss of Threshold Detect will cause loss of Clear-to-Send, which marks the line in Answer Mode and turns the carrier off in the Originate Mode.

If $\overline{ESD}$ is high the modem will transmit data until hang-up occurs 3 s later. Transmit Break is clamped 150 ms following the Data Terminal Ready interrupt.

INPUT/OUTPUT FUNCTIONS

Figure 3 shows the I/O interface for the low speed modem. The following is a description of each individual signal:

Receive Carrier (Rx Car)

The Receive Carrier is the FSK input to the demodulator. The local Transmit Carrier must be balanced or filtered out prior to this input, leaving only the Receive Carrier in the signal. The Receive Carrier must also be hard limited. Any half-cycle period greater than or equal to 429 ± 1.0 μs for the low band or 235 ± 1.0 μs for th high band is detected as a space.

Ring Indicator ($\overline{RI}$)

The modem function will recognize the receipt of a call from the CBT if at least 20 cycles of the 20-47 Hz ringing signal (low level ⩾ 50% of the duty cycle) are present. The CBS $\overline{RI}$ signal must be level-converted to TTL according to the EIA RS-232 specification before interfacing it with the modem function. The receipt of a call from the CBS is recognized if the $\overline{RI}$ signal is present for at least 51 ms. This input is held high except during ringing. A $\overline{RI}$ signal automatically places the modem function in the Answer Mode.

Switch Hook ($\overline{SH}$)

$\overline{SH}$ interfaces directly with the CBT and via the EIA RS-232 level conversion for the CBS. An $\overline{SH}$ signal automatically places the modem function in the Originate Mode.

$\overline{SH}$ is low during origination of a call. The modem will automatically hang up 17 s after releasing $\overline{SH}$ if the handshaking routine has not been accomplished.

Threshold Detect ($\overline{TD}$)

This input is derived from an external threshold detector. If the signal level is sufficient, the $\overline{TD}$ input must

e low for 20 μs at least once every 32 ms to maintain normal operation. An insufficient signal level indicates the absence of the Receive Carrier; an absence for less than 32 ms will not cause channel establishment to be lost; however, data during this interval will be invalid.

If the signal is present and the level is acceptable at all times, then the threshold input can be low permanently.

Loss of threshold for 51 ms or longer results in a loss of Clear-to-Send. The Transmit Carrier of the originate modem is clamped off and a constant Mark is transmitted from the answer modem.

Receive Data Rate (Rx Rate)

The demodulator has been optimized for signal-to-noise performance at 300 bps and 600 bps. The Receive Data Rate input must be low for 0-600 bps and should be high for 0-300 bps.

Transmit Data (Tx Data)

Transmit Data is the binary information presented to the modem function for modulation with FSK techniques. A high level represents a Mark.

Data Terminal Ready ($\overline{DTR}$)

The Data Terminal Ready signal must be low before the modem function will be enabled. To initiate a disconnect, $\overline{DTR}$ is held high for 34 ms minimum. A disconnect will occur 3 s later.

Break Release ($\overline{Brk\ R}$)

After receiving a 150 ms space signal, the clamped high condition of the Receive Break output can be removed by holding Break Release low for at least 20 μs.

Transmit Break ($\overline{Tx\ Brk}$)

The Break command is used to signal the remote modem to stop sending data.

A Transmit Break (low) greater than 34 ms forces the modem to send a continuous space signal for 233 ms. Transmit Break must be initiated only after $\overline{CTS}$ has been established. This is a negative edge sense input. Prior to initiating $\overline{Tx\ Brk}$, this input must be held high for a minimum of 34 ms.

Enabled Space Disconnect ($\overline{ESD}$)

When $\overline{ESD}$ is strapped low and $\overline{DTR}$ is pulsed to initiate a disconnect, the modem transmits a space for either 3 s or until a loss of threshold is detected, whichever occurs first. If $\overline{ESD}$ is strapped high, data instead of a space is transmitted. A disconnect occurs at the end of 3 s.

Enable Short Space Disconnect ($\overline{ESS}$)

$\overline{ESS}$ is a strapping option which, when low, will automatically hang up the phone upon receipt of a continuous space for 0.3 s. *$\overline{ESS}$ and $\overline{ELS}$ must not be simultaneously strapped low.*

Enable Long Space Disconnect ($\overline{ELS}$)

$\overline{ELS}$ is a strapping option which, when low, will automatically hang up the phone upon receipt of a continuous space for 1.5 s.

Crystal (Xtal)

A 1.0-MHz crystal with the following parameters is required to utilize the on-chip oscillator. A 1.0-MHz square wave can also be fed into this input to satisfy the clock requirement.

Mode:	Parallel
Frequency:	1.0 MHz ±0.1%
Series Resistance:	750 ohms max
Shunt Capacitance:	7.0 pF max
Temperature:	0-70°C
Test Level:	1.0 mW
Load Capacitance:	13 pF

When utilizing the 1.0-MHz crystal, external parasitic capacitance, including crystal shunt capacitance, must be ⩽9 pF at the crystal input.

Test Clock (TST)

A test signal input is provided to decrease the test time of the chip. In normal operation this input *must be strapped low.*

Self Test ($\overline{ST}$)

When a low voltage level is placed on this input, the demodulator is switched to the modulator frequency and demodulates the transmitted FSK signal. Channel establishment, which occurred during the initial handshake, is not lost during self test. The Mode Control output changes state during Self Test, permitting the receive filters to pass the local Transmit Carrier.

$\overline{ST}$	$\overline{SH}$	$\overline{RI}$	Mode
H	L	H	H
H	H	L	L
L	L	H	L
L	H	L	H

Answer Phone (An Ph)

Upon receipt of Ring Indicator or Switch Hook signal and Data Terminal Ready, the Answer Phone output goes high $[(\overline{SH} + \overline{RI}) \bullet \overline{DTR}]$. This signal drives the base of a transistor which activates the Off Hook and Data Transmission control lines in the data coupler. Upon call completion, the Answer Phone signal returns to a low level.

Mode

The Mode output indicates the Answer (low) or Originate (high) status of the modem. This output changes state when a Self Test command is applied.

Clear-To-Send ($\overline{CTS}$)

A low on the $\overline{CTS}$ output indicates the Transmit Data input has been unclamped from a steady Mark, thus allowing data transmission.

Receive Data (Rx Data)

The Receive Data output is the data resulting from demodulating the Receive Carrier. A Mark is a high level.

Receive Break (Rx Brk)

Upon receipt of a continuous 150 ms space, the modem automatically clamps the Receive Break output high. This output is also clamped high until Clear-to-Send is established.

Digital Carrier (FO)

A test signal output is provided to decrease the chip test time. The signal is a square wave at the transmit frequency.

Transmit Carrier (Tx Car)

The Transmit Carrier is a digitally-synthesized sine wave.

Mode	Data	Transmit Frequency	Tolerance*
Originate	Mark	1270 Hz	-0.15 Hz
Originate	Space	1070 Hz	0.09 Hz
Answer	Mark	2225 Hz	-0.31 Hz
Answer	Space	2025 Hz	-0.71 Hz

*The reference frequency tolerance is not included.

The proper output frequency is transmitted within 3.0 μs following a data bit change with no more than 2.0 μs phase discontinuity. The typical output level is 0.35 V (RMS) into a 100 k-ohm load impedance.

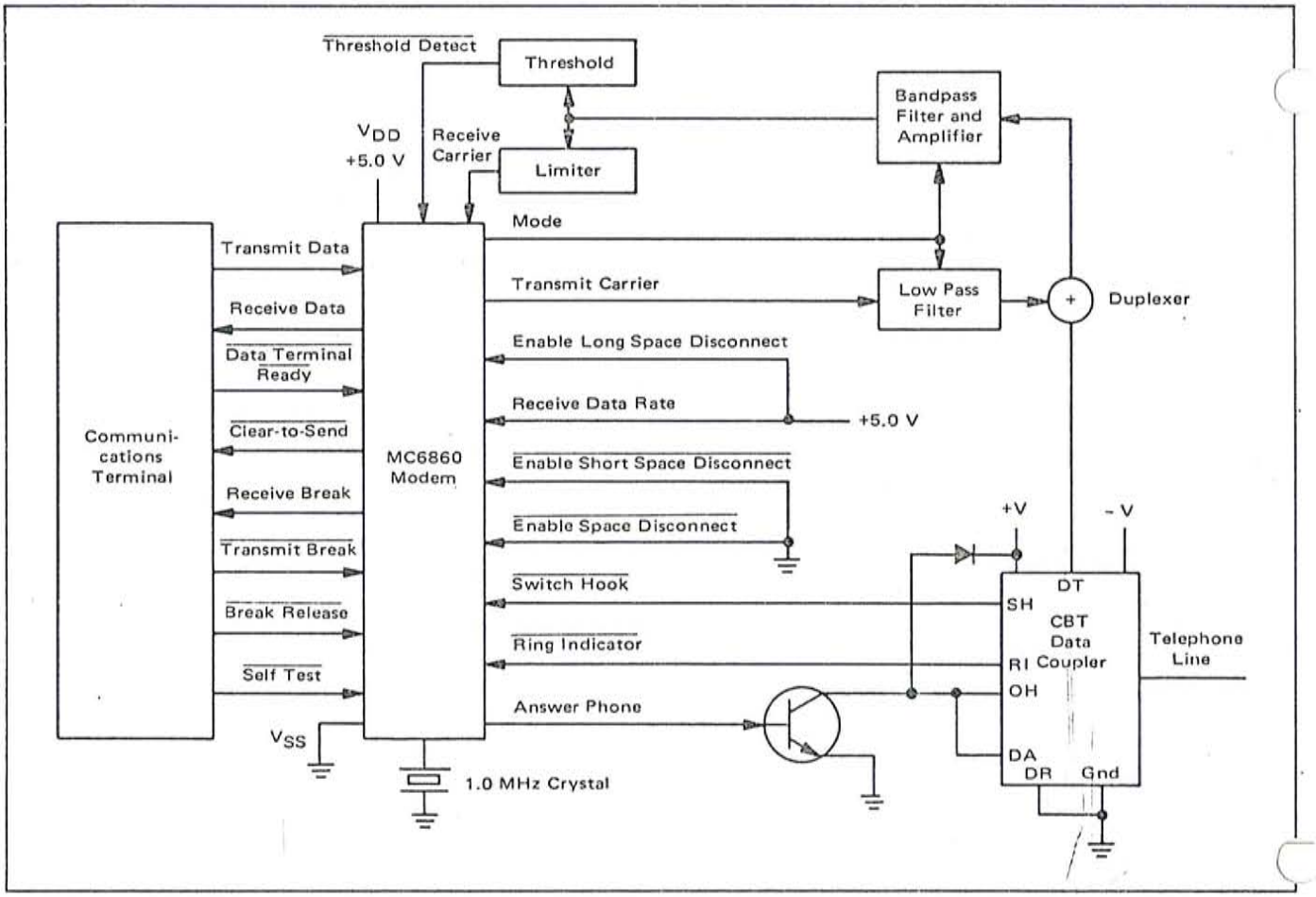
The second harmonic is typically 32 dB below the fundamental.

POWER-ON RESET

Power-on reset is provided on-chip to insure that when power is first applied the Answer Phone output is in the low (inactive) state. This holds the modem in the inactive or idle mode until a $\overline{SH}$ or $\overline{RI}$ signal has been applied. Once power has been applied, a momentary loss of power at a later time may not be of sufficient time to guarantee a chip reset through the power-on reset circuit.

To insure initial power-on reset action, the external parasitic capacitance on $\overline{RI}$ and $\overline{SH}$ should be < 30 pF. Capacitance values > 30 pF may require the use of an external pullup resistor to V_{DD} on these inputs in addition to the pullup devices already provided on chip.

FIGURE 3 – I/O INTERFACE CONNECTIONS FOR MC6860 (ORIGINATE/ANSWER MODEM)



For further data refer to the Systems Reference and Data Sheets booklet and to the M68ØØ Microprocessor Applications Manual.

APPENDIX F

User Groups, Periodicals, Books

User Groups

Association for Computing Machinery
211 East 43rd Street
New York, New York 10017

Association for Educational Data Systems
1201 16th Street N.W.
Washington, D.C.

Association of Data Processing Service Organization
947 Old York Road
Abington, Pa.

Business Equipment Manufactures Association
235 East 42nd Street
New York, New York

Data Processing Management Association
505 Busse Highway
Park Ridge, Illinois 60068

Systems and Procedures Association
7890 Brookside Drive
Cleveland, Ohio 44138

Southern California Computer Society
P.O. Box 987
South Pasadena, California

Oklahoma City Club
% Bill Cowden
2412 SW 45th
Oklahoma City, Oklahoma 73119

Personal System Newsletter
% Dave Cook
San Diego Computering Coop.
10137 Caminito Jovieal
San Diego, California 92126

Global News
"The Sphere Newsletter"
791 South 500 West
Bountiful, Utah 84010

User Groups

The Computer Hobbyist
Box 295
Cary, North Carolina 27511

Chicago Area Microcomputer Users Group
c/o Bill Precht
1102 S. Edson
Lombard, IL. 60148

Amateur Computer Society of New Jersey
c/o George Fischer
72 So. Railroad Ave.
Staten Island, NY 10305

The Amateur Computer Society
Stephen B. Gray
260 Noroton Ave
Darien, CT 06820

Homebrew Computer Club Newsletter
Fred Moore
558 Santa Cruz Ave
Menlo Park, CA 94025

Micro-8 User Group Newsletter
Hal Singer
Cabrillo Computer Center
Cabrillo High School
Lompoc, CA 93436

The Digital Groups
P.O. Box 6528
Denver, CO 80206

Staccato Notes
Derel McCall
1715 Havemeyer
Redondo Beach, CA 90278

Magazines

Byte Magazine
Petersbord, New Hampshire
03458

Computer Decisions
Hayten Publishing Co, Inc.
50 Essex St.
Rochelle Park, NJ
07662

Computer Design
221 Baker Ave
Concord, MA
01742

Computer Magazine
McGraw-Hill Publications Inc.
1221 Avenue of the Americas
New York, NY
10020

Data Communications
McGraw-Hill Publications, Inc.
1221 Avenue of the Americas
New York, NY
10020

Datamation
1801 S. La Cienega Blvd
Los Angeles, CA
90035

Digital Design
Benwill Publishing Corp.
167 Corey Rd
Brookline, MA
02146

E.D.N.
221 Columbus Ave
Boston, Mass,
02116

Electronic Design
Hayden Publishing, INC.
50 Essex St
Rochelle Park, NJ
07662

Electronics Magazine
McGraw-Hill Publications, INC.
1221 Avenue of the Americas
New York, NY
10020

Electronic Products
645 Stewart Ave
Garden City, NY
11530

Elementry Electronics
Davis Publications, INC.
229 Park Ave So.
New York, NY
10003

Infosystems
Hitchcock Building
Wheaton, Ili
60187

Modern Data
P.O. Box 369
Hudson, Mass
01749

Popular Electronics
Ziff-Davis Publishing Co.
Editorial & Executive Offices
One Park Ave
New York, NY
10016

Popular Mechanics
Hearst Corp.
224 West 57th St
New York, NY
00019

Popular Science
380 Madison Ave
New York, NY
10017

Radio Electronics
Gernback Publications, INC.
200 Park Ave So.
New York, NY
10003

Creative Computing
Ideametrics
P.O. Box 789-M
Morristown, NJ
07960

IEEE Spectrum
The Institute of Electrical and
Electronics Engineers, INC.
345 East 47 St.
New York, NY
10017

Computer & People
815 Washington St.
Newtonville, MA
02160

Data Processing
134 N. 13th St.
Philadelphia, PA
19107

Peoples Computer Company
P. O. Box 310
Menlo Park, Calif. 94025

Business Automation
Business Press International, Inc.
Elmhurst, Illinois

Computer Characteristics Quarterly
Charles W. Adams Associates, Inc.
Bedford, Mass.

Computer Digest
Detroit, Michigan 48226

Computers and Automation
Newtonville, Mass.

Data Processing Digest
Los Angeles, California

Data Processing Magazine
North American Publishing
Philadelphia, Pa.

E D P Weekley
Industry Reports, Inc.
Washington, D.C.

Information Processing Journal
Cambridge Communications Corp.
Washington, D.C. 20006

BOOKS TO READ

Basic — Robert L. Albrecht, LeRoy Finkel, and Jerald R. Brown
John Wiley & Sons, INC.

Microprogramming Handbook
Microdata

My Computer Likes Me
Dymax

101 Basic Computer Games
Digital Equipment Corp.

What to do After You Hit Return — Peoples Computer Company

Structured Programming — O.J. Dahl; E.W. Dijkstra; C.A.R. Hoare
1972 Academic Press, New York

The Logical Design of Operating Systems — Alan C. Shaw
1974 Prentice-Hall, INC, Englewood Cliffs, NJ

Fundamental Algorithms-The Art of Computer Programming — Donald E. Knuth
Addison-Wesley Pub. Co. INC. Menlo Park, CA

Switching & Finite Automat Theory — Zvi Kohavi
McGraw-Hill Book Co. New York

Introduction to Computing — T.E. Hull
1966 Prentice-Hall

Computer Architecture — Carton C. Foster
1970 Van Nostrand Reinhold Co.
450 W. 33rd St, NY, NY

Introduction to Artificial Intelligence — Philip C. Jackson
Petrocelli Books, New York

Introduction to Computer Organization & Data Structure — Harold S. Stone
McGraw-Hill 1972

Books To Read cont'd.

The Elements of Programming Style	Brain W. Kernighan/P.J. Plauger McGraw-Hill, New York, N.Y. 1974
Designing Logic Systems Using State Machines	Christopher R. Clare McGraw-Hill New York, New York 1973
T T L Cookbook	Don Lancaster, Howard W. Sams & Co. Indianapolis, Indiana 1974
Computer Lib/Dream Machines	Theodore H. Nelson Hugo's Book Service, Box 2622, Chicago, Ill.
Practical Digital Electronics an Introductory Course	Juris Bluhis & Mark Baker Hewlett-Packard Co., 1501 Page Mill Road Palo Alto, California 94304

NEWS PAPERS

Computer World
797 Washington Street
Newton, Mass. 02160

Electronic News
821 Market Street
San Francisco, California 94103
or
7 East 12 St., New York, N.Y. 10003

Mini-Computer News
167 Corey Road
Brookline, Ma. 02146

Computer Weekly
46 - 49 Porter Street, Prahran
Victoria 3181

Electronic Engineering Times
280 Community Drive
Great Neck, New York 11021

Business Automation News Report
O A Business Publications
New York, New York

Peoples Computer Company
P. O. Box 310
Menlo Park, Calif. 94025

APPENDIX G

KIT ASSEMBLY INSTRUCTIONS

GENERAL INFORMATION

WARNING 1

SOLDER EACH PIN CAREFULLY AND COMPLETELY FROM THE BACK OF THE BOARD WITH A SOLDERING IRON (NOT A SOLDERING GUN) AND DO NOT LEAVE A HOT TIP ON THE IC PIN OVER 5 SECONDS.

WARNING 2

WHENEVER PIA (40 PIN IC), 1702 PROM (24 PIN IC) OR 4K DYNAMIC MEMORIES (22 PIN CHIPS) ARE REMOVED OR HANDLED USE CAUTION TO PREVENT STATIC CHARGE. USE SPECIAL PLASTIC SHIPPING PACKAGE TO HOLD THE IDLE IC CHIP SINCE IT IS CONDUCTIVE AND PREVENTS STATIC CHARGE BUILDUP. IT IS ADVISABLE TO "GROUND" YOURSELF PRIOR TO HANDLING THIS IC, ESPECIALLY IF YOU ARE ON A RUG.

_____ CHECK PARTS Procedure: When Kit arrives check all items against the Parts List. If any part appears to be missing, check against any extra parts. Some equivalent substitutes may have been sent to avoid delays in shipping. If a shortage exists, SPHERE will mail any actual shortages you request within two weeks. All parts that are in parenthesis on the parts list are optional not provided by SPHERE, but available for the building up of the Television circuitry.

NOTE 1

The printed circuit board is marked with labels to identify location of parts. Each label includes a letter prefix followed by a number. Each letter identifies a particular type of component. A list of component types and identifiers is included below:

C CAPACITORS
D DIODES
E INTEGRATED CIRCUITS
L INDUCTOR
Q TRANSISTOR
R RESISTOR
X CONNECTOR SOCKET

To aid location of parts, a parts layout is provided for each board.

NOTE II

All sockets and IC's are to be mounted with PIN 1 located in the lower left corner (with printed circuit board writing in normal position). PIN 1 is designated by the notched end or the dot to the left.

_____ CLEANING Procedure: Using alcohol, clean the bottom of the PC board.

ASSEMBLY INSTRUCTION FOR TERMINAL (MECHANICAL)

WARNING

IN BELOW ASSEMBLY, CARE SHOULD BE TAKEN NOT TO SCRATCH UNIT

_____ CHECK PARTS Procedure: When kit arrives, check all items against the parts list.

_____ KEYBOARD SUBASSEMBLY Procedure: Assemble stand offs on PC Board Assembly as follows: Place stand offs on part side of card (observing locations of fiber washers) and attach with pan head screws.

_____ KEYBOARD ASSEMBLY Procedure: Establish Top and Bottom by placing bent section in front of you with counter sink holes up. Above subassembly mounts on rear with flat head screws.

_____ BASE ASSEMBLY Procedure: Establish Top, Bottom, Front, and Rear by placing metal in front of you with all sides and rear metal bends up. Front has no metal in it, but has three screw holes.

_____ Metal Card Rack front and rear on unit may be established by placing it in front of you and looking at front and rear metal pieces that bend up. Front metal is higher than rear.

_____ Card Rack mounts to rear of base in the right corner with four screws.

_____ Terminal Block mounts to rear of base in the left corner with two screws.

_____ Keyboard subassembly may be mounted to front of base with three pan head screws, and two flat head screws.

WARNING 1

DO NOT REMOVE 40 PIN LARGE IC FROM PACKAGE UNTIL 40 PIN SOCKET HAS BEEN SOLDERED IN PLACE ON THE P.C. BOARD. MOS Devices may be damaged by static charge. This chip can not be returned if it has been soldered at all.

WARNING 2

SOLDER EACH PIN CAREFULLY AND COMPLETELY FROM THE BACK OF THE BOARD WITH A SOLDERING IRON (NOT A SOLDERING GUN) AND DO NOT LEAVE A HOT TIP ON THE IC PIN OVER 5 SECONDS.

_____ CHECK PARTS Procedure: When Kit arrives check all items against the Parts List. If any part appears to be missing, check against any extra parts. Some equivalent substitutes may have been sent to avoid delays in shipping. If a shortage exists, SPHERE will mail any actual shortages you request within two weeks.

NOTE

The printed circuit board is marked with labels to identify location of parts. Each label includes a letter prefix followed by a number. Each letter identifies a particular type of component. A list of component types and identifiers is included below:

C	CAPACITORS
D	DIODES
E	INTEGRATED CIRCUITS
L	INDUCTOR
Q	TRANSISTOR
R	RESISTOR
X	CONNECTOR SOCKET

To aid location of parts, a parts layout is provided for each board.

_____SOCKET INSERTION Procedure: Place 14 pin dual-in-line sockets on front (right reading symbols) side of board in positions X1, X2, X3, X4 and X6. Carefully solder sockets in place. Check with ohm meter.

_____40 PIN SOCKET Procedure: Put 40 pin socket into E6 position and solder carefully and completely. Check with ohm meter.

_____DIODES AND RESISTORS Procedure: Carefully bend leads of resistors and diodes to required length and insert on front side of board in positions shown on layout sheet. Solder carefully and cut off extra lead lengths.

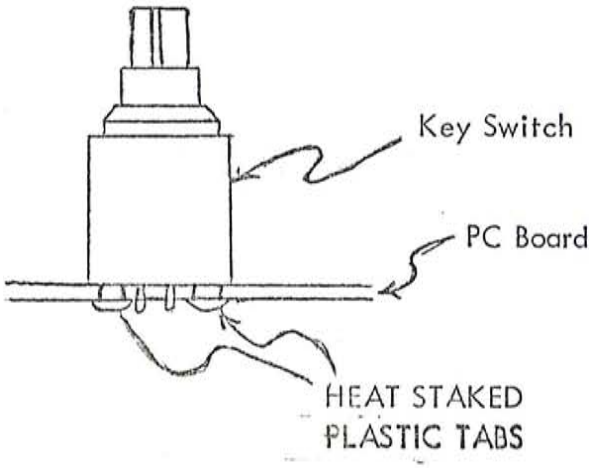
_____INTEGRATED CIRCUITS Procedure: Most Integrated Circuits (IC's) come with the leads spaced a little wider than the printed circuit board layout holes. Remove each of the small IC's one at a time and press carefully one side of the pins on a hard surface. Repeat on the other side and fit into P.C. board position holes. If IC is still too wide, repeat this process carefully until the parts fit into their proper locations. BE SURE PIN 1 IS IN THE PROPER PLACE. (The notched end to the left or the dot by the lower left corner.)

_____KEY SWITCH INSPECTION Procedure: Using an ohm meter, check the operation of each keyswitch. Failures to check for include: failure to make contact (open), always in contact (closed), and contact closure to near top of travel (i.e. wiggle switch plunger without depressing while watching ohm meter, no activity should occur).

NOTE

As keyswitches are the only mechanical parts in the kit, extra precautions should be taken to insure their proper operations. EXTRA SWITCHES ARE PROVIDED in case these precautions find faulty switches.

_____KEYSWITCH INSERTION Procedure: Place keyswitches in board one by one. While doing so, "heat stake" switches to P.C. board. This procedure involves melting the plastic tabs that stick through the P.C. board so that a flattened piece of plastic holds the key in place.



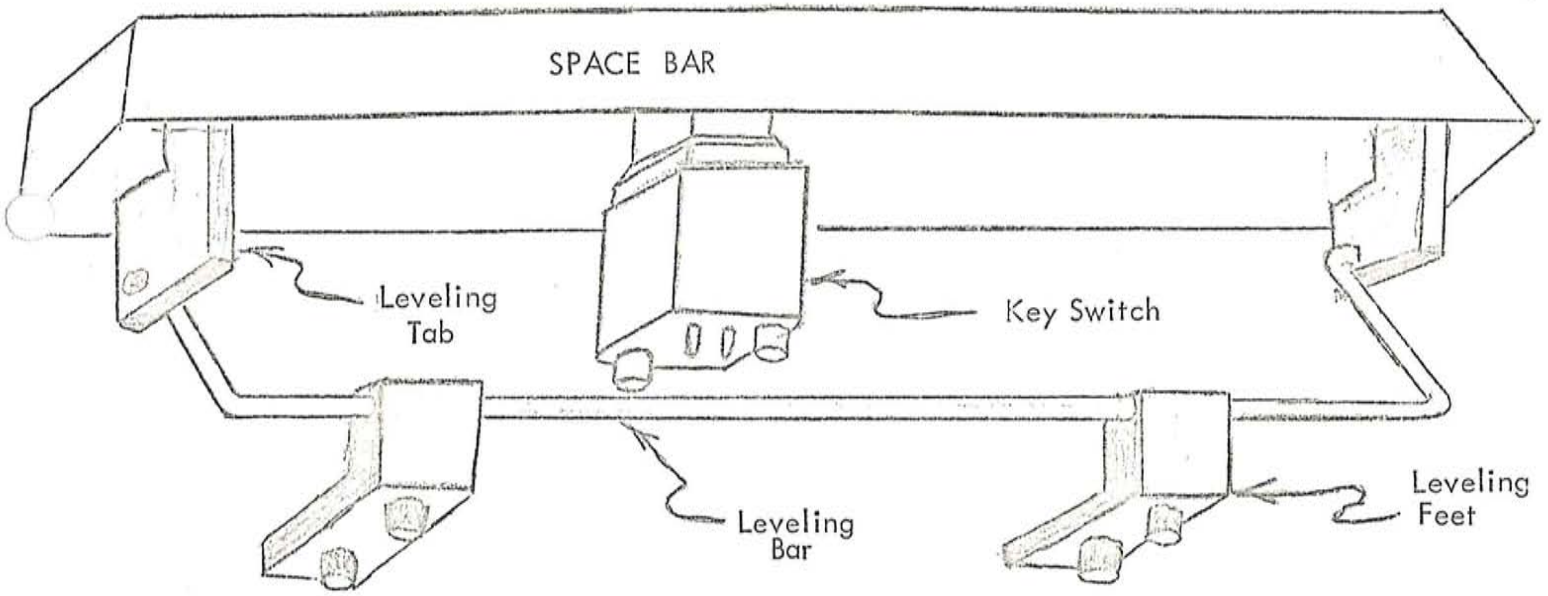
----- SOLDER SWITCHES IN PLACE

WARNING

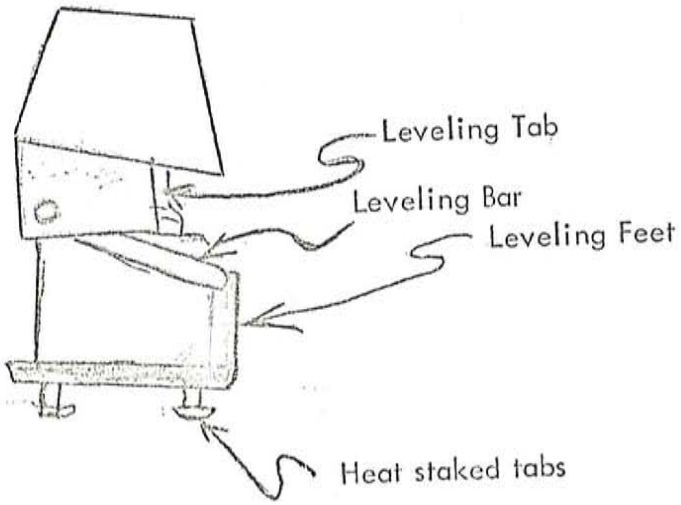
USE MINIMUM AMOUNT OF TIME TO SOLDER SWITCHES AS INTERNAL DAMAGE MAY RESULT.

_____LEVELING MECHANISM The space bar keyswitch should already be inserted, heat staked, and soldered. Insert leveling feet into PC Board in holes provided about 2 1/2 inches to each side of space bar keyswitch then heat stake each of the tabs. Now insert a leveling tab into one end of the space bar as shown in diagram. Insert the leveling bar into the leveling tab and into remaining tab which may now be inserted into the space bar. The leveling bar should now be snapped onto the leveling feet. The space bar now may be snapped onto the keyswitch.

LEVELING MECHANISM



SIDE VIEW



____CLEANING Procedure: Using alcohol, clean the bottom of the PC Board. The 40 pin PIA should not be in its socket yet.

____INSPECTION:

NOTE

CAREFUL INSPECTION WILL SAVE MANY HOURS OF DEBUGGING LATER.

Procedure: Using a magnifying glass examine each solder joint and printed circuit trace for problem "splashes" or "cold" joints.

____KEY CAPS Procedure: Put on Keycaps in positions shown on the layout sheet. They mount on plunger.

____PERIPHERIAL INTERFACE ADAPTER (PIA)

WARNING 3

WHENEVER PIA (40 PIN IC) IS REMOVED OR HANDLED USE CAUTION TO PREVENT STATIC CHARGE. USE SPECIAL PLASTIC SHIPPING PACKAGE TO HOLD THE IDLE IC CHIP SINCE IT IS CONDUCTIVE AND PREVENTS STATIC CHARGE BUILDUP. IT IS ADVISABLE TO "GROUND" YOURSELF PRIOR TO HANDLING THIS IC, ESPECIALLY IF YOU ARE ON A RUG.

Procedure: Carefully remove PIA (MC6820) 40 pin chip from package and insert it with the raised dot (Pin 1) in lower left corner.

____ATTACH POWER BUS Procedure: Power for this board is applied thru X6 with +5 volts on pin 14 and Ground pin 1.

____ATTACH ADDRESS & DATA BUSSES Procedure: X1 and X2 will receive the cables for the address Bus and the Ø 2 clock from the CPU/1 Module. X3 is the Data Bus.

____OPTIONAL RESET DISABLE Procedure: Your keyboard comes with the ability to generate a system reset by depressing the Shift, Control (CTRL), and Reset Keys. This option can be removed if this keyboard is to be a Satellite without System Control. To disable the Keyboard reset option cut the etch between the 2 feed thru's by "r" on the upper left side of the board by D2.

Assembly Instructions for KBD/1A cont'd.

--------OPTIONAL ADDRESS FOR KEYBOARD Procedure: Your keyboard comes pre addressed to F000 (hex notation). If your programs want to use another address or if you use this board as a Satellite you can change its address as shown in the following table. Note that A2, A3, and A4 straps are just below X2 and that the Solid etch must be cut before the dashed line is jumpered in. A_0 and A_1 go directly to the PIA and select 1 of 4 registers inside the PIA.

TABLE 1

Address	Change Lines
F000 - F003	none
F004 - F007	A2
F008 - F00B	A3
F00C - F00F	A2 and A3
F010 - F013	A4
F014 - E017	A4 and A2
F018 - F01B	A4 and A3
F01C - F01F	A4, A3, and A2

CHECKOUT

NOTE

THE THEORY OF OPERATION DETAILS CONTAINED IN THE OPERATOR AND REFERENCE MANUAL SHOULD BE USED IN CONJUNCTION WITH THE SCHEMATIC TO DEBUG THIS MODULE.

KEYBOARD PARTS LIST

ITEM	QNTY	PART #	DESCRIPTION	MANUFACTURER	DESIGNATION
1	1	SN74123	Dual Monostable Multivibrator	TI or Equiv.	E1
2	1	SN7442	4 line to 10 line decoder	" " "	E2
3	1	SN7430	8 input NAND gate	" " "	E4
4	2	SN7404	HEX Inverter	" " "	E5, E8
5	1	MC6820	Peripheral Interface Adapter	Motorola	E6
6	1	SN7400	Quad 2 input NAND gate	TI or Equiv.	E7
7	3	SN7427	Triple 3 input NOR gate	" " "	E9, E17, E3
8	1	SN7474	Dual D F/F	" " "	E10
9	1	SN7407	HEX Buffer/Driver w/open col.	" " "	E11
10	1	SN7410	Triple 3 input Pos. NAND gate	" " "	E12
11	2	SN7493N	4 Bit Binary Counter	" " "	E13, E16
12	1	SN7486	Quad Exclusive/OR gate	" " "	E14
13	1	SN74150	One of sixteen Multiplexer	" " "	E15
14	2	1N914	Diode		D1, D2
15	10	.1uf	.1uf ceramic capacitor		C1, C2, C3, C4, C5, C6, C7, C8, C9, C11
16	1	841283-100/10/8112	100uf 10 V DC Tantalum cap.		C10
17	18	RC07GF102	1K, 1/4W Resistor		R2-R19
18	1	RC07GF473	47K, 1/4W Resistor		R1
19	1	RC07GF471	470 1/4W Resistor		R20
20	5	314-AG39D	14 pin I.C. socket		X1-X4, X6
21	1	KBD/1	P.C. Board		
22	73		Key Switches	Mech. Enter.	
23	69		Key top, 1X (different symbols)	" "	
24	2		Key Top, 1 1/2X	" "	
25	1		Key top, 2X	" "	
26	1		Key top, 8X	" "	
27	1		Leveler Mechanism	" "	

ASSEMBLY INSTRUCTION FOR 04K/1 Thru 16K/1

REFERENCES:

MEM 1	MODULE LAYOUT SHEET
MEM 1	PARTS LIST AND DESIGNATION
MEM 1	SCHEMATIC

_____ 14 PIN SOCKET INSERTION Procedure: Place 1C 14 pin sockets in positions X1, X2, X3 and X4 on front (writing) side of board. Solder sockets carefully in place. Verify good solder joints with an ohm-meter.

_____ INTEGRATED CIRCUITS Procedure: Most Integrated Circuits (IC's) come with the leads spaced a little wider than the printed circuit board layout holes. Remove each of the small IC's one at a time and press carefully one side of the pins on a hard surface. Repeat on the other side and fit into P.C. board position holes. If IC is still too wide repeat this process carefully until the parts fit into their proper locations. BE SURE PIN 1 IS IN THE PROPER PLACE. (The notched end to the left or the dot by the lower left corner.) See layout sheet and Parts List for positions.

_____ DIODES AND RESISTORS Procedure: Carefully bend leads of diodes and resistors to required length and insert on front side of board in positions shown on layout sheet. Solder carefully and cut off extra lead lengths. Leave the 2 Ohm Resistors slightly off the board as they get warm.

_____ TRANSISTORS Procedure: Insert transistors on the front side of the board in positions shown on layout sheet. Be aware of the correct positioning of the transistor before inserting it into the circuit board.

_____ CAPACITORS Procedure: Insert capacitors on front side of board in positions shown on layout sheet. Solder carefully and cut off extra lead lengths. Note the polarity of the larger capacitors, (100uf 10v, 47uf 16v.)

_____ 4K DYNAMIC MEMORY (22 PIN CHIPS) Note: Your memories have been pretested at the factory and at Sphere. Procedure: Carefully remove Memory Chips from packing tube and insert into the first column of 8 designated positions. Be sure to take precautions to ground yourself before inserting them into the circuit board. Test board with first bank in place. If a failure should be noted, your grounding precautions may not be adequate. A single chip accounts for one bit in an 8 bit word and can usually be detected easily by writing into an address all Ones and reading back, then writing in all zeros and reading back. Note that all banks are common and a failure would affect the other 3 columns also. For this reason it is best to insert and test 1 bank of 8 at a time. You may order replacements for $8.00 each. (abt. 1/2 normal cost). Use a grounded soldering iron. An alternative to this procedure is to use 22 pin IC sockets. Insert and solder the sockets in place. Then carefully insert the memory chips. (Sockets are 50¢ each.)

MEM/1 MODULE

ITEM	QTY	PART NO.	DESCRIPTION	DESIGNATION
1	1	MEM/1	P.C. board 8x10x1/16	MEM/1
2	1	SN74123	One Shot	E7
3	1	SN7400	NAND Gates	E6
4	1	SN7402	NOR Gates	E3
5	1	SN7404	INVERTER	E1
6	1	SN7408	AND Gate	E5
7			SPARE	E2, E4
8	1	SN7483	Full ADDER	E8
9	2	DM8098	BUFFER	E10, E11
10	1	SN74156	2 line to 4 line DEMUX	E9
11	32	ZA-0248	4KX1 Dynamic RAM	E12-E43
12				
13	4	2N2369A	TRANSISTOR	Q1 - Q4
14	3	IN4001	DIODE	D3 - D5
15	1	IN914	DIODE	D1
16	1	IN5225B	3.0V ZENER	D2
17	4		RESISTOR, 130 , 2W	R6 - R9
18	9		RESISTOR 3.3KI/4W	R2 - R5, R11 - R15
19	1		RESISTOR 33 K 1/4W	R1
20	1		RESISTOR 100 1/4W	R10
21	1		Capacitor, Ceramic 33Pf	C4
22	46		Capacitor, Ceramic .1uf	C1, C2, C3, C5-C7, C14-C53
23	1		Capacitor, 100uf-10VDC	C12
24	5		Capacitor, 47uf 16V	C8 - C11, C13
25	4	314-AG39D	14 pin socket	X1-X3, X6

ASSEMBLY INSTRUCTIONS FOR POWER SUPPLY

WARNING

THE PRIMARY ENERGY SOURCE, 115 VAC, CAN PRODUCE A SEVERE SHOCK

_____ CHECK PARTS Procedure: When Kit arrives check all items against the Parts List.

_____ TERMINAL STRIPS Procedure: Using the bag of nuts and bolts place the 4 small terminal strips along the bottom row of holes of the main chassis sheet. The 6 larger terminals are then mounted along the Top row of holes.

(See Assembly Drawing.)

_____ TERMINAL BLOCKS Procedure: Mount TB 1 on the bottom (center back) section and TB 2 on the outside of one end piece. (See Assembly Drawing.)

_____ T 03/220 SOCKETS Procedure: Mount the Diamond Shaped Sockets on the inside of the main chassis as shown on the Assembly Drawing. These sockets are for mounting E1, E2, Q1, and Q2. Refer to the parts list for Description of Parts.

_____ COMPONENT MOUNTING Procedure: Wire the components between the Terminal Strips as shown on the Assembly drawing. Fasten wire leads firmly to terminal. The metal chassis is ground for the DC Voltage outputs. When mounting the TIP 34 A Transistors (Q1, Q2) use insulating MICA Washer. The case of these transistors is the collector and are NOT grounded. Check with an ohm-meter to insure the case isn't shorted to the chassis after mounting Q1 and Q2. E1 and E2 do have their cases tied to ground, therefore no Mica insulators are needed.

Mount the Switch and Fuseholder next. DO NOT MOUNT TRANSFORMERS YET.

_____ WIRING Procedure: Using the Schematic wire the components together. Omit transformer wires until last. Be careful to note the Emitter, base, in, and out designations on the Assembly drawing. Solder terminal after component leads and wires are all in place. Pull the Power Cord through the Strain Relief and insert in the hole IN The end plate (opposite the one with the terminal block TB2.) wire to the fuse, switch, and TB 1 as shown on the Schematic.

Assembly Instructions for Power Supply cont'd.

_____ ASSEMBLE END PLATES Procedure: Bolt on the two end plates. Put the little black grommet in the hole above TB 2 to protect wires coming from inside the supply chassis.

_____ MOUNT AND WIRE TRANSFORMERS Procedure: Last of all mount the Transformers and wire them as shown on the Schematic. The striped wire is the center Tap and indicates the Secondary side (red wires). The black wires are the Primary Leads.

_____ INSPECTION Procedure: Visually and with an ohmmeter double check all connections. Be sure diodes are wired in with the right Polarity. Remember twice checked is four times safer. An error may be fatal to a component after power is applied.

_____ TEST Procedure: After careful checking, plug in the power cord to a Standard 115 VAC wall outlet. With a volt meter check each of the D C voltages. With a Scope verify that there is less than 10 MV ripple on the top of the DC voltages. IF ripple is present, a .1uf cap from the regulator input to ground (Physically close to the regulator - LM 309 or LM 340) will normally solve the problem. The +12 volt and +5 volts must be within 10% while the -12 volt can be -10 to -15 volts and -5 volt can be ± 20%. The ON/OFF Switch should control the output voltages.

_____ CLOSE CASE Procedure: bolt on top and back Protective Screen.

_____ SYSTEM INTERCONNECTION Procedure: The 5 wire cable included should be used to connect TB 2 to the Terminal Block in the Keyboard chassis. Then the open 20 inch end of the flat ribbon power cable(PCB) can be split out and tied into that same Terminal Block on the Keyboard metal chassis. Use VOLTmeter (before attaching the ribbon cable to the P.C. boards) to verify voltages on X6 connectors.

PIN #	VOLTAGE
1,2,8,9	Ground
3,10	+ 12 V
4,11	- 5 V
5,12	- 11 to -14 volts
6,7,13,14	+ 5 V

POWER SUPPLY PARTS LIST

ITEM	QNTY	PART #	DESCRIPTION	DESIGNATION	CHECK
1	1		Metal Plate		
2	1		Metal Plate		
3	1		Metal Plate		
4	1		Metal Plate		
5	1	F23U	Transformer-Triad	T1	
6	1	F40X	Transformer-Triad	T2	
7	1	F55X	Transformer-Triad	T3	
8	1	LM309K	Voltage Regulator	U2	
9	1	LM340K-12	Voltage Regulator	U1	
10	2	TIP 34 A	Transistor, PNP Power	Q1-Q2	
11	1	VK048	Bridge Rectifier	BCR1	
12	4	3SM2(V352)	Diode, 3A Power	D5-D8	
13	1	1N4743	Zener 1.0W 13.0V	D1	
14	1	1N5348B or 1N5340B	Zener 5W 11.0V	D4	
15	1	1N5339B	Zener 5W 5.6V	D2	
16	1	1N5338B	Zener 5W 5.1V	D3	
17					
18					
19	1	4700uf-25V	Capacitor	C-6	
20	4	4700uf-16V	Capacitor	C1-C4	
21	1	2200uf-25V	Capacitor	C-5	
22					
23	3	BWH-2-.1-10%	Resistor ,1 ohm, 2W	R1, R3, R2	
24	1		Resistor 3 ohm, 1/2W	R-5	
25	1		Resistor 10 ohm, 1/2W	R-4	
26	3		Resistor 1K, 1/4W	R-6-R-7-R-10	
27	1		Resistor, 100, 2W	R8	
28	1	VL3-27-10%	Resistor, 27, 3W or 25w, 5W	R9	
29	1ea.	6677-C	Strain Reliefs		
30	1ea.		Power Cord		
31	1ea.	3190-0002	Toggle Switch		
32	3 ea.	6-140	Terminal Block, Cinch Jones	2 in P.S., 1 in chassis	
33	4 ea.	53-A	Terminal Strip, Cinch Jones		
34	6 ea.	56-C	Terminal Strip, Cinch Jones		
35	1	342004-A	Fuse Holder		
36	1	3AG/25L	Fuse, SLO-BLO 2 Amp	F1	
37	4	4606	TO3/TO-220 Socket		
38	5 ft.		Wire, Black #20 AWG		
39	5 ft.		Wire, Yellow #20 AWG		
40	5 ft.		Wire, Green #20 AWG		
41	5 ft.		Wire, Red #20 AWG		
42	10 ft.		Wire, Blue #20 AWG		
43	1 oz.	SN63	Solder, Kester		
44	1 pkg.		Hardware		
45	1 ea.		Grommet 5/8 in.		
46	4	2194	Cabinet Bumpers		
47	10'	Cable	Multi-conductor		

ASSEMBLY INSTRUCTIONS FOR CRT/1A

REFERENCES:		
	CRT/1	MODULE LAYOUT SHEET
	CRT/1	PARTS LIST AND DESIGNATIONS
	CRT/1	SCHEMATIC

_____ 14 PIN SOCKET INSERTION Procedure: Place IC 14 pin sockets in positions X1, X2, X3, X6 on front (writing) side of board. (X5 is optional on this board). Solder sockets carefully in place. Verify good solder joints with ohm-meter.

_____ INTEGRATED CIRCUITS Procedure: Most Integrated Circuits (IC's) come with the leads spaced a little wider than the Printed circuit board layout holes. Remove each of the small IC's one at a time and Press carefully one side of the pins on a hard surface. Repeat on the other side and fit into P.C. Board position holes. If IC is still too wide, repeat this process carefully until the parts fit into their proper locations. BE SURE PIN 1 IS IN THE PROPER PLACE. (The notched end to the Left or the Dot by the lower left corner). Solder each pin carefully and completely from the back of the board with a soldering iron (Not a Soldering Gun) and do not leave a hot tip on the IC pin over 10 seconds.

See the Layout sheet and Parts List for positions.

_____ CHARACTER GENERATOR 2513 Procedure: Carefully remove 2513 chip from foam, and insert into position E8. Be sure to take precautions to ground yourself before inserting this into the circuit board.

WARNING

DO NOT REMOVE 24 PIN LARGE IC's FROM PACKAGE UNTIL 24 PIN SOCKETS HAS BEEN SOLDERED IN PLACE ON THE P.C. BOARD. MOS Devices may be damaged by static charge. These Chips can not be returned if they have been soldered at all.

_____ 24 PIN CHIPS Procedure: Carefully remove MEMORIES (6810) 24 pin chips from packages and insert with raised dot (normally writing will also be in proper position when Braille dot is) at lower left corner.

WARNING

WHENEVER MEMORY CHIPS (24 PIN IC's) ARE REMOVED OR HANDLED, USE CAUTION TO PREVENT STATIC CHARGE. USE SPECIAL PLASTIC SHIPPING PACKAGE TO HOLD THE IDLE IC CHIP SINCE IT IS CONDUCTIVE AND PREVENTS STATIC CHARGE BUILDUP.

Assembly Instructions for CRT/1A cont'd.

_____ DIODES AND RESISTORS Procedure: Carefully bend leads of resistors and diodes to required length and insert on front side of board in positions shown on layout sheet. Solder carefully and cut off extra lead lengths.

_____ TRANSISTORS (attached on back)

_____ CAPACITORS Procedure: Insert on front side of board in positions shown on layout sheet. Solder carefully and cut off extra lead lengths Note the polarity of the larger capacitors, (100 F 10v, 47 F 16v) There will be a "+" on the capacitor and a "+" on the PC Board which must be aligned.

_____ INSPECTION Procedure: Following assembly, inspect carefully all Solder joints and lines for problem splashs or "cold" joints.

_____ VIDEO MONITOR Procedure: If a Video Monitor is used, connect composite VIDEO wires to the pads labeled "COMP VIDEO OUT and GND". Add a jumper from +5v to D4 along the dotted line shown.

_____ TELEVISION TRANSMITTER Procedure: If you want to connect up to a commercial TV, the TV XMIT circuit area must be built. Parts for it are shown in ( ) in the parts list and the Schematic shows the circuit. By placing a shield can over the completed TV circuit after laying a section of TV antenna Twin Lead along the shown path, you will eliminate most radiation. Components will not be supplied by SPHERE Corp. The circuit was not implimented because FCC regulations make this form of implementation impractical. If implemented you do so at your risk. Solder the twin lead antenna to the feed throughs at the bottom of the "Ground" and "Antenna" etch paths.

_____ POWER

Power for this board is applied thru X6 with +5 volts on pin 14 and Ground pin 1. Plus 12 is on pins 3 and 10. Minus 5 is on pins 4 and 11. Minus 12 is on pins 5 and 12.

_____ ADDRESS AND DATA BUS

X1 and X2 will receive the cables for the address Bus and the Ø2 clock from the CPU/1 Module. X3 is the Data Bus.

_____ ADDRESS INTERFACING

Your CRT interface board comes pre-addressed for EØØØ to EIFF (hex notation). If your programs want to use another address or if you use this board as a Satellite you can change its address as shown in the following table. Note that A9, A10, and A11 straps along the left side and that the Solid etch must be cut before the dashed line is jumpered in.

TABLE 1

Address	Change Lines
E000 - E1FF	none
E200 - E3FF	A9
E400 - E5FF	A10
E600 - E7FF	A9 and A10
E800 - E9FF	A11
EA00 - EBFF	A11 and A9
EC00 - EDFF	A11 and A10
EE00 - EFFF	A11, A10, & A9

_____ TRANSISTORS Procedure : Insert transistors on front side of board in positions shown on layout sheet. Be aware of the correct positioning of the transistor before inserting it into the circuit baard.

CRT PARTS LIST

ITEM	QNTY	PART #	DESCRIPTION	DESIGNATION	✓
1	1	CRT/1	P.C. Board		
2	4 (1)	314-AG39D	14 Pin Socket	X1-X3, X6, (X5)	
3	2	SN7400	Quad NAND gate	E24, E19	
4	2	SN7404	HEX Inverter	E13, E17	
5	1	SN7405	HEX Inverter	E16	
6	2	SN7408	Quad AND gate	E26, E27	
7	1	SN7409	Quad & Gate	E12	
8	1	SN7420	Dual NAND gate	E18	
9	2	SN7430	NAND gate	E7, E15	
10	1	SN7451	Dual AND/OR gate	E28	
11	1	SN7474	Dual D F/F	E23	
12	1	SN7490	4 bit Dec. Cntr.	E22	
13	3	SN7493	4 bit Bin. Cntr.	E6, E10, E11	
14	2	SN7495	4 bit Shift Reg.	E9, E21	
15	1	SN74123	Dual Monostable	E31	
16	3	SN74157	Quad MUX	E3, E 4, E5	
17	1	NE555	Timer	E29	
18	1	2513N (CM2140)	ASC II Char. Gen.	E8	
19	2	DM8833	Quad T/R	E1, E2	
20	4	MCM6810	128 x 8 Static RAM	E14, E20, E25, E30	
21	1	2N2222A or 2N5129	Transistor	Q1	
22	(1)	2N918	Transistor	(Q2)	
23	(1)	*L1	Inductive Coil	(L1)	
24	4 (1)	IN914	Diode	D1-D3, (D4), D5	
25	1	47 uf	Cap. 16 vdc	C35	
26	1 (1)	100 uf	Cap. 10 vdc	C1, (C31)	
27	24	.1 uf	Capacitor	C2-C14, C17-C19, C21-C22 C26-C28, C30, C33, C36	
28	1	.01 uf	Capacitor	C29	
29	3	.001 (1K)	Capacitor	C20, C34, C38	
30	(1)	27 pf	Capacitor	(C23)	
31	1 (1)	47 pf	Capacitor	(C16), C37	
32	1 (2)	470 pf	Capacitor	C15, (C24), (C32)	
33	(1)	8-25 pf	Capacitor, var	(C25)	
34	1	20K	Resistor, var	R19	
35	1	5K	Resistor, var	R20	
36	1	50K	Resistor, var	R22	
37	(1)	22	Resistor	(R13)	
38	2	20K	Resistor	R17, R16	
39	(1)	470	Resistor	(R4)	
40	8 (2)	1K	Resistor	R2, R3, R6-R9, (R12) (R15), R18, R23	
41	1	47, 1/2w	Resistor	R10	
42	1	100, 1/2w	Resistor	R11	
43					
44	2	10K	Resistor	R1, R5	
45	(1)	2.2K	Resistor	(R14)	
46	1	3.3K	Resistor	R21	

ASSEMBLY INSTRUCTIONS FOR CPU/1A

REFERENCES:
- CPU/1 MODULE LAYOUT SHEET
- CPU/1 PARTS LIST AND DESIGNATIONS
- CPU/1 SCHEMATIC

_____ SOCKET INSERTION Procedure: Place IC 14 pin sockets in positions X1, X2, X3, X6 on front (writing) side of board (If you did not order a 16D Kit, you will not have I C pin sockets for X4, X5, or E3) Solder sockets carefully into place. Verify good soldering technique with ohm-meter.

_____ INTEGRATED CIRCUITS Procedure: Most Integrated Circuits (IC's) come with the leads spaced a little wider than the Printed circuit board layout holes. Remove each of the small IC's one at a time and Press carefully one side of the pins on a hard surface. Repeat on the other side and fit into P.C. board position holes. If IC is still too wide, repeat this process carefully until the parts fit into their proper locations. BE SURE PIN 1 IS IN THE PROPER PLACE. (The notched end to the Left or the Dot by the lower left corner). Solder each pin carefully and completely from the back of the board with a soldering iron (Not a Soldering Gun) and do not leave a hot tip on the IC pin over 10 seconds. See the Layout sheet and Parts List for positions.

_____ 22 PIN SOCKETS (for Memory) Procedure: Put 22 Pin Sockets into positions E2, E5, E11, E17, E19, E26, E32, and E34. Solder carefully. Check with ohm-meter for continuity of every pin.

_____ 4K DYNAMIC MEMORIES Procedure: Remove 22 Pin chips carefully from the Shipping Container and insert into positions E2, E5, E11, E17, E19, E26, E32, and E34. Be sure to take precaution to ground yourself before inserting these into sockets on the circuit board. If memories are soldered directly, be sure to ground the soldering iron tip.

_____ 24 PIN SOCKETS Procedure: Put 24 pin sockets into positions E6, E12, E20, E33 and solder carefully and completely. Check with ohm-meter.

_____ 1702 PROM Procedure: Carefully remove 1702 PROM 24 pin chip from package and insert with Pin 1 mark (normally writing will also be in proper position when Pin 1 Mark is) at lower left corner.

_____ 40 PIN SOCKET Procedure: Put 40 pin socket into E47 position and solder carefully and completely.

WARNING 1

DO NOT REMOVE 40 PIN LARGE IC FROM PACKAGE UNTIL 40 PIN SOCKET HAS BEEN SOLDERED IN PLACE ON THE P.C. BOARD. MOS DEVICES MAY BE DAMAGED BY STATIC CHARGE. THIS CHIP CAN NOT BE RETURNED IF IT HAS BEEN SOLDERED AT ALL.

_____ CPU (XC6800) 40 PIN CHIP Procedure: Carefully remove CPU (XC6800) 40 pin chip from package and insert with raised dot (normally writing will also be in proper position when Braille dot is) at lower left corner.

_____ DIODES AND RESISTORS Procedure: Carefully bend leads of resistors and diodes to required length and insert on front side of board in positions shown on layout sheet. Solder carefully and cut off extra lead lengths.

_____ TRANSISTORS (attached on back) Procedure: Insert on front side of circuit board in positions shown on layout sheet. Note the polarity of larger capacitors, (100 F 10v, 47 F 16u). Solder carefully and cut off extra lead lengths.

_____ INSPECTION Procedure: following assembly inspect carefully all solder joints and lines for problem splashes or "cold" joints.

_____ INTERFACING INFORMATION

Power for this board is applied thru X6 with +5 volts on pin 14 and Ground pin 1.

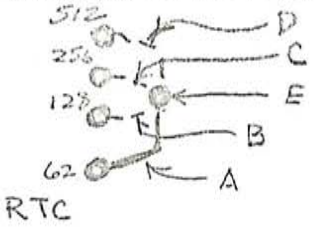
X1 and X2 will receive the cables for the address Bus and the Ø2 clock from the CPU/1 Module. X3 is the Data Bus.

Your CPU/1 Board comes with the ability to generate a system reset when power is first turned on.

Your CPU/1 Board with 4K of Memory located in addresses ØØØØ to ØFFF (hex notation). Address ØØØØ and ØØØ1 are NOT available, however.

16 D Addresses are FØ4Ø - FØ43 (optional)

_____ REAL - TIME CLOCK Information:
The real-time clock is located on the CPU Board. It is located immediately left of E8 and is denoted by the symbols "RTC". Below is the layout of the clock on the PC Board:



_____ MODIFICATION Procedure: The clock is pre-strapped to 62 interrupts per second. If 128, 256, or 512 interrupts per second are desired, the etch marked "A" must be carefully cut with an Exacto or Razor Blade Knife. An insulated wire (20 to 30 Gage) Should be cut to 1 inch and 1/4 inch insulation should be stripped from each end. One end should be soldered from the back side of the board in hole E. The other end should be inserted and soldered into the appropriate hole at the end of the dotted lines marked B, C, or D.

CPU12 PARTS LIST

REF	REF	G1		SCHEMATIC	D000019		
1	1	60		PRINTED CIRCUIT BOARD	E000020		
1	1	59		TRANSISTOR	2N2369A	Q1	
2	2	58		CAPACITOR CERAMIC 68pf 1KV	DD-680	C29,30	CRL
3	3	57		CAPACITOR CERAMIC 33pf 1KV	DD-101	C21,27,31	CRL
3	3	56		CAPACITOR CERAMIC 100PF 1KV	DDM-103	C17,22,23	CRL
2	2	55		CAPACITOR CERAMIC .01uf 150V	DDM-103	C16,26	CRL
1	1	54		CAPACITOR TANTULUM 100uf 10VDC		C5	SIEMIENS
9	9	53		CAPACITOR 47uf 16 VDC	EK47/16	C2,3,10,11,13,14,28,29,32	EDC
12	11	52		CAPACITOR CERAMIC .1uf 50V	CK-104	C1,4,6-9,12,15,19,20,25	CRL
2	2	51		RESISTOR 4.7K Ω 1/4 W.	RC07GF472	R24,27	
2	2	50		RESISTOR 10 Ω 1/4 W	RC07GF100	R23,26	
2	2	49		RESISTOR 22 Ω 1/4 W	RC07GF220	R22,25	
3	3	48		RESISTOR 10K Ω 1/4 W	RC07GF103	R21,28,30	
4	3	47		RESISTOR 1K Ω 1/4 W.	RC07GF102	R17,19,31,32	
1	-	46		RESISTOR 510 Ω 1/4 W	RC07GF511	R16	
2	2	45		RESISTOR 470 Ω 1/4 W.	RC07GF471	R15,33	
1	1	44		RESISTOR	RC07GF561	R14	
1	1	43		RESISTOR 33K Ω 1/4 W	RC07GF333	R13	
1	1	42		RESISTOR 11K Ω 1/4W 1%	RNC65H1102F	R12	
1	1	41		RESISTOR 27.1K Ω 1/4 W. 1%	RNC65H2712F	R11	
1	1	40		RESISTOR 15 K Ω 1/4 W.	RC07GF153	R10	
1	-	39		RESISTOR 2K Ω 1/4 W.	RC07GF202	R9	
5	5	38		RESISTOR 3.3K Ω 1/4 W.	RC07GF332	R7,8,18,20,29	
1	1	37		RESISTOR 130 Ω 2 W.	RC42GF131	R6	
1	-	36		RESISTOR 820 Ω 1W.	RC32GF821	R5	
1	-	35		RESISTOR 1.1K Ω 1W.	RC32GF112	R4	
2	2	34		RESISTOR 36K Ω 1/4 W.	RC07GF363	R2,3	
1	1	33		RESISTOR 100 Ω 1/4 W.	RC07GF101	R1	
6	4	32		DIODE	IN4001	D5,6,7,8,10,11	
5	5	31		DIODE	IN914	D2,3,4,9,12	
1	1	30		ZENER DIODE 3V.	IN5225B	D1	
8	8	29		22 PIN IC SOCKETS	CA 22-CSITSD		CA
2	1	28		40 PIN IC SOCKETS	340-A639D	XE3,50	AUGAT
2	4	27		24 PIN IC SOCKETS	324-A639D	XE6,12,20,35	AUGAT
6	4	26		14 PIN IC SOCKETS	314-A639D	X1-X6	AUGAT
1	-	25		QUAD RECEIVER-RS232	MC1489	E52	
1	1	24		QUAD TRANSISTOR	MPQ6842N	E51	
1	1	23		MICROPROCESSOR	MC6800	E50	
1	1	22		QUAD EXCLUSIVE OR	SN7486N	E49	
1	-	21		QUAD TRANSMITTER-RS232	MC1488N	E46	
2	2	20		QUAD TRI-STATE TRANSCEIVER	DM8833N	E47,43	
2	2	19		HEX INVERTER W/OC	SN7405N	E40,41	
1	1	18		DUAL ONE SHOTS	SN74123N	E38	
2	2	17		QUAD 2-INPUT NAND GATE	SN7400N	E32,E39	
2	-	16		PHOTO COUPLER	4N33N	E30,31	
1	1	15		HEX INVERTER	SN7404N	E29	
1	1	14		QUAD 2 INPUT AND GATE	SN7408N	E28	
-	-	13		(SPARE)		E25	
5	5	12		TRI-STATE HEX BUFFER	DM8097N	E15,21,44,47,48	
2	2	11		TRIPLE 3-INPUT AND GATE	SN7411N	E19,18	
1	1	10		4-LINE TO 10 LINE DECODER	SN7442N	E13	
3	3	9		QUAD 2-INPUT NOR GATE	SN7402N	E10,16,23	
2	2	8		DUAL D FLIP FLOP	SN7474N	E9,37	
2	2	7		DUAL ONE SHOT	9602(DM8602N)	E8,45	
3	3	6		4 BIT BINARY COUNTER	SN7493N	E7,22,27	
2	4	5		EPROM 256 X 8 BIT	MM1702A	E6,12,20,35	
2	2	4		8-INPUT NAND GATE	SN7420N	E4,24	
1	-	3		PERIPHERAL INTERFACE ADAPTER (PIA)	MC6820	E3	
8	8	2		4K DYNAMIC RAM MEMORY	2A0248	E2,5,11,14,19,26,34,36	
2	2	1		TRI-STATE HEX INVERTER	DM8098N	E1,33	
-02	-01			ASSEMBLY			

ASSEMBLY INSTRUCTIONS FOR SIM/1 BOARD

REFERENCES:

SIM/1	SCHEMATIC
SIM/1	MODULE LAYOUT SHEET
SIM/1	PARTS LISTS

_____ 14 PIN SOCKET INSERTION Procedure: Place IC 14 pin sockets in positions X1, X2, X3, X24, X25, and X6 on front (nomenclatured) side of board. Solder carefully in place. Verify good solder connections.

_____ 24 PIN SOCKET INSERTION Procedure: Same as for 14 pin sockets with the following positions depending on options purchased.
Modem option - E34
Cassette I - E19
Cassette II - E32
Extra EPROM - E3

_____ INTEGRATED CIRCUITS Procedure: Most of the integrated circuits (dual in-line packages) may now be soldered into place. Note the dot or notch on the end of the chips denotes pin 1.

Pin 1 goes to the lower left on all positions on this board (components and writing side facing you). See Layout Sheet and/or Parts List for placement. The 24 pin chips need not be removed from their packages until the assembly is complete and the initial power on tests are run. The only other chips requiring careful handling are the 4000 series CMOS. See General Soldering Instructions at the beginning of Appendix G.

_____ RELAY MOUNTING Procedure: Mount and solder the three relays into positions K1, K2, and K3. Be sure the form C type goes into K3 on the lower right side of the board. K1 and K2 are identical form A contacts.

_____ DIODES AND RESISTORS MOUNTING Procedure: Carefully bend leads of the diodes and resistors to required lengths one by one and mount as shown by Parts List and/or Layout Diagram. Solder carefully into place. Verify resistor values and placement carefully. Due to the many options on this board and the number of descrete components, great care must be used in determining which sections to build up. R74 is used as a trimmer value to establish accurate frequency rates. See Description of Options.

_____ TRANSISTOR MOUNTING Procedure: Q1, Q2, Q4, and Q5 are plastic packages associated with the two cassette interface sections vertically through the center of the board. These 4 transistors mount with the flat side (top view) facing to the right of the board (X1-X6 at the top). Q3 has a triangular lead configuration and mounts in the normal manner. Solder all transistors in place carefully. You may leave the cases above the board as much as 1/4 inch.

_____ CAPACITOR MOUNTING Procedure: Most capacitors on a digital logic board are associated with noise bypassing on the power supply lines. This board has several capacitors for this purpose; however, many others are associated with timing functions and active filter sections, and the values and types used are more critical. Follow the Layout Sheet and/or Parts List carefully. For better modem operation at 600 Baud, use good temperature compensated capacitors for C28, C29, C30, and C31. At the standard preselected 300 Baud rate, these values are not so critical. C47 and C48 are critical to establish the basic frequency of the Baud rates. These two capacitors should have reasonable temperature and tolerance specifications for teletype operation.

_____ CRYSTAL MOUNTING Procedure: Place the 1mHz crystal in the two holes by "XTAL.". Leave the case about 1/4" to 1/16" above the board.
NOTE: Ignore the dash line jumper option directly under the crystal. A future design will feature crystal controlled CPU clocks and a 1mHz signal will be brought across the address bus line X2-11. At that time, this crystal will be omitted and the jumper added at the factory. Also at that time, R74, R75, C47, C48 and E43 will be eliminated. The dashed lines in the lower left corner will be used to select either 625kHz (divides down for 300 Baud rate) or 455kHz (divides down to 110 Baud rate for teletypes). These two frequencies will be on the address bus lines X1-8 and 9 respectively.

_____ TRIMPOT MOUNTING Procedure: Place the trimpots in R20 and R63 in the 3-hole pattern provided. These are the key adjustment elements to get the cassette interface to operate satisfactorily.

OPTION SELECTION AND SET UP

_____ BOARD ADDRESSING Procedure: In the center of the upper left quadrant are three address lines that provide for 8 unique binary board addresses. A1, A2, and A3 are prestrapped to address the top ACIA (asynchronous communications interface adaptor), E19, as address F050 in HEX notation. This automatically places the bottom ACIA, E32, at address F060. If a second SIM/1 board is used, the etch by A1 should be cut and a jumper placed across the dotted lines. This moves the ACIA addresses up to F052 and F062. In a like manner, A2 could be changed (cut etch and add jumper) instead of A1 with the resulting change in addresses to F054 and F064. The ultimate selection would be with all three addresses changed (etches by A1, A2, and A3 cut and jumpers soldered in place). This would place the ACIA's respectively at F05E and F06E.
NOTE: The PROM socket at E3 is not selectable and will respond to FB00 through FBFF only. If a second board is added, E3 must be left open or else you will have two PROM's residing at the same address and your system will falter. However, you may select any of the addresses for your cassettes or modems and/or use the PROM on any one of the boards. It will still reside at the addresses just below the 4 PROMs on the CPU/2 board.

NOTE: ALL OPTION SETUP PROCEDURE ARE BASED ON P.C. BOARD IN ORIGINAL STATE.

____ TELETYPE OPTION - Setup Procedure: First select 110 baud rate clock at the bottom left of the board. To do this cut the etch on 300 and add a jumper to 110/150 selection. Then add jumper TTY1 in the Lower Left hand corner of the Board above E43. With a scope probe at the 9600 baud rate node above E44, select R74 to yield a 2.18 us period as close as possible. R74 will normally exceed 91K if needed at all. Next cut the etch at TTY2 to the right and below E28 (or remove the modem chip E34) and put in the jumper across the dotted lines. This ground will let the ACIA begin operations. The data input will be wire or'd with the RS232, modem, cassette, TTL direct, and 20 ma. TTY current loop. Since most of these inputs normally hold the receive data line low, some of them must be jumpered out.

For the Teletype option, add jumpers RS232 and TTL by above E5 and below E13 respectively. The 20 ma will NOT be jumpered.

MODEM OPTION - SETUP Procedure: If a modem option was not purchased, you will not have parts for the right hand 1/3 of the board. If you desire modem operation and have assembled this section, the following jumpers and cuts will be needed: put in the 20 ma, RS232, and TTL jumpers by or below X25. Verify a 1.6 microsecond period at the 9600 Baud rate node with a scope (lower left corner of the board) R74 may be changed to insure a 1.58 to 1.62 microsecond period. The board comes set for 300 Baud modem operation in half duplex. Connect X25-8 to another board's X25-1 and vice versa to set up the two way lines. Now decide if your board is to be the originate or answe modem. If you are initiating the call, ground the switch hook signal (SH is on X25-11). If you are receiving the call, ground the ring indicator (RI is on X25-2).
If you wish to use an acoustic data coupler (Full duplex), the following additional jumpers must be set up: DCP1 is cut and a jumper put along the dotted line. DCP2 and DCP3 are set up the same way. Now X25-8 is the data line (DL) and ground is X25-7 and X25-14 (DR and GND). The OFF hook and Data Answer are on X25-4 and 12 respectively. X25-6 is for TV.

To operate at 600 Baud, change the jumper on Baud Rate from 300 (cut etch) to 600 and make the cut and jumper change at 300/600 by the crystal. If a modem is not used (i.e. cassette #1 or teletype, etc. are used), it is best not to put the modem MC6860 on the board.
There are no tune-up pots on the modem section. If the filter sections are carefully assembled, it should run with less than .0001% error at 300 baud and about .001% errors at 600 baud. Excessive errors are normally caused by resistor values outside the 1% range, or capacitors outside their 5% tollerance range.

CASSETTE #1 OPTION - SETUP Procedure: To use ACIA #1 (E19) as a cassette interface, you must insert Jumpers 20 ma, RS232, TTL, CAS 1, CAS RXC, and CAS TXC. These latter 2 jumpers require 2 etch cuts while the former 3 are jumpers only. See description of cassette II for tune up involving the Trimpot R20.

X24-1 is the connection from the recorder output or "Earphone" jack. X24-13 is Re connection to the recorder input or "mike" jack.

X24-4 and 5 are the relay contacts to control the cassette on/off function. X24-10 will provide 24 ma of +12 volts to drive a separate relay to control the AC power line if necessary.

Jumper TTY 2 by E32 must be added and the modem chip (MC6860) E34 must be removed. Keep E34 safe in conductive foam when not in use.

____ SECOND CASSETTE INTERFACE OPTION - Procedure: E32 and the associated circuitry is designed for a 300 baud "Kansas City" standard cassette interface and plays no part in teletypes or other serial interfaces. If your Kit has this option, you will have the following parts to be added, otherwise leave these items open and build the other sections of the board.

E27, 28, 29, 30, 32, 33, 39, 40, 47, 48,
R48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 62, 64, 65, 66, 79, 80, 81, 82
C37, 38, 39, 40, 41, 42, 43, 44, 54, 56, 57, 58
Q4, Q5
Potentiometer R63
Note that E38, R76, R77, C50, and C51 comprise the 300 baud rate clock for both cassette sections and must be included unless no cassettes are to be used.

_____ TUNE UP - Procedure: R63, the variable TRIMPOT must be set carefully for error-free cassette operation. One procedure is to look at the collector of Q5 (Q2 on cassette #1) with a scope. Set the vertical on 1 volt/division with the base line 2 1/2 volts below the center (when grounded). Now begin to receive from a cassette and select the time base such that several bits (O's and 1's have slightly different amplitude as well as density on the screen) are present. Now vary the Pot until the tops of the high amplitudes bits are above the center line and the tops of the low amplitude bits are below the center line. This should be very close for good data transmission.

X24-11 is the input from the cassette recorder ("Earphone") Jack.
X24-12 is the output to the recorder "Mike" or "Aux" jack.
X24-2 and 3 are the contacts of a 10 watt relay to control the recorder on/off jack.
X24-14 will provide 24 ma of +12v to drive a separate relay if the small on board relay can not handle your load.

TABLE OF CONTENTS cont'd

TABLE OF CONTENTS cont'd