A CLOCK TIMER HARDWARE UNIT FOR AN INTEL 8080 CPU BASED COMPUTER SYSTEM

by

MICHAEL FRANZBLAU

B.S. SUNY at Stony Brook, 1975

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1977

Approved

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to thank Dr. Myron Calhoun and Earl Harris for the patience and help they provided for this "novice" who was building his first hardware project. Their suggestions and experience proved invaluable and this project would not have been completed without their help.

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

1.0 INTRODUCTION

The puropse of this report is to describe the implementation of the clock counter for an Altair 8800 microcomputer. This report will contain the design, construction and testing procedures which were used for the clock counter, and may be helpful to others who wish to build a similar unit themselves. Many of the concepts involved are very general so that the design presented here could be implemented on many different computers with only small modifications.

The clock counter is a small hardware unit which gives the computer the ability to keep track of the passage of real time. With this capability the computer can control how long programs should "run". There are many applications in which this is desireable. An operating system must be able to enforce its timeslicing policies in order to insure that no single job monopolizes the computer's facilities. This is especially true in a timesharing environment where each user must be given attention for a specified amount of time. Computer control systems may require the gathering of data from external sources (i.e. sensors, etc.) at certain specified points in time. Clearly a timing device would be beneficial for these applications.

There are many reasons why this project was chosen. This unit is a part of a larger design project (see Chapter 7) which was too complex for a Master's project. The Altair computer in the Computer Science Department does not have this facility and some of the other computers do. It is also a project suited to a novice in the hardware construction area since there are a minimum number of control signals to deal with when constructing an interface. The unique design of the Altair 8800

computer allows a user to hold it in several intermediate stages of the execution of an instruction and thereby allows a simple method of testing to be used. Control signals, data bus values and address bus values are also frozen and can be easily checked. This has proven an invaluable aid in the testing and verification phases of the project.

No hardware unit is useful without documentation. This report contains complete circuit diagrams of the clock counter circuit, complete descriptions of the command sequences involved in using the clock counter, and a "cookbook" style listing of the use of the clock counter commands. Some sample test programs are also provided to check the unit.

It is necessary to discuss the major architecture of the Altair 8800 computer first to determine the limitations of the computer. Then the design of the counter will be described followed by a description of the implementation. The last sections will describe the testing of the unit and the command sequences necessary to program and use it.

2.0 ALTAIR 8800 ARCHITECTURE CONCEPTS

Before beginning the design phase the basic architecture of the target computer must be studied. Some of the limitations of the computer might be very severe and may thus affect the design at an early stage.

The Altair 8800 computer is a very powerful microcomputer based on the Intel 8080 CPU chip and other Intel 8000 series support chips. The computer is basically a byte oriented one which is driven by three busses.[1] The data bus consists of two groups of eight unidirectional data lines (eight lines conveying data from the CPU to the external world, the eight

[1] See Block Diagram Appendix A page A-1

"data out" lines, and eight lines carrying data to the CPU from the external world, the eight "data in" lines) for a total of sixteen lines. The address bus consists of sixteen lines which communicate memory location addresses and I/O device numbers to the outside world. The control bus contains important signals which indicate the status of the computer at all times. There are two different methods of producing some of the control signal depending on which support components of the Intel 8000 series chips are used. The selection of components used will affect the implementation of the clock counter since one scheme generates more powerful control signals for interfacing than the other. Some of the control signals indicate communication to an I/O device or memory while others are involved with the generation and acknowledgement of interrupts. One control line input to the Intel 8080 CPU is the HOLD indicator which is used to freeze the CPU during the execution of certain instructions for an indefinite period of time. Other miscellaneous bus lines include such things as power lines and three inputs from the clock system which are used to drive the 8080 CPU chip. All of these lines are grouped together in a special order which is a defacto standard called the Altair Bus (S-100 bus for short). A description of the S-100 bus signals and the Intel chips used in this project can be found in Appendices C and B respectively. The reader should refer to these when necessary.

In summary, the Altair 8800 computer has a basic word size of eight bits (a byte) and has control lines for communication to I/O devices and can handle interrupts. In addition, the outputs of the system clock are available for whatever purposes are desired.

With this preliminary data the design of the clock counter can now be described.

3.0 DESIGN OF THE CLOCK TIMER

At the topmost level of design the clock unit should consist of two basic sections. The counting unit is the section that keeps track of the passing of time. The communication unit is responsible for communication between the computer and the counting unit. The block diagram in Fig. 1 below shows the basic structures and concepts involved. In order to avoid confusion, the entire unit will be called the timer or clock timer and the counting unit simply the counter or clock counter.

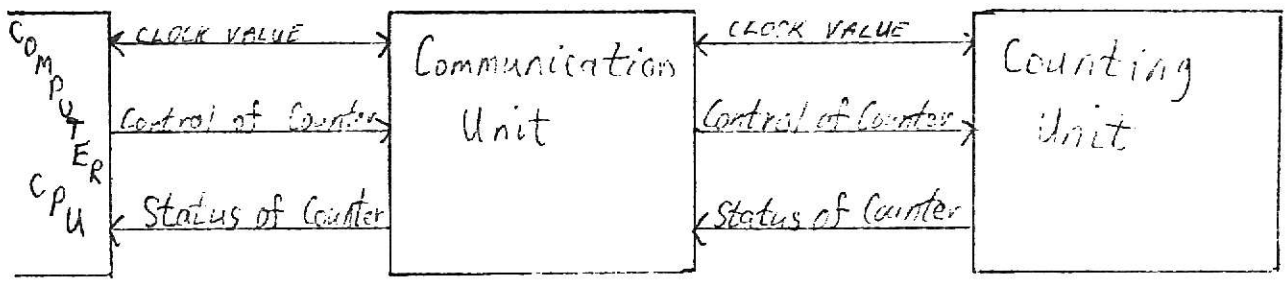


Fig 1 SIMPLIFIED BASIC DIAGRAM OF CLOCK TIMER

It is necessary to be able to instruct the counter when to start or stop counting and from what value to start counting and at what value to stop counting so that a specified time interval can be counted out. It might be desirable to know the value the counter is at for any given moment as well as the rate at which it is counting. When the counter reaches some predetermined value, the selected time has been counted out and thus a timeout has occurred. It is desirable to have a large counting range from small times in the order of microseconds to large times of a few seconds.

At this time some of the results of the discussion of the Altair architecture will be considered. The basic word size is eight bits and to control the timer device with a minimum number of commands

requires the designer to carefully consider the number of bits required for the counter. In order to reduce the size of the counter, the clock signals already provided by the computer system clock can be utilized (although this is not a necessary condition). The standard clock system used for the Intel 8080-based system uses an 18 megahertz crystal and the clock lines of the system have pulse rates of 18 megahertz and 2 megahertz. The 2 megahertz clock rate (which is 500 nanoseconds per cycle) requires a division by 2^8 to yield a time of 128 microseconds per cycle and a division by 2^{24} to yield a time of about eight seconds. Not all systems have this clock timing rate but they are on the same order of magnitude. All calculations and values used in this report will be based on the assumption that there are 500 nanoseconds between the clock pulses of the basic system clock (the $\phi 2$ clock pulses).

From these values a simple counting scheme emerges. A basic counter of size eight bits will be employed. This counter can be loaded to any desired value between 0 and 255 (00000000_2 to 11111111_2 or 000_8 to 377_8 or 00_{16} to FF_{16}). The basic clock pulses will be fed to a divider network capable of dividing by 2^8 or 2^{16} based on the desires of the user. Thus the counter can be programmed to count at different rates which are 2^8 or 2^{16} times the system clock pulse times any integer value from 1 to 256. Thus we can divide the basic system clock by any value from 2^8 to 2^{24} in multiples of 2^8. At the assumed system clock rate the timer can time from 128 microseconds to 8 seconds in multiples of 128 microseconds. We choose the value 0 for the maximum value of the counter (i.e., if the counter should reach a value of 0 when counting then a timeout has occured). With this scheme only one output

instruction is required to load a value into the clock and one output instruction to set the proper counting rate. Of course instructions are needed to start and stop the counter. At this time the basic clock diagram can be redrawn to include the details discussed up to this point, and is shown in Fig. 2 below.

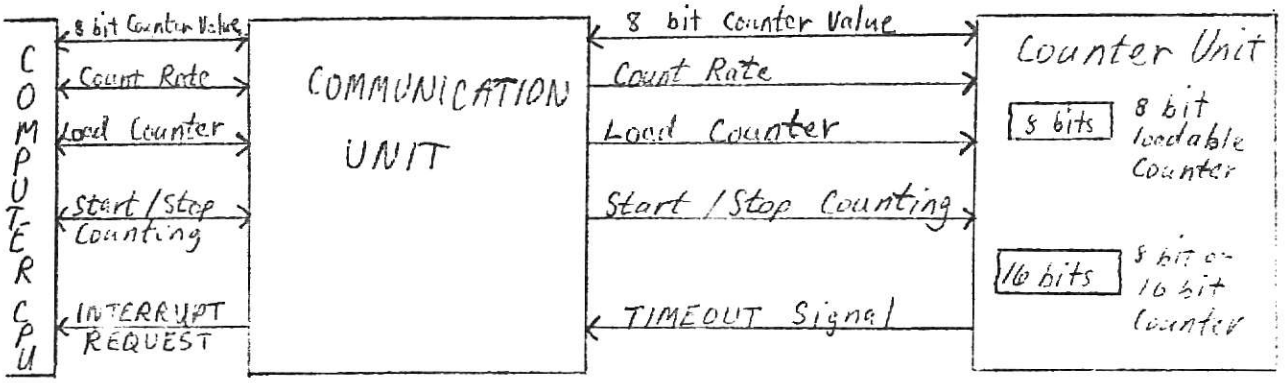


Fig 2 BLOCK DIAGRAM OF CLOCK TIMER WITH CONTROL SIGNALS

The communication unit will be responsible for coordination of all activities between the counter unit and the computer's CPU. It therefore must be a hardware interface device with the capability to communicate between two separate devices.

At this point the design has proceeded far enough to begin the selection of components and to match the capabilities of the implementation against the original design.

4.0 IMPLEMENTATION OF THE CLOCK TIMER

In this chapter the actual implemetation will be described. The components which were selected for each section of the clock timer will be described. Diagrams in the appendix section (Appendix B) give descriptions of the IC chips and pin designations used in this project. In some cases a sample circuit using the chip has been presented. After the discussion of the chip selection, the circuits using them will be presented and discussed along with some possible variations. Restrictions imposed by the Altair 8800 architecture will be discussed (for example it was necessary to add an extra chip to the Altair so that it would respond to and accept an interrupt). Full circuit diagrams are also provided so that almost anyone can build the clock timer[2].

4.1 THE COUNTER UNIT

The counter unit is basically two counters, one of 8 bits which can be preset to a desired value and the other of 16 bits to divide the basic clock pulses (the 2^8 divider network can be derived from this 2^{16} divider network). This 2^{16} dividing network must be presettable to a zero value before counting begins. The TTL 74163 IC chips which are synchronously clearable and loadable 4 bit binary counters are ideal for this purpose. The reader is referred to Appendix B page B-2 for information concerning this chip. These devices can be cascaded together without any extra logic gates to provide counters which have a multiple of 4 bits. For the 8-bit loadable counter two 74163 chips are cascaded as on page B-2. The 16-bit counter consists of four of these devices cascaded together.

[2] This was my first hardware project.

Please refer to the diagram on page D-1 while reading the description of this section of the timer.

At first sight the diagram may appear to be complex but it really is not hard to understand. The lower 74163 chips (IC's 1 thru 4) form a 16 bit counter. The 2^8 and 2^{16} count pulses are combined with the Count Rate input by the 4 gates of IC5. If the Count Rate line is high[3] then the Countup line will pulse whenever the count reaches 2^{16} and similarly when the Count Rate line is low[4] then the Countup line will pulse whenever the count reaches 2^8. The synchronous inputs are tied to ground and the Load inputs (active low) are tied high[3] since they are not needed. When the clock counter is idle because the Enable Count line is low[4] the four counters are held at 0 by the connection to the Clear input (active low).

The two upper 74163 chips (IC's 6 and 7) are cascaded to form the 8-bit loadable counter. The Clear inputs are connected to a logic 1 to keep them disabled. When the clock is counting because the Enable Count line is high the Load input is high and no values are loaded into the counter and it counts up under control of the Countup Line. If the Enable count line is low then any value desired can be loaded into the 8-bit counter from the communication unit. The eight bit count value is always available to the communication unit and the Countup line is input to the communication unit to let it know that the 8 bit counter has a new value whenever it is pulsed. The negative[5] of the desired clock value should be used since the clock counts up instead of down.

The Carry output from the upper four bit section of the counter pulses when the counter has reached its maximum value of 255 (11111111_2) and is about to change to 0. This condition represents a timeout and the

[3] true or logic 1
[4] false or logic 0
[5] two's complement

communication unit has to be informed of this. Thus this Carry signal must be an input to the communication unit.

As a final comment on the circuit one may notice that the three inputs to the counter unit have been labelled as being "amplified". This is because of a restriction on the outputs of the S-100 bus that the bus lines be input to only one TTL load, and also the same load restriction on the output of the chip used for the communication unit. These signals are needed in several places and thus must be amplified. The amplifier merely consists of a logic AND gate with one input tied to a logic 1. This increases the number of connections that can be made from one to ten at a cost of a ten nanosecond delay for average speed AND gate circuits. This small delay is negligible for this circuit.

4.2 THE COMMUNICATION UNIT

According to the design of the communication unit an 8 bit storage register is needed to hold the value which will be loaded into the 8 bit clock counter. Another 8-bit storage register is needed to hold the counter value which is read from the 8-bit clock counter. In addition several lines are needed for the control handshaking signals between the communication unit and the clock counter unit. The 8-bit loadable counter value must be transferred between the CPU and the communication unit so at least another 8-bit storage area is needed as well as the handshaking signals for this kind of transfer. In summary three 8-bit storage areas are needed if all values are to be maintained in separate areas and there must be handshaking control for two separate devices. If one 8-bit storage area serves both as the receiving area from the CPU and the area which sends the value from the communication unit to

the clock counter then only two 8-bit storage areas are needed. These requirements can be met when the Intel 8255 Programmable Peripheral Interface (PPI for short) chip is used properly. The reader should refer to the information in Appendix B pages B4 and B5 for data on the 8255 PPI chip.

When the control register of the 8255 is loaded with 264_8 the PPI takes on the characteristics that are need for the communication unit, namely, it has two 8-bit storage areas or ports and two sets of handshaking signals for these ports. The Appendix on page B4 shows a picture diagram of this configuration and the Appendix on page D-2 shows the circuit diagram of the communication unit.

Port A is a strobed-input port of size 8 bits and is loaded from the 8 output lines of the 8-bit loadable counter whenever the Countup input line (input to PC4) is low) at which time the counter is not counting or changing its value. Port B contains the clock value from the CPU which can be loaded into the 8-bit loadable counter when the Enable Counter line is low. At this time no counting occurs so Countup is low and Ports A and B have the same value, namely, the counter value.

Port C contains the two sets of handshaking signals and two settable output lines (PC6 and PC7) which are used as the Enable Count line and the Count Rate control lines respectively. Port C bits 0, 1, and 2 are handshaking signals which generate an interrupt request to the CPU when a timeout occurs. This mechanism is controlled by Port C bit 2 which must be set to 1 to enable this mechanism to function.

A requirement of the S-100 bus system is that the data bus lines must

be held tristated (effectively removed from the circuit, or in this case removed from the bus) by all devices that are not using the bus at that particular moment. The PPI has only a single bidirectional data bus while the S-100 bus system is based on two unidirectional busses. It is therefore necessary to break the data bus of the 8225 PPI chip up into two sections and tristate each of them when the device is not using them.

The 8255 chip selection circuitry is controlled by three control lines, $\overline{\text{I/O W}}$, $\overline{\text{I/O R}}$, and $\overline{\text{PORT } 374_8 \text{ SELECT}}$. Computer systems based on the Intel 8228 System Controller chip will have the $\overline{\text{I/O W}}$ and $\overline{\text{I/O R}}$ lines on the data bus and these signals must then be amplified by AND gates. Systems based upon the 8212 I/O Port used as a status latch do not have these signals and they must be generated from other control signal provided. The alternate schemes to generate these signals at the proper level are shown in Appendix D page D-3. The Altair 8800 computer used for this project was based on the 8212 status latch and thus external circuitry was needed to generate the $\overline{\text{I/O W}}$ and $\overline{\text{I/O R}}$ signals as shown on page D-3.

The 8255 PPI which uses four I/O ports for communication was placed at ports 374_8 to 377_8 because this device number group can be decoded using only one eight input NAND gate as shown in the communication unit circuit diagram (IC 9). The port selection signal is applied to the 8255 PPI device and to the inputs of two NOR gates (IC12). The other inputs are the $\overline{\text{I/OR}}$ and $\overline{\text{I/O W}}$ signals. The outputs of the NOR gates are the control inputs of the four quad bus buffers which connect to the PPI and the two unidirectional data busses of the S-100 bus. When the PPI is not active the busses are in a tristate mode and are thus removed from the

S-100 bus. When the 8255 PPI is selected because the port selection signal is low and either the $\overline{\text{I/O R}}$ or $\overline{\text{I/O W}}$ signal is active, both the 8255 PPI and the proper direction bus will be activated. The other bus will remain tristated.

The Port A lines are connected directly to the output lines of the 8-bit loadable counter and the Port B lines are connected to the counter inputs. Copying and setting of the counter value occurs under control of the Enable Count (Load/Clear) control line as described earlier. This line and the Count Rate line can be individually controlled by the instruction command group to the 8255 PPI which allows the setting and resetting of the bits of Port C (refer to Appendix B page B-4).

The Port B interrupt request mechanism enable flag described earlier is controlled by Port C bit 2 (PC2). The timing diagram for the PPI configuration used here (see Appendix B page B-5) shows that the interrupt request line (PC0) goes high to request an interrupt on the rising edge of the timeout signal (which is an input tied to PC2). The rising edge transition occurs as the 8-bit loadable counter is about to make the transition from 255 (11111111_2) to 0. An interrupt request will be generated only if PC2 is set to one when this transition occurs. If this transition occurs when PC2 is not set to one then it remains pending and the next moment that PC2 is set to one causes the interrupt to be generated. It is thus imperative to insure that a write to Port B (which resets the interrupt generation mechanism) occur before setting PC2 to one. As long as the Enable Count line is high the write to Port B will not affect the value of the clock counter. Port A works in a slightly different fshion. A value is loaded into Port A when the

Countup line is low. When an interrupt request is accepted and a read from Port A occurs immediately, the value of Port A will be 0. This is one test that is made in the testing phase.

There are several alternatives to the implementation described here. For example, the system clock pulses do not have to be used to drive the dividing network. The builder can design a very precise oscillator circuit to produce clock pulses of any desired frequency. This approach can be used whenever precise timing is required. In certain critical cases it might be necessary to use high speed chips to reduce time delays.

For systems that do not have a power-on-clear pulse to clear I/O devices when power is turned on, the illegal 8255 chip selection combination can be used to provide a program controlled reset of the device.

It is most important to check the timing diagrams of various components when interfacing them to the computer and to each other. There were no problems with the clock pulse rate at which this circuit was built to operate. At higher clock rates where the duration of the pulses might be much smaller there might be some problems because of the delays of the standard speed logic chips and some faster versions may have to be used. As an example of some of the timing requirements the 8255 PPI port signals require a steady period of 500 nanoseconds for the ACK signal and 350 nanoseconds for the STB signal (see timing diagram in Appendix B page B-5 and PPI data on page B-4). The importance of the timing diagrams should not be underestimated.

4.3 SUPPLEMENTARY HARDWARE AND CONSTRUCTION NOTES

The clock timer produces a timeout indicator signal to the computer in the form of an interrupt request. The computer used did not have the capability to respond to interrupts. It was necessary to add hardware to provide this capability. The simplest method is to use an Intel 8212 chip as an interrupt instruction port (see Appendix B page B-6) which will jam a restart instruction (RST 7) onto the data bus when the computer acknowledges the interrupt. The RST 7 instruction causes the program counter to be loaded with the address 070_8 and program execution begins at that location. The old value of the program counter is saved on the stack. The circuit used is the one suggested by Intel and can be found in the Appendix (page B-6).

No electrical circuit will work without power, so the power supply requirements will be discussed now. All of the chips described so far require +5 Volts. Since the computer bus supplies +8 Volts unregulated voltage, it was necessary to build a +5 Volt supply from the +8 Volt line. The circuit used (see Fig. 3 below) is fairly straightforward. The +5 Volt regulator that the builder uses must be able to supply sufficient current for the specific circuit being built, which was 0.8 Amperes for the design presented here.

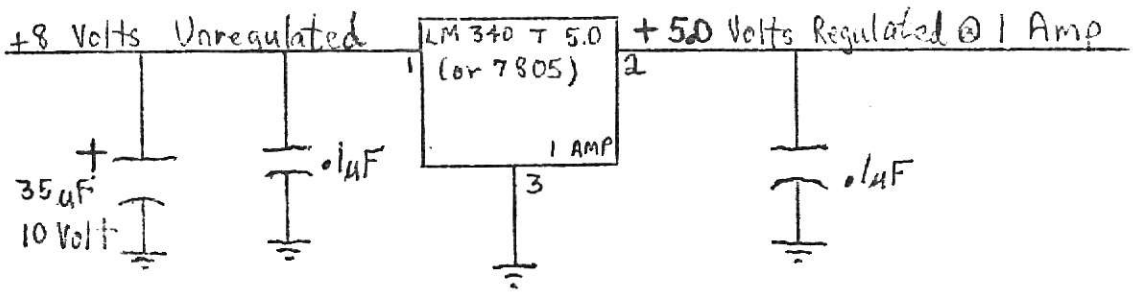


FIG. 3 POSITIVE 5 VOLT REGULATED POWER SUPPLY

For simplicity and ease of construction it was decided that a PC board which is plug-in compatible with the S-100 bus would be used (the 8800V Microprocessor Board from Vector Electronics was selected), and wire wrap sockets would be used for the IC chips to minimize the soldering requirements and make it easier to correct wiring errors.

After installing the voltage supply and verifying that it works properly the sockets should be installed and 0.1 microfarad capacitors tied between the voltage supply line and ground at every socket position to minimize electrical noise problems. The next step is to make the wiring connections and then check the connections with some kind of continuity checking device such as a buzzer to verify that the connections are correct. At this point power should be applied to the board and voltage levels checked to verify that the voltage levels are properly set. Lastly the chips should be installed (with the power off, of course) and power should be applied. If no problems are evident the circuit should then undergo further testing.

The actual circuit implemented is a combination of the diagrams in Appendix D pages D-1, D-2, D-3 and Fig. 3 . All of the numbered IC chips appear on the board that was constructed.

5.0 TESTING PROCEDURES

The purpose of this section is to present some guidelines and procedures by which the clock timer unit can be tested. A logic probe will prove to be an invaluable aid to the testing phase. It is assumed that the chips have been installed and the unit is installed in the computer with the power on.

For systems with a power-on-clear signal the PPI should be reset. The clock counter will probably be counting. If the system does not have the power-on-clear feature then manually reset the port by executing the instruction INP 377_8 (this uses the illegal 8255 chip select configuration). At this time the instruction sequence given in Fig. 4 below should be loaded into memory.

```
    *0
    MVI A,264₈
    OUT 377
LOOP: JMP LOOP
```

Location	Contents
000	076
001	264
002	323
003	377
004	303
005	004
006	000

FIG. 4 TEST PROGRAM A

This small program should be executed starting at location zero. It will loop indefinitely at location 004_8 until it is stopped manually by depressing the stop switch. The control word 264_8 conforms the 8255 PPI to the format described earlier and all of the Port C bits should be set to zero. Thus the clock should not be counting. At this time the values of the Port C bits should be checked and verified as being at zero. The 16 bit counter should have all outputs at zero. If these actions have

not occurred then execute the test program manually using the single-step switch. Freeze execution in the middle of the OUTput instruction. At this time all chip select signals should be verified as being active and the input buffer lines should also be active in the proper direction. Check the value present on the PPI data bus pins (D0 thru D7) against the value on the data bus (which should be 264_8). Make sure that all of the data bus connections are correct and that none have been switched with others. The address bus should contain all ones at this point (since there is a write to port 377_8=11111111_2) and the address bus connections should also be verified. If necessary, trace the values of all gates responsible for communication between the CPU and the 8255 PPI chip.

For the next test, the value at location 001_8 should be changed from 264_8 to 017_8. This is the command to set PC7 to 1 (selecting the slower counter rate). Execute the program and check to see that the value of the bit has indeed changed to a 1. If it has not then the checking procedure described above should be followed through once again to locate the trouble. In addition check the wiring to make sure that the PC7 line is not being held low by a wiring error.

The value in location 001_8 should now be changed to 015_8 and the program run again. The counters should now be counting and the value of PC6 should be 1. The clock should then be stopped by placing 014_8 in location 001_8 and the test program run once again. If all of these tests give positive results then check to see if the computer can read and write from the PPI and clock counter. Change the contents of location 003_8 from 377_8 to 375_8. Eight tests should then be run with

the value of location 001_8 being 001,002,004,010,020,040,100,200 (all in octal). In each case the value of the 8 bit loadable counter should agree with the value in location 001_8. If not then check the wiring to the counter from Port B of the PPI and check the buffer and data bus connections. It might be helpful to freeze the computer during the execution of the OUTput instruction so that the control lines can be checked.

To test if the computer can read the value from the counter the following test program should be loaded:

	Location	Contents	Location	Contents
*0	000	076	006	002
MVI A,?	001	?	007	020
OUT 375_8	002	323	010	000
IN 374_8	003	375	011	303
STA 20_8	004	333	012	011
LOOP: JMP LOOP	005	374	013	000

FIG. 5 TEST PROGRAM B

The values used should be the same as the eight values listed above. In each case location 001_8 and 020_8 should match.

The last test involves loading the first test program and executing it to reset the PPI. The program should then be run again after changing location 001_8 to 005_8. Finally, change location 001_8 to 015_8. When the test program is executed this time the clock will start counting from 0. Within a short time (dependent on the clock pulses used to drive the divider network) the interrupt request line should pulse. If it does not then check the Carry line from the 8 bit loadable counter. Before repeating this test be sure to reset the PPI by using code 264_8.

As a final check an oscilliscope could be used to verify that the counter is keeping the correct time.

After completing these tests succesfully the unit is ready for use. The next chapter discusses how to program the unit and presents some sample programs.

6.0 PROGRAMMING THE CLOCK TIMER UNIT

This section of the report contains information describing how to use the clock timer unit. A command guide is provided (Appendix F) as an additional aid and summarizes the discussion here. The table below lists the command instructions for the clock timer. In this chapter and in the command guide all values are in octal unless otherwise stated and the base indicator 8 will be omitted. The mnemonics given here (as well as in the previous chapter on testing) are the standard mnemonics for the 8080 processor instruction set (see Appendix E).

Mnemonics	Octal Code	Function
OUT 377	323 377	writes to the PPI control register allowing the intialization of the PPI and the setting and resetting of the bits of Port C thus controlling the clock.
OUT 375	323 375	writes an 8 bit value into Port B which is used to turn off the interrupt control line or set the clock value.
IN 376	333 376	loads the values of Port C into the accumulator of the CPU for subsequent inspection.
IN 374	333 374	transfers the contents of Port A (the clock counter value) to the CPU accumulator.

Fig. 6 CLOCK TIMER COMMANDS

The following table lists the control word values for the OUT 377 command

Control Word	Function
004	set PC2 to 0 thus disabling PPI interrupt requests
005	set PC2 to 1 thus enabling PPI interrupt requests
014	set PC6 to 0 thus stopping the clock and loading the value in Port B into the counter
015	set PC6 to 1 thus starting the clock
016	set PC7 to 0 indicating the faster counting rate
017	set PC7 to 1 indicating the slower counting rate
264	configure the PPI to the proper format

Fig. 7 CONTROL WORDS FOR THE OUT 377 COMMAND

There are certain guidelines to follow when using these commands to insure that control is maintained using the fewest possible instructions. The configuration control word 264 should be written to the PPI first before attempting any other activities with the timer. After this is done the unit is ready to be programmed. As long as power is maintained the communication unit will maintain the proper configuration and thus the control word 264 does not have to be used again. The counting rate should be set next via control words 016 and 017. For applications which will use only one clock rate this operation need be done only once. Next the clock value should be written to Port B. With the clock stopped, this value will automatically be loaded into the 8 bit loadable counter. For applications which required timing to be the same each time the clock is used this operation need only be done when an interrupt request has to be cleared (if the interrupt system is being used). This reduces control down to the command word values 014 and 015 to start and stop the clock which require one output instruction each.

If interrupts are to be used they should be turned on at this time before the clock is started by use of control word 015. The clock can be stopped whenever necessary through the use of the control word 014. The interrupt system uses an RST 7 instruction causing the interrupt handler routine at location 070 to be invoked. The sample programs presented here will use this convention. These programs will be written in standard mnemonics for the Intel 8080 processor.

The first sample program sequence illustrates the sequence to initially configure the 8255 PPI and then set up clock with a value and then starting it counting without using the interrupt system. Whenever the clock value reaches some predetermined target value a counter will be incremented and the clock counter will be allowed to continue timing throughout this process.

```
START:    MVI A,264       ; PPI configuration command word
          OUT 377         ;  configure the PPI to the proper format
               .          ;
               .          ;  it is assumed at this point that the clock
               .          ;  unit has not yet been set in any way and
               .          ;  is idle.
               .          ;
          LDA RATE        ;  see which clock rate we use (0 is slow one)
          JNZ FAST        ;  if not zero then we are using the fast rate
SLOW:     MVI A,17        ;  control word for slow clock rate
          JMP SETRATE     ;
FAST:     MVI A,16        ;  control word for fast clock rate
SETRATE:  OUT 377         ;  set clock rate
          LDA INITIAL     ;  put initial counter value into accumulator
          OUT 375         ;  write counter value to PPI and clock counter
          LDA TARGET      ;  put target clock value into accumulator
          MOV B,A         ;  place target value in B register
          LXI H,0         ;  set our program's 16 bit counter to zero
          MVI A,15        ;  control word to start the clock
          OUT 377         ;  start the clock
LOOP1:    IN 374          ;  obtain current clock value
          CMP B           ;  see if it matches the target value
          JNZ LOOP1       ;  loop around if it does not match
          INX H           ;  increment the program's counter by one
LOOP2:    IN 374          ;  obtain clock value
          CMP B           ;  see if are still at the target value
          JZ LOOP2        ;  loop until we are at the next target value
          JMP LOOP1       ;  continue counting and waiting for the target
                          ;  value.
```

Fig. 8 PROGRAM SEQUENCE 1

If it was desired to reset the counter to its initial value when the target value is reached then the lines from LOOP2 and down should be replaced by:

```
LOOP2:    MVI A,14        ;  stop counter and load Port B value into count
          OUT 377         ;
          MVI A,15        ;  command word to start the counter
          OUT 377         ;  start the clock counter
          JMP LOOP1       ;
```

The sample sequence below is a much better example of how the capabilities of the clock can be used to allow timing control with concurrent program execution. Included is a subroutine ("CLOCK") to set the clock, which might be useful in applications which require different clock rates and values.

```
          .               ;
          .               ;
          LDA RATE        ;  put clock rate indicator in B register
          MOV B,A         ;
          LDA CLKVAL      ;  obtain clock value
          MOV C,A         ;  place clock value in C register
          CALL CLOCK      ;  set the clock up and start it running
          .               ;
          .               ;  other processing and calculations
          .               ;
*70                       ;  the following starts at location 70
          JMP INTHND      ;  jump to the interrupt handler routine
          .               ;
          .               ;
CLOCK:    MOV A,B         ;  put clock rate indicator in accumulator
          JZ  FAST        ;  a 0 value indicates the fast clock rate
SLOW:     MVI A,17        ;  control word for a slow clock rate.
          JMP SETRATE     ;
FAST:     MVI A,16        ;  control word for fast clock rate
SETRATE:  OUT 377         ;  set the clock rate
          MOV A,C         ;  get clock value (should be complement of
                          ;    desired clock value)
SETVAL:   OUT 375         ;  set clock counter value
          MVI A,5         ;  command control word to enable interrupt
                          ;    request system of the PPI
SETINT:   OUT 377         ;  enable interrupt request system of PPI
ENINT:    EI              ;  allow 8080 processor to respond to interrupts
          MVI A,15        ;  command word to start clock counter
          OUT 377         ;  start clock
          RET             ;  resume normal processing
```

Fig. 9 CONCURRENT TIMING AND PROGRAM EXECUTION

The interrupt handler code was not placed in the above figure since its structure depends on the environment of the computer system. If the timer is the only device which will be handled by the interrupt handler at location 70 then its form is simple. It is possible however that other devices may use the same RST 7 instruction and thus must be serviced by the same interrupt handler. The figure below shows the alternative forms of the interrupt handlers to take care of these cases.

```
     Single Device          Multiple Devices

INTHND:PUSH PSW        INTHND:PUSH PSW          ; save status of computer and
                                                ;   accumulator value
     PUSH B                 PUSH B              ; save register b contents
     IN 374                 IN 374              ; obtain counter value from PPI
     MOV B,A                MOV B,A             ; save counter value in B reg.
     LDA CLKVAL             LDA CLKVAL          ; load a clock value
     OUT 375                OUT 375             ; reset the interrupt request line
                            MOV A,B             ; get counter value
                            CPI 0               ; see if the value is 0 indicating
                                                ;   that a timeout has occurred
                            JNZ OTHER           ; no timeout, check other devices
TIMEOUT:    .          TIMEOUT:    .            ;
                                                ; processing for timeout
                                                ;
                       OTHER:      .            ; other processing as needed
                                                ;
     POP B                  POP B               ; restore B register value
     POP PSW                POP PSW             ; restore flags
     EI                     EI                  ; enable the CPU to accept interrup
     RET                    RET                 ; return to normal processing
```

Fig. 10 INTERRUPT HANDLER ALTERNATIVE FORMS

There are many possible variations of the sample program sequences and the user's guide (Appendix F) contains some others.

The reader might notice that there is a slight delay in responding to the timeout since no interrupt is acknowledged during execution of an instruction. The maximum time delay is dependent upon the time for a memory reference and is thus system dependent. However an average delay

time value can be obtained by some test programs. This delay is very small for the average Altair 8800 computer being on the order of about 3 microseconds. For very fast counting systems this value might be taken into account when calculating the desired clock value.

A computer system employing the clock should have a set of subroutines to drive the clock in its various modes of operation to insure proper use of the clock with a minimal knowledge of the commands needed to drive it. This would give all users of a system the ability to employ the clock using some high-level commands, leaving the details to the system control program or whatever clock subroutines have been provided.

7.0 CONCLUSIONS

This report describes the design and implementation of a hardware clock timer unit for an Altair 8800 microcomputer. The structure of the Altair microcomputer affected the original design and those changes have been discussed. The inclusion of the testing procedures should help the builder to correct his unit even if he deviates from the design presented here since many of the concepts that are used should be the same as the ones presented here. Block diagrams have been used to illustrate the design principles so that the builder could choose his own group of integrated circuits to realize the timer.

The user's guide in Appendix F should help the programmer develop his own software for efficient use of the clock timer and there is a great deal of freedom in the commands of the timer. This freedom should allow the timer to be used in many different applications.

My original Master's proposal used the clock timer. The proposal contained a description of several hardware modules which would enhance the power of the Altair 8800 computer. The Status and Control Module keeps track of whether the computer is in a Supervisor or User mode. Several condition flags for different types of interrupts must also be maintained. This module is also responsible for signalling interrupts to the microprocessor and handling the necessary I/O interfacing to the processor which would allow processor control over all of the new hardware modules. The Counter Unit is necessary for it gives an operating system the ability to enforce timeslicing. A Memory Protection module is needed in a multiuser environment to provide security between separate user programs which may be in memory at the same time. This module could also extend the addressability

of the computer by providing extra lines on the address bus. The RIS (restricted instruction set) module checks all instructions as they are fetched from memory and generates interrupts when certain "sensitive" instructions are executed in User mode. These instructions (I/O instructions and interrupt system control instructions) must be prevented from occuring in User mode so that the operating system can enforce its control over the computer system. For example, a program should not be able to change the clock value in the clock timer (or it may try to give itself an infinite amount of time to run) so that output instructions must not be allowed in User mode.

The additional hardware modules would give the Altair computer the ability to support a multiuser environment, which is another example of how the Counter unit might be used.

REFERENCES

1. Intel 8080 Microcomputer System User's Manual, Intel Corp. 1976

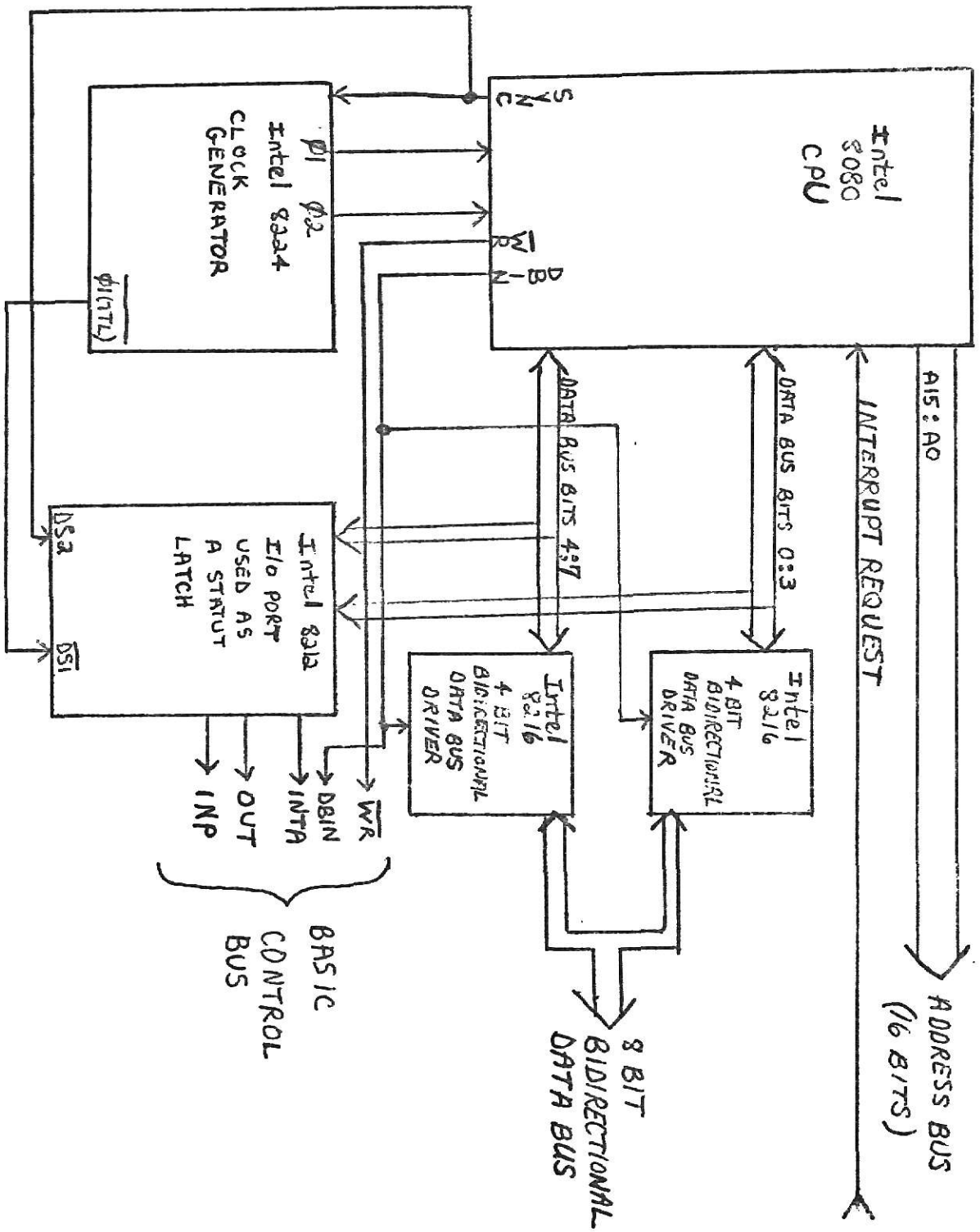
2. The TTL Data Book For Design Engineers, Texas Instrument 1976

3. MITS Altair 8800 Theory of Operation, MITS Corp. Albequerque, N.M., 1976

APPENDICES

APPENDIX A

BLOCK DIAGRAM OF AN ALTAIR 8800 COMPUTER SYSTEM

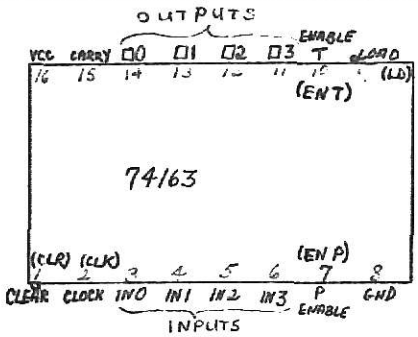


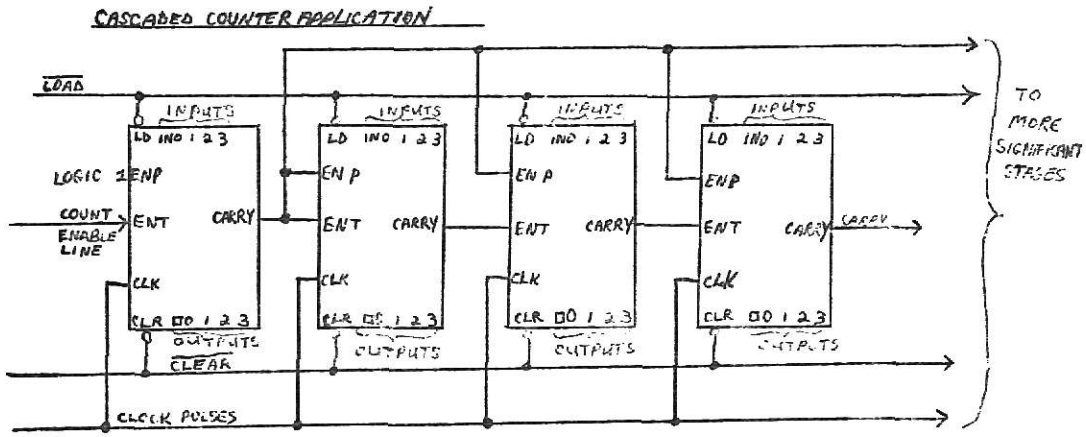
APPENDIX B

LEGEND OF SYMBOLS USED IN CIRCUIT DIAGRAMS

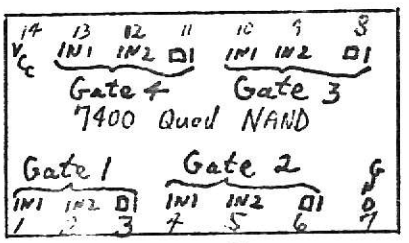
1,2,3,... etc.	IC Pin Numbers
V_{cc} , +5V	Supply Voltage - +5 Volts
GND , ⏚	Ground Connection - 0 Volts
□0,□1,... etc.	Output lines - higher numbers indicate greater significance
IN0,IN1,... etc.	Input lines - higher numbers indicate greater significance
o	This input or output causes some action when its value is logic 0 (active low)
1	A Logic 1 value
PA3, PB0, PB7, etc.	Specified bits of the named ports
D0,D1,...,D7	The 8 lines of a bidirectional data bus
DI0,DI1,...,DI7	The 8 data input lines to the CPU
D□0,D□1,...,D□7	The 8 data output lines from the CPU
X	A don't care value - could be logic 0 or 1
A0,A1,...,A7	The low order 8 address bus lines from the

74163 PIN DESIGNATION AND SAMPLE 16 BIT CASCADED COUNTER CIRCUIT

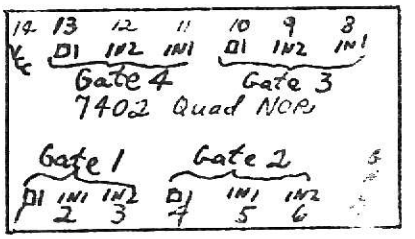




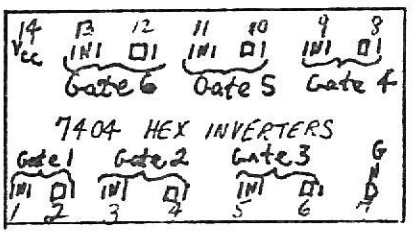
7400 SERIES CHIP PIN DESIGNATION AND LOGIC GATE DIAGRAMS



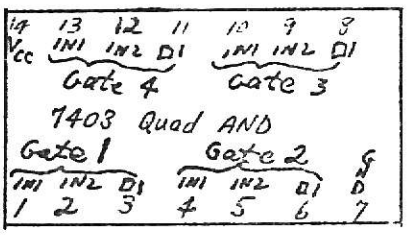
$O1 = \overline{IN1 \cdot IN2}$



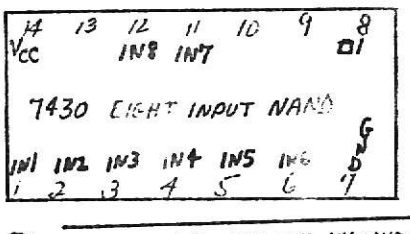
$O1 = \overline{IN1 \vee IN2} = \overline{IN1} \cdot \overline{IN2}$



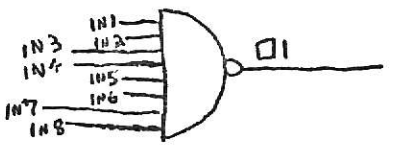
$O1 = \overline{IN1}$

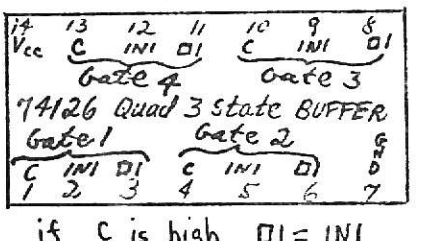


$O1 = IN1 \cdot IN2$



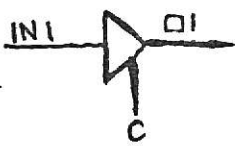
$O1 = \overline{IN1 \cdot IN2 \cdot IN3 \cdot IN4 \cdot IN5 \cdot IN6 \cdot IN7 \cdot IN8}$



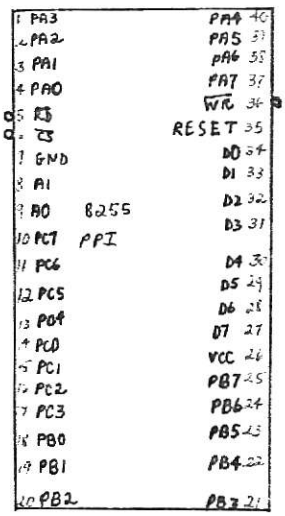


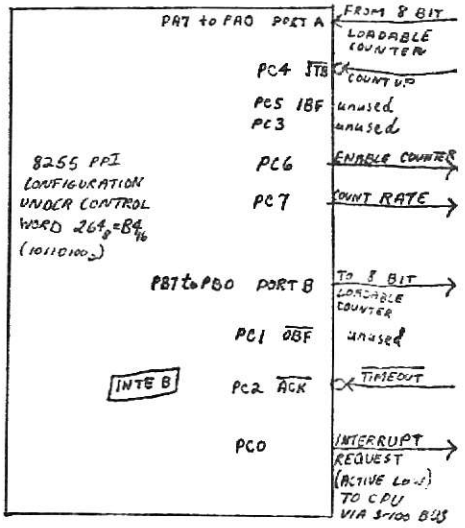
if C is high $O1 = IN1$

if C is low O1 is tri-state



INTEL 8255 PROGRAMMABLE PERIPHERAL INTERFACE - PIN DESIGNATION AND BASIC COMMANDS





RD (ACTIVE LOW) — INDICATES CPU WISHES TO READ DATA FROM DEVICE.

WR (ACTIVE LOW) — INDICATES CPU WISHES TO WRITE DATA TO DEVICE.

CS (ACTIVE LOW) — INDICATES THIS DEVICE HAS BEEN SELECTED FOR AN I/O OPERATION.

RESET (ACTIVE HIGH) — WHEN HIGH PUTS ALL 3 PORTS IN THE INPUT MODE.

Basic Operation

A_1	A_0	RD	WR	CS	OPERATION
0	0	0	1	0	PORT A → DATA BUS
0	1	0	1	0	PORT B → DATA BUS
1	0	0	1	0	PORT C → DATA BUS
1	1	0	1	0	ILLEGAL *
0	0	1	0	0	DATA BUS → PORT A
0	1	1	0	0	DATA BUS → PORT B
1	0	1	0	0	DATA BUS → PORT C
1	1	1	0	0	DATA BUS → CONTROL REG.
X	X	X	X	1	DATA BUS IS 3 STATE (DEVICE NOT ACTIVE)

SETTING BITS IN PORT C

USE $0\ X\ X\ X\ P_0P_1P_2\ Y_2$ where

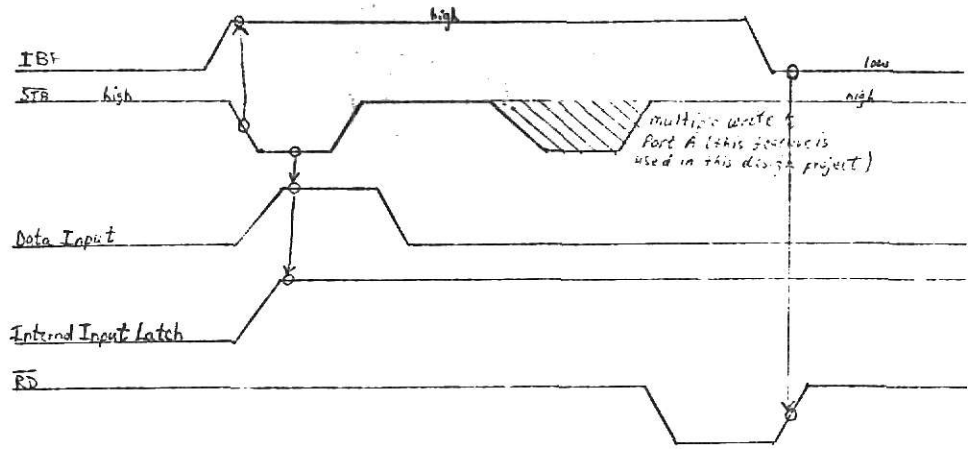
$P_0P_1P_2$ is the binary number of which bit to set

Y indicates if the bit is to be set (Y=1) to 1 or reset (Y=0) to 0

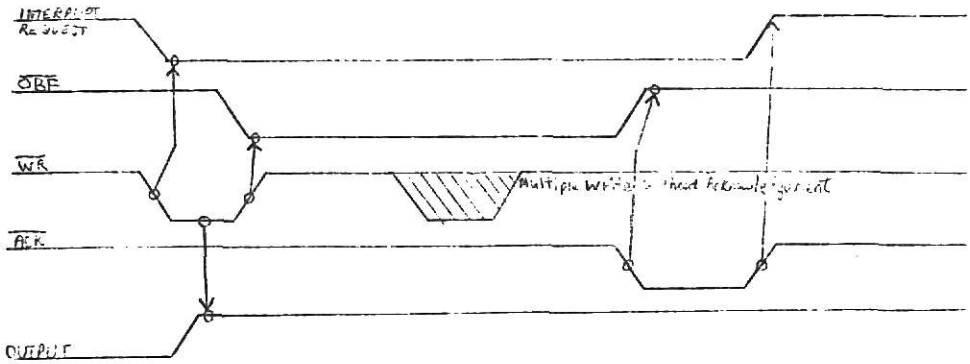
* THIS COMBINATION COULD BE USED TO RESET THE DEVICE IF NO OTHER MEANS ARE AVAILABLE FOR THAT PURPOSE.

8255 TIMING DIAGRAMS - PORT A STROBED-INPUT AND PORT B STROBED-OUTPUT

Port A Strobed Input (mode 1) [WRITE FROM CPU TO 8255]



Port B Strobed Output (mode 1) [READ TO CPU FROM 8255]



NOTES

a) The STB pulse must be at least 350 ns in width

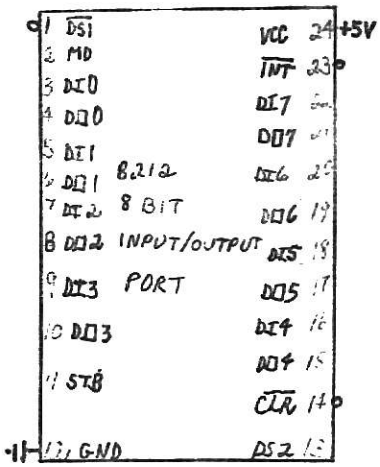
b) The data input must remain stable for 150 ns after STB returns high

c) The output data is not stable until a maximum of 500 ns after WR goes low.

d) OBF is reset high a maximum of 500 ns after ACK goes active.

e) The ACK pulse must remain low for at least 500 ns.

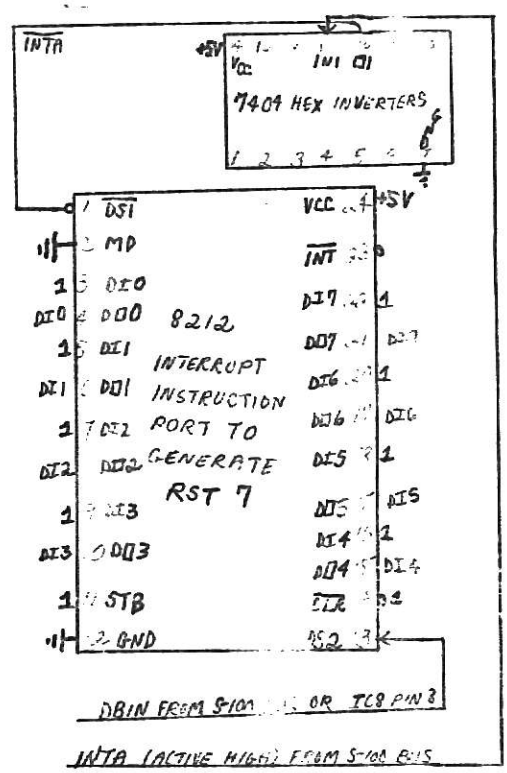
INTEL 8212 I/O PORT AND SAMPLE USE AS AN INTERRUPT INSTRUCTION PORT (RST 7)



OPERATION

STB	MD	$\overline{DSI} \cdot DS2$	DATA OUT EQUALS
0	0	0	3 STATE
1	0	0	3 STATE
0	1	0	Data Latch Value
1	1	0	Data Latch Value
0	0	1	Data Latch Value
1	0	1	Data In
0	1	1	Data In
1	1	1	Data In

$\overline{CLR}$ RESETS THE DATA LATCH



NOTE: ON SYSTEMS WHERE $\overline{INTA}$ IS AN ACTIVE LOW INTERRUPT ACKNOWLEDGE SIGNAL, PUT THE $\overline{INTA}$ SIGNAL DIRECTLY TO PIN 1 ($\overline{DSI}$).

OPERATION

WHEN $\overline{DSI}$ IS LOW FROM AN INTERRUPT ACKNOWLEDGE SIGNAL, THE DATA BUS WILL BE JAMMED WITH AN RST 7 INSTRUCTION ($377_8 = FF_{16}$) WHEN DBIN GOES HIGH.

AT ALL OTHER TIMES THE OUTPUTS ARE 3 STATED.

APPENDIX C

8800 SYSTEM BUS STRUCTURE

The 8800 system bus structure consists of 100 lines. These are arranged 50 on each side of the plug-in boards.

The following general rules apply to the 8800 system bus:

SYMBOLS: "P" prefix indicates a processor command/control signal

"S" prefix indicates a processor status signal

LOADING: All inputs to a card will be loaded with a maximum of 1 TTL low power load.

LEVELS: All bus signals except the power supply are TTL

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
1	+8V	+8 volts	Unregulated input to 5v regulators
2	+16V	+16 volts	Positive unregulated voltage
3	XRDY	External Ready	For special applications: Pulling this line low will cause the processor to enter a WAIT state and allows the status of the normal Ready line (PRDY) to be examined
4	VIO	Vectored Interrupt Line #0	
5	VI1	Vectored Interrupt Line #1	
6	VI2	Vectored Interrupt Line #2	
7	VI3	Vectored Interrupt Line #3	
8	VI4	Vectored Interrupt Line #4	

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
9	VI5	Vectored Interrupt Line #5	
10	VI6	Vectored Interrupt Line #6	
11	VI7	Vectored Interrupt Line #7	
12 to 17	TO BE DIFINED		
18	$\overline{\text{STA DSB}}$	$\overline{\text{STATUS DISABLE}}$	Allows the buffers for the 8 status lines to be tri-stated
19	$\overline{\text{C/C DSB}}$	$\overline{\text{COMMAND/CONTROL DISABLE}}$	Allows the buffers for the 6 output command/control lines to be tri-stated
20	UNPROT	UNPROTECT	Input to the memory protect flip-flop on a given memory board
21	SS	SINGLE STEP	Indicates that the machine is in the process of performing a single step
22	$\overline{\text{ADD DSB}}$	$\overline{\text{ADDRESS DISABLE}}$	Allows the buffers for the 16 address lines to be tri-stated
23	$\overline{\text{DO DSB}}$	$\overline{\text{DATA OUT DISABLE}}$	Allows the buffers for the 8 data output lines to be tri-stated
24	Q2	Phase 2 Clock	
25	Q1	Phase 1 Clock	
26	PHLDA	Hold Acknowledge	Processor command/control output signal which appears in response to the HOLD signal; indicates that the data and address bus will go to the high impedance state

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
27	PWAIT	WAIT	Processor command/control output signal which acknowledges that the processor is in a WAIT state
28	PINTE	INTERRUPT ENABLE	Processor command/control output signal indicating interrupts are enabled: indicates the content of the CPU internal interrupt flip-flop; F-F may be set or reset by EI and DI instruction and inhibits interrupts from being accepted by the CPU if it is reset
29	A5	Address Line #5	
30	A4	Address Line #4	
31	A3	Address Line #3	
32	A15	Address Line #15	
33	A12	Address Line #12	
34	A9	Address Line #9	
35	DO1	Data Out Line #1	
36	DO0	Data Out Line #0	
37	A10	Address Line #10	
38	DO4	Data Out Line #4	
39	DO5	Data Out Line #5	
40	DO6	Data Out Line #6	
41	DI2	Data In Line #2	
42	DI3	Data In Line #3	
43	DI7	Data In Line #7	

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
44	SM1	M1	Status output signal that indicates that the processor is in the fetch cycle for the first byte of an instruction
45	SOUT	OUT	Status output signal which indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{PWR}$ is active
46	SINP	INP	Status output signal which indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when PDBIN is active
47	SMEMR	MEMR	Status output signal which indicates that the data bus will be used for memory read data
48	SHLTA	HLTA	Status output signal which acknowledges a HALT instruction
49	$\overline{CLOCK}$	$\overline{CLOCK}$	Inverted output of the 2MHz oscillator that generates the 2 phase clock
50	GND	GROUND	
51	+8V	+8 volts	Unregulated input to 5v regulators
52	-16V	-16 volts	Negative unregulated voltage
53	$\overline{SSW\ DSB}$	$\overline{SENSE\ SWITCH\ DISABLE}$	Disables the data input buffers so the input from the sense switches may be strobed onto the bidirectional data bus right at the processor
54	$\overline{EXT\ CLR}$	$\overline{EXTERNAL\ CLEAR}$	Clear signal for I/O devices (front panel switch closure to ground)

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
55 to 67	TO BE DEFINED		
68	MWRT	MEMORY WRITE	Indicates that the current data on the Data Out Bus is to be written into the memory location currently on the address bus
69	$\overline{PS}$	$\overline{PROTECT\ STATUS}$	Indicates the status of the memory protect flip-flop on the memory board currently addressed
70	PROT	PROTECT	Input to the memory protect flip-flop on the memory board currently addressed
71	RUN	RUN	Indicates that the RUN/STOP flip-flop is Reset
72	PRDY	READY	Processor command/control input that controls the run state of the processor; if the line is pulled low the processor will enter a wait state until the line is released
73	$\overline{PINT}$	$\overline{INTERRUPT\ REQUEST}$	The processor recognizes an interrupt request on this line at the end of the current instruction or while halted. If the processor is in the HOLD state or the Interrupt Enable flip-flop is reset, it will not honor the request.
74	$\overline{PHOLD}$	$\overline{HOLD}$	Processor command/control input signal which requests the processor to enter the HOLD state; allows an external device to gain control of address and data buses as soon as the processor has completed its use of these buses for the current machine cycle

BUS DEFINITION

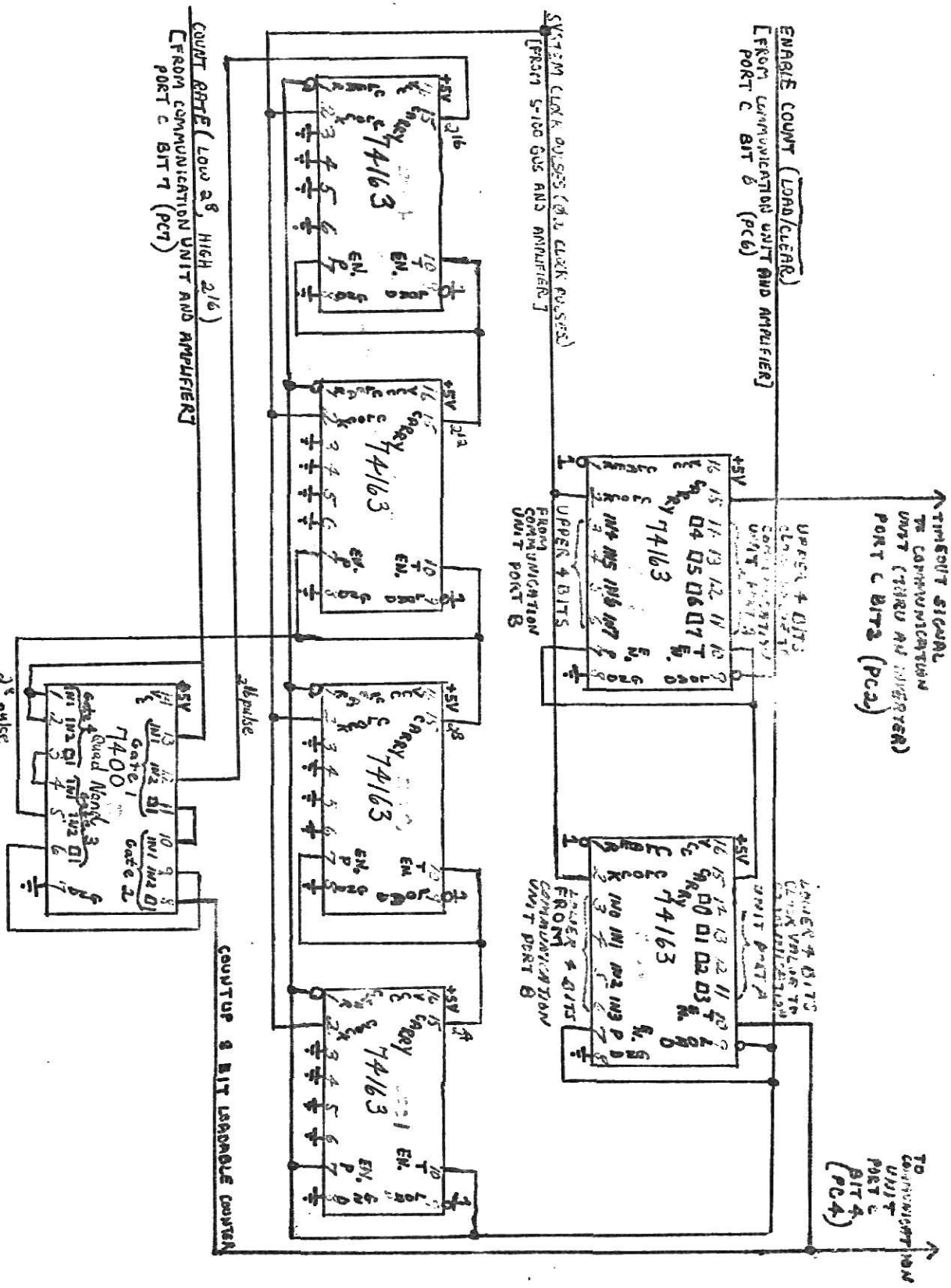
No.	SYMBOL	NAME	FUNCTION
75	$\overline{\text{PRESET}}$	$\overline{\text{RESET}}$	Processor command/control input; while activated the content of the program counter is cleared and the instruction register is set to 0
76	PSYNC	SYNC	Processor command/control output provides a signal to indicate the beginning of each machine cycle
77	$\overline{\text{PWR}}$	$\overline{\text{WRITE}}$	Processor command/control output used for memory write or I/O output control: data on the data bus is stable while the $\overline{\text{PWR}}$ is active
78	PDBIN	DATA BUS IN	Processor command/control output signal indicates to external circuits that the data bus is in the input mode
79	A0	Address Line #0	
80	A1	Address Line #1	
81	A2	Address Line #2	
82	A6	Address Line #6	
83	A7	Address Line #7	
84	A8	Address Line #8	
85	A13	Address Line #13	
86	A14	Address Line #14	
87	A11	Address Line #11	
88	DO2	Data Out Line #2	
89	DO3	Data Out Line #3	
90	DO7	Data Out Line #7	
91	DI4	Data In Line #4	

BUS DEFINITION

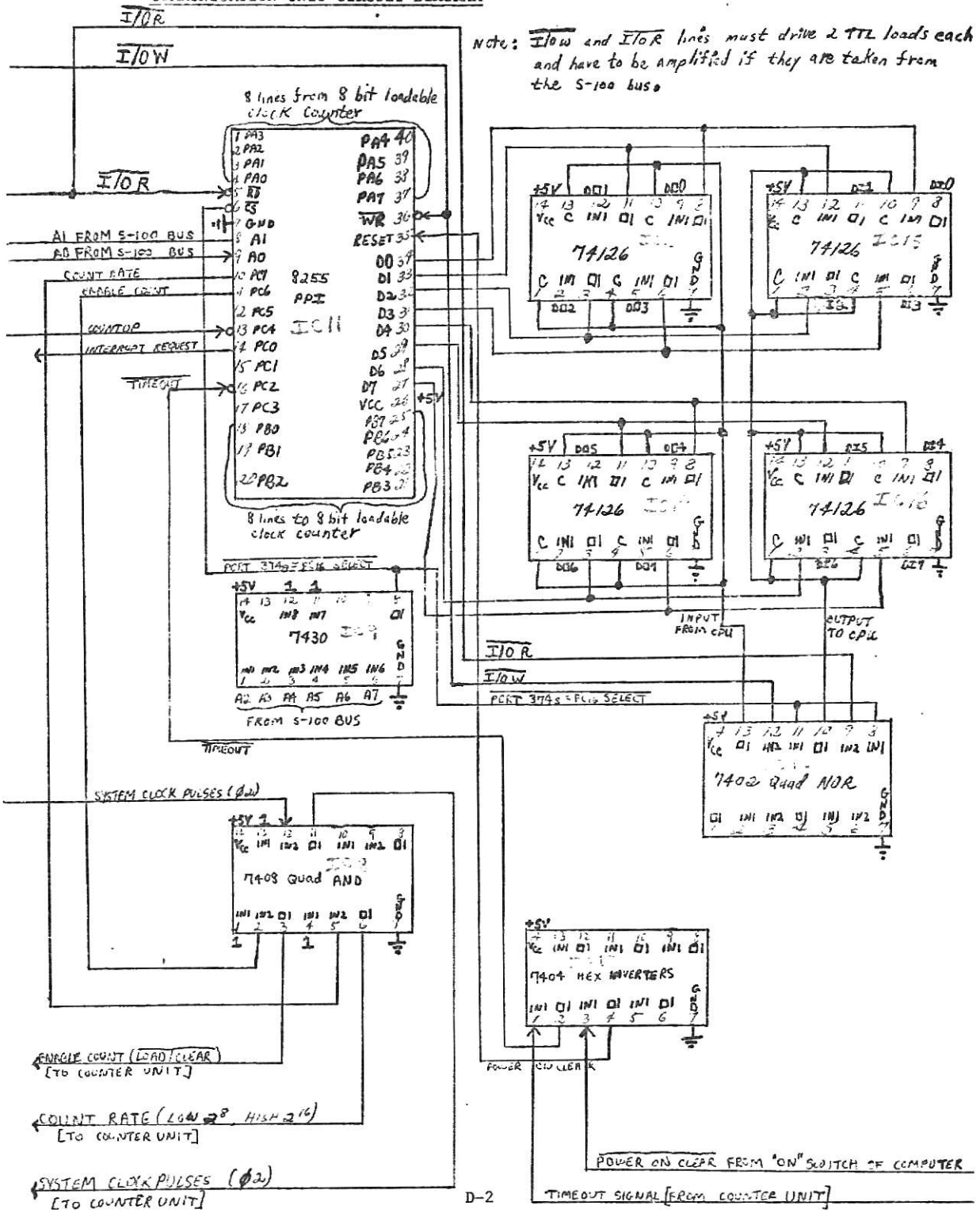
No.	SYMBOL	NAME	FUNCTION
92	DI5	Data In Line #5	
93	DI6	Data In Line #6	
94	DI1	Data In Line #1	
95	DI0	Data In Line #0	
96	SINTA	INTA	Status output signal to acknowledge signal for INTERRUPT request
97	SWO	WO	Status output signal indicates that the operation in the current machine cycle will be a WRITE memory or output function
98	SSTACK	STACK	Status output signal indicates that the address bus holds the pushdown stack address from the Stack Pointer
99	$\overline{\text{POC}}$	Power-On Clear	
100	GND	GROUND	

APPENDIX D

CLOCK COUNTER CIRCUIT DIAGRAM

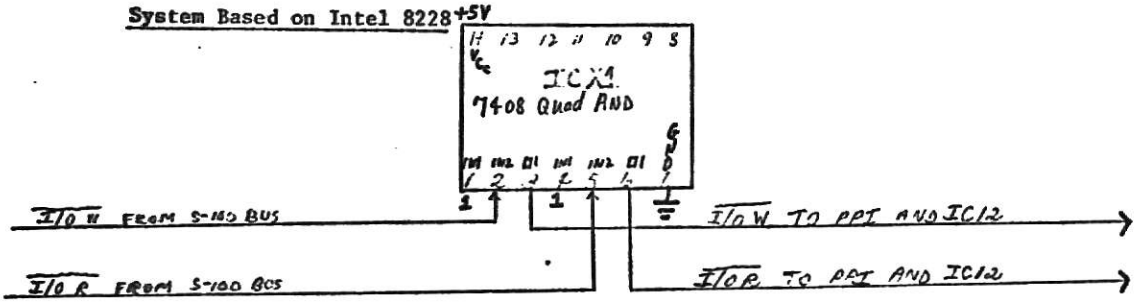


COMMUNICATION UNIT CIRCUIT DIAGRAM

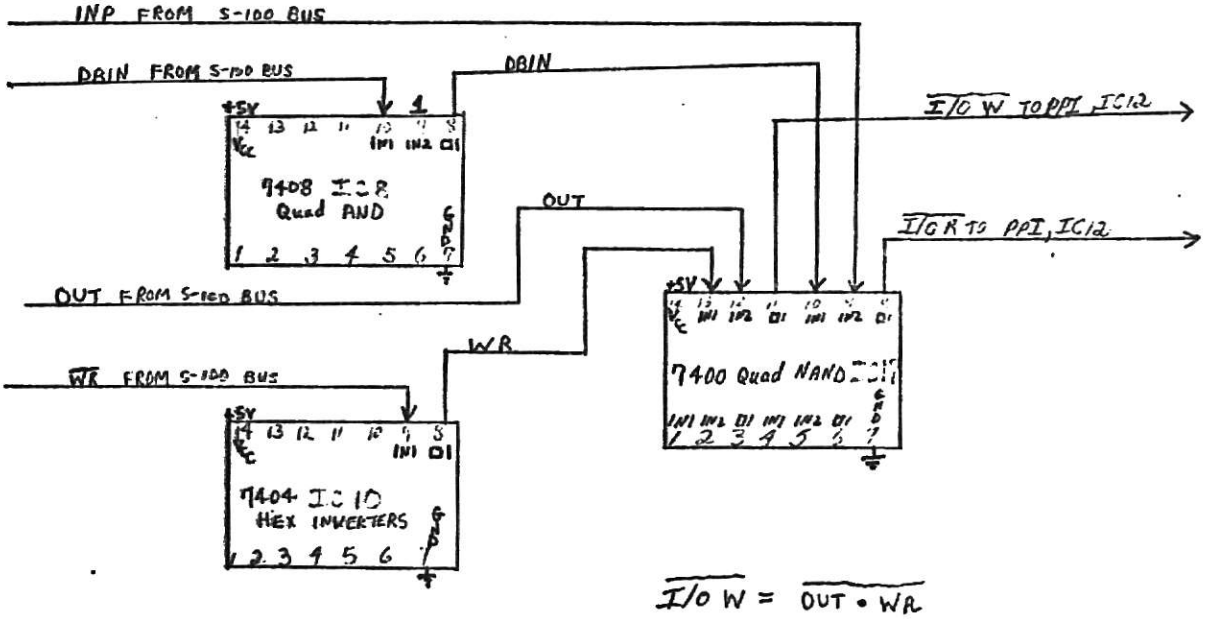


CIRCUITS SHOWING GENERATION OF $\overline{\text{I/O R}}$ AND $\overline{\text{I/O W}}$ SIGNALS

System Based on Intel 8228



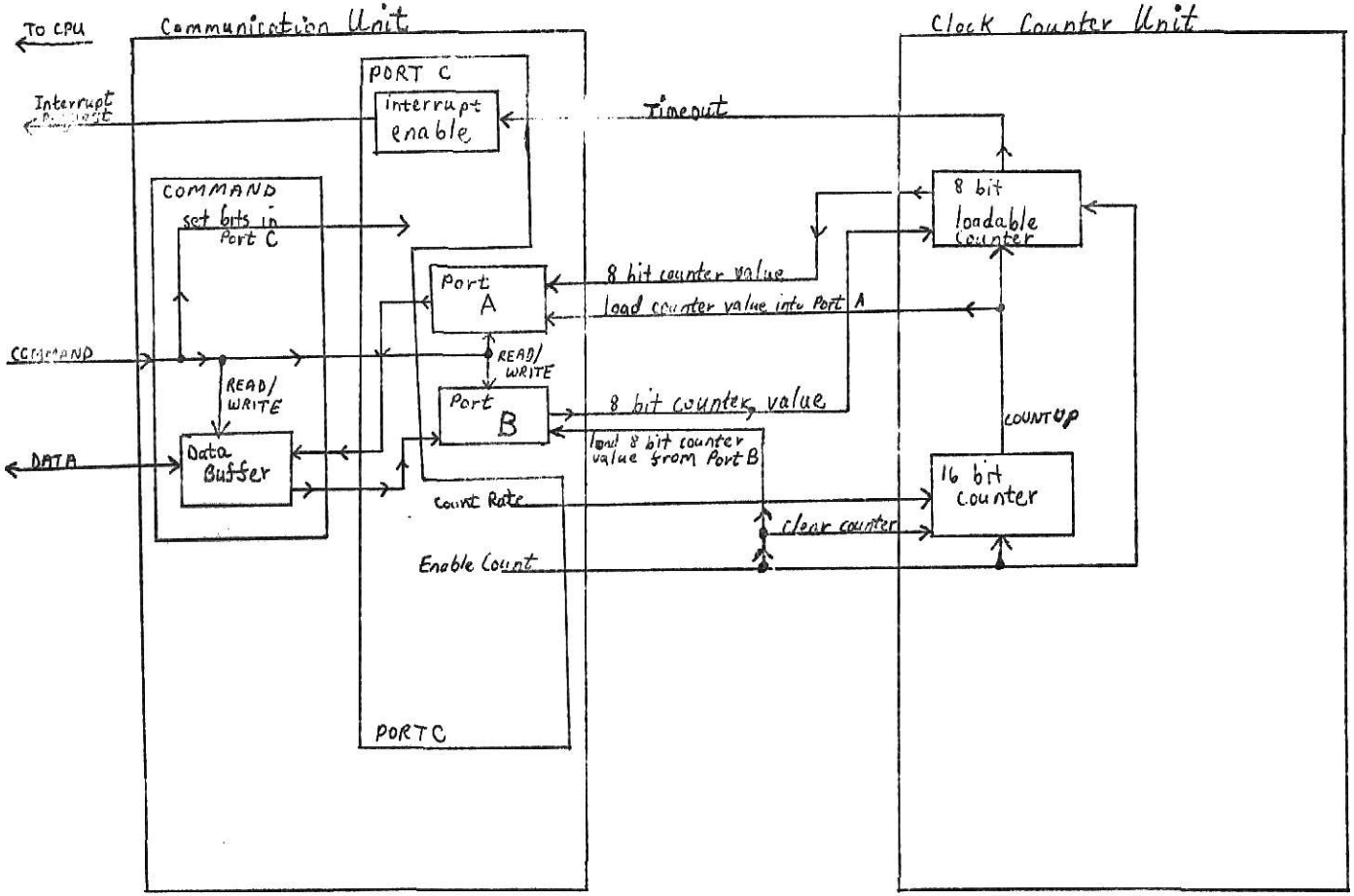
System Based On Intel 8212 Used As A Status Latch



$\overline{\text{I/O W}} = \overline{\text{OUT} \cdot \text{WR}}$

$\overline{\text{I/O R}} = \overline{\text{INP} \cdot \text{DBIN}}$

BLOCK DIAGRAM OF CLOCK TIMER UNIT



APPENDIX E

SILICON GATE MOS 8080A

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
MOV r1,r2	Move register to register	0	1	D	D	D	S	S	S	5
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock Cycles
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP B	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4

NOTES 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

APPENDIX F

CLOCK TIMER USERS MANUAL

I. Control Sequence Summary

Control Sequence	Function
MVI A,264 OUT 377	Set the 8255 PPI to the proper configuration
MVI A,17 OUT 377	Set the clock timer to count at the slow rate
MVI A,16 OUT 377	Set the clock timer to count at the slow rate
MVI A,5 OUT 377	Enable the communication unit to request an interrupt
MVI A,4 OUT 377	Disable the communication unit interrupt request facility
MVI A,15 OUT 377	Start the clock counter counting
MVI A,14 OUT 377	Stop the clock counter from counting, then load the value in Port B into the clock counter
INP 374	Load the value of Port A into the accumulator
INP 376	Load the value of the Port C control bits into the accumulator
LDA CLOCK_VALUE OUT 375	Load Port B with the value in the accumulator. This also turns off the interrupt request line if it was active without changing the clock value if the clock was running
LDA CLOCK_VALUE OUT 375 MVI A,14 OUT 377 MVI A,15 OUT 377	Load a new clock value in the clock counter and continue counting

II. Notes

1. The clock rate bit PC7 and the clock count in Port B remain constant unless explicitly changed by writing new values. Thus if the same clock rate or value will be used several times the values need be set only once.

2. Immediately after the interrupt handler gains control a value should be written to Port B to clear the interrupt request line. As long as the clock counts, writes to Port B will not affect it.

3. The clock counter counts up instead of down so the negative of the desired clock value should be loaded into Port B and the counter.

4. On a system with many devices being serviced by the same interrupt handler as the clock, the clock value should be read from Port A. If the value of the clock is 0 then the clock needs servicing.

III. Sample Programs

The following sample program illustrates usage of the clock timer to se some external device at constant time intervals.

```
START:  LXI SP,10          ; set the stack pointer address to 10(8)
        MVI A,264          ; code to configure the 8255 PPI
        OUT 377            ; configure the 8255 PPI to the proper f
        MVI A,17           ; (or 16) set proper clock rate
        OUT 377            ; set clock rate
        LDA CLKVAL         ; accumulator gets negative of desired
                           ; clock value
        OUT 375            ; set clock value
        MVI A,5            ; code to enable interrupt facility of P
        OUT 377            ; allow the PPI to generate interrupts
                           ; initialize external device(s)
        EI                 ; enable the 8080 CPU to accept interrup
        MVI A,15           ; code word to start the clock counting
        OUT 377            ; start the clock
LOOP:   JMP LOOP           ; wait for timeout

*70                        ; the next instruction is placed at loca
                           ; 070(8)
        JMP INTHND         ; transfer control to interrupt handler
INTHND: PUSH PSW           ; save flags and accumulator value
        LDA CLKVAL
        OUT 375            ; reset interrupt request line
        MVI A,14           ; command code to stop clock and load va
        OUT 377            ; load value into clock after stopping i
        MVI A,15           ; code to start the clock counting
        OUT 377            ; set the clock counting
                           ;
                           ; service devices(s)
                           ;
        POP PSW            ; restore flags and accumulator
        EI                 ; enable the 8080 CPU interrupts
        RET                ; exit interrupt handler
```

Note : The execution time of the device handler must be less than the time on the clock or else some clock timeouts may be missed.

The following sample program illustrates the program structure which coul be used to keep track of real time.

```
START:  LXI SP,10           ; set stack pointer to address location 10
        MVI A,264
        OUT 377             ; configure the 8255 PPI to the proper for
                            ;
                            ; initalize program clock values for secon
                            ; minutes and hours of elapsed real time.
                            ;
        MVI A,16            ; command for the fast count rate
        OUT 377             ; set the clock to count at the fast rate
        MVI A,374           ; set the clock to count out 4 groups of
                            ; pulses each of size 2**8 times the clock
         EI                 ; rate.  On a system with a clock rate of
                            ; 500 ns this times out 512 microseconds
        OUT 375             ; set the clock value into the clock count
        MVI A,15
        OUT 377             ; start the clock
           .                ;
           .                ; do any desired processing
           .                ;
*70
        JMP INTHND

INTHND: PUSH PSW            ; save flags and accumulator value
                            ; save any registers necessary
        MVI A,374
        OUT 375             ; reset the interrupt
        MVI A,14            ; stop the clock and load the clock value
        OUT 377
        MVI A,15
        OUT 377             ; start the clock counting
                            ;
                            ; update the clock values properly
                            ;
                            ; restore any saved register values
        POP PSW             ; restore flags and accumulator
        EI
        RET
```

A CLOCK TIMER HARDWARE UNIT FOR AN INTEL 8080 CPU BASED COMPUTER SYSTEM

by

MICHAEL FRANZBLAU

B.S. SUNY at Stony Brook, 1975

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1977

This report describes a hardware project design and implementation of a programmable clock timer unit for the Computer Science Department's Altair 8800 microcomputer. Included in the report are block diagrams describing the logical structure of the unit and electrical diagrams which can be used to construct the same or a similar device. There is a section which outlines a series of tests to pinpoint and correct errors in the circuit. The last section contains an explanation of the commands which control the clock. A user's guide is included as a summary of these commands along with some sample programs.