



An 8-bit microcomputer system processor, memory and I/O
by Darrell Boots Irvin

A thesis submitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE
in Electrical Engineering
Montana State University
© Copyright by Darrell Boots Irvin (1976)

Abstract:

This thesis presents an 8-bit microcomputer based on the INTEL 8008 microprocessor. Suitable applications for the system include data acquisition and digital control systems formerly realized with dedicated hardware or larger and more expensive minicomputer systems.

Processor, memory and I/O subsystems are first described from an architectural viewpoint with the aid of block diagram. The instruction set for the system is introduced in an informal manner within the main text while Appendix B formalizes the instruction definitions. Finally, each circuit card used in the prototype system is documented with schematics and operational descriptions of the logic used on each circuit card.

STATEMENT OF PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Montana State University, I agree that the Library shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by my major professor, or, in his absence, by the Director of Libraries. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature Danell Boots-Lurin

Date July 21, 1976

AN 8-BIT MICROCOMPUTER SYSTEM

PROCESSOR, MEMORY AND I/O

by

DARRELL BOOTS IRVIN

A thesis submitted in partial fulfillment
of the requirements for the degree

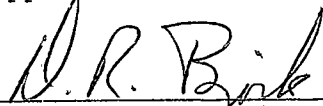
of

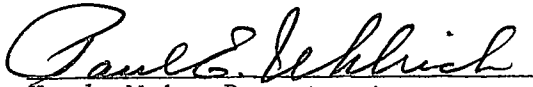
MASTER OF SCIENCE

in

Electrical Engineering

Approved:


Chairperson, Graduate Committee


Head, Major Department


Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

July, 1976

ACKNOWLEDGEMENT

The development and construction of a prototype system of the size and complexity represented by this thesis requires the cooperation and creative talent of many individuals. The author wishes to thank the U.S. Forest Service, Forest and Range Experiment Station, Bozeman, Montana, and in particular Mr. William Hartsog for supplying the funding which made this thesis possible.

Finally, the author is especially grateful to Mr. Donald Bjork for his support of the project and a special thanks is due Mr. John Somppi for his invaluable assistance in constructing and debugging the prototype.

D.B.I.

TABLE OF CONTENTS

<u>Chapter</u>		<u>Page</u>
1	INTRODUCTION.....	1
	DESIGN GOALS.....	2
	LOGIC CONVENTIONS.....	3
	DATA FORMATS.....	5
2	MACHINE ARCHITECTURE.....	7
	PROCESSOR.....	9
	PROCESSOR INSTRUCTION SET.....	18
	MEMORY.....	26
3	I/O STRUCTURE.....	30
	POLLED I/O.....	33
	INTERRUPT I/O AND PRIORITY CHAIN.....	35
	REAL TIME CLOCK AND PRIORITY CHAIN.....	41
4	PROCESSOR.....	43
	INSTRUCTION EXECUTION SEQUENCE.....	44
	PROCESSOR INTERRUPTS.....	49
	WAIT STATE.....	51
	PROCESSOR HARDWARE.....	52
5	MEMORY SYSTEM.....	64
	READ ONLY MEMORY.....	65
	OPERATION.....	66

<u>Chapter</u>	<u>Page</u>
PROGRAMMING OF PROM CHIPS.....	69
OPTIONS.....	70
RANDOM ACCESS MEMORY.....	71
RAM OPERATION.....	72
RAM READ OPERATION.....	75
RAM WRITE OPERATION.....	76
6 SYSTEM I/O.....	77
INTERRUPT AND PRIORITY LOGIC.....	78
8 BIT PARALLEL I/O INTERFACE.....	82
CONTROL AND STATUS WORDS.....	83
DATA TRANSFER SEQUENCES.....	83
TRIGGERING INTERRUPT DATA TRANSFERS.....	86
PIO DATA HARDWARE.....	87
REAL TIME CLOCK.....	92
REAL TIME CLOCK CONTROL WORD.....	93
REAL TIME CLOCK TIMING LOGIC.....	96
SERIAL DATA INTERFACE.....	103
UART MODE SELECTION.....	105
BAUD RATE SELECTION.....	105
SERIAL DATA LINES.....	107
SIO CONTROL LOGIC.....	107

<u>Chapter</u>	<u>Page</u>
SIO CONTROL AND STATUS WORDS.....	108
OUTPUT DATA TRANSFER SEQUENCE.....	108
INPUT DATA TRANSFER SEQUENCE.....	111
7 CONCLUSIONS AND RECOMMENDATIONS.....	113
APPENDIX	
A DEFINITION OF BUS SIGNALS, BUS PIN NUMBERS AND BUS SCHEMATIC.....	130
B FORMAL DESCRIPTION OF INSTRUCTION SET.....	143
C INTEGRATED CIRCUIT POSITION KEYS.....	156

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	I/O CHANNEL ASSIGNMENTS	34
2	STATE CODES	44
3	CYCLE CODES	45
4	PROCESSOR STATE SEQUENCES	46
5	PROM ADDRESS RANGE SELECTION JUMPERS	70
6	RELATIVE PROM SOCKET ADDRESS	71
7	PIO CONTROL AND STATUS WORD FORMATS	84
8	PIO CHANNEL SELECT JUMPERS	90
9	REAL TIME CLOCK CONTROL WORD	94
10	REAL TIME CLOCK CHANNEL SELECT JUMPERS	101
11	UART MODE SELECT CODES AND SERIAL DATA FORMAT	106
12	SIO CONTROL AND STATUS WORD FORMATS	109
13	BUS SIGNALS AND PIN NUMBERS	140

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	HEX AND OCTAL BIT FIELDS.....	6
2	SYSTEM BLOCK DIAGRAM.....	8
3	PROCESSOR BLOCK DIAGRAM.....	10
4	INSTRUCTION FORMATS.....	19
5	ROTATE ACCUMULATOR INSTRUCTIONS.....	21
6	JUMP, CALL AND RETURN.....	23
7	PROM BLOCK DIAGRAM.....	27
8	RAM BLOCK DIAGRAM.....	28
9	GENERALIZED DATA I/O INTERFACE BLOCK DIAGRAM.....	31
10	PRIORITY CHAIN.....	36
11	PROCESSOR SCHEMATIC.....	53
12	TIMING DIAGRAM.....	54
13	PROM SCHEMATIC.....	67
14	RAM SCHEMATIC.....	73
15	SINGLE LEVEL OF INTERRUPT AND PRIORITY CONTROL LOGIC....	79
16	PIO DATA INPUT SEQUENCE.....	85
17	PIO DATA OUTPUT SEQUENCE.....	86
18	PIO INTERFACE SCHEMATIC.....	88
19	REAL TIME CLOCK SCHEMATIC.....	97

<u>Figure</u>	<u>Page</u>
20 SERIAL DATA INTERFACE.....	104
21 SYSTEM BUS SCHEMATIC.....	139
22 INTEGRATED CIRCUIT POSITION KEYS.....	157

ABSTRACT

This thesis presents an 8-bit microcomputer based on the INTEL 8008 microprocessor. Suitable applications for the system include data acquisition and digital control systems formerly realized with dedicated hardware or larger and more expensive minicomputer systems.

Processor, memory and I/O subsystems are first described from an architectural viewpoint with the aid of block diagram. The instruction set for the system is introduced in an informal manner within the main text while Appendix B formalizes the instruction definitions. Finally, each circuit card used in the prototype system is documented with schematics and operational descriptions of the logic used on each circuit card.

CHAPTER 1

INTRODUCTION

Since the perfection of Large Scale Integration (LSI) technology to the point where economics allow marketing of a single chip microprocessor (uP) with a versatile instruction set and a sophisticated architecture, the digital designer has new options in design. It has become economically possible to incorporate programmable hardware into smaller systems, resulting in a host of new applications in the areas of microcomputers and new realizations for digital control systems formerly constructed as dedicated hard-wired logic.

Presented is a modular 8-bit microcomputer system based on a microprocessor, the INTEL 8008. By selecting the desired amount and type of memory, the necessary Input/Output (I/O) channels and developing suitable software, the system may be configured to the complexity of the task at hand. Typical uses for this system could include data acquisition and preprocessing or digital control.

Features of the system include low power consumption to allow battery operation, and interrupt capability for real time applications. Modular design with separate circuit cards for the processor, memory and I/O allow the system to be expanded easily to the maximum capability of 16k of memory and eight I/O channels.

DESIGN GOALS

The goals and design constraints set for the system in the original specification stages were very general in nature. Rather than define the system with a hard and fast set of specifications, only the desired features and a few of the more general hardware specifications were detailed.

First, the system was to exhibit low power consumption. This was accomplished by designing the system with low power logic families. Low Power Transistor to Transistor Logic (LPTTL) and Complimentary Metal Oxide Semiconductor (CMOS) logic was used wherever possible. In keeping with the desire to conserve power the voltage regulators should run in the switched mode.

Second, the system was to accommodate both Random Access Memory (RAM) and programmable Read Only Memory (PROM). PROM allows non-volatile storage of code necessary to start the system and eliminates the need to enter program loaders by hand. The PROMs used in this system may be electrically reprogrammed and are more correctly referred to as Electrically Alterable Read Only Memories (EAROM). The RAM is static, requiring no refresh clocks but code stored in RAM will be lost if power is removed from the memory.

Third, the system was to have a variety of data I/O options. At this time two are available including an 8-bit Parallel I/O

channel (PIO) and a Serial I/O channel (SIO). Data may be input and output eight bits at a time in parallel through a PIO channel, while the SIO allows a similar transfer of data in serial form at a variety of baud rates and word formats. The SIO interface is compatible with most commonly available serial devices using asynchronous start/stop bit data formats.

Fourth, the system was to have real time capabilities in the form of interrupts. This capability allows servicing of infrequent or short, high priority data transfers while allowing the processor to spend most of the time on longer but less critical routines.

Included in the I/O section, but not handling data is the Real Time Clock (RTC) which allows the processor to be interrupted after a programmable amount of time has passed since the last clock interrupt. The RTC allows software to keep "real time" by interrupting the processor at known intervals. This feature makes programming of systems that must react in real time a much simpler task.

LOGIC CONVENTIONS

Positive logic is used throughout this discussion. Thus a logical 1 (also referred to as a high signal) is defined to be the more positive output voltage (generally 3.5 to 5.0 volts) and a logical 0 (or low signal) to be the more negative voltage (generally

0.0 to 1.5 volts). The actual 0 and 1 voltages developed by a given output will vary as a function of the type of output (LPTTL or CMOS) supplying the drive and the number and type of inputs that comprise the load.

Positive logic convention also applies to the names (mnemonics) used in the discussion of the logic timing and control signals. Each signal may be either true or false. Signals will be called true when they are performing their function and false when they are not. True and false are thus logical designations as opposed to the electrical designations of 0 and 1. A signal may be logically true and electrically a 0 or 1 depending on its sense. Signals that are true when a 1 will be referred to as upright (positive true) and will have names without bars over the top (example: XYZ) while signals that are true when a 0 will be referred to as inverted (ground true) and will have names with a bar over the top (example: $\overline{XYZ}$). Conversion between the two senses is accomplished by the logic inverter. Thus $\overline{XYZ}$ becomes XYZ or vica versa after being inverted.

In the naming of data carrying signals similar conventions apply. Data is transferred eight bits at a time. Thus an eight bit binary integer having a value of 0 to 255 is the basic word (byte) used in the system. When a byte is transferred along a bus

each bit is given its own signal path. Each signal (or bit) is given a numbered name in the order of increasing significance. Thus DATA0 is the Least Significant Bit (LSB) and DATA7 the Most Significant Bit (MSB) of the eight bit byte (or bus) called DATA.

Since each bit in a binary word may assume either a 1 or 0 state depending on the value of the word, naming the upright and inverted senses requires a slightly different convention than that used with the timing signals. When a bit signal mnemonic appears without a bar on top (e.g. DATA) then the sense is upright and the electrical 1 corresponds to a true bit.

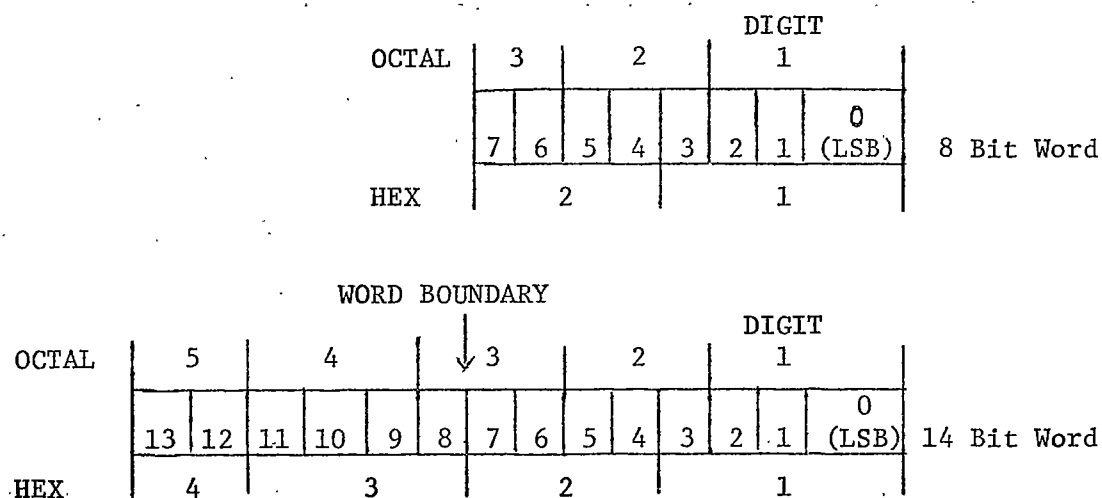
DATA FORMATS

An 8-bit byte can be represented in several ways. It may be written as a string of ones and zeros in binary form. Alternately, a byte may be expressed as a three digit octal or two digit hexadecimal number with each digit representing a three or four bit field respectively. Octal numbers will be written as 123B (or 123₈) while hex numbers will be written as ABC' (or ABC₁₆). A number written in decimal notation will have its digits written down as in 356. Thus each form of notation is unique to eliminate confusion.

When expressing hex numbers, the standard notation of using letters A through F for binary values of 9 through 15, respectively, is observed. The octal notation uses only the digits of 0 through

7. Figure 1 defines the bit fields for the hex and octal representations of both eight and fourteen bit words.

While all of the data transfers in the system take place eight bits at a time in a bit parallel-byte serial form, when the processor is addressing memory it uses a 14 bit address word (H and L bus). This larger word is formed by concatenating two bytes output at different times. This system ignores the top two bits of the most significant word resulting in a 14 bit address.



EXAMPLE:

10110101110111 = 26567B = 2D77'

Or Split Into 6 and 8 Bit Fields:

(101101), (01110111) = 55B, 167B = 2D', 77'

FIGURE 1: HEX AND OCTAL BIT FIELDS

CHAPTER 2

MACHINE ARCHITECTURE

The machine block diagram is presented in Figure 2. Architecturally, the system is composed of a series of modules (circuit cards) interconnected by a bus structure divided into two sections; Memory and I/O. Appendix A defines the bus signal set. Under control of a set of instructions (program) stored in memory, the processor performs a series of data transfers between the I/O interfaces and memory. As the data passes through the processor, a combination of things may happen: data may be modified arithmetically or logically, stored temporarily in a processor register, or used to make a decision about future program flow (conditional branching).

Memory is used for both program and scratch or buffer storage as determined by the programming. The configuration of the memory is expandable to a maximum of 16k words with many different combinations of RAM and PROM possible.

Eight bit data words are passed in and out of the system through the I/O interfaces, each of which provides an input port and an output port. The processor may control up to eight I/O interface channels for a total of 16 ports. Each port present in the system may generate interrupts if desired, forcing the program to execute in real time. Interrupt priority is controlled by position relative to the processor in the priority chain.

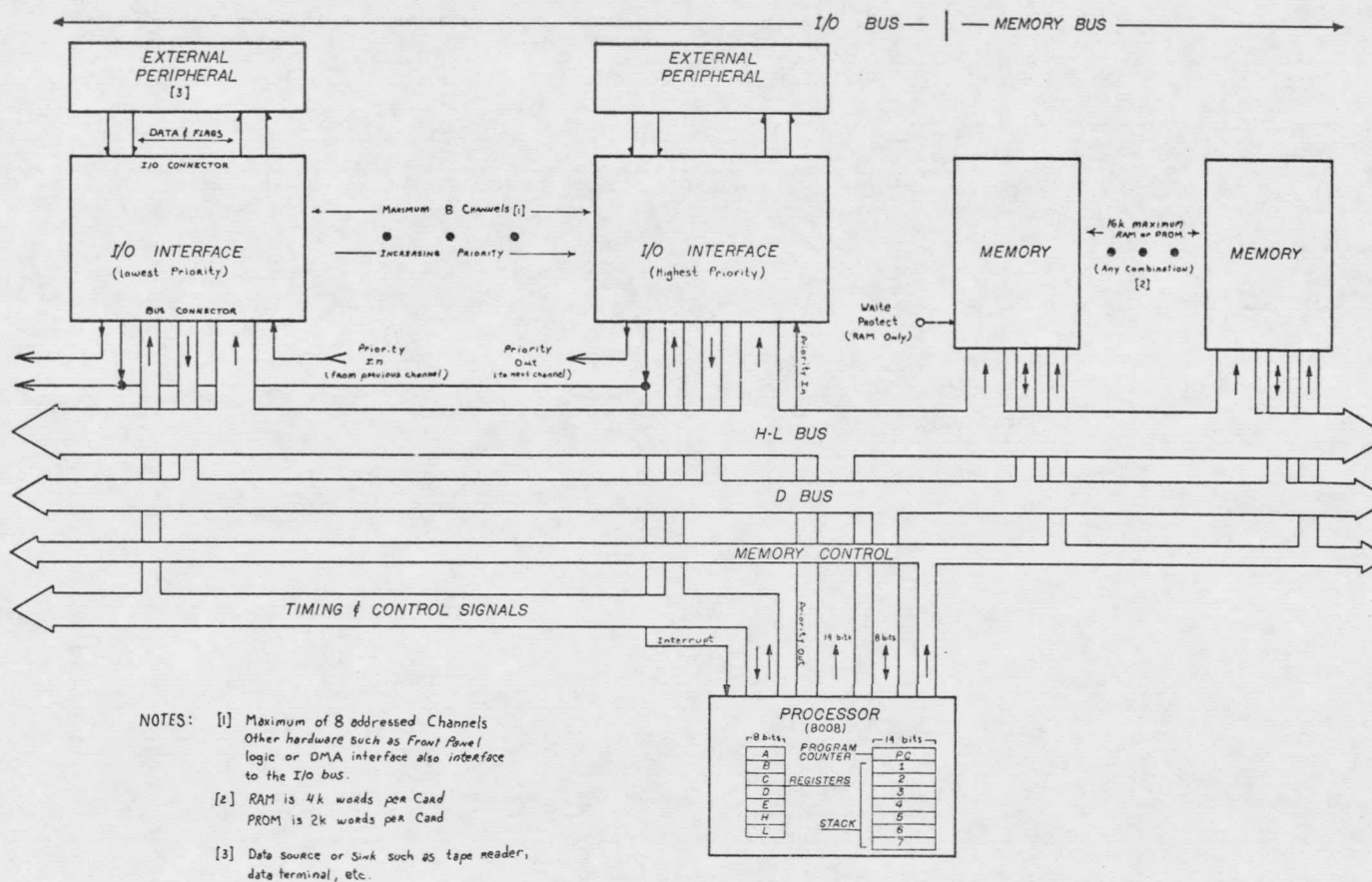


FIGURE 2: SYSTEM BLOCK DIAGRAM

PROCESSOR

The system processor block diagram is presented in Figure 3. An Intel 8008 microprocessor (enclosed within dotted lines) serves to provide the intelligent functions of the processor. The remaining portions of the processor serve as an interface between the 8008 and the bus. Functions performed by this logic include generation of the clock signals, decoding of the cycle and state signals and control of the three data paths within the system (D, H and L buses). Memory control and system timing signals are also generated by the processor.

Internally the 8008 contains seven general purpose registers each of which is eight bits wide. These are named A, B, C, D, E, H and L. Any of these registers may be used as general registers as determined by the program. Additionally register A functions as the accumulator and is always one argument of an arithmetic operation involving two words. Registers H and L also have an additional function. When addressing memory to read or write data (not instructions which are addressed by the Program Counter) they contain the memory address. Register L provides the eight LSB while H supplies the six MSB. The top two bits of H are ignored when using H and L as address registers.

The 8008 also contains a 14 bit wide address stack 8 levels

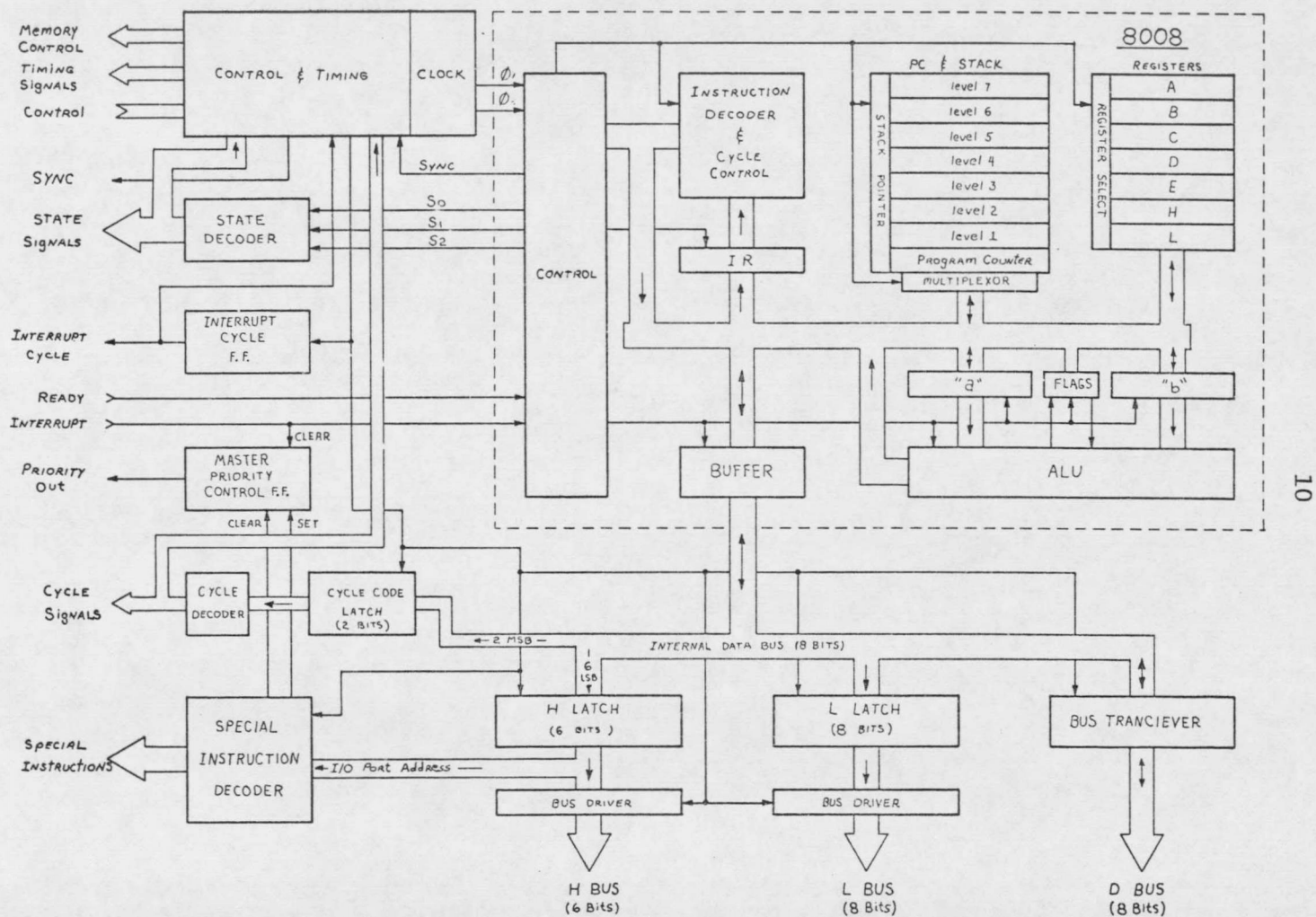


FIGURE 3: PROCESSOR BLOCK DIAGRAM

deep. The top of the stack functions as the program counter (PC) and is used to address the executing program. The remaining seven levels serve as a first in/last out stack used to save return addresses when calling subroutines. Executing a subroutine call instruction (CALL) causes the existing PC to move into Level 1 in the stack. At the same time the contents of Level 1 moved into Level 2 and so forth. The contents of Level 7 are lost. The PC is then set to the address associated with the subroutine and execution begins. Whenever a return from subroutine instruction (RETURN) is executed all levels of the stack move up one level thus restoring the PC to what it was before the last CALL was encountered. Execution then resumes on the main program. Note that software is responsible for saving all machine status (REGISTERS and FLAGS) except for the PC. This is extremely important when subroutine CALLS generated by interrupts may arrive at unknown times. When the capacity of the stack is exceeded (7 levels + current PC) then return addresses will be lost by being pushed out of Level 7 if further CALLS are executed. This is to be avoided at all costs unless the code is written such that return functions are not implemented through the RETURN instruction itself. In general this requires that each routine return to the same point in the main program and that such subroutines do not call one another.

The uP also provides an 8-bit wide parallel Arithmetic Logic Unit (ALU) capable of ten operations; ADD, ADD with Carry, SUBTRACT, SUBTRACT with Borrow, AND, EXOR, OR, COMPARE (& set Flags), INCREMENT and DECREMENT. Temporary Registers "a" and "b" are used as storage for the Accumulator and Operand for ALU operations.

The 8008 also contains four flags which are set by the result of ALU operations. The Carry Flag (CF) is set if the operation resulted in a carry or borrow. The Zero Flag (ZF) is set if the result of the operation is zero. The Sign Flag (SF) will be set if the MSB of the result is a one (negative number in two's complement) while the Parity Flag (PF) is set whenever the parity of the result is even.

The state of any one of the four flags may be tested by the program by executing a conditional CALL, JUMP or RETURN. Either a true or false condition of the flag bit may be used to cause the conditional to succeed.

The behavior of the 8008 is controlled by the instruction input to the Instruction Register (IR). This instruction is decoded and used to control the action of the 8008.

The portions of the block diagram outside the 8008 serve to interface the microprocessor to the rest of the system. This logic also provides special hardware functions not available from

the 8008. These include interrupt control, state and subcycle decoding and demultiplexing of the internal data bus to the three system buses (D, H, and L).

Whenever power is present on the processor card the two phase clock ($\phi 1$, $\phi 2$) is generated and sent to the 8008 which returns the SYNC signal which serves to differentiate between the two clock cycles that make up the basic processor state time. Thus with a clock period of 2-3 us, a state may last 4-6 us.

There are eight possible states for the processor. These are encoded by the 8008 and sent out through the three state signals (S_0 , S_1 , S_2). Six of the states normally occur only during the instruction execution sequence. These are T_1 , T_{1I} , T_2 , T_3 , T_4 , and T_5 . The remaining two states are named STOP and WAIT. These do just what their names imply. Detailed descriptions of the states may be found in Chapter 4. Generally T_1 (or T_{1I}) and T_2 are used to output memory or I/O address. T_3 is used to input or output data from I/O or memory and T_4 and T_5 are used internally by the 8008 to complete execution of the instruction.

The STOP state is caused by executing a HALT instruction and causes the processor to stop execution until restarted by an interrupt. The WAIT state is an optional state that may be placed between T_2 and T_3 by proper control of the READY input. WAIT serves

to suspend processor activity and may be used to interface the processor to slow memory devices by causing a pause between outputting the address and reading or writing the data.

The INTERRUPT input to the processor card allows other hardware on the bus to interrupt the presently executing code and insert (jam) an interrupt instruction. The processor will execute this instruction but does not increment the PC. If this instruction is a CALL, then execution will begin on the subroutine. Otherwise execution resumes normally where interrupted.

In the system interrupts are generally initiated by the I/O interface cards in response to data transfers, or else they may also be used to time the system as in the case of the Real Time Clock.

When executing an instruction the processor runs through a series of cycles each of which is composed of three to five of the T states in order. Depending upon the particular instruction there may be from one to three cycles in the complete instruction execution sequence. The first cycle of any instruction is the Instruction Fetch Cycle (INST) which is used to fetch the instruction from memory. This may be followed by cycles for reading from memory (READ), writing into memory (WRITE) or performing I/O operations (I/O).

Whenever the presently executing instruction was jammed under

interrupt two changes take place in the behavior of the processor. T1 is replaced by T1I in each cycle of the execution sequence and INTERRUPT CYCLE becomes true. Whenever an interrupt is received the PRIORITY OUT signal goes false (interrupting devices require true PRIORITY OUT to interrupt) to indicate to the rest of the system that the processor is busy with an interrupt. This signal is restored to the true state by a Special Instruction to be described later.

A cycle proceeds as follows. First, during T1 (or T1I) the 8008 outputs 8 bits which are stored in the L latch. This may be the 8 LSB of a memory address (PC or Reg. L) or if the instruction was an OUTPUT then the L latch contains the data. During the second state (T2), the microprocessor loads six bits into the H latch and two bits into the Cycle Code latch. The H latch contains either the MSB of a memory address or else the I/O port address, while the two bits of Cycle Code identify the cycle as INST, READ, WRITE, or I/O. Table 3 in Chapter 4 gives the Cycle Codes. Assuming that a WAIT state is not triggered by READY the processor will enter state T3. During this time the 8008 receives or transmits to the D bus. Depending upon instruction and cycle, this bus carries memory read or write data, interrupt instructions or input data from an I/O interface. If T4 and T5 occur they are used internally by the 8008.

During this time the rest of the processor is idle, and no data is transferred outside the 8008.

As shown on the block diagram the processor card contains an internal data bus which connects the 8008 to four other blocks. The Bus Transceiver allows the 8008 to read the D Bus or to drive it with data. The H, L and Cycle Code latches catch and save data output by the 8008 thereby time demultiplexing the outputs of the 8008. The transceiver and latches are controlled by the decoded T states and cycle signals as are the memory control and timing signals.

The remaining portion of the processor card is the Special Instruction Decoder. Whenever one of the eight OUTPUT instructions designated as Special Instructions is executed the decoder generates a pulse on one of eight outputs. Two of these instructions are used internal to the processor card to provide software control of the Master Priority Control flip flop. Some of the remaining Special Instructions are available in the bus. One of these is dedicated to control of the interrupts. The remaining Special Instructions may be used as desired to perform special hardware control functions peculiar to the individual system. None of the three Special Instructions presently used by the system do anything with the data which is output to the D Bus.

The Control and Timing portion of the processor also has connections to the bus. It generates two strobe signals used to synchronize data transfers, INTERRUPT and READY signals as described later. One of these strobes occurs during every processor state while the other happens only during T3. A buffered version of the 8008 SYNC signal is also present in the bus.

The control logic accepts two inputs from the bus. The DISCONNECT MEMORY ADDRESS input (DMA) is used to disconnect the H and L latches from their respective buses by turning off the bus drivers. Other hardware may now drive the H and L buses. The H-L INHIBIT input (HLINH) can be used to prevent the 8008 from loading the H and L latches thus effectively freezing their contents.

One more signal is generated by the processor logic. Whenever power is first applied to the card a CLEAR signal is generated by the processor. This serves as a master reset signal to the other cards present on the bus.

In general one may conclude that the operation of the processor cards external logic is controlled by the 8008. All data manipulations occur entirely within the microprocessor. The system is fairly slow because of the demultiplexing that must take place because of the 8008's one data port. A complete instruction execution sequence may take from three to eleven processor states or 12 to 44

us at 4 us per state. A detailed description of the processor state sequences for each instruction may be found in Chapter 4.

PROCESSOR INSTRUCTION SET

Instructions in this systems may consist of one, two or three successive bytes in the program as shown in Figure 4. The first byte is always the OP code which is used by the instruction decoding logic to control the facilities of the processor and execute the instruction. Some instructions require only an OP code and are classed as Single Byte instructions.

Those instructions composed of two bytes belong to the Immediate group of instructions. In these the OP code is followed by a second byte which will be used as data in the operation specified by the OP code. Immediate instructions allow the creation of needed constants whenever they are to be used at the cost of a memory location.

Instructions that affect the contents of the PC are encoded in three bytes. The two bytes following the OP code serve to define an address. The low order 8 bits are contained in the second byte and the six lower order bits of the third word complete the 14 bit address. The unused two bits in the third word are ignored and may assume any value.

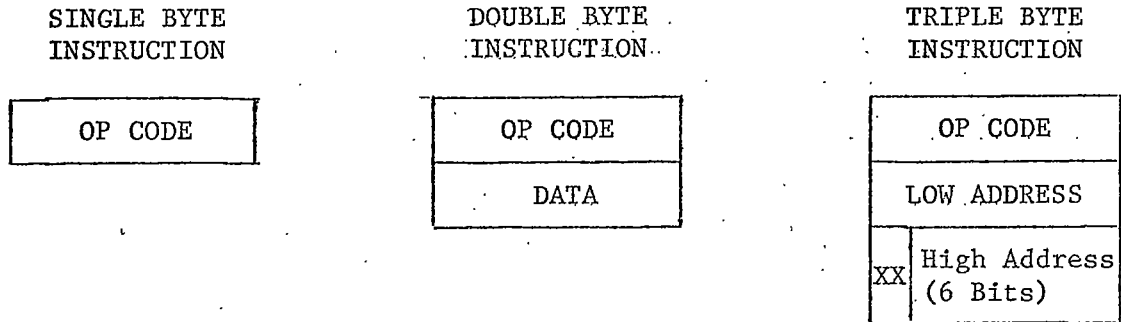


FIGURE 4: INSTRUCTION FORMATS

The instruction set may be broken down into 11 basic groups of instruction types as described below. Appendix B contains a formal description and listing of the OP code and mnemonic assigned to each instruction by the manufacturer of the microprocessor.

Single Register Instructions are encoded in a single byte.

They allow any register except the Accumulator (Reg. A) or a memory location to be incremented or decremented by a count of one. All processor flags except the Carry are affected.

Move Instructions are used to move data. This group of single byte instructions may move data from register to register, memory to register, or register to memory. Only direct transfers between memory locations are not permitted and must be accomplished by using two Move instructions and temporarily storing the data in a processor register. If a memory location is one of the registers involved in a Move instruction, it will be addressed through the

contents of H and L registers. If the contents of a processor register are moved to that same register (these are valid OP codes) then no data is changed and a NOP (No Operation) effectively results. None of the flags are affected by the Move type of instruction.

Register or Memory to Accumulator Instructions are encoded into one byte. The A register and another register (either processor or memory) are used as the arguments for an ALU operation (ADD, ADD with Carry, SUBTRACT, SUBTRACT with Borrow, AND, OR, EXOR, or COMPARE). In each case, except for COMPARE which affects only the flags, the result of the operation is placed in the Accumulator. If the second argument is to be a memory register, it will be addressed through the contents of H and L. These instructions affect all the flags depending upon the particular operation and result. Behavior of the flags is explained in detail in the formal description of the instruction set.

Rotate Accumulator Instructions are single byte instructions that may be used to rotate the contents of Register A either left or right and through or around the Carry Flag. In all cases, the Carry Flag is the only flag affected and will assume the value of the bit that was rotated off the end of the word. Figure 5 shows how the Rotate instructions function.

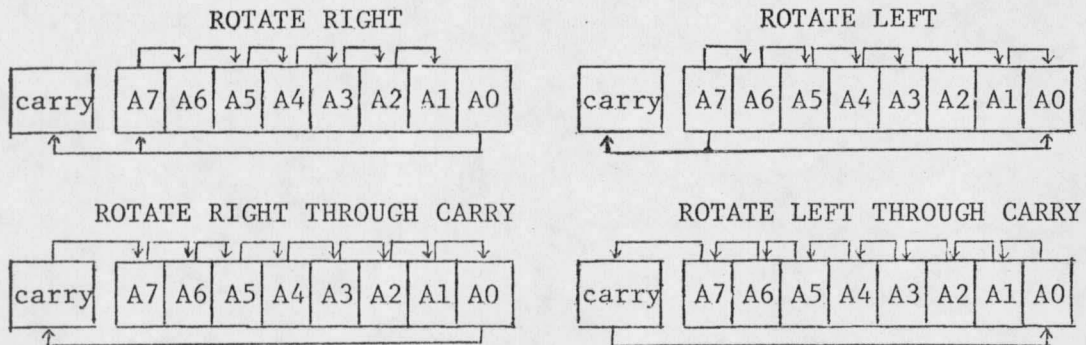


FIGURE 5: ROTATE ACCUMULATOR INSTRUCTIONS

Immediate Instructions are a group of two byte instructions in which the second byte serves as data. Thus the Immediate MOVE instruction loads the processor or memory register with the second byte of the instruction (Data). As usual, the memory location is determined by H and L. ALU operations may also be carried out between the accumulator and an immediate value in an identical manner to the Register or Memory to Accumulator group of instructions. Results are again placed in Reg. A and the flags are affected in an identical manner.

Jump Instructions are encoded in three bytes. The first byte is the OP code while the following two bytes define the address. The second byte is the low portion of the address while the third byte serves as the high portion of the address. A Jump may be unconditional in which case the PC is set to the jump address and program execution continues from this point. If a Jump is conditional

a test will be made to determine the state of a specified processor flag. Only if this flag test succeeds will the jump be made. Otherwise execution will continue on the instruction which follows the Jump. Figure 6 illustrates program flow for Jump instructions. Conditional Jumps may be made on either state (set or clear) of any of the four flags (Carry, Zero, Sign, Parity). Thus program flow may be modified on the basis of past results.

Call Instructions are also three byte instructions where the last two bytes specify an address. Calls perform a similar function to a Jump except that the old PC is not destroyed, rather it is saved by the processor (in the stack) to enable a later return to the instruction following the Call. Thus this class of instruction gives the processor the ability to handle subroutines. No more than seven Calls should be nested without executing a return or else the storage capabilities for return addresses will be exceeded resulting in the loss of older addresses in order to accommodate the seven most recent. Like the Jump instructions, Calls may be either unconditional or dependent on the state of a flag (conditional). None of the Call instructions will affect the state of the flags. Figure 6 shows the program flow for Call instructions.

Restart Instructions are a special group of Call

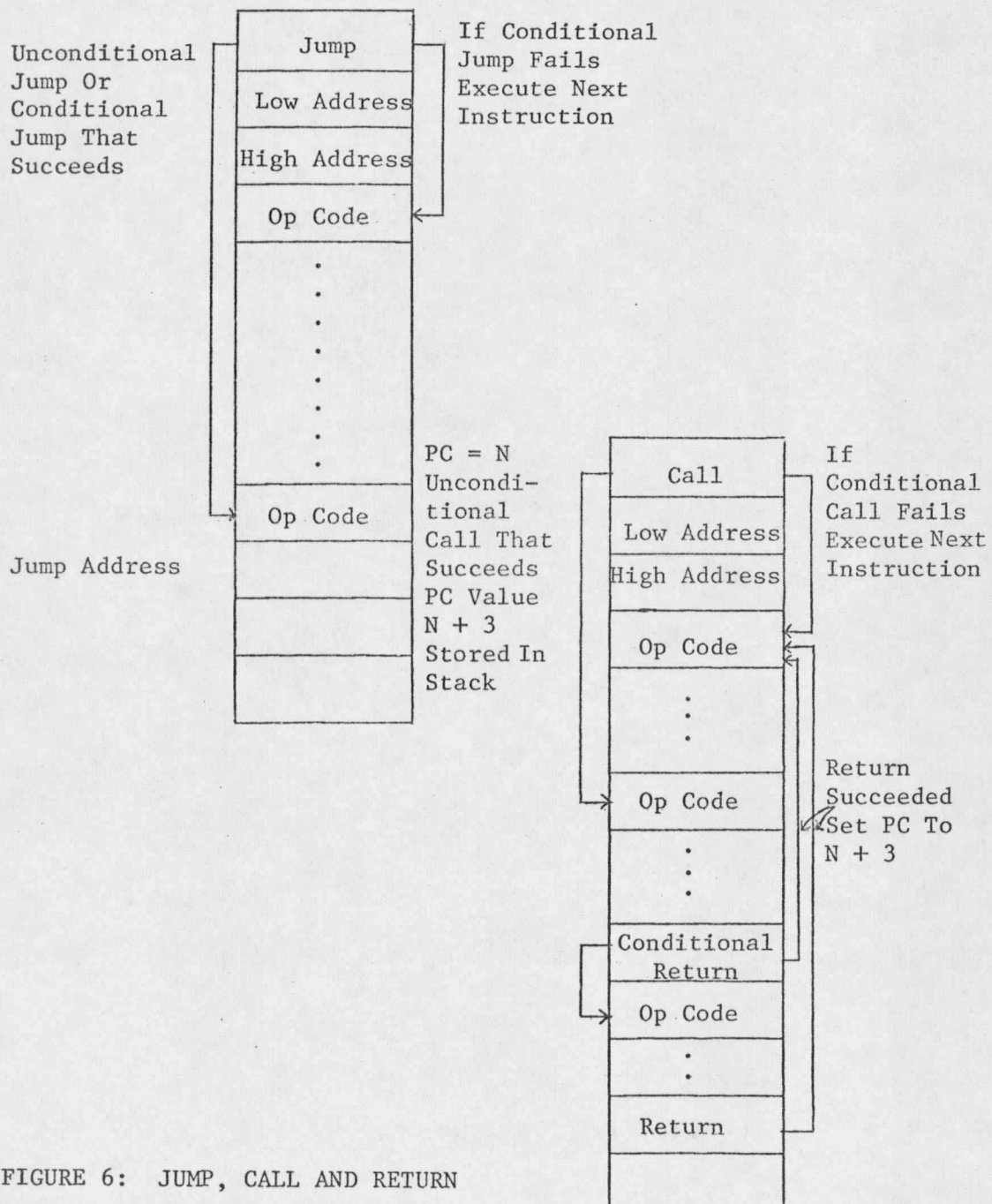


FIGURE 6: JUMP, CALL AND RETURN

instructions that are encoded in a single byte. Since there are no bytes following the OP code, the address must be encoded in the OP code itself. This necessarily limits the number of possible Restart instructions. The OP Code format for the Restart instruction is shown below.

RESTART 00 AAA 101

The address of the routine to be called is coded in the three bits AAA such that a given Restart calls location AAA 000. Thus there are eight possible restart locations (0B, 10B, 20B, 30B etc. to 70B). Restart instructions are unconditional and will always result in the call being performed and the return address pushed into the stack. In general this group of instructions is reserved for use by the I/O interrupt logic to generate calls to interrupt service routines. Except for the limited addressing range and the single byte format, the instruction behaves like any call.

Return Instructions are the complimentary instruction to the Call and must logically be the last statement executed in a subroutine. The Return instruction causes the PC to be restored to what it was prior to the last Call (last value pushed into the stack). Return instructions may also be unconditional or conditional and have no affect on any of the flags. Behavior of the Return instructions is also shown in Figure 6.

Input/Output Instructions are a group of single byte instructions used to pass data in and out of the 32 system I/O ports. Port address is encoded by the RRAAA bits in the OP code for the respective Input or Output instruction. The OP code format is detailed below.

	INPUT/OUTPUT	01	RRA	AA1
RR=00	Data Input Port			(Input instruction)
RR=01	Data Output Port			(Output instruction)
RR=10	Data Output Port			(Special Instruction)
RR=11	Data Output Port			(I/O Interface Control)

Note that there are 8 ports for inputting data and 24 ports that output data as determined by the two bits RR of the OP Code. In this system output ports where RR=10 have been defined as Special Instructions and are used to accomplish hardware control functions not provided for by the standard 8008 instruction set. Similarly those output ports where RR=11 have been set aside for control of the interface circuitry used to pass I/O data in and out of the machine. In all cases the output data is the contents of Register A (Accumulator) and input data will be placed in Register A. None of the flags are affected. Input/Output instructions are identified by their octal port addresses (e.g. INP 6B or OUT 13B).

Machine Instructions (HALT) are a group of three single byte instructions all of which produce the same result. Executing this

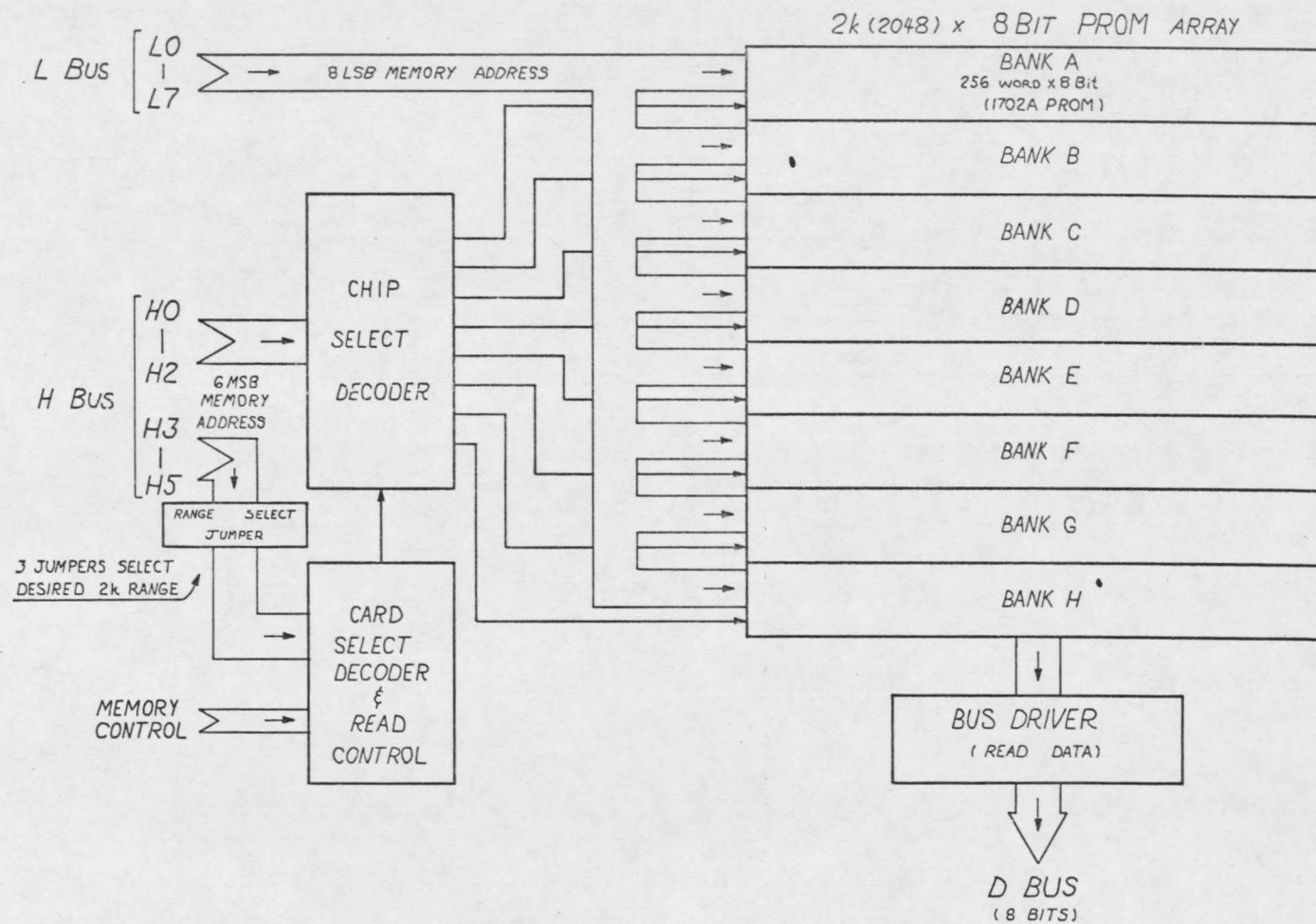
instruction caused the system to enter the STOP state and no more code will be executed until an interrupt is received causing the system to begin running again.

MEMORY

Maximum directly addressable memory is limited to 16k (16,384) 8-bit words. Static RAM and nonvolatile PROM may be mixed in any possible configuration with gaps of nonexistent memory allowed. Figures 7 and 8 block diagram the 2k PROM and 4k RAM cards.

RAM is expandable in banks of 1k words (1024) with each circuit card accommodating up to 4 banks for a total of 4k words. Each RAM card contains decoding logic for 14-bit addresses. A card is first selected to decode in the range of 0 through 7k or 8 through 15k. Each of the four banks on a RAM card may then be decoded into any of the eight possible 1k ranges. Write protection is available on each bank through a pin on the card edge connector enabling user selected banks to be protected by connecting their protect pins together in the bus and grounding the wire. RAM may be cycled as fast as 1 us during DMA transfers. Codes stored in RAM will be destroyed if power is lost from the memory system.

PROM is expandable in banks of 256 words with each card holding up to 2k. Address decoding is also 14 bits wide but decodes only in a 2k range. Thus a PROM card might address 0



27

FIGURE 7: PROM BLOCK DIAGRAM

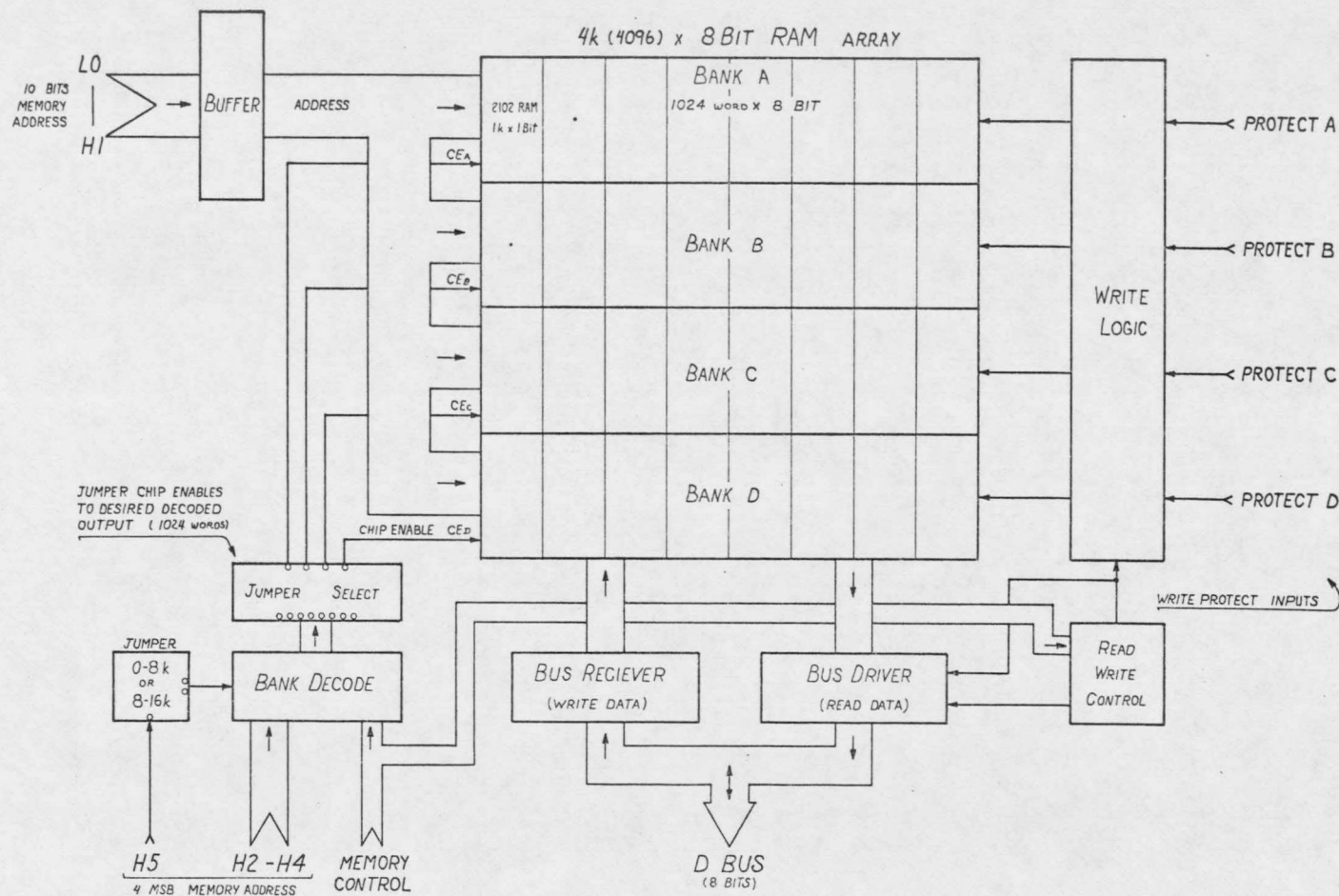


FIGURE 8: RAM BLOCK DIAGRAM

through 2k or 2k to 4k and so forth to the last range of 14k to 16k. Each individual PROM bank may be removed and programmed using special hardware. Like RAM, PROM may be cycled in 1 us. Code stored in these memories is safe from power loss (nonvolatile) and may be changed only by the programming procedure recommended by the chip manufacturer. PROM programming is discussed in more detail in Chapter 5.

If the memory system contains gaps there is a measure of protection against executing these portions of memory. If data from a nonexistent location is used as an OP code the processor will halt. Writing into gaps or PROM locations will result in nothing but wasted time. The processor will perform the motions but nothing will result.

A usable memory may consist of any possible nonoverlapping selection of RAM and PROM banks. A complete 16k PROM may be contained in eight circuit cards. Alternately, 16k of RAM occupies 4 cards. Smaller mixed PROM/RAM systems are possible and probably desirable in most applications.

CHAPTER 3

I/O STRUCTURE

The I/O portion of the machine is composed of a series of up to eight I/O interface cards. These may be data interfaces which allow data to be input and output from the system, or a Real Time Clock used for timing purposes may be substituted for a data interface.

Figure 9 introduces a generalized block diagram for the data interfaces. This diagram represents either a SIO or PIO channel. While there are minor differences between these two I/O interfaces, for the purposes of this section their characteristics may be considered identical.

Control of each I/O interface channel is achieved through the use of a control word. Four bits of the control word are used for External Control and merely appear as levels on the I/O connector. They may be used as desired for performing control functions within the peripheral device.

The remaining four bits comprise a control field that serves to enable Input and Output Interrupts, enable the Status mode, or Initialize the interface. Detailed descriptions of the control words may be found in the chapters devoted to the individual interfaces.

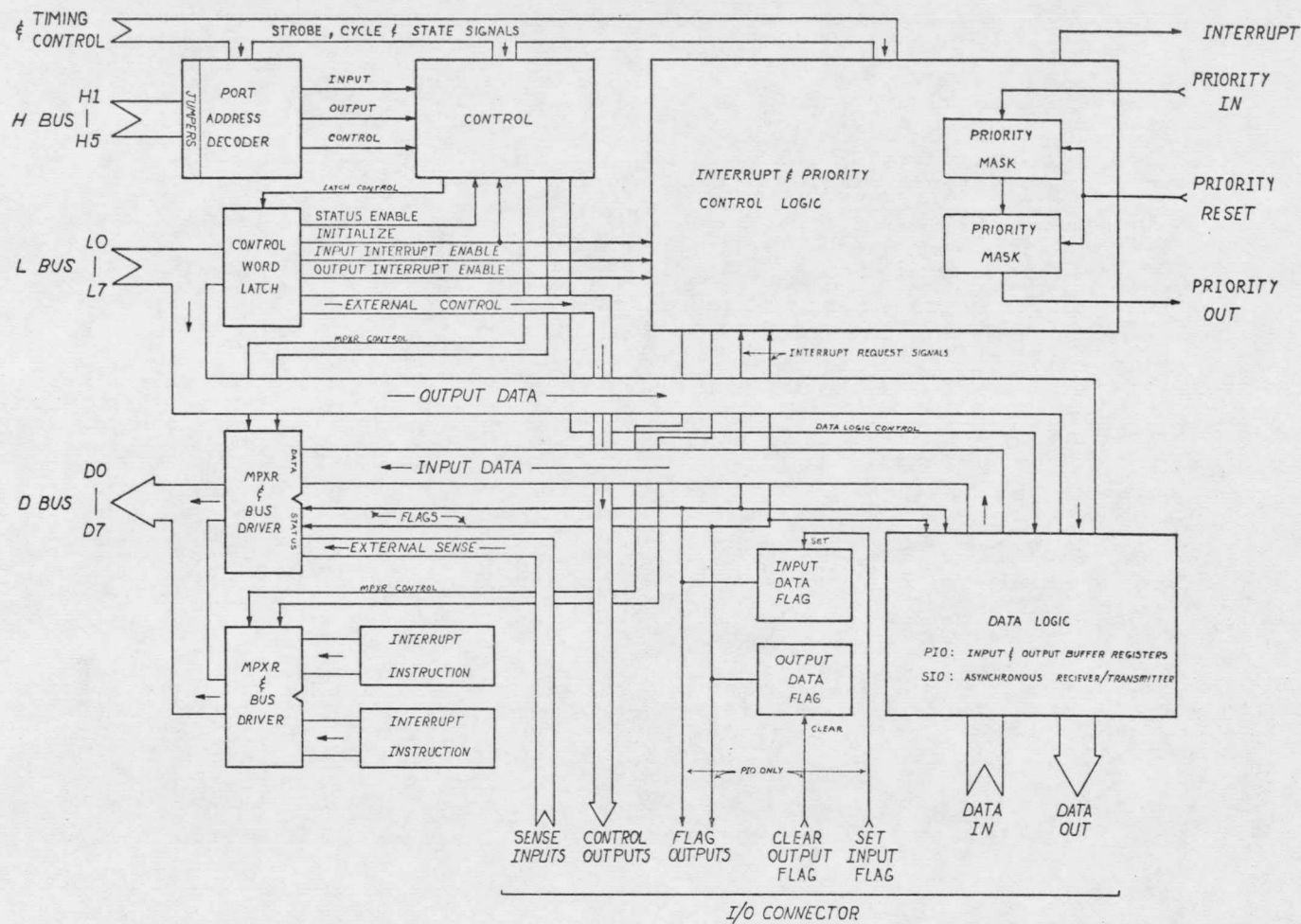


FIGURE 9: GENERALIZED DATA I/O INTERFACE BLOCK DIAGRAM

I/O is accomplished by addressing one of 32 ports with an Input or Output instruction contained in the program. Each I/O card or RTC decodes its own ports making these addresses independent of the position of the card in the bus. Eight of these ports (addresses 0B through 7B) are data input ports which place the input data present at the port into the A register. The remaining 24 output ports output the contents of register A. They are split into three groups serving the following functions. Ports having addresses 10B to 17B are defined as data output ports and pass data out through the I/O interfaces. Ports having addresses 30B and 37B are defined as I/O interface control ports and the output data (A register) is used to control the addressed interface. The remaining eight port addresses (20B to 27B) comprise the Special Instruction group and are used for hardware functions not performed by the standard 8008 instruction set. As mentioned earlier, three of these have already been used for control of the Interrupt and Priority Control logic.

Each data I/O interface requires three ports, two output ports (data and control) and one input port (data). Port addresses are decoded by the interface such that a typical set of ports used by one interface would be Input 0B, Output 10B (data) and Output 30B (control). Such a group of decoded ports used by one

interface card forms an I/O channel with data input, output and control. Each interface contains a set of jumper wires that are used to set the channel address as desired. Table 1 shows the channel assignments that may be decoded by an I/O interface. Since the port address decoding is done on the interface card itself, the address of a given card is independent of its position in the bus allowing cards to be moved in relation to each other without having to change addresses in the software.

POLLED I/O

Each data port (input and output) within a channel may be operated with interrupts enabled or suppressed as set by the control word. In either mode, the processor inputs and outputs data by executing the appropriate I/O instruction addressed to the port through which the data is to pass. Each data interface contains data storage latches enabling the processor to execute an output and load the output data storage register or to execute an input and read the contents of the input data storage latch. In addition to the data storage latches, each interface also contains an output data ready flag and an input data ready flag. The output flag is set when the processor outputs data and is cleared when the peripheral device accepts this data. The input data flag is set

CHANNEL NUMBER	RR				CHANNEL (AAA)
	00	01	10	11	
0	Input 0B	Out 10B	Out 20B	Out 30B	000
1	Input 1B	Out 11B	Out 21B	Out 31B	001
2	Input 2B	Out 12B	Out 22B	Out 32B	010
3	Input 3B	Out 13B	Out 23B	Out 33B	011
4	Input 4B	Out 14B	Out 24B	Out 34B	100
5	Input 5B	Out 15B	Out 25B*	Out 35B	101
6	Input 6B	Out 16B	Out 26B*	Out 36B	110
7	Input 7B	Out 17B	Out 27B*	Out 37B	111
	Data Input	Data Output	Special Inst	Control Output	

*Note: Reserved for control of Interrupt and Priority logic

OUT 25B : POFF (PRIORITY OFF)
 OUT 26B : PON (PRIORITY ON)
 OUT 27B : PRST (PRIORITY RESET)

TABLE 1: I/O CHANNEL ASSIGNMENTS

by the peripheral and cleared by the computer when it reads the contents of the register.

If a control output is performed with the status function enabled, the next time the processor performs an input from that channel it will read status information and not input data. Contained in the status word along with the External Sense inputs and error conditions (serial I/O only) are the two data flags. Thus if the processor is placed into a program loop that repeatedly enables status and checks the state of the data flags, the processor can determine when the I/O port needs servicing. This procedure is called polled I/O. While it is very easy to program, it makes poor use of processor resources since most of the execution time is spent executing the status check loop and not actually inputting or outputting data. The status mode resets after each status input and must be enabled before each use.

INTERRUPT I/O AND PRIORITY CHAIN

When interrupts are enabled, the operation of the port and generation of the interrupt is subject to the control exerted by the Priority Chain. Each interface contains a portion of this logic with priority determined by position relative to the processor.

Figure 10 illustrates a short Priority Chain. In the case

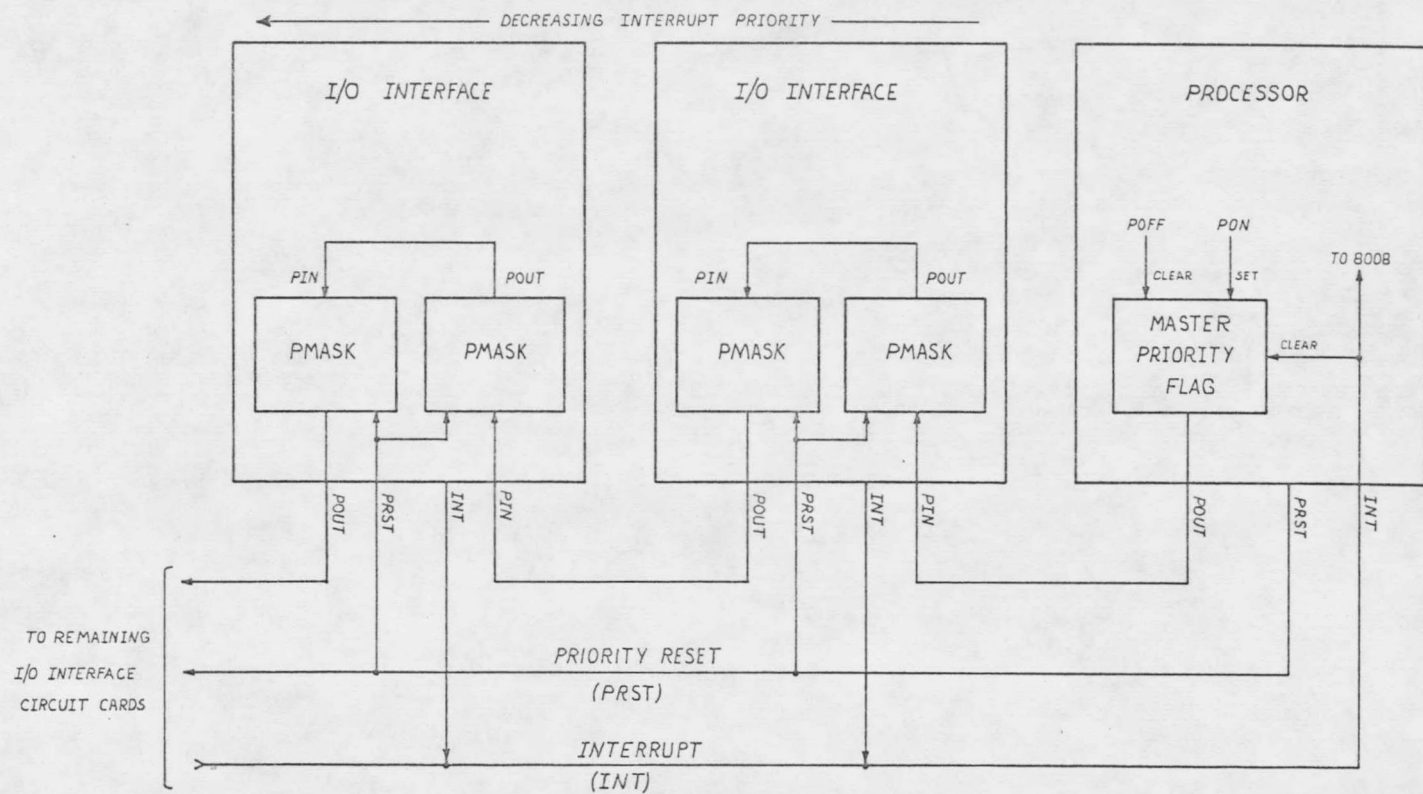


FIGURE 10: PRIORITY CHAIN

where more than two I/O interfaces are present on the bus the chain is just continued further in the lower priority direction as shown in Figure 10.

Since interrupts may arrive at unknown times from different I/O ports, some method of resolving conflicts between interrupts that may arrive at the same time is needed. This task is performed by the Priority Chain logic which is contained in each I/O interface.

The processor develops a master Priority Out (POUT) signal which is fed to the Priority In (PIN) input of the I/O interface located physically next to the processor. Assuming this port does not have its interrupts enabled and has not received a request, then this signal will be passed through the port's Priority Mask (PMASK) to become that port's POUT signal which feeds the PIN input of the next lowest priority data port. Each data I/O interface actually contains two complete interrupt chain structures. Input is considered to be higher priority than output within the same channel.

Now assuming the output port in the highest priority channel is inactive, it will pass its PIN signal through its PMASK to become the PIN signal for the second highest priority I/O channel. Again assuming neither of the PMASKs in this second channel are

set the priority signal will be passed on through the second I/O interface and so forth until the POUT output of the lowest priority port is reached where the chain terminates. The lowest priority POUT signal is not used.

Whenever the processor is outputting the master POUT signal and none of the PMASKs are set, the Priority Chain is in a state where any port may interrupt (if its interrupts are enabled). Assuming the data flag associated with one of these ports generates an interrupt request, several actions take place.

First, as soon as the data flag request is made the PMASK associated with the ports interrupt logic is set. This blocks passage of the priority signal down the chain below the set PMASK thereby preventing any lower priority interrupts. These ports may however make requests and set their PMASKs if their interrupts are enabled thus storing the interrupt request until a later time when they receive the priority signal from higher up the chain.

Second, when the PMASK is set and the board has priority, the interrupt logic waits until interrupt time (less than 5 us) and then signals the processor through the Interrupt (INT) line contained in the system bus. Upon reaching the processor the INT signal clears the master POUT signal preventing any further interrupts from all I/O ports. Once the interrupting routine has

executed its status save code the master POUT signal may be reset by executing Special Instruction Priority ON (PON) and higher priority ports may now interrupt the lower priority interrupt routine if they receive a request. Lower priority is still inhibited by the set PMASK.

The process of causing the processor to execute the interrupt routine is caused by the interrupt instruction (setable with jumpers) contained in the interrupt logic for each port. After the INT signal is received by the processor it finishes execution of the instruction it was working on when the interrupt arrived.

The processor then generates an interrupt instruction fetch cycle. The interrupt logic decodes this cycle and supplies its instruction to the processor in a manner similar to an instruction that came from the program stored in memory. While this interrupt instruction may be any valid machine instruction, the interrupt logic is capable of handling only a single byte. In most cases the RESTART (single byte subroutine CALL) is used to cause execution of the service routine and the RETURN instruction is then used to go back to the program that was interrupted after the service routine is complete. Since the RESTART is a subroutine call and uses the stack, interrupts must not be allowed to nest more than seven levels deep or else trouble will result from lost return

addresses.

Due to the limited number of RESTART instructions available in the 8008 instruction set, if more than eight data ports (four channels) are to run on interrupts, then a single RESTART would be used for both ports in a channel and the processor would perform a status check to determine which port within the channel had generated the interrupt. If less than eight ports are to use interrupts then each port may have its own RESTART and no software checks are needed since each interrupt routine can be called directly.

For added versatility in controlling interrupts, the master POUT signal may also be cleared through software by executing Special Instruction Priority Off (POFF). This allows the program to suppress all interrupts when desired.

Once software is through using a particular peripheral device it is necessary to reset the PMASK in that port's interrupt logic. By executing Special Instruction Priority Reset (PRST) the highest priority PMASK that is set will be cleared thus passing the priority signal to lower priority boards. Only one restriction applies in the use of PRST. The master POUT signal must be present for PRST to affect any PMASKs. Thus PON must be executed before PRST or the mask will not reset.

REAL TIME CLOCK AND PRIORITY CHAIN

While the Real Time Clock is not a data interface it does interface to the I/O section of the machine and uses the Priority Chain.

The RTC contains a single level of interrupt logic and communicates with the processor only with interrupts. Interrupts from the clock do not indicate that a data port needs servicing but rather that a specified amount of time has gone by since the clock was last serviced. Since the purpose of the clock is to keep time its priority would tend to be high (close to the processor) in the total priority chain. A Real Time Clock is necessary in such applications as data acquisition systems where the I/O interfaces must be read at certain intervals. When a clock is used, the processor is relieved of the duty of keeping time.

In summary, several facts apply to the operation of the Interrupt Priority and I/O systems. First, both port address decoding and interrupt instruction are contained in the I/O interfaces and are independent of the position in the bus. Second, interrupt priority is dependent on the position of the interface in the bus relative to the processor. The closer a given port is to the processor the higher its interrupt priority. Third, I/O interfaces that will use interrupts must be inserted into the bus

with no empty connectors between them and the processor so as not to break the Priority Chain. Interfaces to be used only in the polled mode (no interrupts) may occupy any place in the bus. Fourth, the interrupt logic and Special Instructions (PON, POFF and PRST) allow interrupts to be controlled so that they are serviced in the order of their priority and time of arrival. Higher priority interrupts are given preference over those of lower priority, but due to the automatic reset of the master POUT signal upon the occurrence of an interrupt, a lower priority interrupt may be completely serviced before a higher priority interrupt if the lower one was in process first. The higher priority port can interrupt the lower routine when PON is executed thereby restoring priority.

CHAPTER 4

PROCESSOR

Under control of a program composed of a group of instructions (introduced earlier) the processor serves to control the system memory and I/O ports. Data can be modified (arithmetically or logically) or used to make a decision about future program flow. The control and arithmetic functions of the processor are performed by the Intel 8008 uP. Additional logic is included to provide a suitable interface between the 8008 and the rest of the system.

The processor communicates to memory and I/O through the three system buses (see processor block diagram, Figure 3). The D bus (8 bits) is used to carry instruction, memory and I/O input data to the processor. It is also used to carry write data to memory from the processor card. The L bus (8 bits) and H bus (6 bits) are always driven by the processor during normal operation. When addressing memory, the L bus is used to carry the eight least significant bits of memory address while during an Output or Input instruction it contains the contents of register A. The H bus performs a similar function. During memory addressing it carries the six most significant bits of memory address and during Input or Output instructions it contains the I/O port address (Bits 1-5 of the OP Code). The processor card also generates and supplies a series of timing and control signals used to control the memory

and I/O ports. Similar signals are used internally by the processor card to control the H and L bus latches and the bidirectional transceiver that connects the 8008 to the D bus.

INSTRUCTION EXECUTION SEQUENCE

Execution of an instruction by the processor occurs as an ordered sequence of state transitions as shown in Table 4. Depending upon the instruction, complete execution may require from one to three cycles (Instruction, Read, Write, I/O) each of which is composed of three to five T states (T1 or T1I, T2, T3, T4, T5). The T states are decoded from outputs on the 8008 according to the state codes shown in Table 2. These signals are available on the bus for timing purposes.

S0	S1	S2	STATE
0	1	0	T1
0	1	1	T1I
0	0	1	T2
0	0	0	WAIT
1	0	0	T3
1	1	0	STOPPED
1	1	1	T4
1	0	1	T5

TABLE 2: STATE CODES

Instruction cycle always occurs first in the execution sequence. The low and high order bits of the PC are output by the 8008 and stored in the L and H latches during T1 and T2

respectively. The PC is then incremented by one. During T2, the 8008 also outputs (in addition to the 6 high bits of PC) two bits used to encode cycle type. These bits are latched and decoded to the bus. Table 3 lists the codes for the different cycles.

D6	D7	CYCLE
0	0	INSTRUCTION
0	1	READ
1	0	I/O
1	1	WRITE

TABLE 3: CYCLE CODES

T3 is used to generate memory control signals and read the instruction OP code (addressed by the PC) into the 8008. The instruction is decoded and the PC incremented by one. From this point the behavior of the processor depends on the instruction being executed. As Table 4 shows, some instructions can complete execution in the Instruction cycle. T4 and T5 are present only when executing certain instructions and are used by the 8008 for internal operations.

If the instruction is multibyte or requires addition cycles for execution, the Instruction cycle will be followed by a Read, Write, or I/O cycle. The Read cycle is performed exactly like Instruction except that there are two sources for memory address. If the Read is required for the second or third byte of multibyte

INDEX REGISTER INSTRUCTIONS

INSTRUCTION CODING D ₇ D ₆ D ₅ D ₄ D ₃ D ₂ D ₁ D ₀	OPERATION	# OF STATES TO EXECUTE INSTRUCTION	CYCLE ONE (1)					CYCLE TWO					CYCLE THREE				
			T1	T2	T3	T4	T5	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5
11 000 555	Lt ₁ t ₂	(5)	PC _L OUT (12)	PC _H OUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	REG. b TO D00										
11 000 111	LrM	(6)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	(4)	REG. L OUT (15)	REG. H OUT	DATA TO REG. b	X (4)		REG. b TO D00					
11 111 555	LSP	(7)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	REG. L OUT (17)	REG. H OUT	REG. b TO OUT								
00 000 110	LrI	(8)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	DATA TO REG. b	X		REG. b TO D00					
00 111 110	LMI	(9)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	DATA TO REG. b			REG. L OUT (17)	REG. H OUT		REG. b TO OUT		
00 000 000	Inr	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	X	ADD OP-FLAGS AFFECTED										
00 000 001	Dcr	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	X	SUB OP-FLAGS AFFECTED										

ACCUMULATOR GROUP INSTRUCTIONS

10 FFF 555	ALU OP X	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	ALU OP-FLAGS AFFECTED										
10 FFF 111	ALU OP M	(6)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		REG. L OUT (15)	REG. H OUT	DATA TO REG. b	X		ALU OP-FLAGS AFFECTED					
00 FFF 100	ALU OP I	(6)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	DATA TO REG. b	X		ARITH OP-FLAGS AFFECTED					
00 000 010	RLC	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
00 001 010	RRC	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
00 010 010	RAL	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
00 011 010	RAR	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										

PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS

01 xxx 100	JMP	(11)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	LOWER ADD. TO REG. b		PC _L OUT (15)	PC _H OUT	HIGHER ADD. TO REG. a	REG. a TO PC _L	REG. b TO PC _L
01 000 000	JFc	(9 or 11)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	LOWER ADD. TO REG. b		PC _L OUT (15)	PC _H OUT	HIGHER ADD. TO REG. a	REG. a TO PC _L	REG. b TO PC _L
01 100 000	JTc	(9 or 11)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	LOWER ADD. TO REG. b		PC _L OUT (15)	PC _H OUT	HIGHER ADD. TO REG. a	REG. a TO PC _L	REG. b TO PC _L
01 xxx 110	CAL	(11)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	LOWER ADD. TO REG. b		PC _L OUT (15)	PC _H OUT	HIGHER ADD. TO REG. a	REG. a TO PC _L	REG. b TO PC _L
01 000 010	Cfc	(9 or 11)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	LOWER ADD. TO REG. b		PC _L OUT (15)	PC _H OUT	HIGHER ADD. TO REG. a	REG. a TO PC _L	REG. b TO PC _L
01 100 010	Ctc	(9 or 11)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		PC _L OUT (15)	PC _H OUT	LOWER ADD. TO REG. b		PC _L OUT (15)	PC _H OUT	HIGHER ADD. TO REG. a	REG. a TO PC _L	REG. b TO PC _L
00 xxx 010	RET	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	POP STACK (10)	X								
00 000 011	Rfc	(3 or 5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	POP STACK (10)	X								
00 100 011	Rtc	(3 or 5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b	POP STACK (10)	X								
00 AAA 101	RST	(5)	PC _L OUT	PC _H OUT	FETCH INSTR. TO REG. b & PUSH STACK (9 REG. a)	REG. a TO PC _L	REG. b TO PC _L (11)								

I/O INSTRUCTIONS

01 000 001	INP	(8)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		REG. A TO OUT (12)	REG. b OUT	DATA TO REG. b	SEND FF OUT (11)	REG. b TO REG. A					
01 000 001	OUT	(6)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b		REG. A TO OUT (12)	REG. b OUT	X							

MACHINE INSTRUCTIONS

00 000 00X	HLT	(4)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b & HALT (14)											
11 111 111	HLT	(4)	PC _L OUT	PC _H OUT	FETCH INSTR. TO IR & REG. b & HALT (14)											

NOTES

D00 denotes destination register while SSS denotes source register according to the codes below:

REGISTER CODES

000	Reg. A
001	Reg. B
010	Reg. C
011	Reg. D
100	Reg. E
101	Reg. H
110	Reg. L

PPF denotes one of the eight possible ALU operations (not including INCREMENT or DECREMENT)

CC denotes a specific flag bit according to the table below:

FLAG CODES

00	CARRY (Set to one if underflow or overflow)
01	ZERO (Set to one if result is zero)
10	SIGN (Set to one if MSB of result is one)
11	PARITY (Set to one if parity is even)

AAA denotes a RSTART address such that the PC is set to address AAA 000 (binary). Stack behaves in same manner as any CALL instruction.

RR MM enclose the I/O port address within the OP code for the INPUT and OUTPUT instructions. When RR=00 then the instruction is an INPUT. When RR=01, 10 or 11 the addressed port outputs data.

X serves to denote a bit within an OP code whose value may assume either zero or one.

- First cycle is always INSTRUCTION
- PC_L denotes the eight LSB of the PC while PC_H denotes the six MSB of the PC
- Instruction OP code input to the 8008 from memory and decoded during this state.
- These states are skipped
- Memory READ cycle
- X denotes an idle state
- Memory WRITE cycle
- When a conditional JUMP fails, states T4 and T5 are skipped and the processor begins the next INSTRUCTION cycle
- When a conditional CALL fails, T4 and T5 are skipped and the processor starts the next INSTRUCTION cycle. If the condition is true then T4 and T5 are used to push the old PC into the stack and load the two bytes of new PC into PC_L and PC_H.
- When a RETURN condition is true the stack is popped restoring the PC. Otherwise T4 and T5 are skipped and a new INSTRUCTION cycle begins.
- Bits 3 through 5 (AAA in OP code) are loaded into the same positions in PC_L. All other bits in PC are set to zero.
- I/O cycle
- During T4 of an INPUT instruction the flag bits are available on the bus: Sign on D₇, Zero on D₆, Parity on D₅, Carry on D₄.
- When the HALT instruction is decoded the processor remains internally in state T3. Externally SIOP is indicated. The time spent in STOP will be an integral number of state times. An INTERRUPT is required to start another INSTRUCTION cycle.

TABLE 4: PROCESSOR STATE SEQUENCES

instruction then the PC will be output during T1 and T2 just as in the Instruction cycle. After outputting the PC is again incremented. If the Read involves a memory register then the contents of register L and register H will be output during T1 and T2 respectively in place of the PC. Again during T3 the processor card generates the memory control signals to accomplish a Read operation. T4 and T5 are optionally present depending on the instruction.

A Write cycle occurs in a similar manner. The address always comes from registers H and L and during T3 the necessary memory control signals to accomplish a write operation are generated. T4 and T5 never occur in a Write cycle.

During the I/O cycle, the 8008 behaves differently but the external logic behaves in the same manner. When executing either an INPUT or OUTPUT instruction, the contents of register A are output to the L latch during T1 while the six least significant bits of the I/O instruction OP code (containing the port address) are output to the H latch along with the I/O cycle code during T2. The execution of the OUTPUT instruction ends after the T3 state which is used by the I/O port hardware to decode the port address from the H bus and take the data off the L bus. In the INPUT instruction T3 is used to input data to the 8008 from the D bus. T4 and T5 are used internally by the 8008 to complete execution

of the INPUT instruction.

Two interesting things should be noted about the behavior of the INPUT instruction. First the contents of register A are sent to the L bus just as in the OUTPUT instruction. It would be possible to load register A with additional bits of port address which would be replaced by the input data after executing the INPUT instruction. In this manner the number of input ports could easily be expanded from 8 to 2048. Second, the T4 state associated with the I/O cycle belonging to an INPUT instruction is the only time when the four processor flags are available outside the 8008. At all other times the state of the flags can be determined only by use of a conditional instruction (CALL, JUMP OR RETURN). Front panel hardware could store the flag bits for display purposes only during the Input I/O cycle.

In summary several facts may be stated about the behavior of the 8008 and associated logic. During every T1 state the 8008 loads the L latch with some data. If the cycle is either Instruction, Read or Write then memory address data will be loaded. This data is used as the eight least significant bits of address and will come from either the 8 LSB of the PC or register L depending on the instruction. During every T2 the 8008 loads the H latch (6 bits) and the cycle code latch (2 bits). Again if memory is to be accessed then address data will be output from either the 6 MSB of

the PC or the six lower bits of register H. The I/O cycle is treated in an identical manner except the 8008 outputs the contents of register A and the instruction OP code instead of memory address.

T3 is used for varying purposes depending upon instruction. The function of critical importance is the mode of the D bus transceiver which connects the 8008 D port to the D bus. During T3 of Instruction, Read or I/O cycles the 8008 receives from the D bus, while during T3 of Write and every T4 the 8008 transmits to the D bus. In the case of an OUTPUT instruction the data presented to the 8008 from the D bus is ignored. During an Input this port receives the data. By transmitting every T4 availability of the flags on the bus during I/O cycle (T4) is assured. This is the only time the flags are available outside the 8008.

PROCESSOR INTERRUPTS

Whenever power is first applied or after executing a HALT instruction, the processor will enter the STOP state. In this state the system is static and the 8008 only maintains its dynamic registers. In order to leave the STOP state and execute code the processor must be interrupted.

When an Interrupt (INT) signal arrives at the processor the Master POUT signal is cleared and the processor begins an

Instruction cycle where T1 is replaced by T1I. When T1I occurs, the Interrupt Cycle flip flop is set generating bus signal Interrupt Cycle (INTCY). The 8008 does not increment the PC and generation of Memory Control (during T3) is suppressed. During T3 the interrupting device (I/O interface or other) jams (transmits) an interrupt instruction onto the D bus. The term "jam" refers to the process in which an interrupting device transmits its interrupt instruction to the D bus.

This instruction is decoded and executed in the normal manner with T1I replacing T1 and INCY true for each cycle. If the instruction does require additional cycles then execution proceeds in the normal manner. Memory control signals will be generated but the contents of the PC will not be changed unless the instruction was CALL, JUMP, or RETURN.

At the end of execution of the interrupt instruction if the INT signal has been removed then the processor will begin normal program execution at the location of the PC. The first T1 will clear the Interrupt Cycle flip flop. If the INT signal is still present then another Interrupt Cycle will follow.

In order to cause the processor to start execution it is only necessary to interrupt and jam a single NOP instruction (load a register with itself) to the processor. Front panel

operations may be carried out by jamming a series of interrupt instructions written to accomplish the desired function such as loading a register or setting the PC.

The processor may also be interrupted during normal program execution. When the INT signal arrives the processor will finish execution of the current instruction before beginning the Interrupt Cycle. The I/O ports use this feature and jam the single byte subroutine call (RESTART) to cause the machine to execute the appropriate service routine.

WAIT STATE

If before the end of T2 the processor READY input (RDY) is made false, the processor will place a WAIT state between T2 and T3 for as long as RDY is false. During this period the contents of the machine are frozen. The WAIT state may be used either to give slow memory time to react or else the RDY input may be used along with the DMA input to freeze the processor and disconnect it from the buses to allow direct memory access by other hardware.

Two remaining bus signals concern the processor. Whenever power is applied to the processor a Clear pulse is generated ($\overline{\text{CLR}}$). This signal is used to reset the logic in the I/O interfaces. A normally open switch may be connected to the $\overline{\text{CLR}}$ bus line and ground to allow the operator to perform the same initialization

procedure as power up.

If desired, the contents of the H and L latches may be prevented from changing by making the H & L Inhibit (HLINH) input true. As long as this input is true the H L Bus will not change although execution otherwise proceeds in the normal manner.

PROCESSOR HARDWARE

The following section describes the hardware that interfaces the 8008 to the system bus. Familiarity with logic circuitry is assumed and no attempt has been made to describe the operation of the individual Integrated Circuits (ICs). Questions in this area are best answered by reference directly to the manufacturer's data sheets on the IC involved. The INTEL data book on the 8008 uP is included in this category.

Figure 11 presents the schematic diagram of the processor card. Figure 12 (timing diagram) shows the timing of the bus signals involved. Numbers within gates refer to location on the circuit card (see Appendix C).

The processor card requires power supplies of +5 and -9 volts DC. When power is applied to the card the RC time constant (R5, C5) on gate 22B develops a power-on reset pulse which is used to initialize the Master Priority Control flip flop (gates 3A, 3C; interrupts are enabled) and also to start the processor clock. This reset signal is also sent to the bus through gate 31B to become the

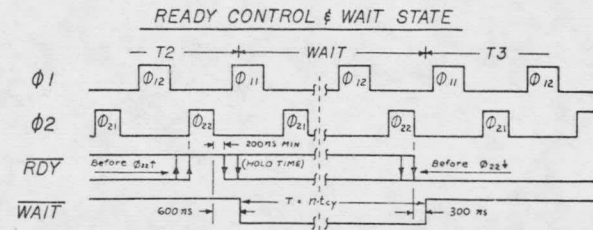
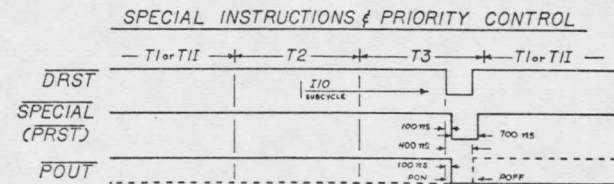
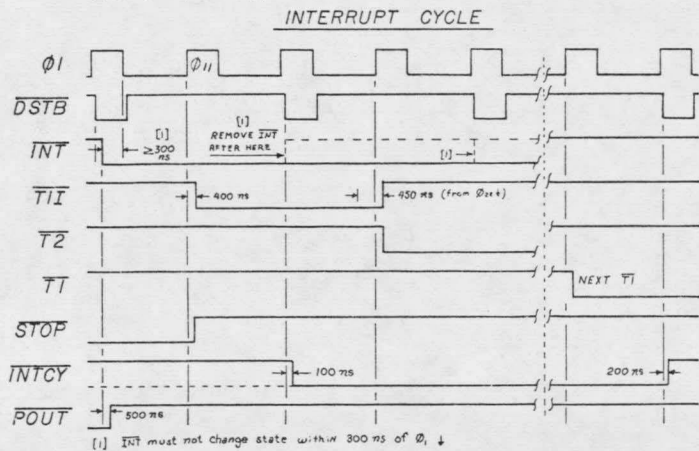
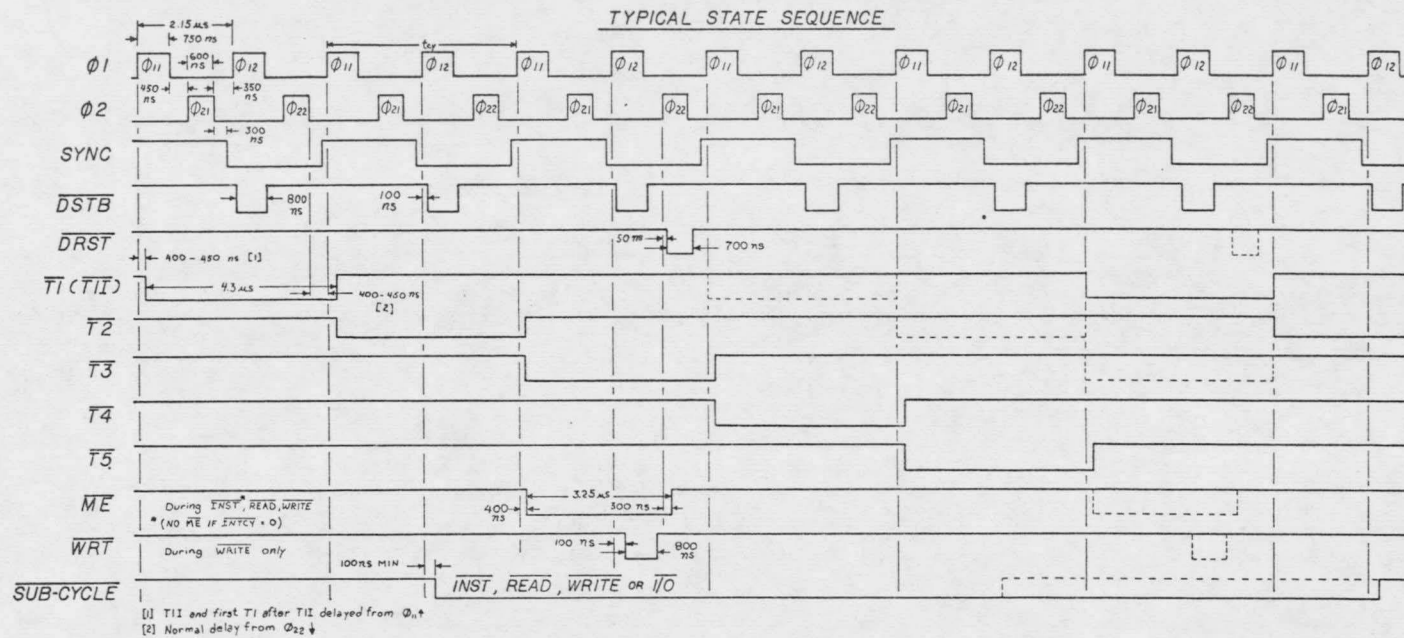


FIGURE 12: TIMING DIAGRAM

CLEAR ($\overline{\text{CLR}}$) signal which is used to initialize the I/O interface hardware. As mentioned earlier, a switch may be connected between the $\overline{\text{CLR}}$ bus line and ground to enable the operator to initialize the I/O and Master Priority Control flip flop manually.

The two-phase clock required by the 8008 is generated by a ring oscillator composed of IC11, 12, 13, and 14. Trimmer resistors allow individual adjustment of each portion of the clock waveform. R1 adjusts the width (time of phase 1 ($\phi 1$), R2 adjusts the time between the end of $\phi 1$ and the start of phase 2 ($\phi 2$). R3 sets $\phi 2$ while R4 controls the time between the end of $\phi 2$ and the start of the next $\phi 1$. Both clock phases are connected to the 8008 which returns the SYNC signal ($\phi 2$ divided by 2) which serves to distinguish between the two clock cycles in each state. The clock timing used in the prototype is shown on the timing diagram although it may be varied within the limits specified in the 8008 data book.

The current state of the processor is indicated by the three state outputs (S0, S1 and S2) as shown earlier. These state outputs are decoded by IC5 to eight individual signals ($\overline{\text{T1}}$, etc.) These are delayed by the 150pf capacitors (C6-C13) and supplied to the bus through two LPTTL inverters. Using two gates in this manner allows both the inverted and upright versions of each state signal to be used by other logic on the processor card.

$\overline{\text{SYNC}}$ and $\phi 1$ are combined in gate 25A to form timing signal DATA STROBE (DSTB) which is passed to the bus ($\overline{\text{DSTB}}$). This signal corresponds to the second $\phi 1$ ($\phi 1_2$) pulse in each processor state and is present at all times the card has power. In a similar manner $\phi 2$ and $\overline{\text{SYNC}}$ are combined with T3 by gates 15A and 25B to form timing signal DATA RESET (DRST). DRST is sent to the bus and also used internally by the processor card. Note that DRST occurs only during T3 and not during the STOP, WAIT or any other T state.

Two four bit latches (ICs 19 & 20) form the L latch. Inverted data from the D port on the 8008 is applied to this latch through the inverters in ICs 7 and 8. Data is stored inverted. Gates 16A, 16D and 24A use T1, T1I and DSTB with the H $\overline{\text{LINH}}$ signal to generate the necessary strobe needed to store the output of the 8008 during T1 or T1I.

Two more four bit latches (ICs 9 & 10) compose the 2 bit Cycle Code latch and the 6 bit H latch. Inputs to these latches are also connected to the 8008 through the inverters at the same point as the L latch. The signal from gate 16D ($\text{DSTB} \cdot \text{H} \overline{\text{LINH}}$) is combined with T2 in gate 16C to form the strobe for the H latch. In a similar manner T2 and DSTB are combined in gate 16B to form the strobe for the Cycle Code latch. A low logic level (zero) on the $\overline{\text{H} \overline{\text{LINH}}}$ input will prevent strobing the H and L latches but has no affect on the

Cycle Code latch.

The H and L latches drive their respective buses through open collector LPTTL gates (ICs 29, 30, 39, 40). As long as the $\overline{\text{DMA}}$ signal is high these gates act as inverters and drive the buses with upright data. A low $\overline{\text{DMA}}$ signal will shut these gates off allowing the H and L buses to be driven by other hardware such as a direct memory access controller.

The bidirectional bus tranceiver is connected between the D bus and the 8008 D port. Open collector gates from ICs 17 and 18 along with inverters from ICs 37 and 38 comprise the receiver. $\overline{\text{WRITE}}$ and $\overline{\text{T3}}$ are combined to generate the one on the output of gate 15D needed to enable the receive function every T3 except during a Write cycle.

The transmitter portion is composed of ICs 27, 28, 7 and 8 and operates in an identical manner. Gates 24C and 24D generate the enable signal such that the transmit function occurs every T4 and during T3 of a Write cycle. Because of the two inverting functions between the 8008 and the D bus, data is received and transmitted upright.

Gates 25C, D and 40C, D make up the Interrupt Cycle flip flop. When processing an interrupt instruction T1I and DSTB set this flip flop generating bus signal Interrupt Cycle ($\overline{\text{INTCY}}$). Once the

interrupt cycles are complete the first T1 and DSTB will clear the flip flop. INTCY is also used by the control logic that generates Memory Enable ($\overline{ME}$).

Whenever the system memory is to be used a $\overline{ME}$ signal must be generated. INST, READ and WRITE are combined with T3 and $\overline{INTCY}$ by gates 23A, B, C and 21A to produce a high level on the output of 23C during T3 when $\overline{ME}$ is required. Gates 22D and 21B along with R14 and C14 form a one-shot that develops a narrow pulse to set the memory enable flip flop (gates 21C & D) at the beginning of this interval. This flip flop is cleared by DRST (through gate 22F) later in T3. The output of this flip flop drives the $\overline{ME}$ bus line through gate 31D. Using an open collector driver permits external direct memory access hardware to generate the $\overline{ME}$ signal when the processor is in the WAIT or STOP states.

If the memory is to be read (INST or READ cycle) then only $\overline{ME}$ will be generated. When the memory is to be used in the write mode (WRITE cycle) then both $\overline{ME}$ and Write Enable ($\overline{WRT}$) will be generated. During T3 of a WRITE cycle DSTB is sent to the bus to become $\overline{WRT}$. Gates 15B and 24B combine the signals while gates 34A and 31A invert the signal and drive the bus respectively. Again the open collector driver allows driving the bus line with external hardware.

During an Interrupt Cycle behavior of the memory control

signals is modified. With the circuit as shown on the schematic, $\overline{ME}$ will not be generated during INST while the Interrupt Cycle flip flop is set. $\overline{ME}$ will be generated during a READ or WRITE. If pin 5 of gate 23B is connected to $\overline{INTCY}$ instead of READ as shown, then $\overline{ME}$ may also be suppressed during an interrupt READ cycle.

This option is needed because of the multibyte instructions used. The 8008 outputs a READ cycle when reading the second or third bytes of a multibyte instruction. It also outputs READ when reading data addressed by registers H and L. If the interrupt READ belongs to the second or third bytes of an instruction then memory control should be inhibited to allow the interrupting device to supply the additional bytes of its instruction without interference from the memory. But if the READ belongs to an instruction which is referencing memory for data then the memory should be allowed to function normally. There are two ways around this problem.

First, $\overline{ME}$ may be suppressed during READ. This allows jamming multibyte instructions with no bus conflicts between memory and the interrupting device. Unfortunately it also causes data reads performed in interrupt cycles to return faulty data. Since memory is inhibited, the processor will see the undriven D bus and always read data that is all ones because of the pullup resistors on the bus lines.

The second solution is more complex but allows all instructions to perform correctly under interrupt. $\overline{ME}$ is inhibited only during INST and any interrupt hardware that uses multibyte instructions must be able to control the $\overline{DMA}$ line. During READ cycles when additional bytes of instruction are to be jammed the $\overline{DMA}$ input must be taken to a logic 0. This causes the H and L busses to assume a condition of all ones (address 3FFF'). The memory register at this address must meet one of two criteria; it must be either undecoded (nonexistent) or contained on a memory card which drives the bus through open collector gates and be loaded with data of all ones. In either case the D bus is left free to be driven by the interrupting hardware.

IC4 and gate 3B form the Special Instruction Decoder. The address inputs (A, B and C) are connected to the outputs of the H latch where the three least significant bits (MMM in the OP code) of port address appear during an I/O cycle. The least significant R bit is used as one enable signal for IC4. The remaining enable is generated in gate 3B from signals I/O, DRST and the remaining R bit of port address.

Whenever an output instruction from the group of Special Instructions is executed, the appropriate output of IC4 will be at a zero during DRST of the I/O cycle. At other times all

ouputs are high. While the decoder recognizes all eight of the Special Instructions, only six have been used on the processor card or connected to the bus. Special Instruction PON and POFF are used to control the Master Priority Control flip flop. PON sets this flip flop enabling interrupts. POFF clears it and inhibits all interrupts. Out 27B is the Priority Reset (PRST) Special Instruction. PRST is sent directly to the bus as an inverted signal ($\overline{\text{PRST}}$) and serves to clear the Priority Mask flip flops in each I/O interface. Three uncommitted instructions are available on the bus as a decoded pulse identical to PRST. These are: SP20 (Out 20B), SP23 (Out 23B) and SP24 (Out 24B) and may be used for control of external hardware. SP21 and SP22 are not present on the bus in decoded form and must be decoded from the port address (H bus) and I/O cycle. As with any output instruction the contents of register A will be available on the L bus and may be used as data or control words if desired. Neither PON, POF or PRST use the output data. Output or special instructions will not change the contents of Register A.

Gates 3A and 3C form the Master Priority Control flip flop which drives the bus through gate 31F ($\overline{\text{POUT}}$). The flip flop may be set (interrupts enabled) by either $\overline{\text{CLR}}$ from the bus or PON. Clearing is accomplished by either POFF or an interrupt. Thus

after receiving an interrupt. POUT must be reenabled by executing PON.

R47 and C15 are used as a differentiator and make the Master Priority Control flip flop sensitive to only the negative edge of the $\overline{\text{INT}}$ signal. D2 serves to protect the input of gate 3A when the INT signal returns to a 1.

When interrupting the processor the $\overline{\text{INT}}$ signal must be synchronized to the 8008 clock. For reliable interrupt performance, INT (measured on the 8008) must not change state within 200 ns of the end of the $\phi 1$ clock (also measured on the 8008). By setting and clearing interrupts on the leading edge of $\overline{\text{DSTB}}$ timing problems are avoided. T11 should be used to enable the removal of the INT signal.

Similarly, to enter a WAIT state, $\overline{\text{RDY}}$ must go high before the last $\phi 2$ clock in state T2. This condition is met by synchronizing to either edge of DSTB or the negative edge of SYNC. The WAIT state will remain until $\overline{\text{RDY}}$ becomes a 0. The processor will then enter into the T3 state at the next $\phi 1_1$ clock and complete execution of the cycle. The RDY input and the WAIT state are useful for synchronizing the processor to slow memories or suspending processor activity to perform direct memory access. They may be also used to step the processor through a program one cycle at a

time. The later use may prove helpful for debugging logic errors in program code.

CHAPTER 5

MEMORY SYSTEM

The system memory is composed of two types; RAM and PROM.

The memory section of the bus provides connections for 14 bits of address (H & L bus), an 8-bit data port (D bus) and three memory control signals; Memory Enable ($\overline{ME}$), Read/Write control ($\overline{WRT}$) and Write Cycle ($\overline{WRITE}$). Power supply voltages of +5 and -9 volts are also available.

Each type of memory card will function correctly in any position on the bus, but if the Write Protect feature on the RAM is used, fixed positions may be necessary because the protect inputs for certain banks of RAM may not be used. Since address decoding is done on the card, changing position will not affect addressing but may result in different and undesired portions of RAM becoming write protected. This will depend on how the protect pins of each bus connector are wired in the individual system.

RAM and PROM may be mixed in any possible combination up to a total of 16k words. The PROM supplies nonvolatile storage. This code will be preserved if power is removed from the memory system. Typical uses for PROM storage would include program loaders, I/O driver subroutines and lookup tables. In general, PROM can be used for any form of code that does not need to be changed. In a system dedicated to a particular task, the entire program may be contained

in PROM with RAM used only for scratch and buffer storage if needed.

RAM provides memory into which data or code may be written and then read at some later time as dictated by the program. This memory may be used for any desired purpose, but since RAM storage is volatile, data will be lost if power is removed from the system. This requires that RAM be loaded through the use of a loader program (possibly stored in PROM) or external hardware after power is applied to the system. An alternate solution would be to supply the system with a backup battery supply to protect the contents of RAM if power is lost.

READ ONLY MEMORY

The PROM card may hold from one to eight 256 word electrically programmable read only memory chips (PROM chips). The card will function with either the INTEL 1702A or National Semiconductor MM520AQ. Since the PROM card need not contain a full compliment of memory chips, the minimum system is 256 words and is expandable in increments of 256 to the full capacity of 2k (2048) words per card. If larger amounts of PROM are needed multiple cards must be used.

In addition to the memory chips, each PROM card also contains a bus driver and a 14 bit address decoder. By inserting three jumper wires, the address decoder may be set up to decode any one

of the eight possible 2k ranges (0-2k, 2-4k, etc.) within the 16k range allowed by the 14 bit address. Within the selected 2k range the memory is organized so that consecutive blocks of 256 words are contained in the same PROM chip. Thus the first 256 words occur in one chip, the second 256 in another chip and so forth through the complete 2k capacity of the card. If a location on a PROM card which contains no PROM chip is read the resulting data will be all ones (data FF' or HALT instruction).

OPERATION

Figure 13 is the schematic for the PROM card. The card requires power supply voltages of +5 and -9 volts along with the 14 bits of address data and memory control signals $\overline{ME}$ and $\overline{WRITE}$. The eight lowest order address bits (L0-L7) are connected directly to the address inputs of each PROM chip on the card and serve to provide the word address for each PROM. The data outputs of each PROM are connected together to form an internal 8 bit data bus to the open collector 7417 bus drivers (ICs 12 and 13).

As long as none of the PROM chips receive a chip enable ($\overline{CE}$) all of their tri-state outputs will be off (in the high impedance state) resulting in a one on the input of each bus driver because of the pullup resistors on each of these lines. Since the 7417 does not invert the signal on its input each of the open collector outputs also

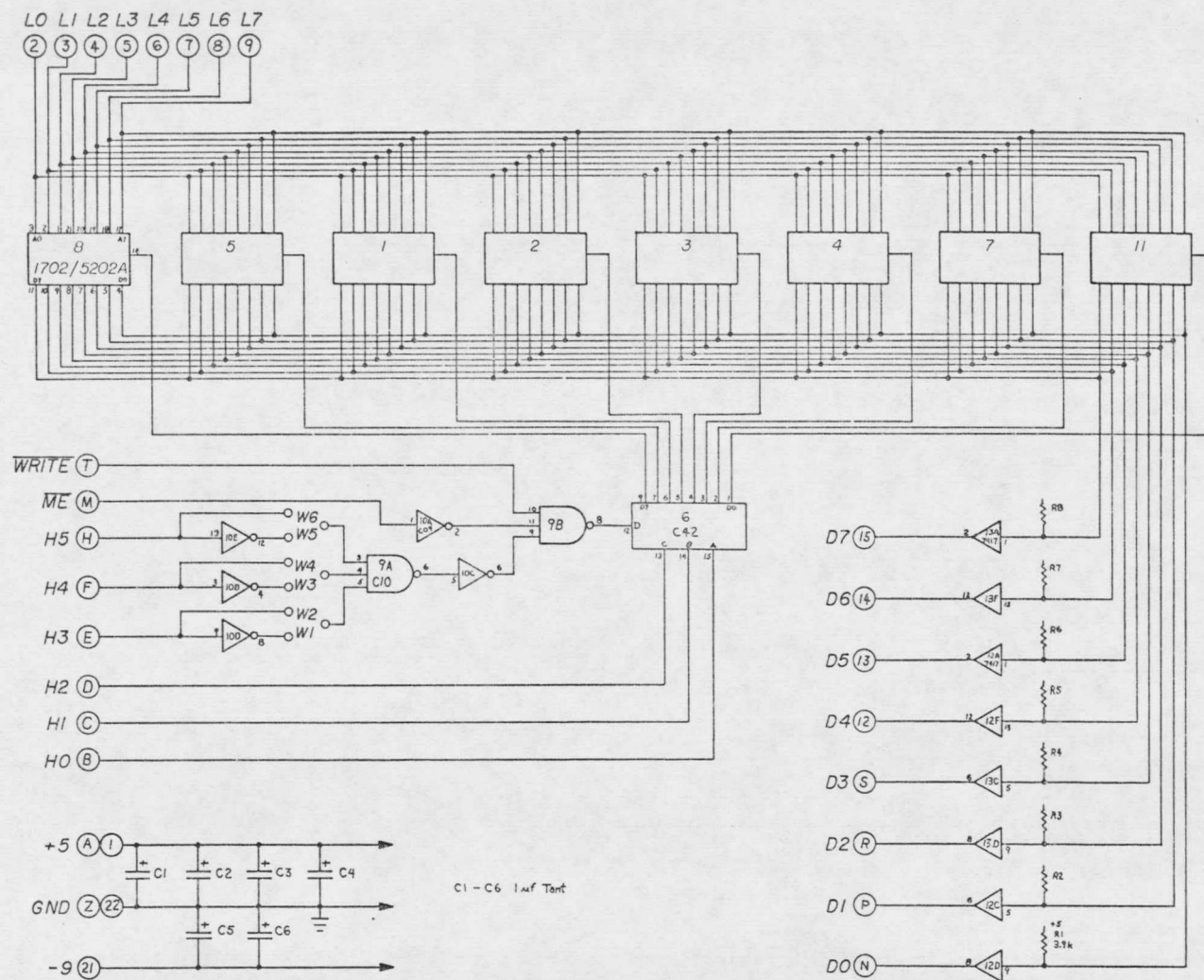


FIGURE 13: PROM SCHEMATIC

assumes a one state and the D bus is free for use by other parts of the system.

Only when a PROM chip receives a low chip enable ($\overline{CE}$) will its outputs assume the active state and place the contents of the data word at the location specified by the L bus on the inputs of the bus drivers. Any bit in this word which is a zero will be passed to the D bus through the 7417 drivers while those bits that are ones will leave their drivers off and that particular bit of the D bus will remain at a one state just as though the bus was not in use. In this manner the data word is passed to the D bus.

Which PROM chip receives the $\overline{CE}$ signal is determined by decoding the remaining 6 bits of address (H0-H5) along with the memory control signals $\overline{ME}$ and $\overline{WRITE}$. This decoding is performed by ICs 6, 9, and 10 in the following manner. The three lowest order bits of the H bus (H0, H1 and H2) are used as the A, B, and C inputs for a 74C42 BCD to Decimal Decoder (IC 6). The D input of this decoder is used as an enable input and must be low for the IC to decode outputs 0 through 7 thus generating the desired $\overline{CE}$. Gate 9B must have all ones on its input to generate the needed low signal for the D input. Thus in order to generate a chip enable, $\overline{WRITE}$ must be a one indicating no write cycle, $\overline{ME}$ must be a zero indicating the memory is being used, and the output of gate 10C must

be a one indicating the address is in the correct 2k range as determined by the jumpers on H3, H4, and H5.

PROGRAMMING OF PROM CHIPS

The programming procedure specified by the manufacturer of the particular memory chips used must be followed. In order to program the PROM chips they must be removed from the memory card. For this reason sockets are recommended for the PROM chips rather than soldering them into the circuit board. While the 1702 family contains several equivalent memories, the prototype used the I1702A chip. This memory may be programmed and then erased by exposure to UV light and then programmed again, a handy feature when debugging code to be placed in PROM. The MM5202AQ is functionally equivalent to the I1702A in the read mode but requires a different programming procedure. Also included in the 1702 family are the Intel 1602A National MM5202AD which contain the same programmable chip only mounted in an opaque package precluding erasure. These chips may be programmed only once. The final member of the 1702 family is the Intel 1302. This chip must be programmed by changes in the chip mask at the time of manufacture. While this memory will function in the PROM card, it is only practical if purchased in large groups with each PROM containing an identical program. Such PROMs would be used only in a production line environment and

and even then only after the program has been proven correct. Debugging of PROM programs is most easily accomplished by first running the program in RAM where mistakes are easily correctable and then transferring the code to PROM.

OPTIONS

The only options available on the PROM card are the jumper wires used to set addressing range and the possibility of not inserting the maximum number of memories. Table 5 lists the jumpers needed for the eight 2k address ranges. Table 6 shows the relative address of each PROM socket within the selected 2k range.

Addressing Range (Hex)	Jumpers
0'- 7FF' (0- 2k)	W1, W3, W5
800'- FFF' (2- 4k)	W2, W3, W5
1000'-17FF' (4- 6k)	W1, W4, W5
1800'-1FFF' (6- 8k)	W2, W4, W5
2000'-27FF' (8-10k)	W1, W3, W6
2800'-2FFF' (10-12k)	W2, W3, W6
3000'-37FF' (12-14k)	W1, W4, W6
3800'-3FFF' (14-16k)	W2, W4, W6

TABLE 5: PROM ADDRESS RANGE SELECTION JUMPERS

Relative Address	Socket Number
0'-FF'	11
100'-1FF'	7
200'-2FF'	4
300'-3FF'	3
400'-4FF'	2
500'-5FF'	1
600'-6FF'	5
700'-7FF'	8

TABLE 6: RELATIVE PROM SOCKET ADDRESS

RANDOM ACCESS MEMORY

The Random Access Memory provides a read and write memory array that may be used as program store or for scratch and buffer store as needed. The RAM card is compatible with the PROM and interfaces to the same bus connections. RAM and PROM may be mixed in any possible combination but must not overlap.

Each RAM card accommodates up to 32 1024 bit fully decoded static RAM chips from the 2102 family (Intel 2102, -1, -2 or National Semiconductor MM2102). Any version of the chip is usable,

but the speed of the memory will vary with the particular chip used. Also included are a 14-bit address decoder, bidirectional bus driver/receiver and write protect circuitry for individual 1k word banks. Using common and inexpensive static RAMs from the 2102 family allows a simple memory design that is expandable in 1k increments to a maximum of 4k per card. By partially loading the card with memory chips, 1, 2 and 3 k cards can be constructed. Multiple cards are used for systems larger than 4k.

While the system is realized in CMOS and LPTTL, the design will accommodate standard TTL to provide a general purpose RAM system that might find use anywhere such a memory is needed.

RAM OPERATION

Figure 14 presents the schematic of the 4k RAM card. Bus connections include 14 bits of address (H & L bus), 8 bits of bidirectional data port (D bus) and the three memory control signals $\overline{ME}$, $\overline{WRT}$ and $\overline{WRITE}$. Five additional inputs allow individual write protection on each of the four 1k banks ($\overline{PROTECT A}$, Etc.) and $\overline{INHIBIT}$ input which may be used to prevent the memory from transmitting to the D bus. Each RAM card represents only a single CMOS load to each bus line and the card requires only a single 5 volt power supply.

The memory array is arranged as four rows or banks each

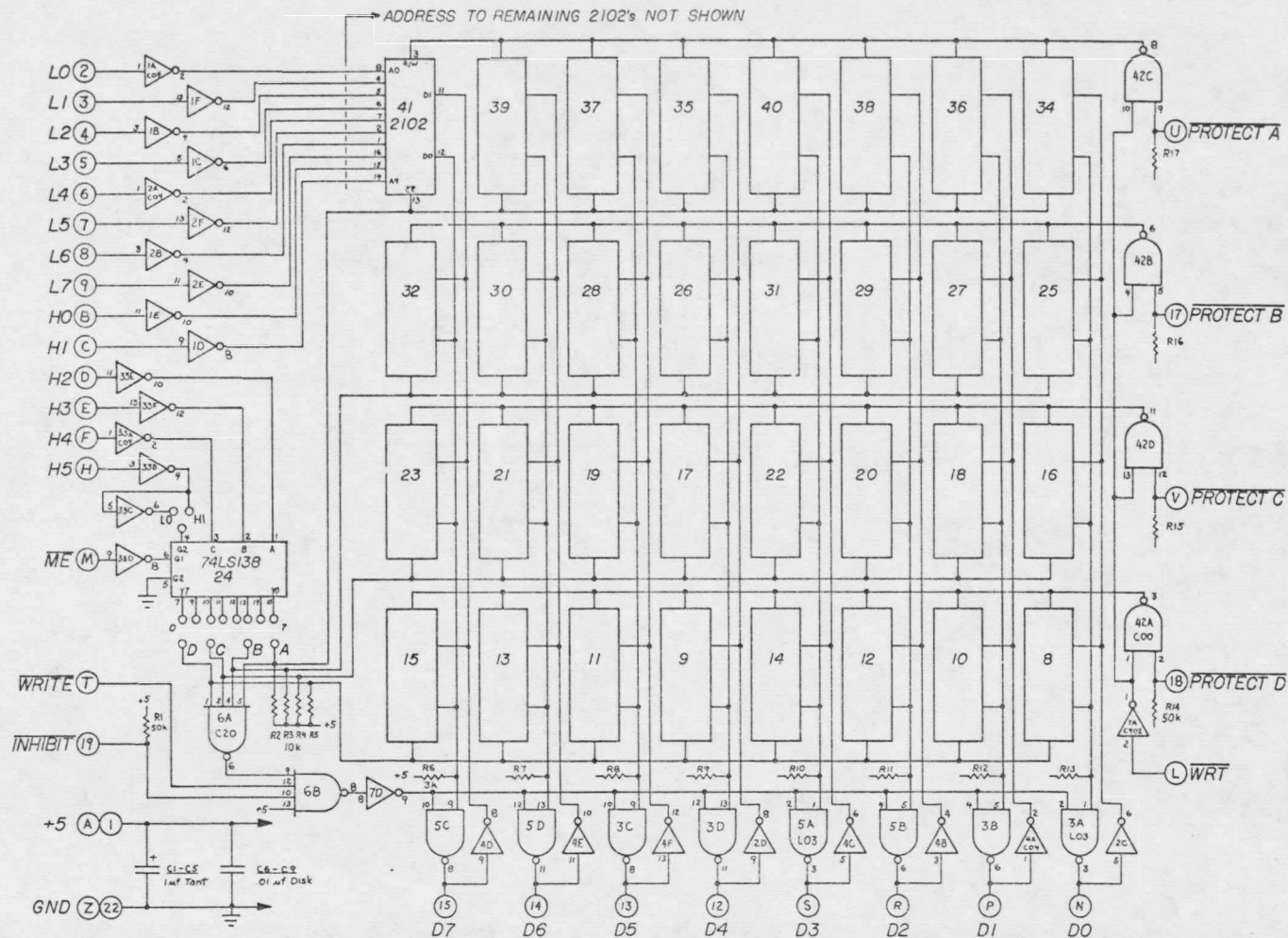


FIGURE 14: RAM SCHEMATIC

composed of eight 2102 RAMs. The card holds up to four of these 1k word banks allowing expansion in 1k increments. The ten least significant address bits (H1-L0) are bussed to every 2102 on the card through the 74C04 address buffers (ICs 1 and 2). $\overline{ME}$ and the remaining bits of address are decoded by a 74LS138 (IC 24) onto one of eight output lines. The individual chip enable lines from each block may be jumpered to the desired 1k range. A jumper on the H5 bit allows operation of the decoder in either the high or low half of the 16k address range. Blocks may be assigned within this 8k range as desired.

The data is passed in and out of the memory array through the bidirectional transceiver consisting of an open collector NAND gate (74L03) and a CMOS inverter (74C04). The data input pins for each 2102 in a corresponding bit location (one from each bank) are connected to the D bus through the inverter. Thus write data is inverted. In a similar manner the 2102 tri-state outputs are connected together to the open collector NAND gate and then to the D bus. The remaining inputs from each NAND gate are connected together to form a control line for the transmitter. The transceiver transmits only when a bank chip enable is low on gate 6A and both $\overline{INHIBIT}$ and $\overline{WRITE}$ are at a high level on gate 6B. Gate 7D provides the enable signal with enough fan out to drive the

eight NAND gates.

RAM READ OPERATION

Read operation is accomplished by establishing an address and then a $\overline{ME}$ signal. $\overline{WRT}$, $\overline{INHIBIT}$, and $\overline{WRITE}$ must all be high to allow the addressed word to be gated onto the D bus. Note that data is stored in the RAMs in inverted form. The static 2102 RAMs allow variable memory timing and data will be present on the D bus as long as $\overline{ME}$ is present.

The read cycle is nondestructive, thus data need not be re-written after a read. If a nonexistent location not decoded to a bank chip enable is read, then a data word of FF^t results because of the open D bus and its pullup resistors. Reading a block that is decoded (jumper inserted to chip enable) but which contains no 2102 RAM chips will cause a data word of 00^t to be transmitted to the bus. The contents of the RAM is volatile and will be destroyed if the power is removed from the memory.

RAM WRITE OPERATION

In order to write, a similar procedure is used. An address is applied and $\overline{ME}$ generated. During the time $\overline{ME}$ is low a $\overline{WRT}$ pulse must be present. Data is stored into the address location on the rising edge of $\overline{WRT}$. Either $\overline{INHIBIT}$ or $\overline{WRITE}$ should be a

zero during the write cycle to prevent the memory from starting to read the old data to the bus. Individual blocks may be protected from writing by grounding (logic 0) the appropriate protect input. When protected, the write cycle appears to the memory as an inhibited read and no data is changed. The RAM write protect feature would most likely be implemented by connecting the protect inputs from the desired RAM banks together and then through a switch to ground. Closing this switch will prevent the memory from writing.

CHAPTER 6

SYSTEM I/O

The I/O portion of the machine accommodates a selection of up to eight I/O interface cards. Any combination of serial or parallel data interfaces, or real time clocks may be used to form the I/O chain. As mentioned earlier, interrupt priority is dependent on position while port address is decoded by the interface and does not change if cards are moved relative to one another in the I/O bus.

Each I/O interface is composed of two major parts; data and control logic and interrupt logic. The function performed by the data and control portion varies with the card. For instance, the Parallel I/O (PIO) interface contains two 8-bit registers to buffer data in and out, while the Serial I/O (SIO) interface used a single MOS LSI chip that performs the complete asynchronous receive/transmit function. In the Real Time Clock (RTC) the data handling function has been replaced by a timing circuit that periodically generates interrupts at a program controllable rate.

The interrupt logic remains identical in function on all cards. In the case of a data I/O port (SIO or PIO) each interface contains two sets of interrupt logic, one for the Input port and one for Output. On these cards Input is always higher priority than Output. The RTC requires only a single level of interrupt

logic since it communicates with the processor only through interrupts.

INTERRUPT AND PRIORITY LOGIC

Figure 15 shows a schematic for a typical single level of interrupt logic. Although the actual schematic of the interrupt logic on each type of I/O interface may differ slightly from Figure 15, the operation is identical. In discussing each interface in particular, the interrupt logic will be skipped and instead explained below. The individual discussion will however include the particular control word formats used for the interface.

The interrupt logic has several bus connections (shown on the left) and connections to the rest of the interface (shown on the right). Both Initialize (INIT) and Interrupt Enable (INT ENB) are control bits from the interface's control port. Request ($\overline{\text{REQ}}$) is used to call for an interrupt. It usually signifies that new data is ready or old has been taken. GATE is used to jam the interrupt instruction on to the D bus. Generally this is a RESTART instruction used to call the interrupt service routine.

Assume that the interrupt logic is in the Initialized state. This is caused by a power-on generated $\overline{\text{CLR}}$ signal (or any $\overline{\text{CLR}}$) or by causing the Initialize one-shot (IC 10) to fire by the appropriate control. Any of these events will cause the four flip flops

BUS CONNECTIONS

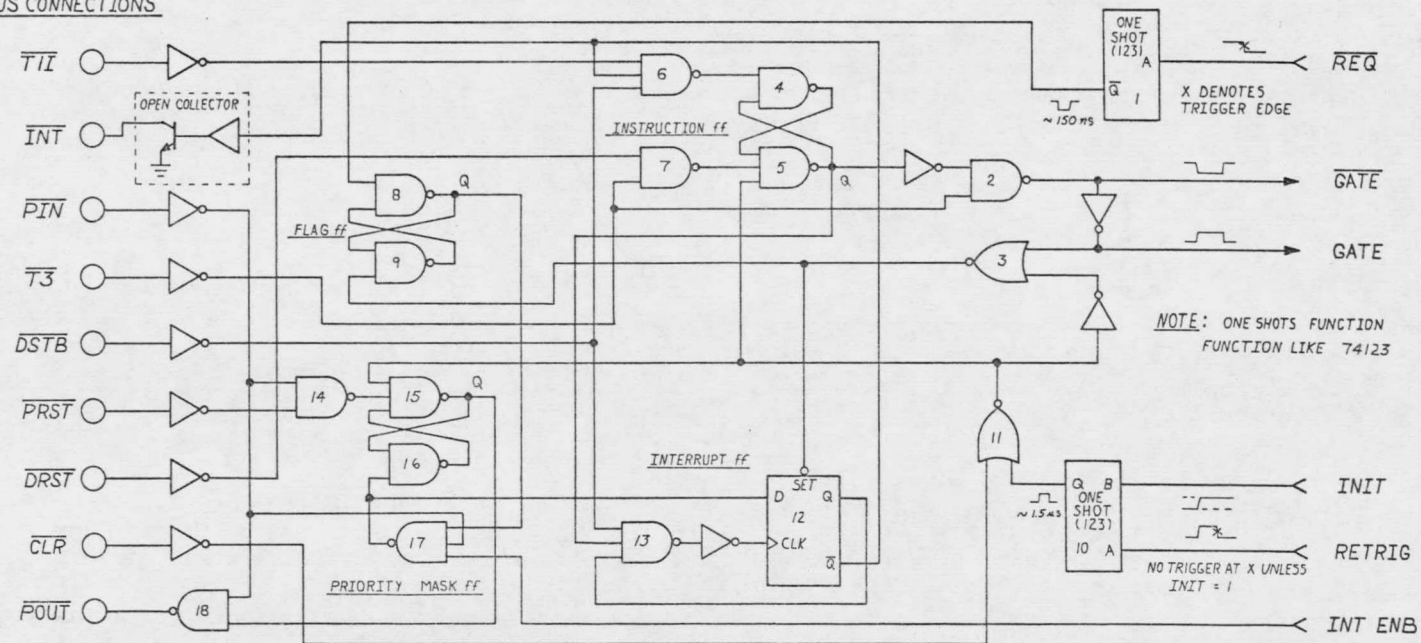


FIGURE 15: SINGLE LEVEL OF INTERRUPT AND PRIORITY CONTROL LOGIC

in the interrupt logic to be initialized. Mask, Interrupt, and Instruction will be set ($Q=1$) while FLAG will be cleared ($Q=0$).

If an interrupt request is generated by the data logic (negative edge on $\overline{REQ}$) then the request one-shot (IC 1) will pulse and set the Flag flip flop. When Flag is set and interrupts are enabled ($INT\ ENB = 1$) and the logic has a true Priority In signal ($\overline{PIN} = 0$) then Mask is cleared asynchronously through gate 17. This causes Priority Out ($\overline{POUT} = 1$) to go false through gate 18 and inhibits any interrupts from lower priority I/O interfaces.

The same signal that clears Mask is also used as data on the D input of the Interrupt flip flop (IC 12). On the leading edge (positive) of the next DSTB, the Interrupt flip flop is clocked to a zero generating an interrupt to the processor through the open collector driver. Interrupt latches in this state.

When the Interrupt signal ($\overline{INT}$) reaches the processor it clears the Master priority flip flop ($\overline{PIN} = 1$) and inhibits any higher priority interrupts. Thus the processor is forced to respond to only the active interrupt logic.

At some later time when the processor recognizes the interrupt, the Instruction flip flop is cleared by DSTB and T1I (gate 6). This generates the enable needed to generate the $\overline{GATE}$ signal through gate 2. $\overline{GATE}$ lasts from the start to T3 until DRST begins at which

time Instruction is set again ($Q = 1$). Signal $\overline{\text{GATE}}$ is used to clear Flag and set Interrupt and all flip flops except Mask are now back in the initial state and one interrupt was generated.

The jamming of the single byte interrupt instruction by the GATE signal causes the processor to begin to execute the service routine associated with the port. This instruction is normally a RESTART. Once the current machine status (registers and flags) has been saved, then the interrupt routine would execute PON (OUT 25B) and reset the Master Priority Control flip flop in the processor. Another interrupt from the same or higher priority port may now occur, but the set Priority Mask still inhibits lower priority interrupts.

Once the service routine has accepted the required number of interrupts from the port the program would then execute PRST (OUT 27B) after executing PON. This will clear the highest priority active Mask flip flop thus enabling lower priority interrupts to occur.

Several things should be mentioned about the response of the Interrupt and Priority Control logic to the control word. The logic is designed to accept logic levels from the control word latch. In data I/O interfaces which contain two levels of Interrupt and Priority Control logic, separate bits of the control word

enable input or output interrupts independently. However the Initialize function is common to both levels. The RETRIG signal, which is also the clock that stores the control word in its latch, allows the Initialize function to be sensitive only to the level of its control bit (INIT). The control word is clocked into its latch on the leading edge of RETRIG and the Initialize one-shot will pulse on the trailing edge of RETRIG only if the INIT bit is a one. If a Control Output changes the bit to a zero, no initialization pulse will occur.

8 BIT PARALLEL I/O INTERFACE

The data portion of the PIO interface performs a bidirectional transfer of parallel 8-bit data words between the computer and an external device that is connected to the cards I/O connector. The PIO provides two 8-bit data ports with input and output storage registers (latches) and two data flags to synchronize data transfers. Through the use of the control port, interrupts may be turned on and off, the state of the data flags and interrupt logic may be initialized, or Status mode may be selected causing the next input addressed to the board to return a status word instead of the regular data word.

CONTROL AND STATUS WORDS

Table 7 shows the formats for the control word and status word. The four unused control bits are available on the I/O connector (External Control Output) and may be used to control the external device connected to the I/O connector on the interface. When Status is used the data word read by the processor contains the two data flags and six bits of External Sense. These Inputs are also found on the I/O connector.

DATA TRANSFER SEQUENCES

An external device may enter data into the PIO board only when the input register is free (empty). This is indicated by the state of the Input Flag. If this flag shows a zero (on the I/O connector or in the Status word) data may be entered by a positive pulse on the Set Input Flag input on the I/O connector. If the flag is in the one state then the interface will not accept data. The Input Flag is used in the following manner; it is set by the external device to enter data into the board and cleared by the processor when the data is input. A $\overline{\text{CLR}}$ signal from the bus or the cards Initialize function will also clear the flag. Figure 16 shows a typical set of waveforms involving one transfer of input data.

A similar procedure occurs when data is output. When an

Bit								Bit
7	6	5	4	3	2	1	0	

CONTROL WORD

BIT 0 INITIALIZE
 BIT 1 INPUT INTERRUPT ENABLE
 BIT 2 OUTPUT INTERRUPT ENABLE
 BIT 3 STATUS ENABLE

BIT 4
 to EXTERNAL CONTROL OUTPUTS (EC4 to EC7)
 BIT 7

NOTE: All control bits are positive true. Outputting a control word with a bit set to one enables a control function or causes an External Control output to assume a logic one state

EXAMPLE:

1	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

I/O board Initialized and Input Interrupts enabled. EC7 is also true.

Bit								Bit
7	6	5	4	3	2	1	0	

STATUS WORD

BIT 0 OUTPUT FLAG
 BIT 4 INPUT FLAG

All remaining bits are External Sense. A logic one (+5) on an External Sense input will show a one in the corresponding bit position of the Status word.

EXAMPLE:

0	0	0	1	0	1	0	1
---	---	---	---	---	---	---	---

New data is ready but not yet input as shown by the set Input Data Ready Flag. Output data is still waiting to be accepted (Output Ready Flag = 1). One External Sense input (ES2) is also true.

TABLE 7: PIO CONTROL AND STATUS WORD FORMATS

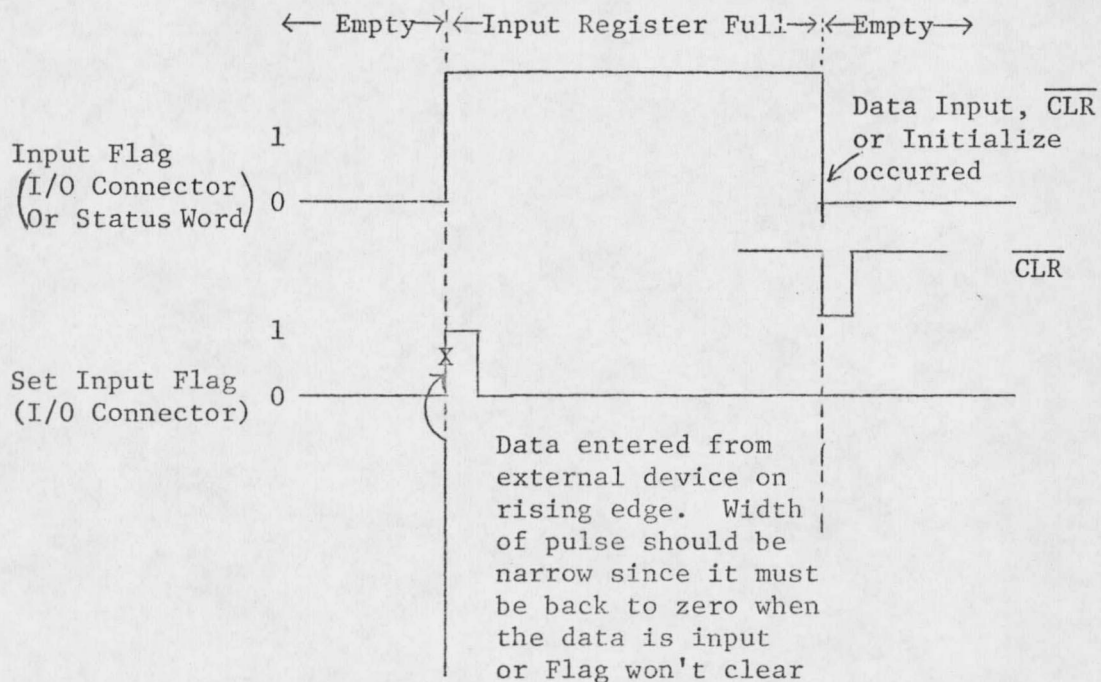


FIGURE 16: PIO DATA INPUT SEQUENCE

Output instruction loads data into the output data latch on the PIO card the Output Flag is also set. This causes the flag to show a one on the I/O connector and in the Status word. When the external device accepts the data it must clear the Output Flag with a short negative pulse on the Clear Output Flag input on the I/O connector. The contents of the output data latch may always be changed by an output instruction even if the flag has not been cleared. Figure 17 illustrates a typical data output sequence. Again $\overline{\text{CLR}}$ or Initialize will clear the flag and $\overline{\text{CLR}}$ will

additionally cause the contents of all PIO board registers to be cleared. This makes all data zero, inhibits all control functions and forces External Controls to zero.

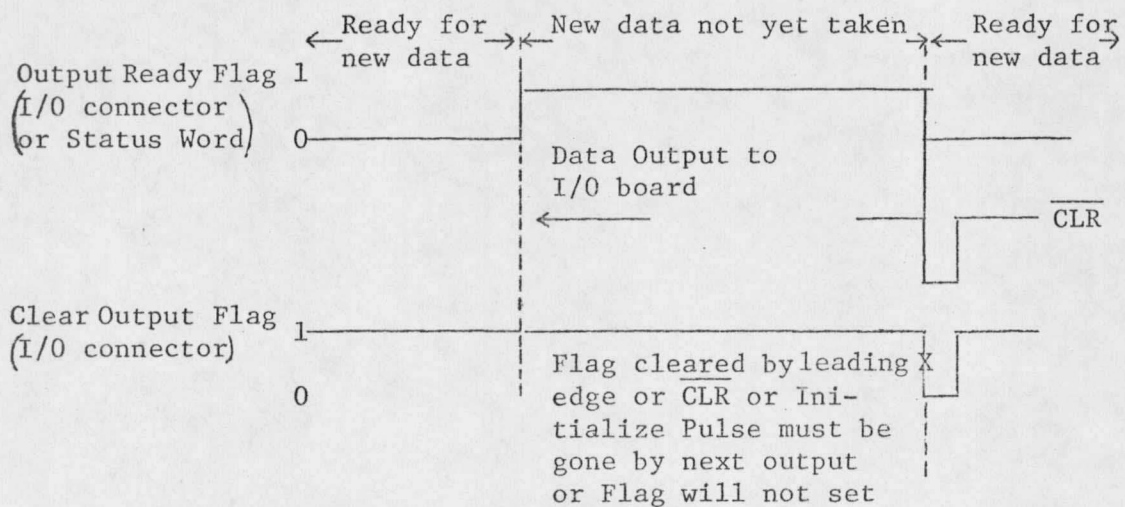


FIGURE 17: PIO DATA OUTPUT SEQUENCE

TRIGGERING INTERRUPT DATA TRANSFERS

When the interrupts are enabled through the control word, the respective data flag (Input or Output) is used to generate an interrupt request for the interrupt logic. If input interrupts are enabled, a request is generated when the Input Flag is set by the external device. The Output Flag generates a request when cleared only if its interrupts are enabled. In this manner the processor may be notified when output data has been accepted (Output Flag cleared) and the next byte may be output to the PIO or that new

data is ready on the input data latch (Input Flag set).

Interrupts and Status make two methods of I/O possible; either the state of the flags may be sensed by the program through the Status input or interrupts may be used to notify the machine that I/O needs attention.

PIO DATA HARDWARE

Figure 18 is the schematic for the 8-bit parallel I/O board. The two levels of interrupt logic are shown on the upper left of the drawing. Data logic and associated controls are on the right and below the interrupt logic.

ICs 30 and 34 are the output data latch while ICs 29 and 33 form the control word latch. The inputs to data and control latches are fed from the L bus through the buffers in ICs 28 and 32. The outputs of the output data latch go to the I/O edge connector as do the four high order bits of the control latch (External Control). The outputs of the four low order bits serve as the control signals to the interrupt logic and status flip flop.

ICs 17 and 18 form the input data latch with data inputs connected to the I/O connector. The outputs of the input data latch feed one set of inputs on an eight bit, two to one multiplexer (ICs 9 and 10). The other set of inputs are connected to the two data

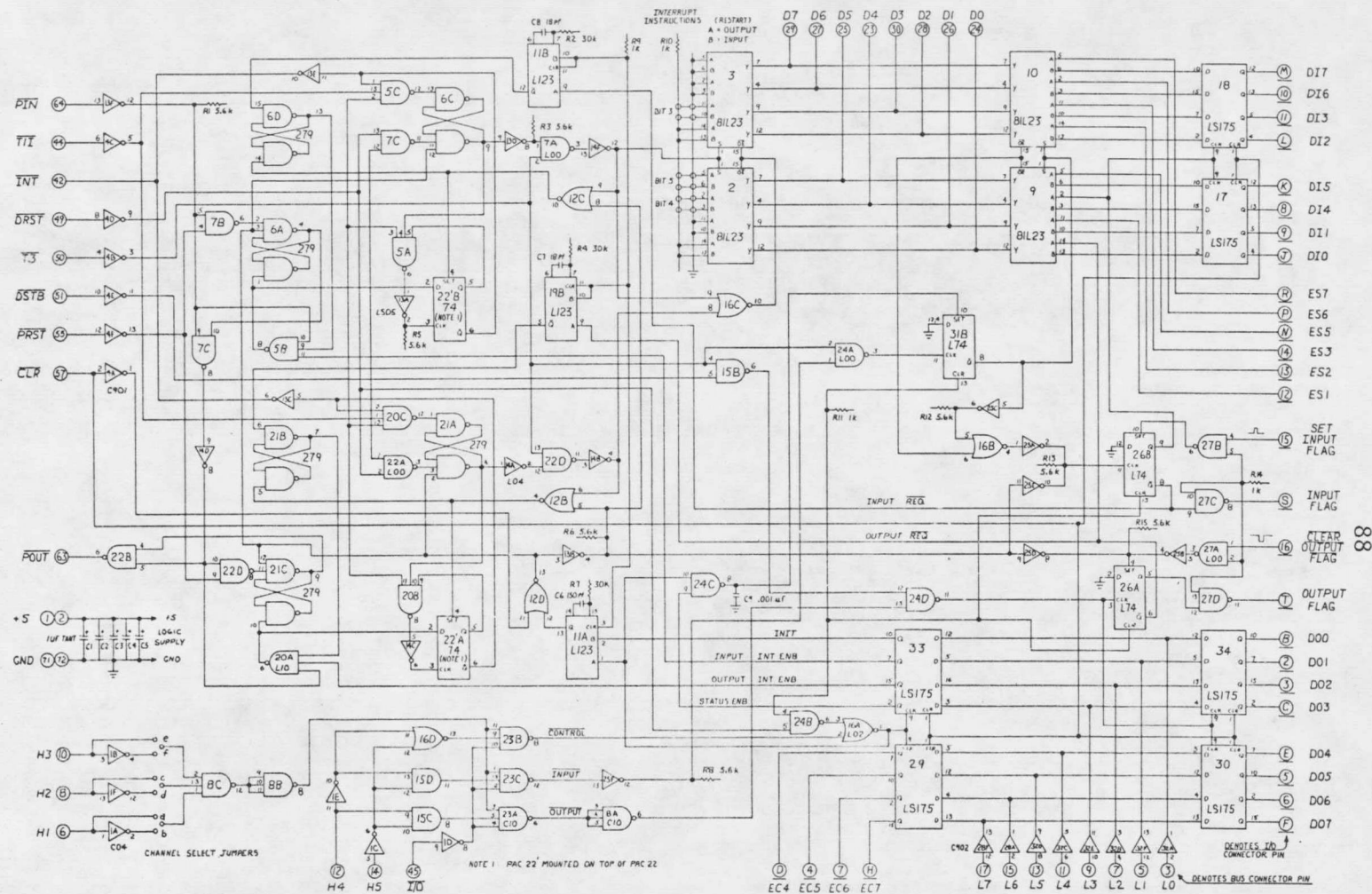


FIGURE 18: PIO INTERFACE SCHEMATIC

flags and six External Sense bits. This multiplexer has tri-state outputs and drives the D bus directly with either input data or status data as selected by the status flip flop.

IC 26 contains the two data flag flip flops. The Output Flag (IC 26A) is clocked to zero (flag is set) when the output data latch is loaded by an output instruction. Either the Clear Output Flag input, $\overline{\text{CLR}}$ or the Initialize function may preset (flag is cleared) this flip flop. Open collector gates 25B and D form a NOR gate to accomplish this function. Gates 27A and D are used as inverting buffers between the flag and I/O connector.

Flip flop 26B functions as the Input Flag. It may be cleared (through the clock input) by either $\overline{\text{CLR}}$, Initialize or the output enable signal for the data/status multiplexer when data is input. Status inputs do not affect the flag in any manner. The three open collector gates from IC 25 (25A, C and E) and gate 16B drive the clock input and act to clear this flag. Again gates 27B and 27C serve as buffers between the flip flop and the I/O connector.

An identical multiplexer (ICs 2 and 3) to the data/status multiplexer functions as the source for the interrupt instructions used by the interrupt logic. The inputs for this multiplexer are a series of connections to a logic one or zero (jumper selected) that form the OP code for the RESTART instructions that point the

processor to the correct service routine. The two GATE signals from the interrupt logic are used to select and enable the multiplexer through gate 16C.

Gates from ICs 1, 8, 15, 16, 23, and 25 from the combinational logic that decodes the three port addresses used by the PIO card (Input, Output and Control). H1, H2 and H3 are jumper selected to set the channel address (see Table 8 below) while H4 and H5 determine if Control, Output or Input is involved. Whenever H1 to H5 contain the correct address during an I/O cycle CONTROL, OUTPUT or INPUT will be generated on the output of the network.

Channel Address (H3,H2,H1)	Jumpers
0	f,d,b
1	f,d,a
2	f,c,b
3	f,c,a
4	e,d,b
5	e,d,a
6	e,c,b
7	e,c,a

Port Addresses within a channel

Out Control = 3 X B

Out Data = 1 X B

Input Data = X B

Where X = Channel Address

TABLE 8: PIO CHANNEL SELECT JUMPERS

When an OUTPUT occurs it is inverted by gate 8A and combined with DSTB during T3 (from gate 15B) by gate 24D. The output of this gate clocks the data from the L bus into the output data latch and sets the Output Flag. Both actions occur at the end of

the DSTB period.

If an INPUT occurs a similar, but wider pulse is generated on the output of gate 24A. This pulse (occurring coincidentally with T3) is used as the enable signal for the Data/Status multiplexer enabling it to drive the D bus. The trailing edge of the pulse is used to clear the Input Flag (when data is being input) or reset the status flip flop to the data mode (when status was input).

Executing an output to the control port on the I/O card causes CONTROL to be generated. This signal is combined with DRST in gate 16A to generate the clock signal for the control latch. This is also the RETRIG signal used by the initialize circuitry in the interrupt logic. The control word is stored in the latches on the leading edge of the clock. The trailing edge of the pulse will trigger the initialize one shot in the interrupt logic only if the initialize control bit is a one.

When both the status enable bit and the control latch clock are both a logic one then gate 24C acts to preset the status flip flop (IC 31) setting the status mode. There is a possibility that when a control output that changes the status bit from one to a zero is executed, an unwanted status mode may be set because of the delay in clocking the zero into the control latch. Capacitor C9 is included to prevent this from happening by slowing the

preset function on the status flip flop to the point where it is slower than the delay through the latch thus preventing this possible hazard from affecting the flip flop.

REAL TIME CLOCK

The RTC does not act to transfer data. Instead a timing or counting function is provided. By addressing the RTC's Data and Control ports a program may load a count word and control word. When the clock is activated by the count output, the current control word sets the mode and rate at which the count word is counted down to zero at which time an interrupt request is generated. Two modes of operation are possible. Single interrupt mode causes the clock to interrupt once and wait to be restarted while the multiple interrupt mode generates interrupts at a constant rate until further control.

The single level of interrupt and priority control logic on the RTC behave identically to that on the data interfaces. Two bits of the control word are devoted to the Interrupt Enable and Initialize functions. The six remaining control bits provide two bits of External Control and four bits of Mode/Rate control for the clock logic.

A crystal oscillator (4.096 mHz) and divider chain provide three time bases of .001, .1, and 10 seconds (1kHz, 10Hz and 1/10 Hz).

Thus if a count of ten and a time base of .1 sec were used, the interrupt would occur one second later. Any one of eight sources for the time base may be selected with the control word. Three come from the crystal clock, one is a zero frequency or stopped mode, while the remaining four are External Rates (ERA to ERC) that may be input on the cards I/O connector. These inputs allow an event counter mode. An interrupt may be generated when a programmed number of events have occurred on an External Rate input.

The I/O connector also provides outputs from other stages of the divider chain (on the crystal oscillator) and an output that is the current time base (clock) selected by the control word. If desired, the other outputs from the crystal clock may be wired to the External Rate inputs (ERA - ERD).

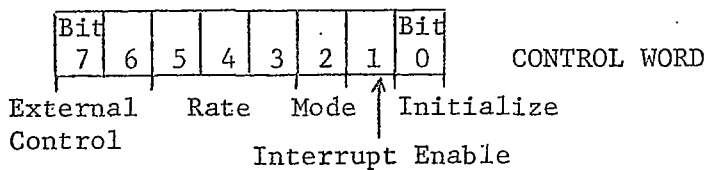
By varying the count and rate, interrupts may be generated in intervals ranging from 1 msec. to 2550 seconds (42 minutes, 40 seconds). These intervals would correspond to counts and rates of 1 and .001 seconds or 255 (FF') and 10 seconds respectively.

REAL TIME TIME CONTROL WORD

The control word format for the RTC is shown in Table 9. The two control bits for the interrupt logic (Interrupt Enable and Initialize) perform the normal function within the interrupt logic. Four bits set the Mode/Rate for the timer and two bits are External

Control.

As shown in the table, three bits are used to select one of eight sources for time base (rate). The remaining bit set the mode in which interrupts are generated.



Initialize and Interrupt Enable are low active.

Outputting a zero bit enables the function. The External Controls also invert. Outputting a zero causes a logic one voltage on the I/O connector.

Bit 2 selects the interrupt mode. When it is a one Multiple mode is selected. A zero selects the Single Interrupt mode.

The source for the time base rate is selected according to the codes below.

RATE CODE (Bits)			SELECTED TIME BASE
5	4	3	
0	0	0	Stopped
0	1	0	10 sec
1	0	0	.1 sec
1	1	0	.001 sec
0	0	1	ERD
0	1	1	ERC
1	0	1	ERB
1	1	1	ERA

TABLE 9: REAL TIME CLOCK CONTROL WORD

When the Single Interrupt mode is selected the clock is started by performing a Control output to set the rate and mode and then a Data output to set the count (in that sequence). A single interrupt will be generated after an interval (starting when the count is output) determined by the product of the count (binary value) and the time base rate. After the interrupt is generated, the clock resets to the stopped state and awaits further control (another count output).

If the Multiple Interrupt mode is selected the clock will generate interrupts at a constant rate determined by the count and rate until either the count data or control word is changed.

Whenever the clock is running and the control word is changed the clock stops and waits for new count data to be output. If the count data is changed while the clock is running, the count is immediately reset to the output value and the clock begins running at the Mode/Rate selected by the control word last output to the board.

If a control word is output to the clock when stopped, the two External Control bits and the Interrupt Enable and Initialize bits become effective when output. The Mode/Rate bits are not active until the clock is started by the next count output.

Like the data I/O interfaces, the state of the RTC is affected

by $\overline{\text{CLR}}$ and the Initialize function. When a control output causes initialization, the interrupt logic is reset. The timing logic is not affected in any manner. When a $\overline{\text{CLR}}$ occurs the clock is immediately stopped and cleared. The counter and the control word are reset to all zeros. Thus interrupts are off and External Control outputs are low (logic zero). The clock is stopped and further control and count outputs must be performed to restart the timer.

REAL TIME CLOCK TIMING LOGIC

Figure 19 illustrates the schematic for the RTC timing logic. The single level of interrupt logic is on the lower left of the drawing. Connections to the computer I/O bus are shown as circles while underlined circles denote pins on the I/O connector.

The time base is driven by a crystal oscillator and buffer composed of inverters 1C, D and E. The trimmer capacitor (C1) allows adjustment of the oscillator frequency to 4.096 MHz.

A series of jumper wires allow the output of the oscillator to be connected to the input of the frequency divider chain (IC2, pin 14). Gate 10D may be used to buffer the 4.096 MHz signal out to the I/O connector (pin H) or by reversing the jumpers pin H may function as a buffered external oscillator input.

The higher frequency half of the divider chain is composed of

three modulo 16 counters (ICs 2, 3, and 7) which reduce the frequency to 1 kHz. An intermediate frequency of 256 kHz may be connected through gate 1B to I/O pin B with a jumper if desired. A 16 kHz output is also available on pin 8 of the I/O connector.

The 1 kHz signal forms the 1 msec time base. This signal is further divided by 10 twice to form the .1 second time base by ICs 8 and 9. Two more divide by 10 stages (ICs 5 and 4) generate the 10 second time base. Intermediate frequencies of 100 kHz or 1 Hz may be connected to I/O pin B instead of 256 kHz by using the appropriate jumper as shown on the schematic.

A reset pin from each of the counters in the time base is connected to a master reset line driven by gate 1A. A high logic level on this line resets all the counters. This occurs whenever the down counter is loaded. Clearing the counters each time a new timing interval is initiated reduces the jitter on the first cycle of the time base to one cycle of the 4.096 MHz oscillator (244 ns).

One of eight sources for the clock (time base) signal is selected by the three rate bits in the control word. IC11, a dual four to one multiplexer, feeds a two to one multiplexer composed of two open collector NAND gates (16A and 16B) with a wire AND connection on the output. Bits 4 and 5 of the control word establish a select code for the two sections of IC11.

Bit 3 (and its compliment) are used to enable one section or the other and the respective NAND gate. Circuit operation is equivalent to an inverting eight to one multiplexer. The External Rate inputs are isolated from the multiplexer by the invertors in IC10 while the selected clock rate is output through gate 6D to the I/O connector. This is the Active Clock output. The counters clock on the negative edge of the signal present on the Active Clock output and on the positive edge of an External Rate signal.

ICs 13 and 12 comprise the 8-bit down counter that performs the timing function. The borrow output from IC12 generates an Interrupt Request by going low (logic 0) when the counter reaches zero. The clock for the counter comes from the output of the rate multiplexer and is coupled to the counters through C3 and R4. This resistor and capacitor serve to differentiate the rate multiplexer output producing a clock signal that is a narrow, positive pulse occurring when the multiplexer output goes high. The diode (D1) serves to clip the negative pulse that would normally result when the multiplexer output goes low. Since the counters themselves clock on the positive edge of the clock and generate the borrow signal on the negative edge of the clock cycle that set the counters to zero, this RC differentiator prevents each timing cycle from being one half time base period longer than was intended.

The clear function for the two counters are connected to the $\overline{\text{CLR}}$ signal from the bus. Thus whenever a $\overline{\text{CLR}}$ occurs, the counters are immediately reset to zero. A negative pulse on the load inputs will cause the outputs of the counters to agree with the data present on the counter's set inputs. This data comes from the outputs of the count latch (ICs 14 and 15) while the load pulse is generated by either gate 16C or 16D. The former loads the counters when an interrupt is requested in the Multiple Interrupt mode while gate 16D generates the load pulse whenever new count data is output.

The port address decoding logic on the RTC functions similarly to that used on the data I/O interfaces. However since the RTC uses no input port, only Output and Control are decoded. The corresponding input port is not used and is left undecoded. The I/O cycle signal and H1 through H5 are decoded to generate $\overline{\text{OUTPUT}}$ and $\overline{\text{CONTROL}}$ signals which are used to enable loading of the respective data storage latch. The port address for the RTC is set by three jumpers. Jumpers and channel addresses are shown in Table 10.

ICs 18 and 19 form the control word latch. Whenever a control output addressed to the RTC is executed, the $\overline{\text{CONTROL}}$ signal enables gate 24D. The DRST occurring during T3 is then used as the clock for the control latch (also the RETRIG signal for the interrupt logic) to store the control word output to the clock. Data to the

CHANNEL ADDRESS (H3,H2,H1)	JUMPERS
0	E,C,A
1	E,C,B
2	E,D,A
3	E,D,B
4	F,C,A
5	F,C,B
6	F,D,A
7	F,D,B

CONTROL PORT OUT 3XB
COUNT DATA PORT OUT 1XB

Where X denotes channel address

TABLE 10: REAL TIME CLOCK CHANNEL SELECT JUMPERS

control latch (and also the count latch) is buffered from the L bus through the inverters from ICs 25 and 26. The contents of the control latch will be set to zero by the occurrence of a $\overline{\text{CLR}}$ signal. This causes the clock to assume the stopped state and further control and count outputs must be performed to restart the timer.

ICs 14 and 15 form the count latch which is used to store the count value for reuse in the Multiple Interrupt mode. This storage register behaves similarly to the control latch except its clock is generated by gate 24C as a combination of $\overline{\text{OUTPUT}}$ and DRST

occurring during T3. Whenever $\overline{\text{CLR}}$ occurs the contents of the count register are destroyed and set to zero.

As mentioned earlier, whenever new count data is output to the clock the new count word is loaded immediately into the counters and the clock starts operation using the control word stored in the control latch. The load pulse is generated by gate 16D and 34A during an $\overline{\text{OUTPUT}}$. $\overline{\text{DRST}}$ is used as the load signal for the counters.

IC 17 forms a four bit register used to delay the effect of the control word until count data is output. This register receives the Mode/Rate bits from the control latch as inputs and the same signal that loads the counters also clocks this latch. The outputs drive the rate multiplexer (rate select bits) and the single or multiple interrupt logic.

When Single Interrupt mode is selected the signal from the interrupt logic that sets the Flag flip flop also acts to clear the contents of this register through gates 20A and B. In the Multiple Interrupt mode clearing is inhibited allowing the clock to begin another timing cycle using the same rate and mode. Whenever $\overline{\text{CLR}}$ occurs the contents of this Mode/Rate delay latch are reset to zero along with the contents of the control latch.

The NAND gates from ICs 32 and 33 are used to gate the interrupt instruction to the bus when the interrupt logic interrupts

the processor. These gates are open collector types so when no enable signal (GATE) from the interrupt logic is present the D bus is left free for other parts of the system. Note that the interrupt instruction is inverted by the NAND gates. Thus it should be selected in inverted form by the instruction pin throughs.

SERIAL DATA INTERFACE

This section introduces a partial design for a full duplex serial data interface. Figure 20 presents a schematic for the proposed SIO card. Major portions of the circuit are represented by blocks. These correspond either to portions of the logic that function in an identical manner to the other data interface (PIO), or portions whose details can not be specified until the actual application of the card is known. The clock and Input/Output level interfacing logic fall into the latter category.

As do all I/O interfaces so far described, the SIO interface contains two levels of Interrupt and Priority Control logic, a port address decoder and multiplexers and bus drivers for the two interrupt instructions and the data or status words.

The interface is controlled by the use of the control port just as in the PIO interface. The same control bits are used for Interrupt Enable, Initialize, and Status mode.

The data portion of the SIO interface is based on a Texas

Instruments Universal Asynchronous Receiver/Transmitter (UART) chip (TMS 6011). This 40 pin IC performs the serial to parallel and parallel to serial conversion. It also creates the needed start and stop bits and may also generate and check even or odd parity if desired. Word lengths may be set at 5, 6, 7 or 8 bits plus parity (if selected). Shorter word formats are composed of the lower order bits from the system's eight bit word. In these cases, the higher order bits are not transmitted or received. Received data is checked for three error conditions by the receiver (see status word format). Both the receiver and transmitter may operate at the same time.

UART MODE SELECTION

The desired mode is selected with a series of five programming jumpers (MODE SELECT on the schematic) as shown in Table 11. With the jumper connected, the input is a zero. Both the transmitter and receiver operate in the same mode. Serial data format is also shown in Table 11 (positive logic).

BAUD RATE SELECTION

The TMS 6011 UART requires a clock whose frequency is 16 times the desired baud rate. Thus to operate at 110 baud, a clock frequency of 1760 Hz would be required. The UART can accept a clock

WLS2	WLS1	WORD LENGTH
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

PI: PARITY INHIBIT

When set to 0 the UART generates parity
When set to 1 no parity is generated
or checked.

PS: PARITY SELECT (active when parity is enabled)

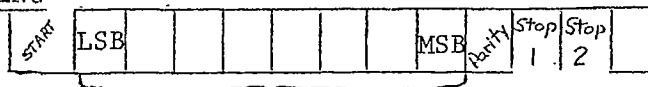
When set to 0 Parity is odd
When set to 1 Parity is even

SBS: STOP BIT SELECT

When set to 0, 1 stop bit is generated
When set to 1, 2 stop bits are generated

NOTE: No pullup resistors are required since the UART contains an internal resistor to Vss (approximately 5k) on each input.

MARKING



Marking or Next
Start Bit

Data Bits
(8 shown)

TABLE 11: UART MODE SELECT CODES AND SERIAL DATA FORMAT

in the range of 0 to 200 kHz. This corresponds to baud rates of 0 to 12,500 baud. The receiver and transmitter may operate independently at different rates within this range. If both receiver and transmitter are to operate at the same rate, then both clock pins on the UART may be connected together and only a single oscillator would be required.

SERIAL DATA LINES

No specific circuitry is shown in Figure 20 for the serial data lines. This logic would depend upon the particular output format chosen. Some common forms that might be used would be current loop or perhaps the EIA (RS232 specifications) format using positive and negative voltages for zero and one levels respectively.

In either case, when the transmitter is not transmitting, the output line will remain in the marking state (logic 1 level) and similarly, when data is not being received, the input line must be held in the marking state.

SIO CONTROL LOGIC

In the SIO, a UART replaces the data registers and flags of the parallel interface. The remaining logic functions in an identical manner to the PIO card.

One of two pin through selected interrupt instructions is

jammed onto the D bus by the active level of interrupt logic. Again input is higher priority than output.

The channel decoder decodes the Input, Output and Control ports used by the card. Channel address is selected with jumpers as in the PIO.

Outputting in the control port loads the control word register in the normal manner. A $\overline{\text{CLR}}$ signal from the bus will cancel the control word inactivating the serial interface.

Status mode operates identically to the PIO interface. Outputting a control word with this bit enabled (see Status word format) causes the next input to return the Status word instead of data. No data flags are reset.

SIO CONTROL AND STATUS WORDS

Table 12 documents the control and status words for the SIO card. The control word is identical to that for the PIO. The status word is different from that returned by the parallel interface. It contains three data flags, three error condition bits and two External Sense bits.

OUTPUT DATA TRANSFER SEQUENCE

Assuming the SIO is not transmitting, both the TBRE and TRE flags will show a one in the Status word. Initializing the SIO

Bit								Bit
7	6	5	4	3	2	1	0	

CONTROL WORD

BIT 0 INITIALIZE
 BIT 1 INPUT INTERRUPT ENABLE
 BIT 2 OUTPUT INTERRUPT ENABLE
 BIT 3 STATUS ENABLE
 BIT 4
 to EXTERNAL CONTROL (EC4 to EC7)
 BIT 7

Bit								Bit
7	6	5	4	3	2	1	0	

STATUS WORD

BIT 0 Transmitter Buffer Register Empty (TBRE) flag.
 A 1 indicates the buffer is empty.
 BIT 1 Transmitter Register Empty (TRE) flag.
 A 1 indicates the transmitter is empty.
 BIT 2 Parity Error (PE) flag.
 A 1 indicates a parity error has occurred
 BIT 3 Framming Error (FE) flag.
 A 1 indicates a Framming error has occurred (incorrect
 Stop bits received)
 BIT 4 Data Ready (DR) flag.
 A 1 indicates new data has been received and is now in the
 receiver buffer.
 BIT 5 Overrun Error (OE) flag.
 A 1 indicates a second character has been received before
 the preceeding character was input. The first character
 is lost and the receiver buffer contains the second char-
 acter.
 BIT 6 External Sense (ES6)
 BIT 7 External Sense (ES7)

TABLE 12: SIO CONTROL AND STATUS WORD FORMATS

card (with a Control output) will also set the UART to this state. If a word is output to the card, it is then loaded into the transmitter buffer and then into the transmitter register. Serial output begins (within one UART clock cycle) and the flags will show an empty buffer and a busy transmitter (TBRE=1, TRE=0). Another character may now be output to the buffer. This will cause TBRE to also show a 0 indicating the buffer is full.

When transmission of the first character (including stop bits) is completed, the UART will load the second character into the transmitter and begin transmission. The buffer is now free to take a third character and so forth. Transmission ceases when both the buffer and transmitter registers are empty.

SIO driver code may be synchronized to the speed of the serial device connected to the interface through either status inputs of TBRE and TRE or through the use of interrupts.

When output interrupts are enabled, a request is generated whenever data is passed from the transmitter buffer to the transmitter register (TBRE 0 to 1 transition).

One word of caution is in order. When outputting data to the SIO interface for the first time be sure to allow at least one complete UART clock cycle before a second character is output. If this rule is not observed, it is possible to destroy the first

character by loading the second on top of the first. It is best to use the TBRE flag or interrupts to trigger the second and all successive data outputs.

INPUT DATA TRANSFER SEQUENCE

If the UART has been Initialized by a control output or has not received a character since the last input addressed to the board, then the DR flag will show 0. Likewise, the three error bits will also be 0.

When a serial character has been received (including all stop bits) the DR flag will go to 1. Parity or Framing errors may also occur. The 1 state on the DR flag may be checked through status, or the 0 to 1 transition can request an interrupt if interrupts are enabled for the input portion of the interface. Inputting the data from the card to the processor will reset the DR flag and any error conditions to 0. The interface may now receive a second character. If a second character is received before the first is input an Overrun Error occurs. This is indicated by OE=1. The receiver buffer will contain the second character and the first is lost. When the UART parity is inhibited by the mode selection, no parity will be generated or checked. The PE flag will remain low.

As can be seen from the description, the SIO interface is

quite flexible. As presented, operation at multiple baud rates, modes and output standards is possible. The UART makes possible a low package count design by integrating the complete transceiver function into a single chip. The designer need concentrate only on the design of the necessary controls for the chip.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

As might be expected of any complex digital system, operation was not correct when power was first applied. Thus it was necessary to identify the sources of trouble and make changes as necessary to correct the faults.

After the processor card was constructed, it was given an initial test by applying power and attempting to execute a series of HALT instructions. If $\overline{\text{INT}}$ is taken to a logic 0 and held there, the processor begins an Interrupt cycle. During T3, the processor will read an OP code of all ones (HALT) since no memory is present. Once execution of the HALT instruction is complete, the processor will enter the STOP state, but if $\overline{\text{INT}}$ is still zero, another interrupt cycle will follow. Thus a state sequence of T1I, T2, T3, STOP should be repeated as long as $\overline{\text{INT}}$ is low.

When this test was first tried a state sequence of T1I, T2, WAIT was observed. Investigation revealed that $\overline{\text{READY}}$ was not "ready". Jumpering the input to ground cured this problem. If the system does not contain logic that controls the $\overline{\text{READY}}$ input, then this jumper must be installed.

Once the initial test was functioning correctly, some RAM was added to the system. Attempting to execute instructions stored in memory exposed further problems in the system concerning

stray capacitance in the bus structure. Input capacitance of the RAM card also proved to be a problem.

The original RAM constructed for the system was a 2k card which contained no buffers on the ten LSB of memory address (L0-H1). Each input on the 2101 RAM chips presents a load of 3 to 5 pf. With no address buffers in the RAM system, each 1k of memory places eight loads in parallel on each of the 10 LSB of memory address. Thus each 1k presents a load of 24 to 40 pf. A complete 16k RAM would load the bus with 384 to 1040 pf. Unfortunately, the Low Power TTL gates used as bus drivers can not drive this much capacitance with rise and fall times suitable for this system. It was necessary to reduce the time constant associated with the bus wires.

Both processor and memory drive the bus with open collector gates. Thus the rise time is dependent on the value of the pullup resistor and capacitance associated with each bus wire. Similarly, the fall time depends mainly on the current that the bus driver can sink in the low state. Fall times on the bus were adequate, but the rise time was much too long due to high valued pullup resistors and too much bus capacitance.

In order to speed up the rise times, both the resistor and capacitor were reduced in value. The pullup resistors were decreased to the minimum allowed by the driver sink current. In

order to reduce the capacitance, a new 4k RAM card was designed incorporating CMOS inverters in each of the ten LSB of memory address. These inverters isolated the input capacitance of the 2102's from the bus. With the new card, 4k of RAM represents about 5 pf of load on each bus line instead of the 96 to 160 pf that would have been present with two cards of the old 2k system.

The PROM card could also have benefited from some address isolation. Only eight address bits (L bus) would require isolation since they are the only ones connected to all 1702 PROM chips. Each 1702 represents an 8-15 pf load. But since the prototype system contained only 256 words of PROM, not much reduction would have been possible by redesigning the PROM. For reasons of limited time and budget, no changes were made. In future systems which might contain larger amounts of PROM, the isolation should be included to prevent problems with the bus.

Just as rise time problems plagued the H and L buses, the D bus experienced similar trouble. Here the problem was not solvable by merely decreasing the pullup resistor because of a fault in the design of the 2k RAM which was identified and corrected on the 4k card.

In the original 2k RAM, only $\overline{ME}$ and $\overline{WRT}$ were used to control the memory during T3. As shown in the Timing Diagram (Figure 12)

$\overline{WRT}$ is a narrow pulse contained within and occurring near the end of $\overline{ME}$. Thus for the period of time extending from the beginning of $\overline{ME}$ to the beginning of $\overline{WRT}$, the 2k RAM was in the read mode and transmitting data onto the D bus. But since the processor is in a WRITE cycle, it too is transmitting data to the D bus at the same time. Since both cards use open collector drivers, the D bus shows the bit by bit wire AND of the data to be written and the old contents of the memory location. The problem occurred whenever the processor tried to change a given bit in a memory location from a zero to a one. Up until the start of $\overline{WRT}$ the memory's bus driver would hold the bus wire at a zero. After the beginning of $\overline{WRT}$, the RAM bus drivers were turned off and the bus wire would start to charge toward the logic one voltage. Even though the data is not written until the end of $\overline{WRT}$, the 800 ns width of the $\overline{WRT}$ pulse did not allow enough time for the bus wire to reach the one voltage. This problem was corrected by using the $\overline{WRITE}$ cycle signal to inhibit the bus drivers in the 4k RAM (and also the PROM). Since $\overline{WRITE}$ appears early in T2, the RAM never gets a chance to place the old data onto the bus and plenty of time is available for the write data to stabilize.

All of the modifications so far described have been aimed at either reducing the load capacitance of a card or changing the

logic to reduce its sensitivity to rise time. Nothing has been said yet about reducing the stray capacitance associated with the physical structure of the bus.

In the prototype system, the bus was composed of edge connectors soldered into an etched mother board that provided the interconnections between connectors. This method of construction resulted in a mechanically strong and reliable structure. However it is almost certain that lower parasitic capacitance could have been attained by wire wrapping the bus. The round cross section of the wire would allow greater conductor to conductor spacing and present less area when viewed as one plate of a capacitor. Both the greater spacing and smaller equivalent area would result in smaller parasitic capacitors than those possible with the flat etched conductors. Future systems should use a wire wrapped bus instead of the etched circuit board.

By adopting a practice of making all gates that drive the bus (data buses and timing signals) a TTL gate and all bus receivers a CMOS gate, a low power bus was realized. Since the CMOS gate requires almost no input current in either a one or zero state, the pullup resistor sets the DC portion of the load and the circuit cards add only capacitance. By keeping the number of inputs connected to each bus wire to a minimum by using buffers, the

capacitive load can be kept to a minimum. In the prototype system each card generally loads the bus with only one CMOS input per bus wire. A minor exception to this rule may be found in the port address decoders on the I/O interface cards and the range decoder on the PROM card. Depending on the jumpers used (channel or 2k range selected) these decoders may load an H bus line with two CMOS inputs. This load is still within reason but a better example of a decoder may be found on the 4k RAM. This decoder presents only the desired single CMOS load.

Once the problems concerning the bus time constants had been solved, other problems involving the logic design became apparent. One of these problems involved timing errors in the decoding of the state signals.

As shown in the processor schematic (Figure 11), the state signals are decoded from the three S signals (S0, S1, S2) output by the 8008. Unfortunately, these signals do not change state exactly at the same time. Thus in state transitions that involve a change of two or more of the S signals, a spurious output (spike) may occur on the wrong decoder output as the S signals momentarily present an invalid code to the input of the decoder. Capacitors C6-C13 on the output of the decoder remove some of the shorter spikes but the longer ones still make it through to the bus.

These spikes make using the edges of state signals a risky venture. Instead, the state signals should be used in combination with a reliable timing signal such as DSTB.

Problems were encountered with the operation of the Interrupt Cycle flip flop (gates 25C and 25D) because of the presence of the spikes. In the original processor circuit T1I and T1 were used directly to set and clear the flip flop. T1I would set correctly, but a spike on T1 at the end of T1I would then clear the flip flop prematurely. When this happened, memory control was not suppressed during the Interrupt Cycle INSTRUCTION cycle. As a result, both memory and the interrupting I/O port would transmit an OP code to the D bus at the same time (T3). The resulting OP code was the bit by bit AND of the two words. Destroying the desired instruction in this manner caused unpredictable performance. This problem was cured by combining DSTB with T1 and T1I in gates 40C and 40D and using the resultant signals to set and clear the Interrupt Cycle flip flop.

Modifications to the Master Priority Control flip flop (gates 3A and 3C) were also required. In case an interrupting device (perhaps a front panel) was to execute Special Instruction PON to reset priority. In the original processor circuit this flip flop was cleared directly by the $\overline{\text{INT}}$ signal. Thus executing this

instruction would not reset the priority since the presence of $\overline{\text{INT}}$ would override the narrow pulse generated by the Special Instruction. Adding R47, D2 and C15 to the processor made the Master Priority Control flip flop sensitive to only the negative edge of $\overline{\text{INT}}$. On this edge the resistor and capacitor generate a negative pulse (by differentiating $\overline{\text{INT}}$). This pulse clears the Master Priority Control flip flop. The positive pulse that would be generated when $\overline{\text{INT}}$ returns to one is clipped by the diode.

The second modification made to this flip flop was necessary because during the debugging, the processor developed the nasty habit of stopping with the priority reset. This inhibited all interrupts and froze the system. Rather than turn the power off and then back on (thereby destroying RAM contents) to reset the system, a connection was made to the $\overline{\text{CLR}}$ bus signal to enable the operator to restore control to the system. If the system freezes, merely closing a normally open switch connected between the $\overline{\text{CLR}}$ bus wire and ground will reset priority and the I/O interfaces. The interrupt system is then in the same state as power up produces but the RAM contents have not been changed. Processor registers and PC are also preserved.

The final modification to the processor was made to accommodate the front panel circuitry used with the system. Other systems

might require the other mode of operation that is described below.

A question arose concerning whether or not memory control should be suppressed during INSTRUCTION and READ or only INSTRUCTION cycles belonging to an interrupt instruction ($\overline{\text{INTCY}}=0$). A conflict arises because there are really two types of READ cycles. One type occurs during the second (or third) bytes of multibyte instructions. Since interrupt logic would probably be expected to supply the immediate or address data, memory control should be inhibited. But the second type of READ involves an instruction which is supposed to input a memory word to the processor and memory control should occur. Further discussion of this problem and description of a masking technique for overcoming the conflict can be found in the discussion covering processor hardware (Chapter 4).

With gates 23A and 23B connected as shown on the processor schematic, memory control is inhibited only during INSTRUCTION cycles belonging to interrupt instructions. If the connection between pins 4 and 5 on gate 23B is broken and a connection made between pins 5 and 1 on gates 23B and 23C then memory control will be inhibited during READ also.

Just as the processor and memory required modification; the I/O ports and Real Time Clock also required attention. When first attempting to run the system on interrupts, it was discovered

that the processor had a tendency to occasionally process interrupts incorrectly. This fault was traced to faulty synchronization of the $\overline{\text{INT}}$ signal.

For reliable recognition of interrupts, $\overline{\text{INT}}$ must not change state within 300 ns of the negative edge of any ϕ_1 clock pulse as measured on the 8008. When this rule is not observed, the 8008 may destroy the contents of registers H and L or the PC. Interrupting at the wrong time appears to interfere with the refresh of the dynamic registers within the 8008.

If all interrupts are set and cleared on the leading edge of DSTB then no problems should be encountered. When the machine is in the STOP state, the Interrupt Cycle (TII) will start on the next ϕ_1 clock (ϕ_{1_1} , which occurs about 2.15 us later). The Timing Diagram (Figure 12) illustrates this sequence. If the interrupt occurs when the machine is running or waiting, then the Interrupt Cycle will begin when execution of the current instruction is complete. $\overline{\text{INT}}$ should return to a one sometime after ϕ_{1_1} in TII belonging to the last interrupt instruction in a sequence, but before the end of the last state of the interrupt instruction.

The original circuit used in the Interrupt and Priority Control logic did not synchronize on the leading edge of DSTB. Because a Set-Reset flip flop was used for the Interrupt flip flop, $\overline{\text{INT}}$ could

go low at any time during DSTB. When this happened toward the end of DSTB, erratic recognition of interrupts was noted. This problem was corrected by adding an edge sensitive D flip flop (7474) to replace the Set-Reset flip flop. This had to be done on all levels of Interrupt and Priority Control logic. Once changes in $\overline{INT}$ were confined to the leading edge, the system performed correctly.

The PIO card exhibited one remaining fault. Occasionally a false Status mode would be set when changing the control bit from a one to a zero. As mentioned in the discussion on PIO hardware (Chapter 6), the delay in clocking the zero bit into the control word latch causes a narrow clear pulse to be sent to the Status flip flop. This narrow pulse was easily removed by loading the set input of the flip flop with a capacitor (C9 on PIO schematic). No modification to the Real Time Clock was required since it does not have a Status mode.

Once the Interrupt and Priority Control logic on the RTC had been modified as described earlier, the clock appeared to function correctly. However, further testing revealed that all timing intervals were longer than they should have been by one half the time base rate. Investigation revealed that the problem lay in using the borrow output of the RTC's down counter (IC's 12 and 13) to generate the Interrupt Request ($\overline{REQ}$).

The clock signal from the time base that clocks (decrements) the down counter is a 50% duty cycle square wave. The counter changes state on the positive edge of this clock, but the borrow signal is not generated until the negative edge (once the counter has reached zero). This caused each timing interval to be a half cycle long. This fault was cured by adding R4, D1 and C3 to the RTC. These components differentiate the clock, generating a narrow positive pulse on the rising edge of the time base signal. D1 clips the negative pulse that would normally occur on the falling edge. Thus the borrow occurs only a short time after the counter reaches zero.

Once the problems just mentioned were corrected, the system was completely functional. Yet further changes and modifications could be made to improve the system in other ways.

The processor clock could stand improvement. Both the frequency and the width of the two phase clock are critical. The Intel data book on the 8008 contains detailed specifications on the clock waveforms. If the specified limits are not observed, the 8008 will not function correctly. A crystal controlled clock would provide better stability than the one-shot ring oscillator now used. Also, the two clock phases should be a part of the bus signal set. This would enable running multiple processors off of the same clock. It

would even be possible to accommodate multiple processors sharing one memory and I/O system if some priority logic were designed to resolve bus conflicts.

Another improvement could be applied to synchronizing interrupts with the ϕ_1 clock. Instead of relying on each level of Interrupt and Priority Control logic to synchronize $\overline{\text{INT}}$, a better method would be to let $\overline{\text{INT}}$ occur asynchronously in the bus and synchronize the interrupt signal to the 8008 with a single flip flop located on the processor. This method would result in interrupt logic that was not sensitive to the delays imposed on the timing signals by the bus drivers and the interrupt logic. Simpler interrupt logic would probably result and it would be possible to interrupt on any ϕ_1 pulse instead of only DSTB (ϕ_{12}) as the present interrupt logic does.

Further improvement in the rise times on the bus could be obtained by replacing the open collector drivers with tri-state drivers. At this stage of development of the system, only the PIO interface uses tri-state logic. Since these outputs can source current in the one state when enabled, a faster rise time results since the output itself can help the pullup resistor charge the capacitive part of the bus load.

The system, as presented in this thesis, represents the major

parts of a complete, general purpose microcomputer. Additional components needed to round out the system include a front panel, power supplies, and packaging. Fabrication of the serial interface (SIO) would allow the system to interface through a modem to a variety of other systems using standard communications channels such as a telephone line. Remote control of the system would now be practical as would remote reading of data collected by the machine in a data acquisition application.

The first application planned for the system is programmable data acquisition. Research is continuing in the development of the hardware necessary to accomplish this function. An analog multiplexor and analog to digital conversion subsystem is under development. This circuitry interfaces to the microcomputer through a PIO interface. A front panel and read/write digital cassette tape have already been developed and debugged. The cassette also interfaces to the microcomputer through a PIO interface. Packaging and power supplies should also be complete in the near future. These portions of the system will be documented in a thesis being written by Mr. John Somppi which should be available in the Montana State University library late in 1976. When these additions are complete the system will be configured as a general purpose data acquisition system.

In conclusion, the system was designed, constructed and debugged. Much was learned during these three stages. Practical experience and knowledge were gained in both logic design and construction of a prototype. The finished system functions as desired and is applicable to many uses formerly handled by special purpose hardwired logic or larger and more expensive minicomputer systems.

APPENDIX A

DEFINITION OF BUS SIGNALS, BUS PIN NUMBERS AND BUS SCHEMATIC

DATA AND ADDRESS BUS GROUP

Data Bus (D Bus) (D0 through D7)

The eight signals that comprise the D Bus serve as a bidirectional data path and carry all data into the processor including instructions, memory read and I/O input data. Additionally, the D bus carries write data to memory from the processor.

Low Address Bus (L Bus) (L0 through L7)

The L Bus carries the eight least significant bits of memory address and output data for I/O ports.

High Address Bus (H Bus) (H0 through H7)

The H Bus carries the eight most significant bits of memory address and also I/O address. Because the existing processor can address only 16k words of memory and the number of I/O channels is limited to 32, H6 and H7 are not used by the system. Provision has been left on the system bus to accommodate these signals if needed.

PROCESSOR CYCLE GROUP

Depending on the particular instruction, the processor may take one, two or three cycles to complete execution of an instruction. All cycle types are available as decoded signals output from

the processor to the system bus.

Instruction ($\overline{\text{INST}}$)

INST is always the first cycle of any instruction. During this time the processor fetches from memory or interrupt source the instruction OP code.

Read ($\overline{\text{READ}}$)

The READ cycle indicates that the processor is reading something other than an OP code byte out of memory. These included the second and third bytes of multibyte instructions.

Write ($\overline{\text{WRITE}}$)

WRITE indicates the processor is writing into memory.

Input/Output ($\overline{\text{I/O}}$)

I/O cycle indicates the processor is performing an Input or Output operation. I/O will also show during execution of the Special Instructions used by this system since they are really I/O instructions.

Interrupt Cycle ($\overline{\text{INTCY}}$)

INTCY is an additional flag output by the processor. It may occur along with any of the four cycle types and serves to indicate that a particular cycle belongs to an instruction being executed under interrupt.

PROCESSOR STATE GROUP

Just as each machine cycle is composed of one, two or three cycles, each cycle contains three, four or five processor states occurring sequentially in time. Every cycle will have a T1 (or T1I), T2 and T3 while some may also have T4 and T5. The remaining two states (WAIT and STOP) represent special processor states and result in a suspension of processor activity.

T1I ($\overline{\text{T1I}}$)

The first state of any cycle belonging to an instruction executing under interrupt is T1I.

T1 ($\overline{\text{T1}}$)

T1 is the first state of any cycle belonging to an instruction not executing under interrupt. Functionally T1I and T1 are identical except that they distinguish between interrupt and noninterrupt instructions.

T2 ($\overline{\text{T2}}$)

T2 is the second state of any processor cycle.

T3 ($\overline{\text{T3}}$)

T3 is the third and final required processor state for a cycle.

T4 ($\overline{\text{T4}}$)

The fourth cycle is T4. It may be or may not occur depending on the instruction being executed.

T5 ($\overline{T5}$)

T5 is the last regular processor state that may occur within a cycle.

Stop ($\overline{STOP}$)

Whenever a HALT instruction is executed the processor responds with a STOP state and then waits for an interrupt to restart execution of program code. A STOP state may last for any integral number of SYNC time periods.

Wait ($\overline{WAIT}$)

If the Ready signal is made false the processor places a WAIT state between the next T2 and T3 for as long as the READY signal remains false. When READY returns to a true state then the processor continues execution of the program. Ready may be used to "wait" for slow memory or to suspend processor activity to allow a DMA transfer.

PROCESSOR TIMING GROUP

Sync (SYNC)

The SYNC signal is a divided by two version of the $\phi 2$ processor clock. When the processor is in a T state, SYNC is true during the second half of the state time. SYNC is present during all states including WAIT and STOP.

Data Strobe ($\overline{\text{DSTB}}$)

DSTB is a timing pulse that occurs coincidentally with every ϕ_{12} processor clock.

Data Reset ($\overline{\text{DRST}}$)

DRST is a timing pulse that occurs coincidentally with processor clock ϕ_{22} during T3 only.

MEMORY CONTROL GROUP

Memory Enable ($\overline{\text{ME}}$)

In order to read or write, the memory requires a true Memory Enable signal during T3.

Read/Write ($\overline{\text{WRT}}$)

When a WRT pulse occurs during a ME then the memory will write data from the D bus into the location addressed by the H and L buses. Data is written at the end of the WRT pulse.

SPECIAL INSTRUCTION GROUP

In the system those Output ports having addresses 20B to 27B are designated as a Special Instructions and used to implement hardware functions not standard in the 8008 instruction set.

Four of these Special Instructions are available as individual decoded pulses on the bus. Each pulse occurs during DRST and I/O cycle when the particular output instruction is executed. In all

cases the I/O address and the output data will appear on the H-L Bus in the normal manner and may be decoded from the bus signals if desired.

Special 20 ($\overline{\text{SP20}}$)

SP20 is caused by executing an output to port 20B.

Special 23 ($\overline{\text{SP23}}$)

Executing and output to port 23B causes SP23.

Special 24 ($\overline{\text{SP24}}$)

SP24 occurs when an output is addressed to port 24B.

Priority Reset ($\overline{\text{PRST}}$)

PRST is a dedicated Special Instruction using output port 27B. It serves to reset the Priority Masks in an I/O interface after an interrupt routine has ended. The output data appears on the L bus but is ignored.

Out of the four remaining Special Instructions, PON (OUT 26B) and POFF (OUT 25B) are used internally by the processor for interrupt control. The remaining two (OUT 21B and 22B) are undecoded and uncommitted and may be decoded from the bus and used for any applicable purpose.

OTHER BUS SIGNALS

Ready ($\overline{\text{READY}}$)

When READY is made false the processor responds by entering the WAIT state until READY becomes true. Uses include interfacing the processor to slow memory and suspending activity for a DMA transfer.

Interrupt ($\overline{\text{INT}}$)

Taking the INT signal true enters a request for an interrupt with the processor which will finish executing the present instruction and then respond with a TII and look for an instruction from the interrupting device.

Disconnect Memory Address ($\overline{\text{DMA}}$)

Enabling this processor input disconnects the processor's drivers from the H-L bus. This allows an external device to control the address buses. Typical uses would be to allow addressing during a DMA transfer.

Priority Out ($\overline{\text{POUT}}$)

POUT is the I/O priority signal used to schedule interrupts from the I/O channels. This signal is passed along a chain originating at the processor and terminating at the lowest priority I/O interface in the machine. The priority of a given I/O port decreases the farther down the chain from the processor it is located. Each POUT signal becomes the Priority Input signal for the next lowest I/O interface present in the machine. The master

POUT signal is generated by the processor and without this signal no interrupts are possible. Thus the processor is considered the highest priority device on the bus.

Priority Input ($\overline{\text{PIN}}$)

PIN originates as the POUT signal from the adjacent higher priority I/O port or processor. This signal is required for an interface to generate an interrupt.

H-L Inhibit ($\overline{\text{HLINH}}$)

HLINH will inhibit the processor from changing the contents of the H and L buses. This feature is not used in the prototype system.

Clear ($\overline{\text{CLR}}$)

CLR is a master reset signal developed when power is applied to the machine and alternately by a front panel switch connected between the bus wire and ground. This signal resets the processor master POUT signal to enable interrupts and also resets all I/O interfaces to an inactive state with interrupts turned off.

RAM WRITE PROTECT FEATURE

Random Access Memory contains a Write Protect feature. Each of the four 1k banks on each 4k RAM card may be protected by a logic zero (grounding) on the appropriate pin ($\overline{\text{PROTECT A}}$, $\overline{\text{PROTECT B}}$,

PROTECT C AND PROTECT D) on the edge connector that the memory card is plugged into (see RAM schematic, Figure 15). These pins are not bussed between edge connectors and must be interconnected as needed to protect the desired portions of RAM. If this feature is used, the positions of the RAM cards in the bus must be fixed in order to prevent different portions of RAM from becoming protected if the cards were moved to a different connector. If no Write Protect is used, these protect inputs may be left unconnected, otherwise, a switch connection to ground may be used to implement the feature.

The RAM memory has one more input which is not used by this system. The INHIBIT input may be used to prevent the memory from transmitting data to the D bus. When INHIBIT is at a logic zero, the RAM's bus drivers are disabled.

Power Supplies

The system bus also carries power to the different cards plugged into the bus. Power supply voltages of +5, -12 and -9 volts DC are available.

BUS CONNECTIONS AND SCHEMATIC

Figure 21 and Table 13 contain a schematic of the bus and list the pin numbers, mnemonic name and termination resistor for each bus signal. Note that there are several spare pins on each edge.

connector.

The numbers assigned to the different pins are those given by the manufacturer of the edge connectors used in the prototype. Since different manufacturers may number their connectors different, the schematic has been drawn as the edge connector appears when viewed from the rear (terminal side). The particular connector used has been noted on the drawing to aid in selecting alternate but equivalent connectors.

TABLE 13: BUS SIGNALS AND PIN NUMBERS

<u>SIGNAL MNEMONIC</u> (Bus Sense)	<u>I/O PIN</u> <u>NUMBER</u>	<u>MEMORY PIN</u> <u>NUMBER</u>	<u>TERMINATION</u> <u>RESISTOR</u>
<u>D BUS</u>			
D0	24	N	1.8k
D1	26	P	1.8k
D2	28	R	1.8k
D3	30	S	1.8k
D4	23	12	1.8k
D5	25	13	1.8k
D6	27	14	1.8k
D7	29	15	1.8k
<u>L BUS</u>			
L0	3	2	1.8k
L1	5	3	1.8k
L2	7	4	1.8k
L3	9	5	1.8k
L4	11	6	1.8k
L5	13	7	1.8k
L6	15	8	1.8k
L7	17	9	1.8k
<u>H BUS</u>			
H0	4	B	1.8k
H1	6	C	1.8k
H2	8	D	1.8k
H3	10	E	1.8k
H4	12	F	1.8k
H5	14	H	1.8k
H6	16	J	1.8k
H7	18	K	1.8k
<u>PROCESSOR CYCLE GROUP</u>			
<u>INST</u>	39	-	1.8

TABLE 13 (continued)

<u>SIGNAL MNEMONIC</u> (Bus Sense)	<u>I/O PIN</u> <u>NUMBER</u>	<u>MEMORY PIN</u> <u>NUMBER</u>	<u>TERMINATION</u> <u>RESISTOR</u>
<u>READ</u>	41	-	1.8k
<u>WRITE</u>	43	T	1.8k
<u>I/O</u>	45	-	1.8k
<u>INTCY</u>	53	-	1.8k
<u>PROCESSOR STATE GROUP</u>			
<u>TLI</u>	44	-	1.8k
<u>TI</u>	46	-	1.8k
<u>T2</u>	48	-	1.8k
<u>T3</u>	50	-	1.8k
<u>T4</u>	52	-	1.8k
<u>T5</u>	54	-	1.8k
<u>WAIT</u>	56	-	1.8k
<u>STOP</u>	58	-	1.8k
<u>PROCESSOR TIMING GROUP</u>			
<u>SYNC</u>	47	-	1.8k
<u>DRST</u>	49	-	1.8k
<u>DSTB</u>	51	-	1.8k
<u>MEMORY CONTROL GROUP</u>			
<u>ME</u>	22	M	2.2k
<u>WRT</u>	20	L	2.2k
<u>SPECIAL INSTRUCTION GROUP</u>			
<u>SP20</u> (uncommitted)	37	-	1k
<u>SP23</u> (uncommitted)	59	-	1k
<u>SP24</u> (uncommitted)	61	-	1k

TABLE 13 (continued)

<u>SIGNAL MNEMONIC</u> (Bus Sense)	<u>I/O PIN</u> <u>NUMBER</u>	<u>MEMORY PIN</u> <u>NUMBER</u>	<u>TERMINATION</u> <u>REGISTER</u>
<u>PRST</u> (SP27) (used by I/O interfaces)	55	-	1k

OTHER BUS SIGNALS

<u>RDY</u>	40	-	2.2k
<u>INT</u>	42	-	2.2k
<u>DMA</u>	19	-	2.2k
<u>POUT</u>	63	-	none
<u>PIN</u>	64	-	none
<u>HLINH</u>	38	-	none
<u>CLR</u>	57	-	2.2k

SPECIAL RAM CONTROL FEATURES
 (these signals not bussed)

<u>PROTECT A</u>	-	U	none
<u>PROTECT B</u>	-	17	none
<u>PROTECT C</u>	-	V	none
<u>PROTECT D</u>	-	18	none
<u>INHIBIT</u>	-	19	none

POWER SUPPLY VOLTAGES

+5 VDC	1 & 2	1 & A	-
-12 VDC	70	Y	-
-9 VDC	69	21	-
Ground	71 & 72	22 & Z	-

APPENDIX B

FORMAL DESCRIPTION OF INSTRUCTION SET

FORMAL DESCRIPTION OF INSTRUCTION SET

Abbreviations Used:

A	The Accumulator (Register A)
A _n	The Nth bit of the Accumulator where n varies between 0 and 7
ADDR	Any Memory Address
Carry	The Carry Bit or Flag
CODE	An Instruction Byte (8 bits) (Also OP CODE)
DATA	Any BYTE that is not an instruction
DDD	Three bit code for destination register (note 1)
DST	Alternate name for the DDD code (Destination)
M	Any memory byte
R	Any register byte
Parity	The Parity bit or flag
PC	Program Counter
REGM	Any Register or Memory byte (R or M inclusive)
Sign	The Sign bit or flag
SSS	Three bit code for source register (note 1)
Zero	The Zero bit or flag
STK	Level 1 in the PC stack (one level below PC)
()	Contents of the register or memory byte enclosed
←	Operator, replace the left hand side with the result of the right hand side.

SRC Alternate name for the three bit SSS code (Source)

RR Top two bits of output port Address (RR $\neq$ 00)

AAA Lower three bits of output port Address (channel address)

X Denotes a bit whose value is unimportant (don't care)

BB BBB BBB SECOND BYTE of multibyte instruction. May be immediate data or low byte of Address.

XX CCC CCC THIRD BYTE of JUMP or CALL Instruction. Always used as 6 bits of high Address (Note two Don't Care bits)

Note 1: DDD and SSS codes

A(000)
B(001)
C(010)
D(011)
E(100)
H(101)
L(110)
M(111)

Codes permissible for R include A through L while REGM may include any of the eight.

SINGLE REGISTER INSTRUCTIONS

Single byte instructions operating on a single operand.

MNEMONIC	CODE	DESCRIPTION
INR	00 DDD 000	$(R) \leftarrow (R) + 1$ Increment Register R
DCR	00 DDD 001	$(R) \leftarrow (R) - 1$ Decrement Register R

Flags Affected: Zero, Sign, Parity

Note: $R \neq A$ (HALT Instruction)

Execution Time: 5 States

MOV INSTRUCTIONS

Single byte instructions used to move data between registers or registers and memory.

MNEMONIC	CODE	DESCRIPTION
MOV	11 DDD SSS	$(DST) \leftarrow (SRC)$ Load DST with SRC

Flags Affected: None

Note: DDD and SSS not both to be M (HALT instruction)

Execution Time: Register to Register 5 states (MOV A,B)

Register to Memory 7 states (MOV M,A)

Memory to Register 8 states (MOV A,M)

REGISTER OR MEMORY TO ACCUMULATOR INSTRUCTIONS

Single byte instructions using the accumulator and another operand from either a register or memory. Memory registers are

addressed by H & L registers. Results end up in the Accumulator except for Compare instructions which affect only the flags.

MNEMONIC	CODE	DESCRIPTION
ADD	10 000 SSS	$(A) \leftarrow (A) + (\text{REGM})$ Add REGM to A
ADC	10 001 SSS	$(A) \leftarrow (A) + (\text{REGM}) + (\text{Carry})$ Add REGM plus Carry to A
SUB	10 010 SSS	$(A) \leftarrow (A) - (\text{REGM})$ Subtract REGM from A
SBB	10 011 SSS	$(A) \leftarrow (A) - (\text{REGM}) - (\text{Carry})$ Subtract REGM and Carry from A
ANA	10 100 SSS	$(A) \leftarrow (A) \text{ AND } (\text{REGM})$ Logical AND between REGM and A
XRA	10 101 SSS	$(A) \leftarrow (A) \text{ EXOR } (\text{REGM})$ Exclusive OR function
ORA	10 110 SSS	$(A) \leftarrow (A) \text{ OR } (\text{REGM})$ Inclusive OR function
CMP	10 111 SSS	All condition Flags set by the result of $(A) - (\text{REGM})$. Compare function does not change A or REGM.

Flags Affected: ADD, ADC, SUB, SBB: Carry, Sign, Zero, Parity
 ANA, XRA, ORA: Sign, Zero, Parity, Carry reset to zero (cleared)
 CMP: Zero set if $(A) = (\text{REGM})$
 Carry reset if $(A) \geq (\text{REGM})$
 Carry set if $(A) < (\text{REGM})$

Note: REGM denotes the SSS code

Execution Time: SSS denotes a register 5 states (CMP B)
 SSS denotes memory 8 states (CMP M)

ROTATE ACCUMULATOR INSTRUCTIONS

Single byte instructions that operate on the accumulator.

MNEMONIC	CODE	DESCRIPTION
RLC	00 000 010	$(\text{Carry}) \leftarrow A_7, A_{n+1} \leftarrow A_n, A_0 \leftarrow A_7$ Set Carry to A_7 , rotate Accumulator left
RRC	00 001 010	$(\text{Carry}) \leftarrow A_0, A_n \leftarrow A_{n+1}, A_7 \leftarrow A_0$ Set Carry to A_0 , rotate Accumulator right
RAL	00 010 010	$A_{n+1} \leftarrow A_n, (\text{Carry}) \leftarrow A_7, A_0 \leftarrow (\text{Carry})$ Rotate Accumulator left through Carry
RAR	00 011 010	$A_n \leftarrow A_{n+1}, (\text{Carry}) \leftarrow A_0, A_7 \leftarrow (\text{Carry})$ Rotate Accumulator right through Carry

Flags Affected: Carry
 Execution Time: 5 States

IMMEDIATE INSTRUCTIONS

Double (Two) byte instructions where the second byte is used for the data needed by the operation. Move instructions place the second byte in the DST register. ALU operations use the second byte as the operand with A and leave results in A. Again the compare instructions affect only the flags.

MNEMONIC	CODE	DESCRIPTION
MVI	00 DDD 110 BB BBB BBB	$(REGM) \leftarrow DATA$ Load REGM with DATA (second byte)
ADI	00 000 100 BB BBB BBB	$(A) \leftarrow (A) + DATA$ Add immediate Data to accumulator
ACI	00 001 100 BB BBB BBB	$(A) \leftarrow (A) + DATA + (Carry)$ Add immediate Data plus Carry to Accumulator
SUI	00 010 100 BB BBB BBB	$(A) \leftarrow (A) - DATA$ Subtract Data from Accumulator
SBI	00 011 100 BB BBB BBB	$(A) \leftarrow (A) - DATA - (Carry)$ Subtract Data and Carry from Accumulator
ANI	00 100 100 BB BBB BBB	$(A) \leftarrow (A) \text{ AND } DATA$ AND accumulator with Data
XRI	00 101 100 BB BBB BBB	$(A) \leftarrow (A) \text{ EXOR } DATA$ Exclusive OR of Data and Accumulator
ORI	00 110 100 BB BBB BBB	$(A) \leftarrow (A) \text{ OR } DATA$ Inclusive OR of DATA and Accumulator
CPI	00 111 100 BB BBB BBB	All condition flags set by the results of $(A) - DATA$ Neither A nor Data are changed

Flags Affected: MVI None
 ADI, ACI, SUI, SBI: Zero, Sign, Parity, Carry
 ANI, XRI, ORI: Zero, Sign, Parity, Carry reset to zero
 CPI Zero set if $(A) = DATA$
 Carry reset if $(A) \geq DATA$
 Carry set if $(A) < DATA$

Note: BB BBB BBB is the second byte of the instruction and is used for DATA

Execution time: 8 states

JUMP INSTRUCTIONS

Three byte instructions that force the PC to be set to another value. Nothing is saved in the stack. The address follows the OP code. Second byte is the low address and the third byte (6 bits) acts as the high address.

MNEMONIC	CODE	DESCRIPTION
JMP	01 XXX 100 BB BBB BBB xx CCC CCC	$(PC) \leftarrow ADDR$ Jump to ADDR
JC	01 100 000 BB BBB BBB xx CCC CCC	If(Carry)=1, $(PC) \leftarrow ADDR$ Jump to ADDR if If(Carry)=0, $(PC) \leftarrow (PC)+3$ carry
JNC	01 000 000 BB BBB BBB xx CCC CCC	If(Carry)=0, $(PC) \leftarrow ADDR$ Jump to ADDR if no carry If(Carry)=1, $(PC) \leftarrow (PC)+3$
JZ	01 101 000 BB BBB BBB xx CCC CCC	If(Zero)=1, $(PC) \leftarrow ADDR$ Jump if Zero If(Zero)=0, $(PC) \leftarrow (PC)+3$
JNZ	01 001 000 BB BBB BBB xx CCC CCC	If(Zero)=0, $(PC) \leftarrow ADDR$ Jump if no Zero If(Zero)=1, $(PC) \leftarrow (PC)+3$
JP	01 010 000 BB BBB BBB xx CCC CCC	If(Sign)=0, $(PC) \leftarrow ADDR$ Jump if positive If(Sign)=1, $(PC) \leftarrow (PC)+3$
JM	01 110 000 BB BBB BBB xx CCC CCC	If(Sign)=1, $(PC) \leftarrow ADDR$ Jump if minus If(Sign)=0, $(PC) \leftarrow (PC)+3$

JPE	01 111 000 BB BBB BBB xx CCC CCC	If(Parity)=1, (PC) $\leftarrow$ ADDR Jump if even Parity If(Parity)=0, (PC) $\leftarrow$ (PC)+3
JPO	01 011 000 BB BBB BBB xx CCC CCC	If(Parity)=0, (PC) $\leftarrow$ ADDR Jump if odd Parity If(Parity)=0, (PC) $\leftarrow$ (PC) +3

Flags Affected: None

Note: CCC CCC BB BBB BBB equals ADDR

Execution Time: Performing any Jump requires 11 states if the processor actually jumps. If the test fails then the instruction takes only 9 states.

CALL INSTRUCTIONS

A group of three byte instructions that set the PC to a new value and also save the old PC in STK. Again as in the Jump instructions the OP code is followed by two bytes of address.

MNEMONIC	CODE	DESCRIPTION
CALL	01 XXX 110 BB BBB BBB xx CCC CCC	(STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR Call subroutine and push return address into stack.
CC	01 100 010 BB BBB BBB xx CCC CCC	If(Carry)=1, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Carry)=0, (PC) $\leftarrow$ (PC)+3 Call subroutine if Carry set
CNC	01 000 010 BB BBB BBB xx CCC CCC	If(Carry)=0, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Carry)=1, (PC) $\leftarrow$ (PC)+3 Call subroutine if Carry reset

CZ	01 101 010 BB BBB BBB xx CCC CCC	If(Zero)=1, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Zero)=0, (PC) $\leftarrow$ (PC)+3 Call subroutine if Zero is set
CNZ	01 001 010 BB BBB BBB xx CCC CCC	If(Zero)=0, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Zero)=1, (PC) $\leftarrow$ (PC)+3 Call subroutine if Zero is reset
CP	01 010 010 BB BBB BBB xx CCC CCC	If(Sign)=0, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Sign)=1, (PC) $\leftarrow$ (PC)+3 Call subroutine if Sign is positive
CM	01 110 010 BB BBB BBB xx CCC CCC	If(Sign)=1, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Sign)=0, (PC) $\leftarrow$ (PC)+3 Call subroutine if Sign is minus
CPE	01 111 010 BB BBB BBB xx CCC CCC	If(Parity)=1, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Parity)=0, (PC) $\leftarrow$ (PC)+3 Call subroutine if Parity is even
CPO	01 011 010 BB BBB BBB xx CCC CCC	If(Parity)=0, (STK) $\leftarrow$ (PC), (PC) $\leftarrow$ ADDR If(Parity)=1, (PC) $\leftarrow$ (PC)+3 Call subroutine if Parity is odd

Flags Affected: None

Note: CCC CCC BB BBB BBB equals the subroutine call address

Execution time: Executing a Call takes 11 states if the Call is actually made. To test and fail requires 9 states.

RETURN INSTRUCTIONS

A group of single byte instructions that force the processor to pop up one level in the stack and resume executing the main program at the instruction after the last Call.

MNEMONIC	CODE	DESCRIPTION
RET	00 XXX 111	(PC) $\leftarrow$ (STK) Return from Subroutine
RC	00 100 011	If(Carry)=1, (PC) $\leftarrow$ (STK) If(Carry)=0, (PC) $\leftarrow$ (PC)+3 Return if Carry is set
RNC	00 000 011	If(Carry)=0, (PC) $\leftarrow$ (STK) If(Carry)=1, (PC) $\leftarrow$ (PC)+3 Return if Carry is reset
RZ	00 101 011	If(Zero)=1, (PC) $\leftarrow$ (STK) If(Zero)=0 (PC) $\leftarrow$ (PC)+3 Return if Carry is set
RNZ	00 001 011	If(Zero)=0, (PC) $\leftarrow$ (STK) If(Zero)=1, (PC) $\leftarrow$ (PC)+3 Return if Zero is reset
RM	00 110 011	If(Sign)=1, (PC) $\leftarrow$ STK If(Sign)=0, (PC) $\leftarrow$ (PC)+3 Return if sign is minus
RP	00 010 011	If(Sign)=0, (PC) $\leftarrow$ (STK) If(Sign)=1, (PC) $\leftarrow$ (PC)+3 Return if sign is plus
RPE	00 111 011	If(Parity)=1, (PC) $\leftarrow$ (STK) If(Parity)=0, (PC) $\leftarrow$ (PC)+3 Return if parity is even
RPO	00 011 011	If(Parity)=0, (PC) $\leftarrow$ (STK) If(Parity)=1, (PC) $\leftarrow$ (PC)+3 Return if parity is odd

Flags Affected: None

Execution Time: To execute any Return function takes 5 states if the return succeeds. To test and fail takes 3 states.

RESTART INSTRUCTIONS

A group of single byte subroutine Calls that can access only eight addresses in the first 100B words of memory.

MNEMONIC	CODE	DESCRIPTION
RST	00 AAA 101	(STK) $\leftarrow$ (PC) (PC) $\leftarrow$ 00000000AAA000 Call subroutine at address

Flags Affected: None

Note: This Call can access only those addresses described above.

Thus RST 0B calls location 0B. RST 1B calls location 10B and so forth for the eight allowed addresses.

Execution Time: 5 states.

This group of single byte instructions is used to input and output data through the I/O ports.

MNEMONIC	CODE	DESCRIPTION
IN	01 00A AA1	(A) $\leftarrow$ Input Port Data Read Data at input port to accumulator
OUT	01 RRA AA1	Output Port Data $\leftarrow$ (A) Send Accumulator to output port

Flags Affected: None

Note: The system defines OUT instructions in the following manner:

RR=00 (Input instruction)
 RR=01 Data Output Port (Channel=AAA)
 RR=10 Special Instruction
 RR=11 Interface Control Port. (Channel=AAA)

In the case of control ports, (A) serves as the control word. Special instructions may or may not use the data that is output.

Execution Time: IN 8 states; OUT 6 states.

MACHINE INSTRUCTION

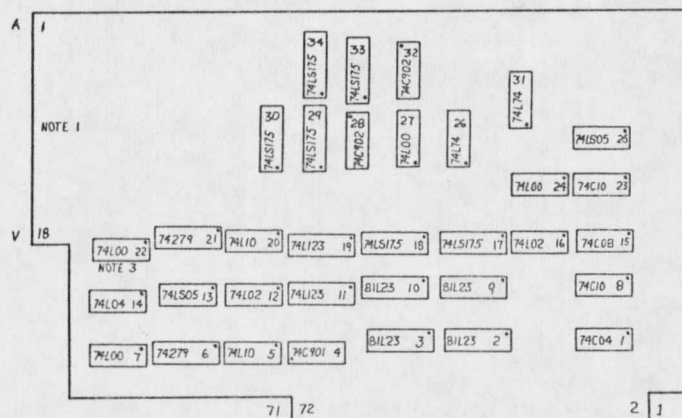
Single byte instructions that stop the processor and place it in the STOP state until interrupted.

MNEMONIC	CODE	DESCRIPTION
HLT	00 000 00X or 11 111 111	Finish execution of Halt instruction and enter STOP state until interrupted.

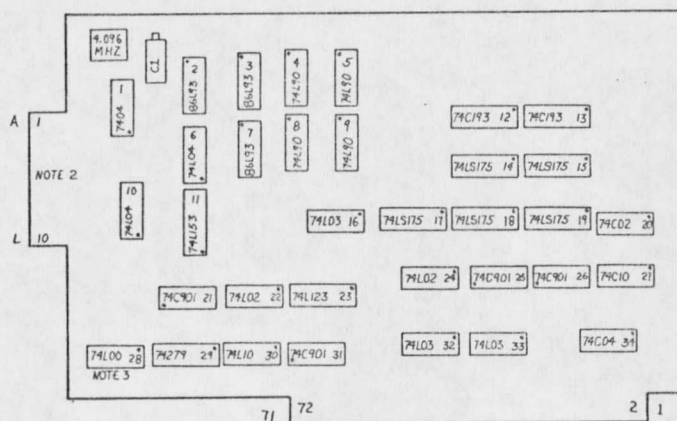
Flags Affected: None
 Execution Time: 5 states

APPENDIX C

INTEGRATED CIRCUIT POSITION KEYS



8 BIT PARALLEL INTERFACE (PIO)



REAL TIME CLOCK (RTC)

NOTE 1: PIO CONNECTOR DOUBLE 18 , .156 inch pin to pin
(ELCO 600T or Equiv.)

NOTE 2: RTC CONNECTOR DOUBLE 10 , .156 inch pin to pin
(VIKING 2VK10D or Equiv.)

NOTE 3: 7474 MOUNTED ON TOP OF IC 22 (PIO) AND IC 28 (RTC)
TO MODIFY INTERRUPT LOGIC

FIGURE 22: INTEGRATED CIRCUIT POSITION KEYS

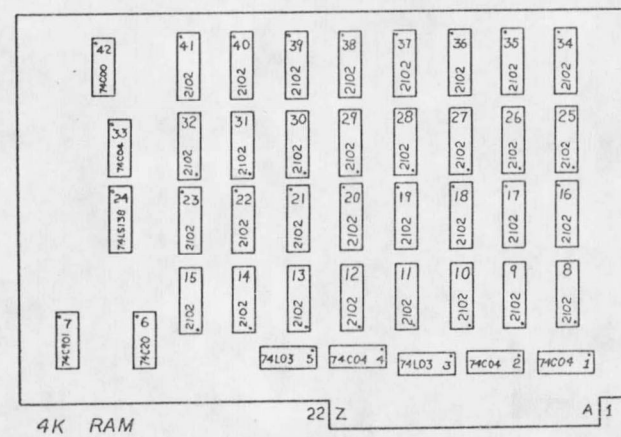
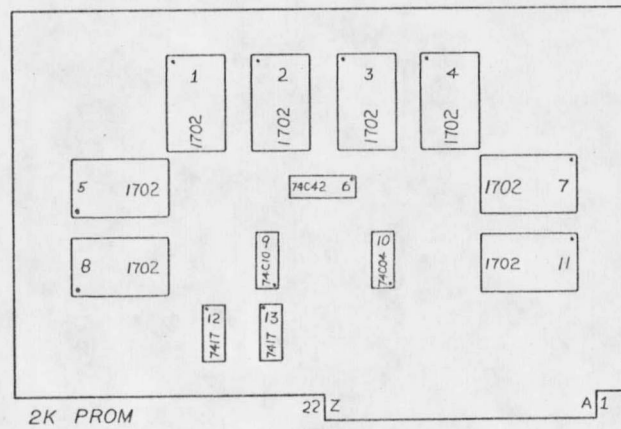
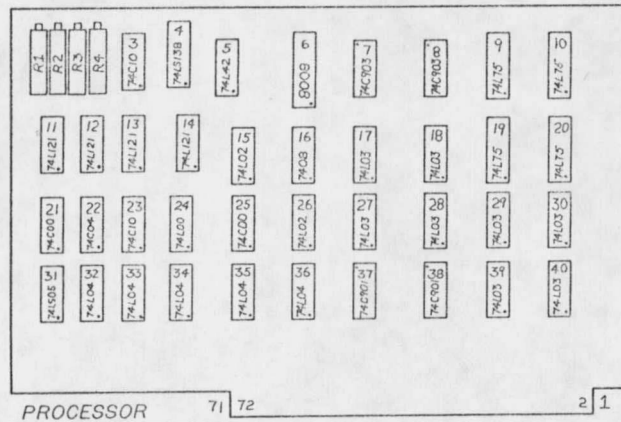


FIGURE 22: (continued)

MONTANA STATE UNIVERSITY LIBRARIES

3 1762 10014546 3

N378 Irvin, Darrell B
Ir8 An 8-bit microcomputer
cop.2 system processor, memory
 and I/O

[illegible]