MOTOROLA MC68701 MICROCOMPUTER INTERFACE FOR THE HP6940B MULTIPROGRAMMER

by

NAGABUSHAN K. BHASKARA

B.E., Bangalore University (India), 1980

A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1983

Approved by

M S Lucas

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

A. MOTIVATION.

This report describes the use of the Motorola MC68701 microcomputer to provide an interface between a Hewlett Packard Multiprogrammer and a North Star Horizon II computer. The MC68701 interface provides access to the Multiprogrammer via either a computer or a teletype. The interface board performs RS-232C to 16-bit conversion between the Horizon computer and the Multiprogrammer [7].

Fig. 1 shows the set-up of the MC68701 interface system. The Multiprogrammer has parallel input and parallel output ports whereas the computer has a serial output port. Thus for data transfer, serial to parallel and parallel to serial conversions are necessary. Also the Multiprogrammer requires TTL/DTL logic levels whereas the computer and teletype are RS-232C. A conversion in the levels has to be accomplished for their communication. Finally the Horizon computer expects the inputs to be ASCII (American Standard Code for Information Interchange) coded while the Multiprogrammer expects the inputs to be binary and returns a binary output.

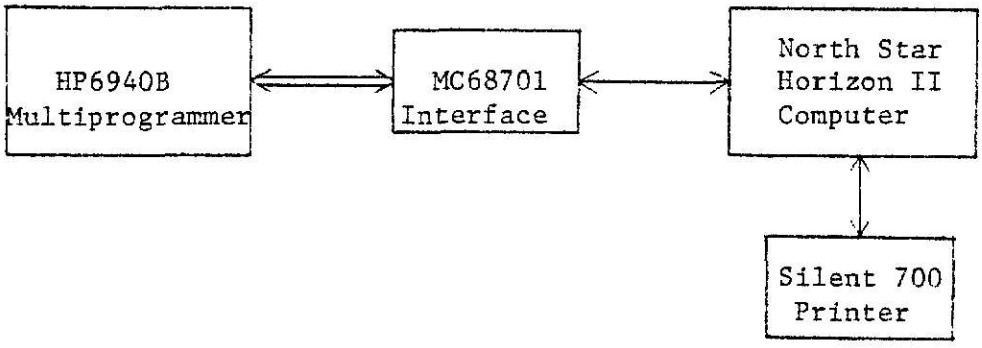


Fig. 1

Block Diagram of the Interface.

The serial to parallel conversion and vice versa are accomplished by the Serial Communications Interface (SCI) in the MC68701 processor. The level change from RS-232C to TTL is accomplished through the hardware. Level translators are used for this purpose. The quad line driver, MC 1488 is used for the transfer from TTL to RS-232C and the quad line receiver, MC 1489 is used for the transfer from RS-232C to TTL. The binary to ASCII and ASCII to binary conversions are handled in the software.

B. REDUCTION IN HARDWARE BY USING THE MC68701

In a microcomputer based system the required task can be performed only by an appropriate selection of the microprocessor. The selection criteria are of importance for the successful completion of the job. An enormous reduction of hardware can be achieved by selecting a suitable processor.

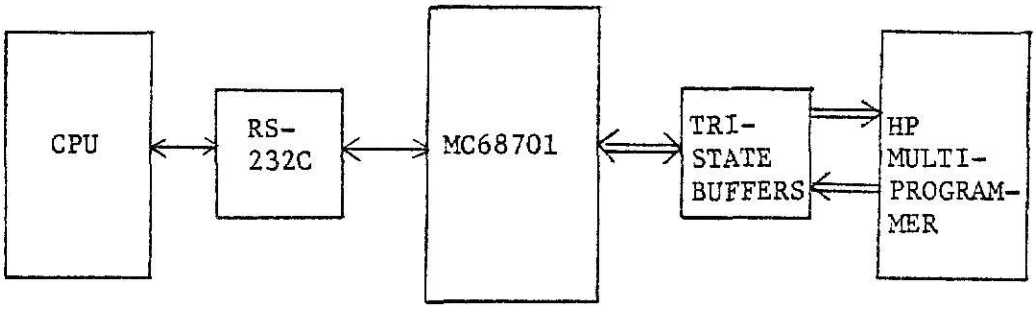


Fig. 2a

Interface Using the MC68701 Microcomputer

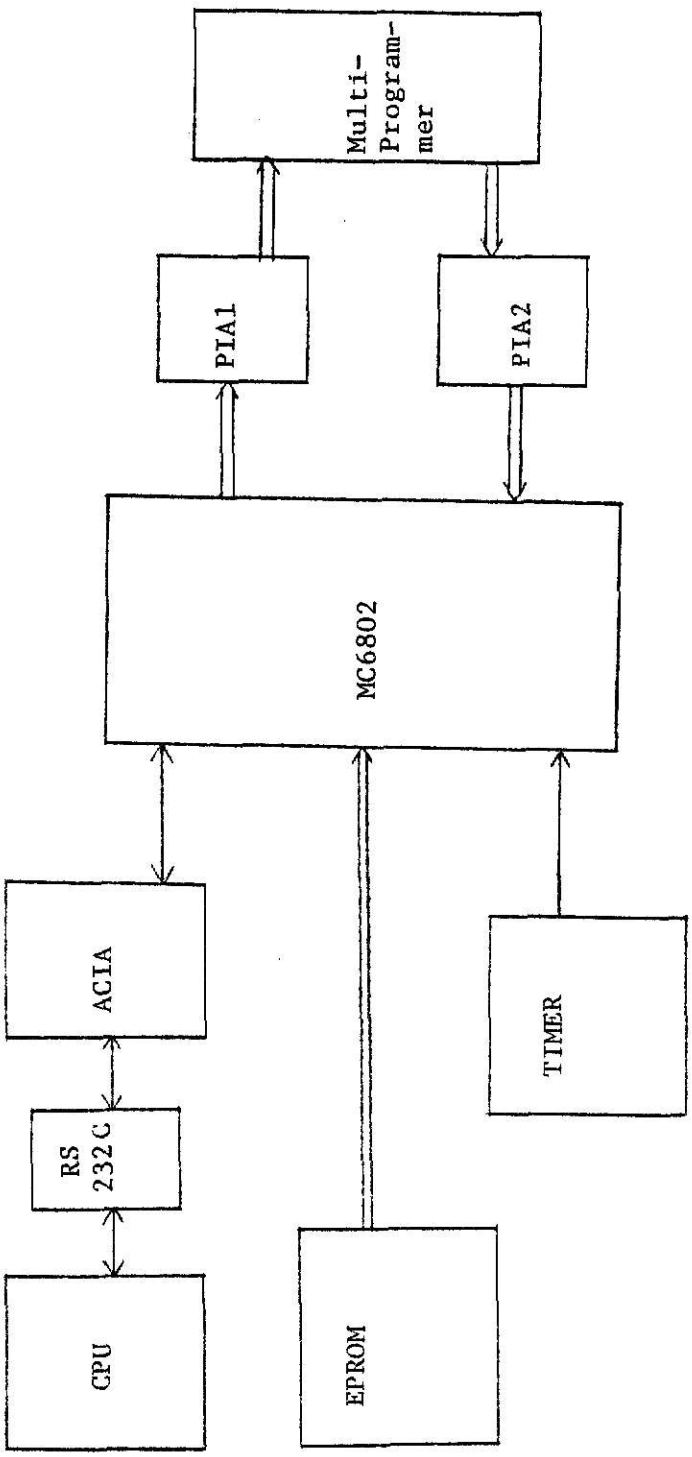


Fig. 2b

INTERFACE USING THE MC6802 MICROPROCESSOR

The Motorola MC68701 microcomputer unit is chosen here for interfacing the HP Multiprogrammer to the North Star computer, where it replaces an earlier system composed of:

1. MC 6802 8-bit microprocessor
2. MC 6850 Asynchronous Communications Interface Adapter
3. Two MC 6821 Peripheral Interface Adapters
4. MC 6840 Programmable Timer
5. 2716 Erasable Programmable Read Only Memory.

The MC68701 is strictly to be considered as a microcomputer unit rather than a microprocessor unit as it possesses RAM, EPROM, SCI and parallel I/O ports.

Fig. 2a shows the interface for the Multiprogrammer using the MC68701 microcomputer. Fig. 2b shows the interface using the MC6802 microprocessor. It can be seen that there is a great reduction in the hardware by using the former.

C. MULTIPROGRAMMER

The Multiprogrammer is a unit that holds power supplies, relay cards and has several slots for housing hardware modules such as a voltage/frequency converter, digital/analog converter, counter and signal generator. It has a 16-bit input and a 16-bit output, controlling up to 240 12-bit I/O channels. The computer communicates with the Multiprogrammer through four types of 16-bit words:

1. Control Word
2. Data Word
3. Address Word
4. Return Data Word.

CONTROL WORD

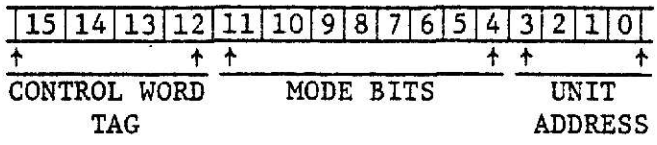


Fig. 3.

Control Word Format

The Control Word format is shown in Fig. 3. To select an input or output mode of operation, control word is sent to the Multiprogrammer from the computer, indicating to the mainframe the slot that is to be selected. Bits 0-4, the unit address portion of the control word specifies which of the 16 mainframe units is to be selected. Bits 4-11 indicate the status during operation - Timing mode Enable, System Enable, Data transfer Enable, etc.

DATA WORD

The Data Word format is shown in Fig. 4. The control word

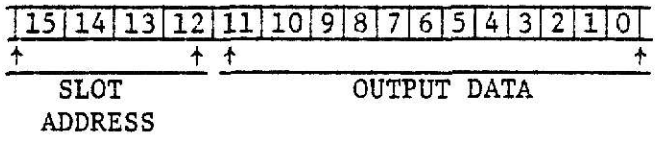


Fig. 4

Data Word Format.

selects an output card in the mainframe. A data word is sent to the output card from the computer. One of the input/output slots in the mainframe is selected by the four MSBs of the data word. Bits 0-11 contain the data that are to be stored on the output card.

ADDRESS WORD

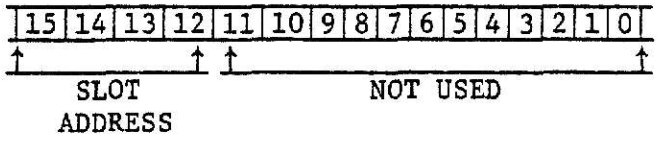


Fig. 5

Address Word Format.

A slot is selected by an address word. This slot holds an input card in the mainframe specified by the control word. The four MSBS of the word specify the slot address and the other 12 bits are unused.

RETURN DATA WORD

Data are sent to the input register of the computer from the input card of the Multiprogrammer with a return data word. The most significant bit is the interrupt request bit. Bits 12 through 14 are not used and bits 0 through 11 hold the input data. The return data word format is shown in Fig. 6.

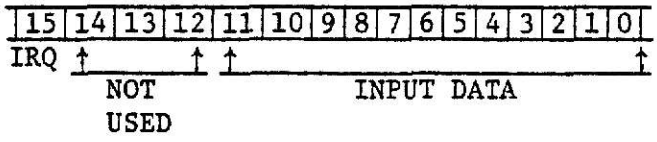


Fig. 6.

Return Data Word Format

CHAPTER II

HARDWARE DESCRIPTION

The block diagram of the interface hardware is shown in Fig. 7. It consists of:

1. RS-232C level translators.
2. Mode Select and Reset Logic.
3. MC68701 microcomputer unit.
4. Quad three state bus transceivers.

A. RS-232C STANDARD

The signals going in and coming out of a processor are not adequate to communicate with a peripheral controlled by the processor. Thus the signals are seldom sent directly to the interfaces. The terminals transfer characters of data in the ASCII form. Special hardware is used to form a standardized communication bus to which memory and peripherals are interfaced [3]. A widely used standard wherein the terminal uses serial communication is the Electronic Industry Associates (EIA) specification RS-232C standard.

All RS-232C signals must be within the limits shown below [2].

1. +3 Volts to +15 Volts for a zero.
2. -3 Volts to -15 Volts for a one.

Thus level translators are required between the TTL levels and the MODEMs. The level translators require additional power supplies (-12 V and +12 V) to generate the required signal voltages. The two popular translators are:

1. MC 1488: Quad TTL to RS-232C. This is for the data going to the MODEM.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

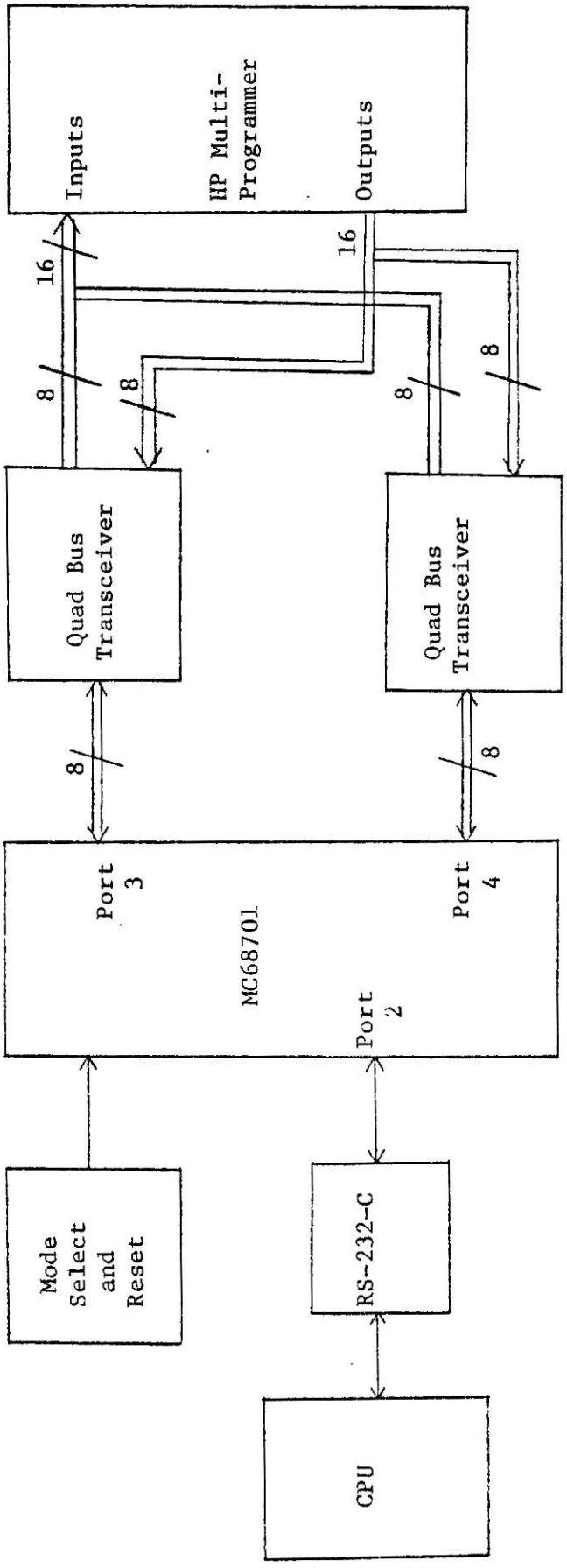


Fig. 7.

BLOCK DIAGRAM OF THE MC68701 INTERFACE HARDWARE.

2. MC 1489: Quad RS-232C to TTL. This is for the data coming from the MODEM to the device.

In all, the RS-232C interface consists of 25 data lines. Quite a few of these are undefined. In most computer terminals only 3 to 5 data lines are required for operation.

B. DESCRIPTION OF THE MC68701 MICROCOMPUTER

The MC68701 is a single chip microcomputer that is architecturally compatible with the 6800 family parts [10]. The onboard chip resources of the MC68701 microcomputer include:

1. 128 bytes of internal RAM
2. 2-K bytes of EPROM
3. Serial Communications Interface (SCI) - analogous to the Asynchronous Communications Interface Adapter
4. Parallel I/O ports - that could be used as Peripheral Interface Adapter.
5. Programmable Timer.

The Internal RAM is located at the hexadecimal addresses 0080 through 00FF and the internal EPROM resides at F800 through FFFF. The memory locations 0000 through 001F are reserved for the internal registers. The MCU has an access to a 64-K byte address space. A few hardware features of the processor are:

1. 8-bit Word Size
2. 16-bit Address Field
3. An internal oscillator with a divided by four circuit
4. M6800 Bus Compatible.

The instruction set for the MC68701 is the same as M6800 but it has a few added instructions. Accumulators A and B are concatenated to form accumulator D, which is 16-bit wide.

In all, the MC68701 has 29 parallel I/O lines shared by four ports 1,2,3 and 4. A simple block diagram of the MCU displaying its ports is shown in Fig. 8.

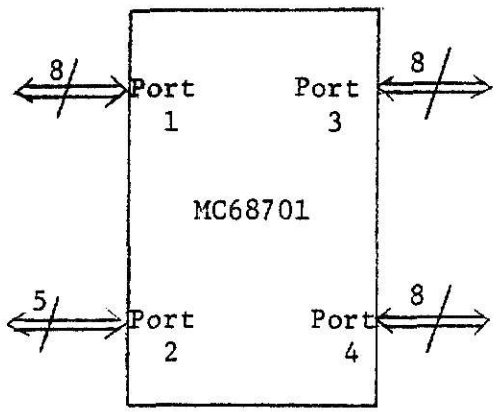


Fig. 8.

PORTS OF THE MC68701

An I/O device is interfaced to the microprocessor through an I/O port. The data transfer is accomplished through these ports. The ports can be programmed in order to control the direction of the data flow.

Port 1 and Port 4 have 8 I/O lines each. Port 3 besides having 8 I/O lines, has two control pins SC1 and SC2. The function of SC1 and SC2 depend upon the mode of operation of the MC68701. In the expanded multiplexed mode, SC1 functions as an Address Strobe (AS) and SC2

functions as a Read/$\overline{\text{Write}}$ (R/$\overline{\text{W}}$). The ports can be programmed by the user's program. Each port has a data register and a data direction register. The ports act as inputs or outputs depending upon the configuration of the bits in the data direction register. A one in the data direction register indicates that it is an output bit and a zero indicates that it is an input bit.

Port 2 is a 5-bit I/O port unlike the other ports which are 8-bit. This port is used as an interface for the Serial Communications Interface and the timer. The three least significant bits of the data register of Port 2 provide the operating mode for the MC68701. Port 3 in addition to the data register and data direction register has a control and status register to configure the control lines SC1 and SC2. Depending upon the mode of operation, Port 3 can be used either as a bidirectional data bus or a multiplexed address/data bus.

C. MODES OF OPERATION OF THE MC68701

The MC68701 can function in eight modes - Mode 0 through Mode 7. Mode 0 is a reserved mode exclusively used for programming the onboard EPROM. The modes are selected by connecting the switches to pins 8,9, and 10 of the processor as shown in Fig. 9. Upon RESET, the levels on the pins P20,P21, and P22 are latched into PC0, PC1 and PC2 bits of the data register of Port 2. The processor then recognizes that it has to operate in the required mode.

The eight operating modes of the MC68701 can be broadly classified under three major modes:

1. Single Chip Mode (Modes 4,7)

2. Expanded Non-Multiplexed Mode (Mode 5)

3. Expanded Multiplexed Mode (Mode 1,2,3 and 6)

Each of the above operating modes has a unique memory map, illustrating the allocation of the 64-K byte memory space. Invariably for all the modes, the first thirty two locations in memory are reserved for the internal registers. In modes 2, 3 and 4 the internal EPROM is removed from the memory map. Depending upon the user's application, RAMs or ROMs could be fitted into the external memory space in any mode. A decode logic circuit distinguishes the internal memory from the external memory.

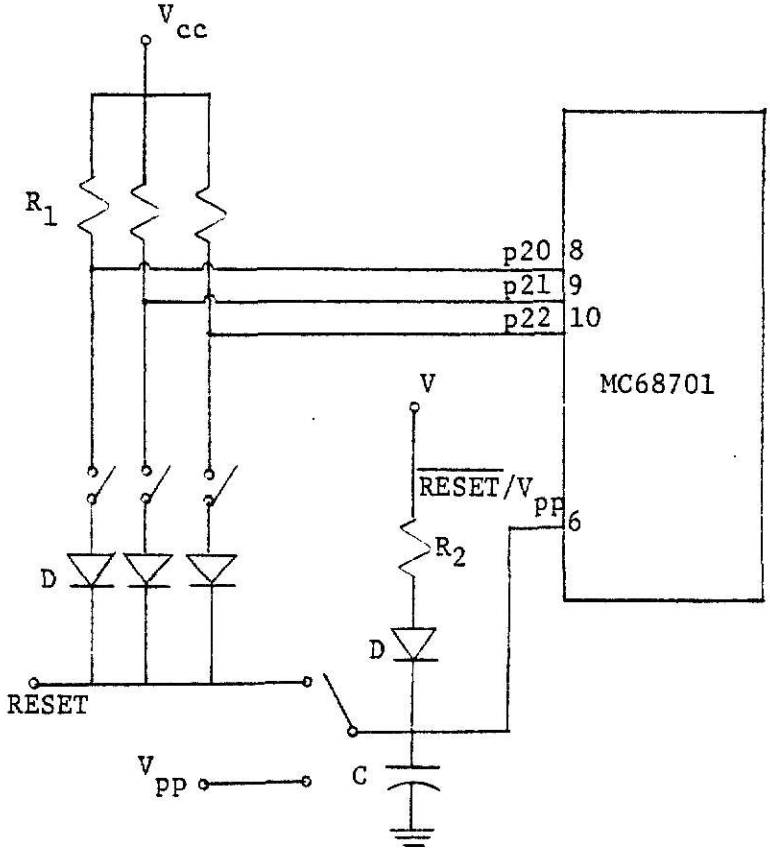


Fig. 9.

RESET LOGIC AND MODE CONFIGURATION.

In some modes the internal EPROM addresses FFF0 through FFFF are not usable as this area is assigned for the external interrupt vectors. Upon RESET, the processor fetches the RESET vector from an external RAM or ROM where the interrupt vectors reside.

D. MODE SELECT

The operational mode of the MC68701 that is selected here for the interfacing is the Single Chip Mode (Mode 7). The data transfer between the MC68701 and the Multiprogrammer requires 32 input/output lines, as the Multiprogrammer is a 16-bit input and 16-bit output unit. But the MC68701 has 3 parallel I/O ports (Ports 1,3, and 4) each having 8 I/O lines resulting in a total of only 24 lines. These ports are to be configured in such a manner that there are 32 lines.

By making use of the tristate buffers, two ports are sufficient for the data transfer between the MC68701 and the Multiprogrammer. In Mode 7, Ports 3 and 4 being the parallel I/O ports, can be used for the above purpose. Besides, there is no need for an external address bus as the external memories are not required. Above all, the onboard EPROM is included in the memory map of Mode 7. Taking into consideration the above factors, the Single Chip Mode was found to be the most feasible.

E. QUAD BUS TRANSCEIVER

A simple block diagram of a quad three state bus transceiver, MC8T26A is shown in Fig. 10. It consists of 4 bus inputs, 4 receiver outputs, 4 driver inputs and 2 control lines (Receiver Enable Input and Driver Enable Input). A low signal on the Receiver Enable Input

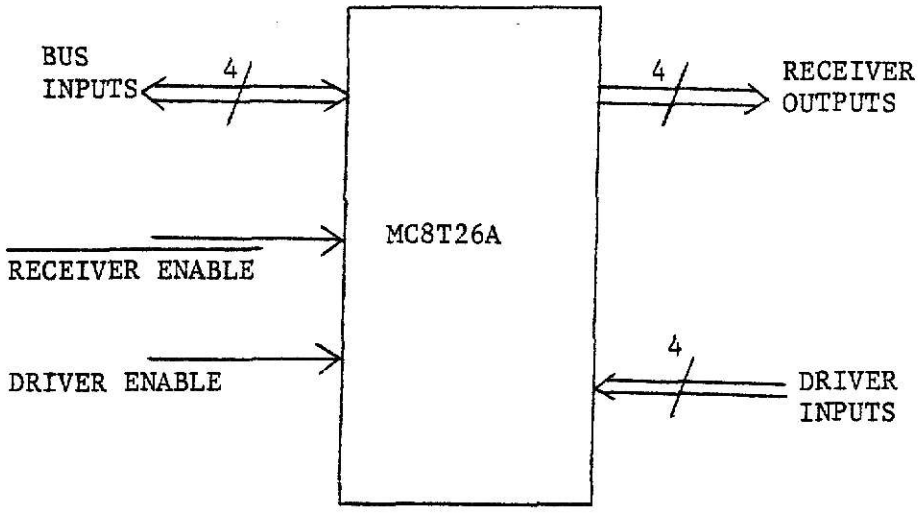


Fig. 10.

BLOCK DIAGRAM OF TRI-STATE TRANSCEIVER.

enables the receiver outputs and a high signal on the Driver Enable Input enables the driver inputs.

A bank of 4 quad bus transceivers is connected between the processor and the Multiprogrammer as shown in Fig. 11. The I/O lines of Port 3 and Port 4 form the bus inputs to the tristate buffers. The receiver outputs are connected to the inputs of the Multiprogrammer and the driver inputs are connected to the outputs of the Multiprogrammer. Thus there are 16 lines going to the Multiprogrammer inputs and 16 lines coming from the outputs. The receiver enable inputs and the driver enable inputs of all the transceivers are shorted.

The software determines if the control pin is to be high or low. When the Receiver Enable Input is low, the receiver outputs are enabled and the data on the ports 3 and 4 is transferred to the inputs of the Multiprogrammer. At the same time the Driver Enable Input

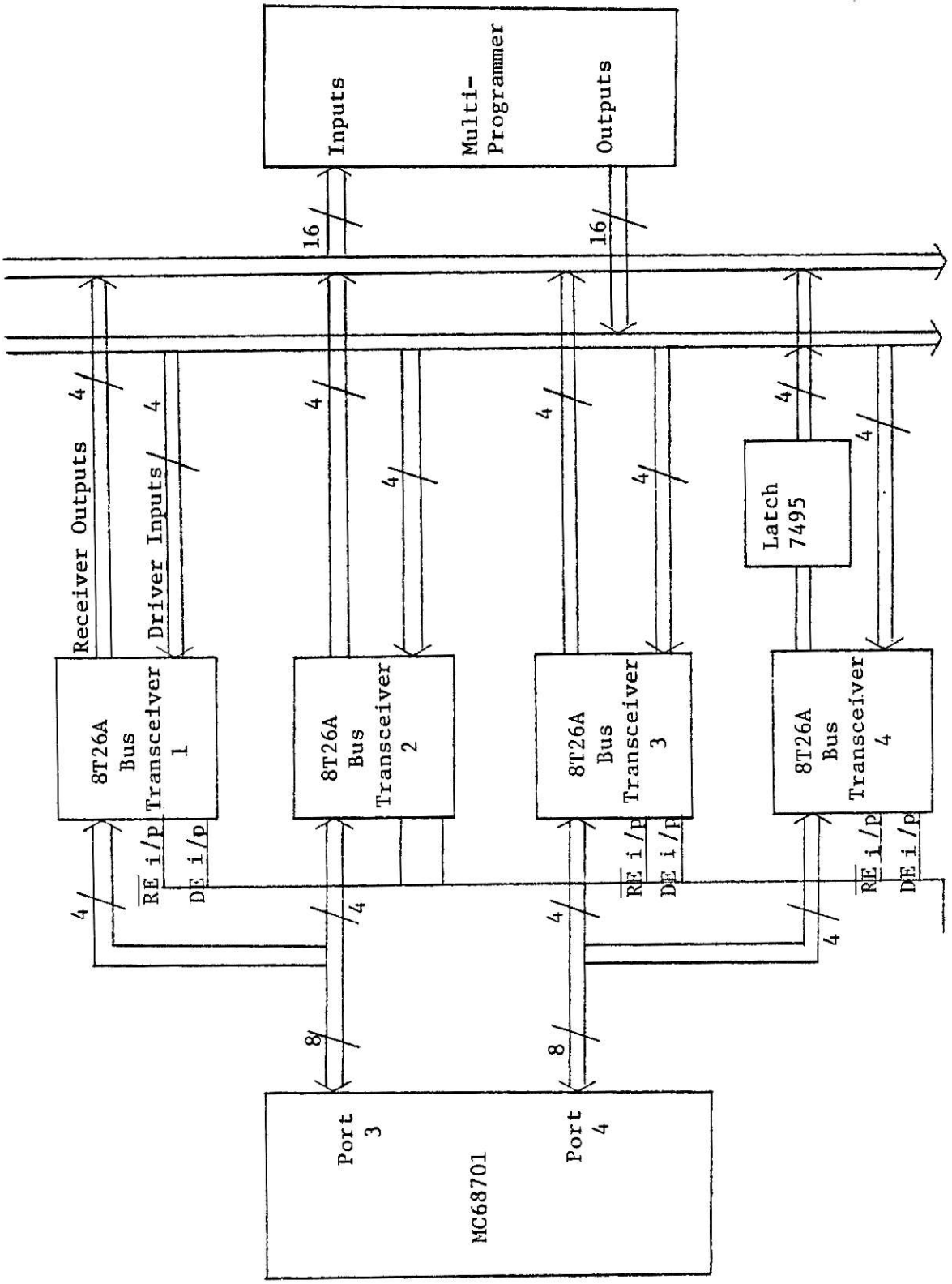


Fig. 11.

CIRCUIT ILLUSTRATING THE TRISTATE BUFFERS CONNECTION

(being shorted to the Receiver Enable Input) is low disabling the driver inputs. When there is a high signal on the Driver Enable Input the data at the outputs of the Multiprogrammer are sent to the ports of the processor.

F. CONFIGURATION OF PORT 1.

Port 1 is configured as shown in Fig. 12. Bit 0 is connected to the control of the tristate buffers. Bit 1 is connected to the gate of the Multiprogrammer through a buffer stage. The buffer is designed to provide sufficient current to drive the gate input of the Multiprogrammer. Bit 2 is connected to the flag output of the Multiprogrammer and bit 3 is connected to the clock input of the 7495 latch.

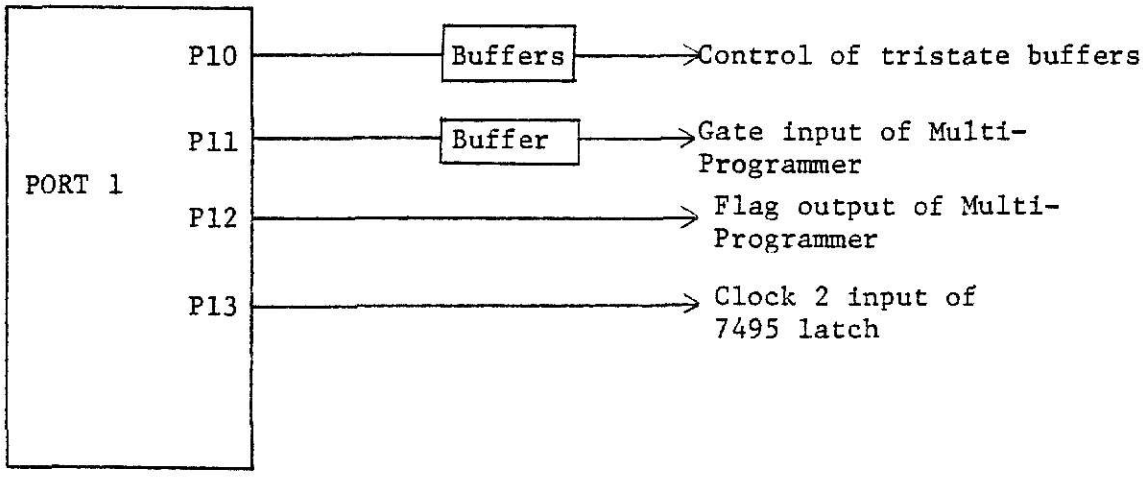


Fig. 12.

CONFIGURATION OF PORT 1.

G. DESCRIPTION OF HARDWARE OPERATION

The schematic diagram of the MC68701 interface is shown in Fig. 13. The resistor-diode circuit connected to pins 8,9 and 10 of the processor is for selecting the required mode of operation. In this application the switches S1, S2 and S3 are open so that the processor operates in Mode 7. A crystal of 4.9152 MHz is connected to pins 3 and 4 of the processor. This frequency is chosen so that, upon proper configuration of the bits in the Rate and Mode Control Register, standard baud rates are available. In particular this frequency also provides a baud rate of 9600. Pins 4 and 5 (being Non-Maskable Interrupt and Interrupt Request) are tied high to disable the interrupts.

Pins 11 and 12 of the processor are connected to the level translators, MC1488 and MC1489 allowing for the Serial Communications Interface (SCI) to accept the serial data characters from the terminal or to transmit the bits serially to an asynchronous serial data device. The SCI has two control registers: Rate and Mode Control Register and Transmit/Receive Control and Status Register. The SCI is initialized by initializing these two registers. The least two significant bits in the Rate and Mode Control Register provide the standard baud rates of 300, 1200, 9600 and 76,800. Depending upon the requirement the bit configuration is done in the software.

Ports 3 and 4 are configured either as input ports or output ports depending on the data transfer between the processor and the Multiprogrammer. One of the I/O lines of Port 1 is used to indicate to the processor the direction of the data flow. If the bit is low, the data flow is from the processor to the Multiprogrammer and if the

ILLEGIBLE DOCUMENT

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

THIS IS THE BEST COPY AVAILABLE

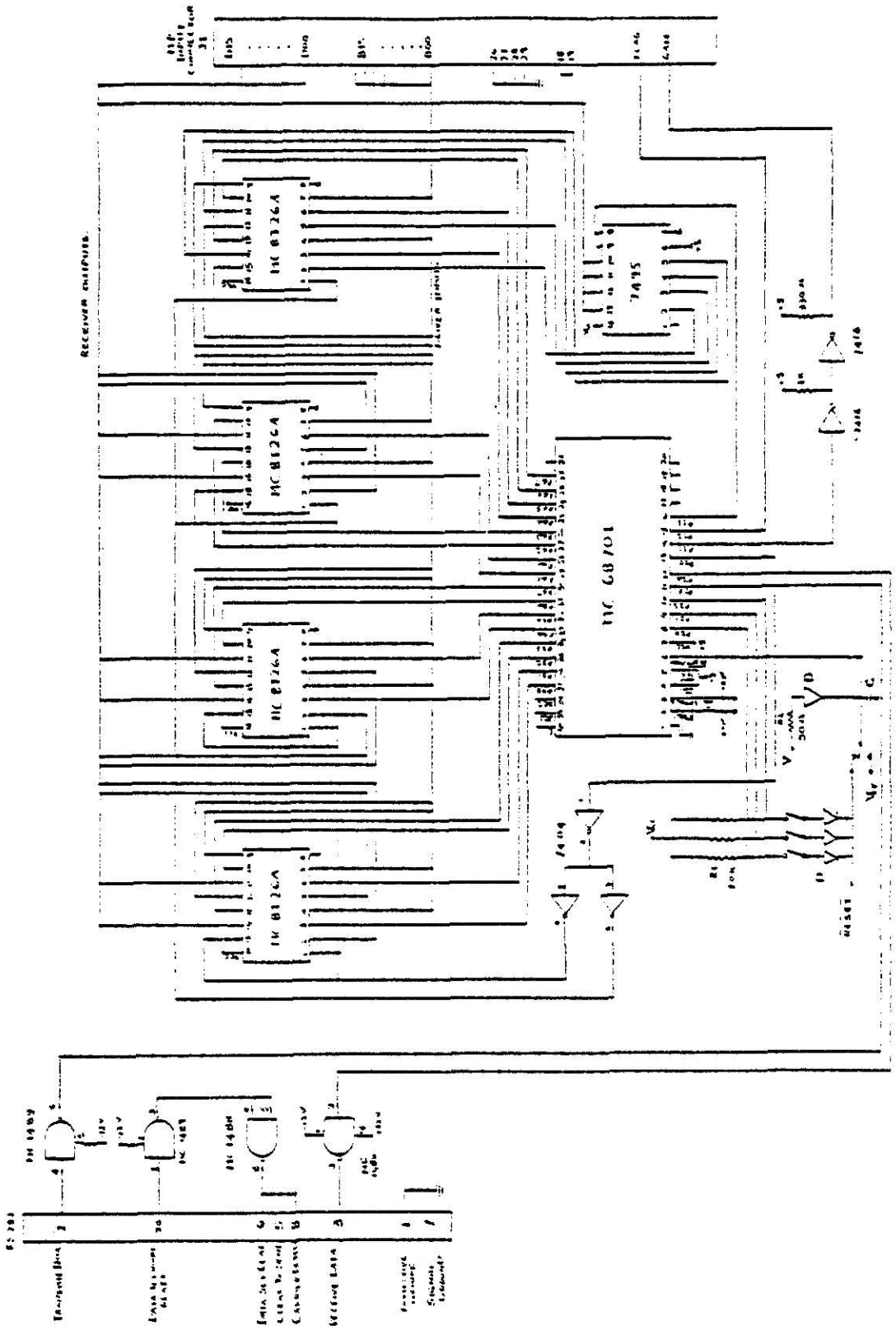


Fig. 13.

SCHEMATIC DIAGRAM OF THE INTERFACE HARDWARE

bit is high the data flow is vice versa. The software determines if this bit is to be high or low. This bit of Port 1 is connected to the control line (i.e., Receiver Enable Input and Driver Enable Input shorted) of the quad bus transceivers.

The two main functions of the interface are:

1. To provide a given binary number to the Multiprogrammer.
2. To read the data from the Multiprogrammer.

The steps involved in providing a binary number to the Multiprogrammer are:

1. The SCI accepts the ASCII characters from the terminal through the RS-232 level translator.
2. The software makes the ASCII to BCD to binary conversion.
3. Ports 3 and 4 are configured as the output ports.
4. Bit 0 of Port 1 is made low. (This drives the Receiver Enable Input of the bus transceivers low, thus enabling the binary data to be sent to the Multiprogrammer.)

The steps involved in reading the data from the outputs of the Multiprogrammer are:

1. Ports 3 and 4 are configured as the input ports.
2. Bit 0 of Port 1 is made high. (This drives the Driver Enable Input of the bus transceivers high, thus enabling the processor to read the data from the outputs of the Multiprogrammer.)
3. The software makes the binary to BCD to ASCII conversion.
4. The SCI sends the bits serially to the computer.

H. SOFTWARE DESCRIPTION

The software controls the direction of the data transfer and is also responsible for the ASCII to BCD to Binary Conversion and vice-versa. If the interface receives an "&", the data transfer is from the computer to the Multiprogrammer, and if it receives a "?", the data transfer is vice-versa. To begin with, all the data registers, data direction registers, control registers and stack pointer are initialized.

The following steps are involved if an "&" is recognized:

1. Port 1 (excepting bit 2) is configured as an output port. Bit 2 is configured as an input.
2. Ports 3 and 4 are configured as output ports.
3. A low signal is sent to the control of the tristate buffers, enabling the receiver outputs of the transceivers. The gate of the Multiprogrammer is set high initially.
4. ASCII to BCD to Binary conversion [7] is performed.
5. The binary data is stored on Ports 3 and 4.
6. A software delay is created so that the data on the interface lines becomes stabilized.
7. The gate of the Multiprogrammer is pulled low, indicating to it that the data is ready.
8. A software loop is set up to see if the flag of the Multiprogrammer is switched back to the READY state from the BUSY state.
9. The gate of the Multiprogrammer is pulled high.

The timing diagram [9] of the handshaking between the Multiprogrammer and the computer is shown in Fig. 14.

The following steps are involved if a "?" is recognized:

1. Port 1 (excepting bit 2) is configured as an output port. Bit 2 is configured as an input.
2. Ports 3 and 4 are configured as input ports.
3. A high signal is sent to the control of the tristate buffers, enabling the driver inputs of the transceiver. The gate of the Multiprogrammer is set high.
4. Binary to BCD to ASCII conversion [7] is performed.
5. The ASCII data are sent serially to the computer.

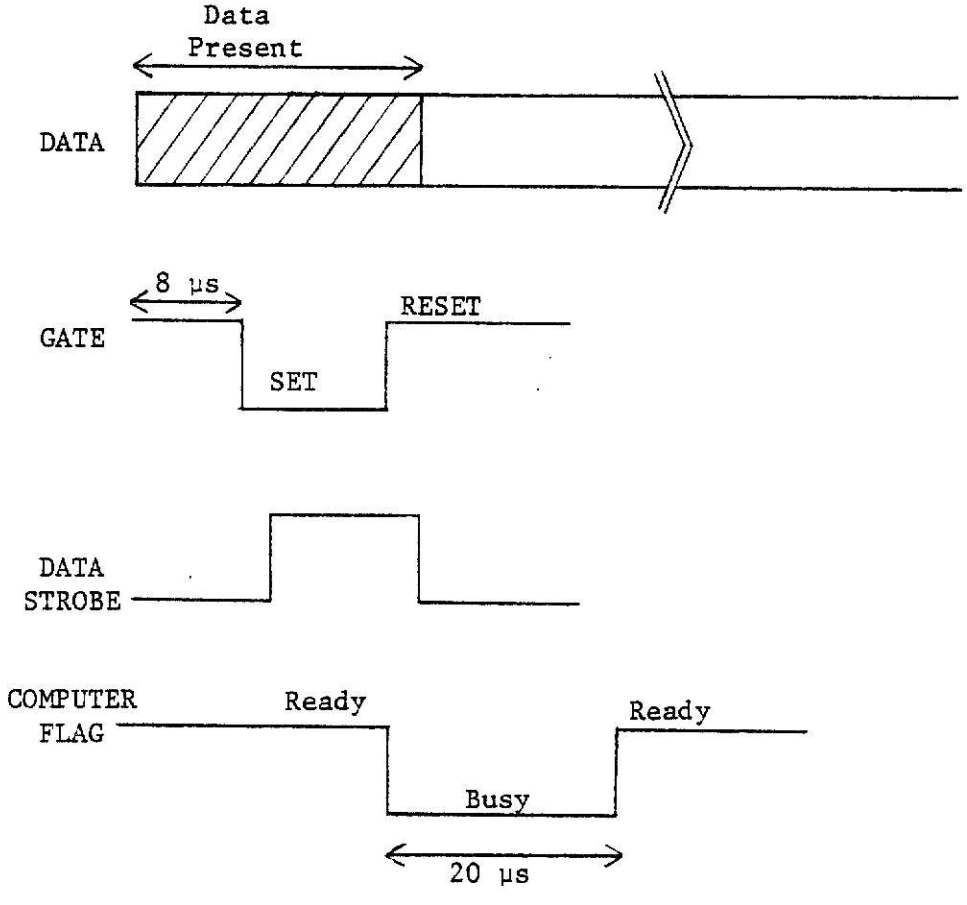


Fig. 14.

HANDSHAKE TIMING DIAGRAM
(Taken from Operating and Service Manual for Multiprogrammer)

CHAPTER III

PROGRAMMING THE MC68701 ONBOARD EPROM.

One of the excellent features of the MC68701 microcomputer [10, 11] is its ability to program itself. The processor has an onboard 2-K byte Erasable Programmable Read Only Memory (EPROM). Mode 0 is exclusively used to program the onboard EPROM. This mode is an expanded multiplexed mode wherein the MCU has an ability to access a 64-K byte memory space.

The memory map of Mode 0 is shown in Fig. 15. The important features of this mode are:

1. 128 bytes of internal RAM
2. 2-K bytes of internal EPROM
3. Port-3 is a multiplexed address/data bus
4. Port-4 is an address bus
5. SC1 is an Address Strobe (AS)
6. SC2 is Read/Write (R/W)

Port 3 functions as a time multiplexed address/data bus, wherein the address is valid on the negative edge of the Address Strobe and the data are valid when the Enable (E) is high. The address lines A0 to A7 are provided by Port 3 by connecting the AS of the processor to the control of an 8-bit D-type latch. The remaining address lines A8 to A15 are provided by Port 4. The timing diagram of the expanded multiplexed mode is shown in Fig. 16.

The interrupt vectors in Mode 0 are external to the processor and are located at the hexadecimal addresses BFF0 through BFFF. In particular, the $\overline{\text{RESET}}$ vector is at BFFE and BFFF.

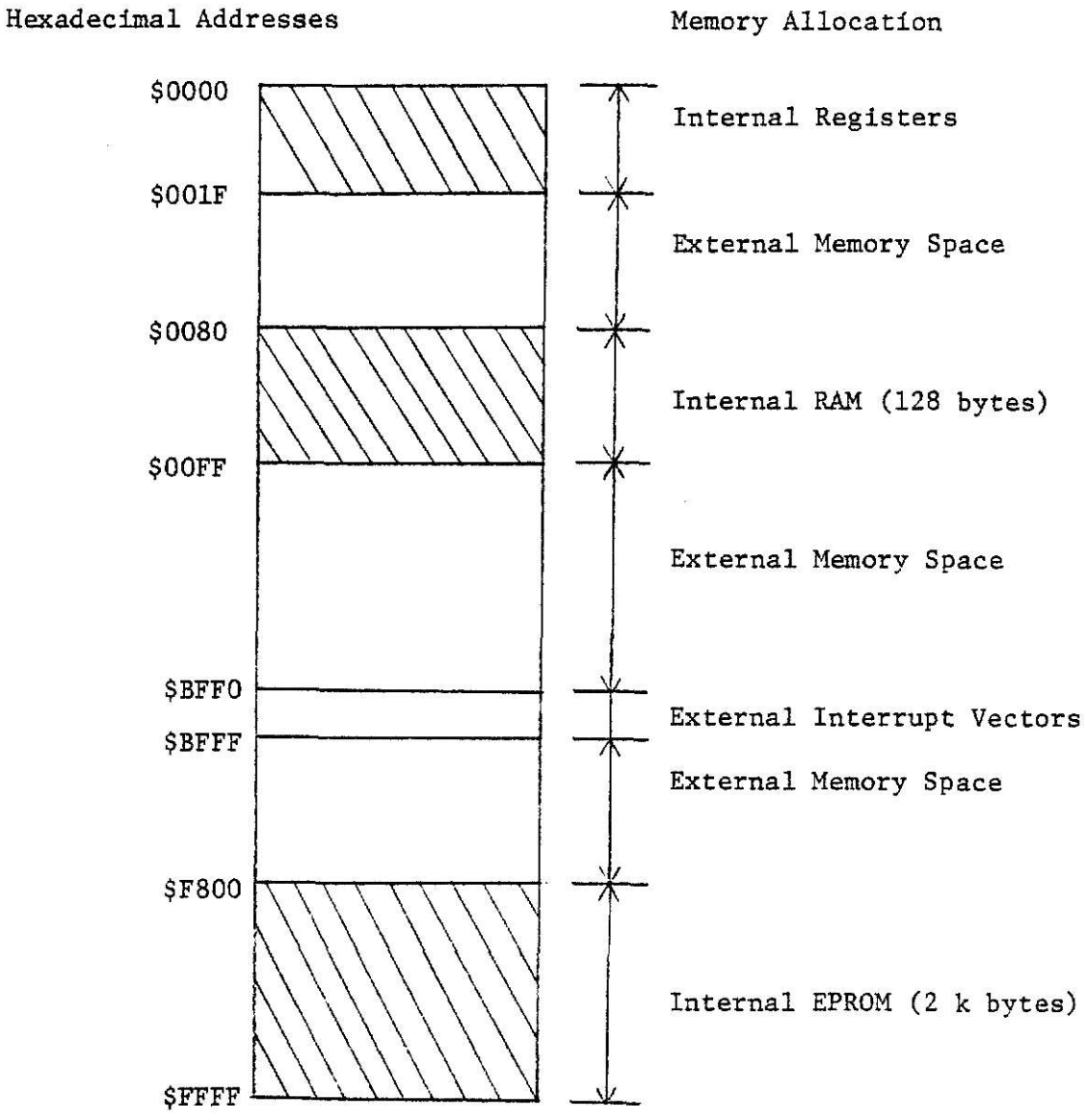


Fig. 15.

MEMORY MAP FOR MODE 0.

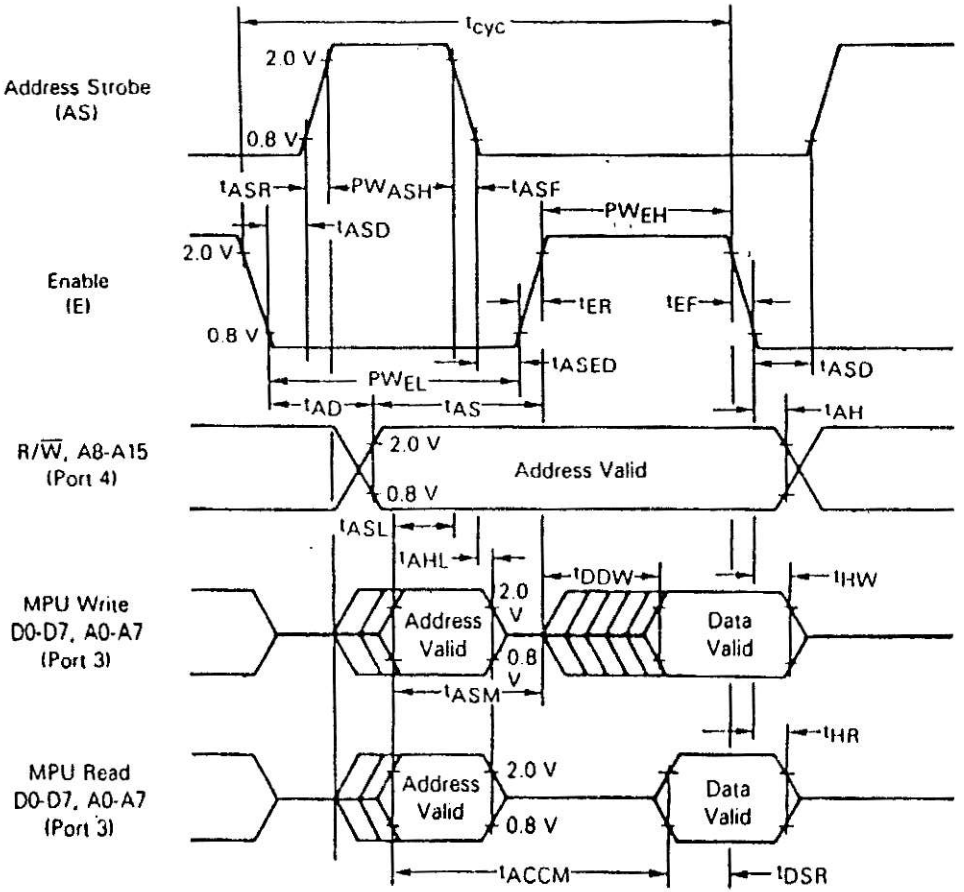


Fig. 16.

TIMING DIAGRAM OF THE EXPANDED MULTIPLEXED MODE
(Taken from Motorola MC68701 Microcomputer Unit Manual)

In order to program the onboard EPROM, the MC68701 is operated in Mode 0. The $\overline{\text{RESET}}$ vector is fetched from the locations BFFE and BFFF and directs the processor to come under the control of an external program, which is the programming routine program that resides in a 2716 EPROM. This bootstrap program will fetch data sequentially from another Data ROM (another 2716 EPROM). The Data ROM contains the program code that is to be programmed into the MC68701.

The programming circuit thus requires two EPROMs to be connected externally [8]. As both the internal and external data buses are connected in Mode 0, care must be taken to see that there is no memory map overlap. In other words, there must not be any bus conflicts.

The hexadecimal address space B800 through BFFF is allocated to the programming routine EPROM. This address space is chosen so that the interrupt vectors can be fetched from BFF0 through BFFF. The Data EPROM is fitted in the memory space 8000 through 87FF.

Thus the memory map consists of five address spaces:

1.	Internal Register Area	0000 through 001F
2.	Internal RAM	0080 through 00FF
3.	Data EPROM	8000 through 87FF
4.	Programming Routine EPROM	B800 through BFFF
5.	Internal EPROM	F800 through FFFF

The memory map in Fig. 17 shows the above allocation clearly.

The most significant bits of the address lines are to be decoded for the proper selection of the required EPROM. It can be seen from Table 1 that the MSBs A13, A14 and A15 are used for decoding.

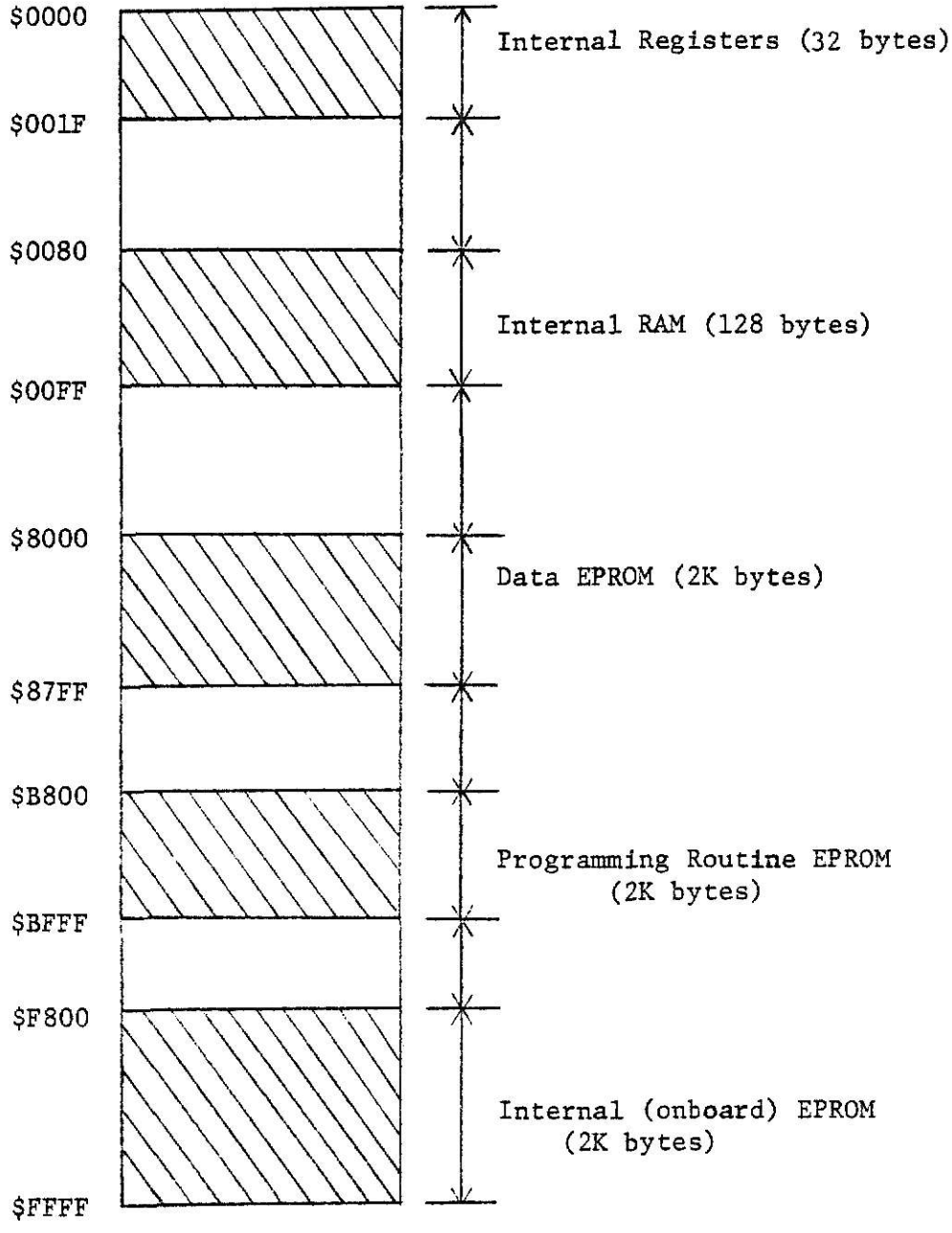


Fig. 17.

MEMORY MAP FOR PROGRAMMING CIRCUIT.

	RAM/ROM	Starting Address	MSBS A15	A14	A13	A12
1	Internal RAM	$0080	0	0	0	0
2	Data EPROM	$8000	1	0	0	0
3	Prog. Routine EPROM	$B800	1	0	1	1
4	Onboard EPROM	$F800	1	1	1	1

Table 1

NAND gates and inverters are used for the address decoding of the two external EPROMs. The Data EPROM is selected with A15=1, A14=0 and A13=0. The programming routine EPROM is selected with A15=1, A14=0 and A13=1. The decode logic for the selection of the EPROMs is shown in Fig. 18.

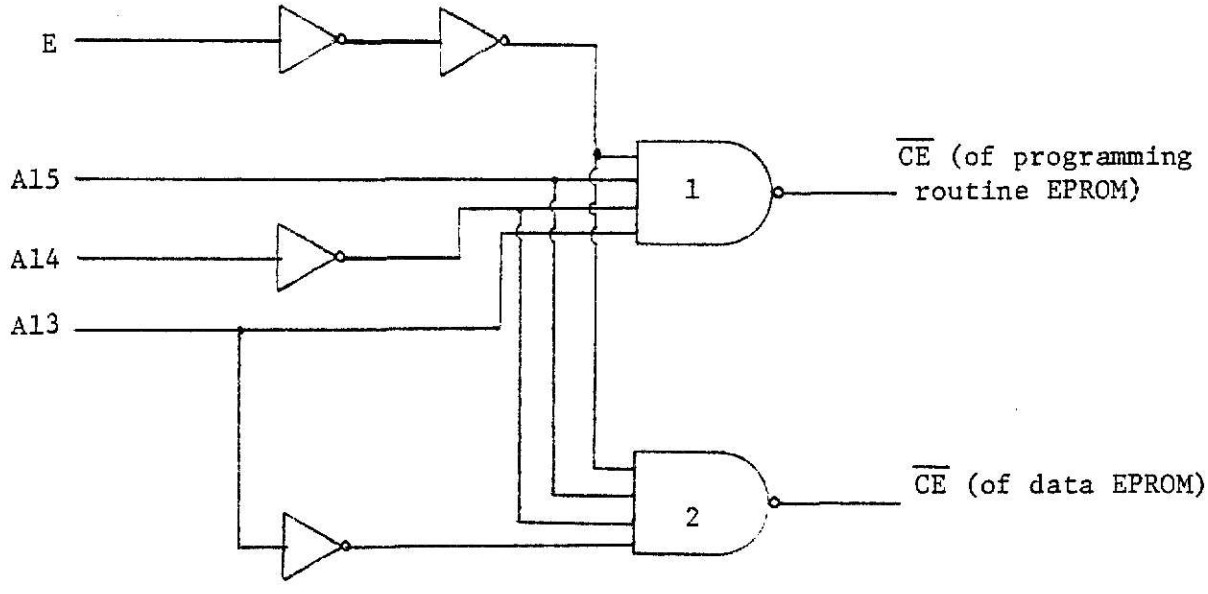


Fig. 18.

Decode Logic Diagram.

Both the EPROMS are selected with high A15 and low A14. A high A13 and high E make the output of NAND gate 1 a zero. The output of NAND gate 1 is connected to the chip select ($\overline{CE}$) of the programming routine EPROM. This being an active low signal is enabled by the low output of NAND gate 1 and the programming EPROM is selected. Similarly a low A13 and high E make the output of NAND gate 2 a zero. The output of NAND gate 2 is connected to the chip select ($\overline{CE}$) of the Data EPROM. The low signal on $\overline{CE}$ selects the DATA EPROM. The decode logic thus assures separation between the off-chip and the on-chip EPROM address spaces.

A. PROGRAMMING CIRCUIT DESCRIPTION

The block diagram of the programming circuit is shown in Fig. 19. It is comprised of:

1. MC68701 microcomputer unit.
2. Intel 2716 EPROM containing the programming routine.
3. Intel 2716 EPROM containing the data that is to be programmed into the MC68701.
4. 74LS373 8-bit latch.

A crystal depending upon the minimum and maximum frequencies of operation of the processor, is chosen. A 4-Mhz crystal is used to provide a 1-MHz operation for the MC68701 processor and a 4.9152-Mhz crystal is used to provide a 1.2288 MHz operation for the MC68701-1 processor. Both the leads of the crystal are tied to the ground through capacitors to ensure stable crystal operation. Alternatively the processor can also be driven by an external TTL clock at pin 3 with pin 2 grounded.

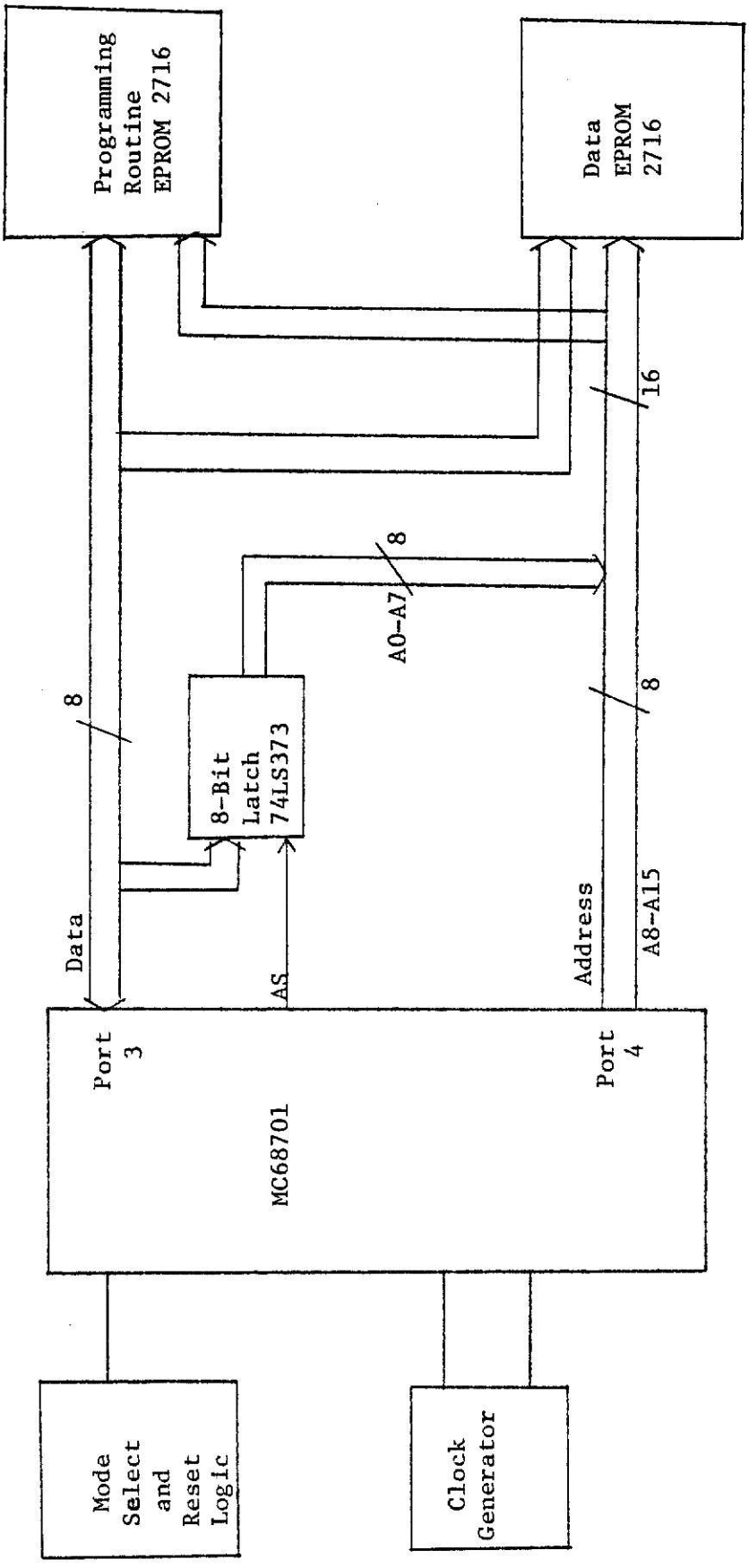


Fig. 19.

BLOCK DIAGRAM OF PROGRAMMING CIRCUIT.

The schematic diagram of the programming circuit is shown in Fig. 20. To begin with, the pins 8,9 and 10 are grounded so that the processor operates in Mode 0 at RESET. In order to disable the external interrupts, pins 4 and 5 (interrupt request and non-maskable interrupt) are tied high. Pin 6 (RESET/VPP) serves as a dual purpose pin - to reset the processor as well as to apply the programming power to the EPROM. The pin is usually +5 Volts during nonprogramming operation and is increased to VPP (21 Volts) during programming of the internal EPROM.

Three LEDs are connected to the I/O port 1, for indicating the state of the MC68701 EPROM during programming.

1. LED 1 connected to pin 13 of the processor, when lit, indicates that the internal EPROM of the MC68701 is initially fully erased.

2. LED 2 connected to pin 14 of the processor, when lit, indicates that the internal EPROM has been programmed and its contents verified with that of the Data EPROM.

3. LED 3 connected to pin 14 of the processor, when lit, indicates that the EPROM was partially erased and hence failed to verify the contents after programming.

An 8-bit latch (74LS373) is used to demultiplex port 3 which serves both as an address port and a data port. The address strobe of the processor tied to the latch enable of the 74LS373 latches the lower order address (A0 through A7) on the negative edge of the Address Strobe. On completion of the address latching, the port is used for transferring data.

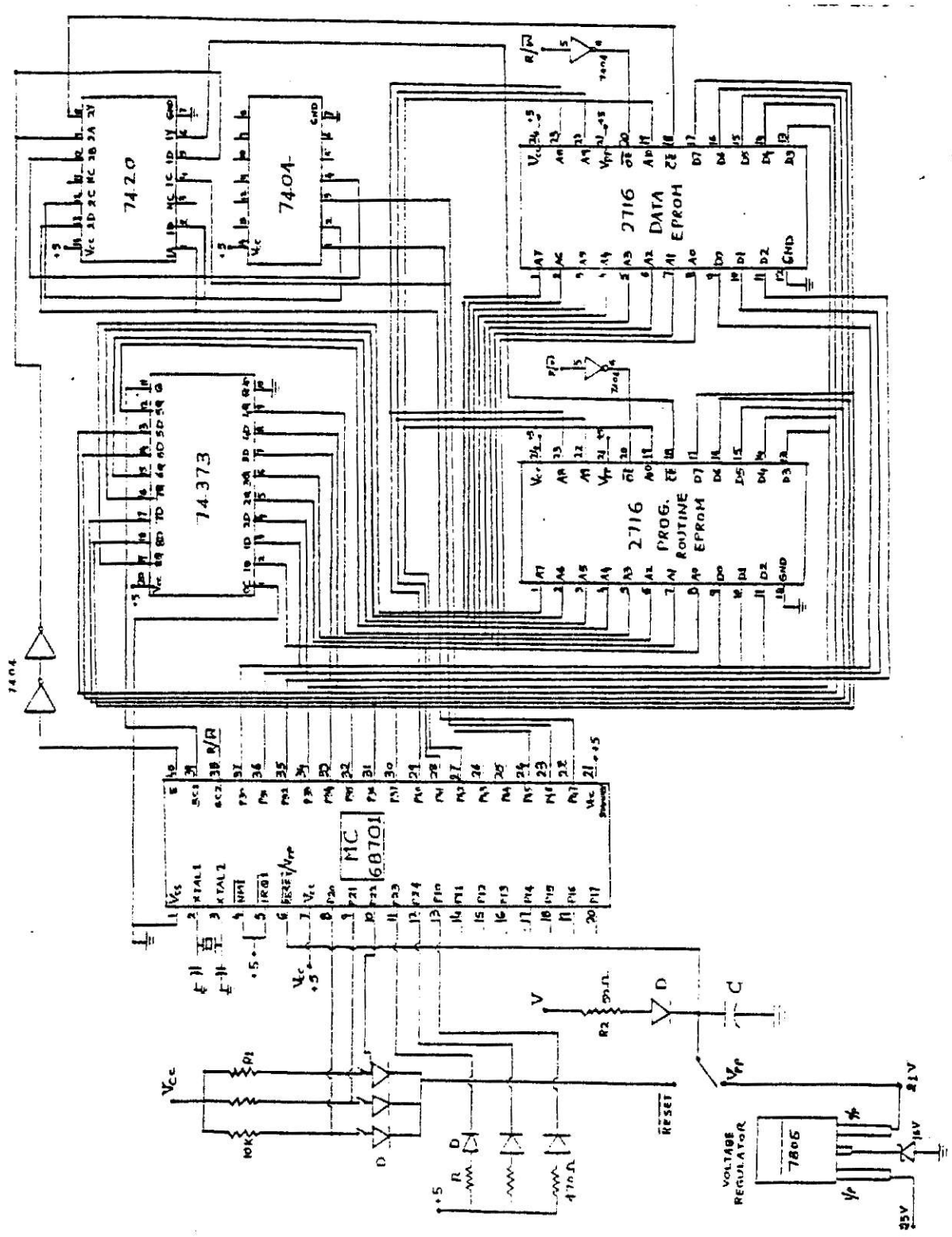


Fig. 20. SCHEMATIC DIAGRAM OF PROGRAMMING CIRCUIT

B. INITIALIZATIONS

Prior to programming, the processor expects a few initializations to be made [8]. The values of four variables namely IMBEG, IMEND, PNTR and WAIT are to be stored in the internal RAM of the processor. These variables are described below.

1. IMBEG: An address pointing to the first byte to be programmed into the onboard EPROM.
2. IMEND: An address pointing to the last byte to be programmed into the internal EPROM.
3. PNTR: A pointer (address) pointing to the first internal EPROM location to be programmed.
4. WAIT: A value which is used to generate the programming time delay.

The value that is to be stored in the variable WAIT is calculated using the relation:

$$\text{WAIT} = \frac{50{,}000 \times (\text{MCU Input Frequency in Hz})}{4 \times 10^6}$$

Fig. 21 gives a clear idea of the above variables.

C. STEPS INVOLVED IN PROGRAMMING

The programming routine EPROM contains the routine that directs the processor to go to the Data EPROM and fetch data sequentially. These data are then burned in the intenal EPROM at the desired addresses. The external EPROM plays an important role in programming. The program [8] written is M6800 assembly language with a few added instructions. This program is also called as the bootstrap program.

The steps involved in programming the MC68701 are:

1. Initialize the stack pointer.
2. Initialize the RAM/EPROM control register.
3. Establish a part of port 1 (least 3 significant bits) as an output port. The LEDs which indicate the state of the MC68701 EPROM during programming are connected to these output bits.
4. Verify if the onboard EPROM is initially erased.
5. Store the values of IMBEG, IMEND, PNTR and WAIT in the internal RAM of the processor.
6. Clear the PLC bit in the RAM/EPROM control register enabling the EPROM address latch. Set the PPC bit disabling the EPROM programming power.
7. Get data from the address pointed by the IMBEG. Store it in the internal EPROM in the address pointed by the PNTR.
8. Clear the PPC bit for the required programming time, 50 mS. (The programming power is applied to the EPROM from the $\overline{\text{RESET}}$/Vpp pin. This allows the data byte to be burned in).
9. Steps 6 through 8 are repeated until all the bytes have been programmed.

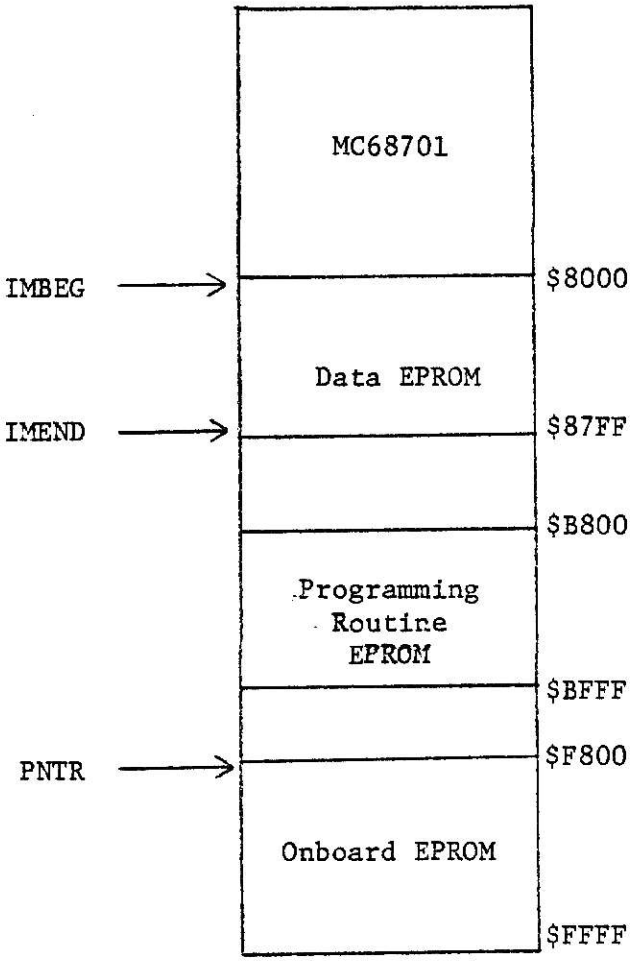


Fig. 21.

MAP ILLUSTRATING INITIALIZATIONS.

IV. CONCLUSIONS

1. The two main functions of the MC68701 interface are:
   a) To provide a given binary number of the Multiprogrammer.
   b) To read the data from the Multiprogrammer.

2. Presently the computer that is being used is the North Star Horizon II. Any computer with an RS-232C Standard could be used with the MC68701 interface.

3. The MC68701 interface is almost a 50% reduction in the hardware, as compared with the MC6802 interface that is currently being used in the Instrumentation Laboratory.

4. The MC68701 interface should be more reliable as it makes use of fewer components.

BIBLIOGRAPHY

1. John B. Peatman, Microcomputer-Based Design. McGraw-Hill, Inc., 1977.

2. J. D. Greenfield and W. C. Wray, Using Microprocessors and Microcomputers: The 6800 Family. John Wiley, 1981, pp. 253-254.

3. Bruce A. Artwick, Microcomputer Interfacing. Prentice Hall, Inc., 1980, pp. 290-292.

4. D. D. Givone and R. P. Roesser, Microprocessor/Microcomputers: An Introduction. McGraw-Hill, Inc., 1980.

5. Aditya P. Mathur, Introduction to Microprocessors. Tata McGraw-Hill, 1980.

6. M. S. P. Lucas, An Introduction to Computer Aided Instrumentation. Dept. of Electrical Engg., Kansas State University.

7. E. E. Creel and J. D. Roberts, A Dedicated Interface for the HP6940B Multiprogrammer. Dept. of Electrical Engg., Kansas State University, 1981.

8. A. J. Morales and D. Ruhberg, Let the MC68701 Program Itself, Motorola, Inc.

9. HP6940B, Operating and Service Manual for Multiprogrammer, June 1979.

10. Motorola MC68701 Microcomputer Unit Manual, Motorola, Inc., 1980.

11. Motorola Probug Support Firmware for 68701/6801/6803. Motorola, Inc., June 1980.

12. Motorola Lilbug Monitor for the MC6801L1. Motorola, Inc., 1980.

13. Motorola Microprocessors Data Manual, Motorola, Inc., 1981.

ACKNOWLEDGEMENTS

I take this opportunity to sincerely thank my major advisor, Dr. M. S. P. Lucas for his excellent guidance, fruitful hours of discussion, helpful suggestions and encouragement during my graduate study program.

Thanks are due to my committee members, Dr. Nasir Ahmed and Dr. Fuller for their suggestions, support and comments. I also express my appreciation to Dr. D. H. Lenhert and Dr. Won Park for their guidance.

Gratitude is due to my parents, whose encouragement and support contributed much toward the successful completion of my graduate program.

Thanks are also due to Brenda Merryman for her superb typing of this report.

APPENDIX I.

TXTEST Program (Test Program)

```
        M68SAM IS THE PROPERTY OF MOTOROLA SPD, INC.
               COPYRIGHT 1974 BY MOTOROLA INC

        MOTOROLA M6800 CROSS ASSEMBLER, RELEASE 1.1

00001                       NAM     TXTEST
00002                   *                                                     *
00003                   *******************************************************
00004                   *                                                     *
00005                   *  THIS PROGRAM TESTS IF THE MC68701 RECEIVES THE CHARACTER TRANS-   *
00006                   *  MITTED FROM THE NORTH STAR COMPUTER. IF THE PROCESSOR RECEIVES    *
00007                   *  THE CHARACTER, THEN THE ASCII EQUIVALENT MUST BE ON PORT3.        *
00008                   *                                                     *
00009                   *******************************************************
00010                   *                                                     *
00011   0000    DDR1    EQU     $00
00012   0002    DR1     EQU     $02
00013   0004    DDR3    EQU     $04
00014   0006    DR3     EQU     $06
00015   0005    DDR4    EQU     $05
00016   0007    DR4     EQU     $07
00017   0010    RMCR    EQU     $10
00018   0011    TRCSR   EQU     $11
00019   0012    RDR     EQU     $12
00020                   *
00021 F800              ORG     $F800
00022                   *
00023 F800 86 FF        LDA A   #$FF        CONFIGURE PORTS 3 AND 4 AS OUTPUT PORTS.
00024 F802 97 04        STA A   DDR3
00025 F804 97 05        STA A   DDR4
00026 F806 86 05        LDA A   #$05        INITIALIZE SCI. BAUD RATE=7200
00027 F808 97 10        STA A   RMCR        CLOCK SOURCE=NRZ FORMAT, INTERNAL.
00028 F80A 86 08        LDA A   #$08        ENABLE THE RECEIVER OF THE SCI.
00029 F80C 97 11        STA A   TRCSR
00030 F80E 96 11        LDA A   TRCSR       DUMMY OPERATIONS
00031 F810 96 12        LDA A   RDR
00032 F812 86 FF        LDA A   #$FF        MAKE PORT1 AN OUTPUT PORT.
00033 F814 97 00        STA A   DDR1
00034 F816 86 0F        LDA A   #$0F        OUTPUT $0F ON PORT1
00035 F818 97 02        STA A   DR1
00036                   *
00037 F81A 96 11    BACK1   LDA A   TRCSR   RECEIVE THE CHARACTER FROM NORTH STAR.
00038 F81C 48           ASL A
00039 F81D 24 FB        BCC     BACK1
00040 F81F 96 12        LDA A   RDR         OUTPUT THE SAME ON PORT3.
00041 F821 97 06        STA A   DR3
00042 F823 86 D6        LDA A   #$D6        OUTPUT SOME GARBAGE ON PORT4.
00043 F825 97 07        STA A   DR4
00044 F827 02       BACK2   NOP
00045 F828 02           NOP
00046 F829 02           NOP
00047 F82A 7E F827      JMP     BACK2
00048                   *
00049 FFFE              ORG     $FFFE
00050                   *
```

```
00051 FFFE F8             FCB     $F8,$00
      FFFF 00
00052                     END

SYMBOL TABLE

DDR1   0000  DR1    0002  DDR3   0004  DR3    0006  DDR4   0005  DR4    0007  RMCR   0010  TRCSR  0011  RDR    0012
BACK1  F81A  BACK2  F827
```

APPENDIX II.

NSTAR Program (Main Program)

```
        M68SAM IS THE PROPERTY OF MOTOROLA SPD, INC.
               COPYRIGHT 1974 BY MOTOROLA INC

        MOTOROLA M6800 CROSS ASSEMBLER, RELEASE 1.1

0C001                       NAM     NSTAR
0C002                   *                                                       *
00003                   *********************************************************
0C004                   *                                                       *
0C005                   *  ORIGINAL PROGRAMMERS: E.E.CREEL AND J.D.ROBERTS.     *
0C006                   *  MODIFIED BY: NAGABUSHAN K. BHASKARA.                 *
0C007                   *    THIS IS THE MAIN PROGRAM WHICH IS THE SOFTWARE SECTION OF THE   *
0C008                   *  INTERFACE.  THIS PROGRAM IS ENTERED IN THE ONBOARD EPROM OF THE   *
00009                   *  MC68701.                                             *
0C010                   *                                                       *
00011                   *********************************************************
0C012                   *                                                       *
0C013       00FF        STAK    EQU     $00FF
00014       F900        DDAT    EQU     $F900       ASCII TO BCD TO BINARY CONVERSION SUBROUTINE.
00015       FA00        IDAT    EQU     $FA00       BINARY TO BCD TO ASCII CONVERSION SUBROUTINE.
0C016       FB00        SUB16   EQU     $FB00       16 BIT SUBTRACTION ROUTINE.
00017       0000        DDR1    EQU     $00         DATA DIRECTION REGISTER OF PORT1.
00018       0001        DDR2    EQU     $01         DATA DIRECTION REGISTER OF PORT2.
00019       0002        DR1     EQU     $02         DATA REGISTER OF PORT1.
00020       0003        DR2     EQU     $03         DATA REGISTER OF PORT2.
00021       0004        DDR3    EQU     $04         DATA DIRECTION REGISTER OF PORT3.
00022       0005        DDR4    EQU     $05         DATA DIRECTION REGISTER OF PORT4.
00023       0006        DR3     EQU     $06         DATA REGISTER OF PORT3.
00024       0007        DR4     EQU     $07         DATA REGISTER OF PORT4.
00025       0010        RMCR    EQU     $10         RATE & MODE CONTROL REGISTER OF SCI.
00026       0011        TRCSR   EQU     $11         TRANSMIT/RECEIVE CONTROL & STATUS REGISTER.
00027       0012        RDR     EQU     $12         RECEIVE DATA REGISTER OF SCI.
0C028       0013        TDR     EQU     $13         TRANSMIT DATA REGISTER OF SCI.
00029       0092        CP      EQU     $92
0C030       00AE        VALL    EQU     $AE
0C031       00AF        VALH    EQU     $AF
0C032       00AC        TMPL    EQU     $AC
0C033       00AD        TMPH    EQU     $AD
00034       00AB        KNT     EQU     $AB
0C035       00AA        MARK    EQU     $AA
00036       00A6        LSD3    EQU     $A6
00037       00A7        LSD2    EQU     $A7
0C038       00A8        LSD1    EQU     $A8
00039       00A9        LSD     EQU     $A9
0C040       00A1        SUBHI   EQU     $A1
00041       00A0        SUBLO   EQU     $A0
00042       00A3        MINHI   EQU     $A3
0C043       00A2        MINLO   EQU     $A2
00044       00A4        NEG     EQU     $A4
00045       00A5        CARRY   EQU     $A5
00046                   *
00047                   *
0C048 F800                  ORG     $F800
0C049                   *
00050 F800 8E 00FF          LDS     #STAK       INITIALIZE THE STACK.
```

```
00051 F803 02              NOP
00052 F804 02              NOP
00053 F805 02              NOP
00054 F806 02              NOP
00055 F807 02              NOP
00056 F808 02              NOP
00057 F809 02              NOP
00058 F80A 02              NOP
00059 F80B 02              NOP
00060 F80C 86 05           LDA A  #$05      SCI INITIALIZATION. SELECT REQUIRED BAUD RATE
00061 F80E 97 10           STA A  RMCR      CLOCK SOURCE=INTERNAL, NRZ FORMAT, BIT2 UNUSED.
00062 F810 86 0A    LKC0   LDA A  #$0A      ENABLE BOTH THE RECEIVER & TRANSMITTER OF SCI.
00063 F812 97 11           STA A  TRCSR
00064 F814 02              NOP
00065 F815 02              NOP
00066 F816 96 11           LDA A  TRCSR     DUMMY OPERATIONS.
00067 F818 96 12           LDA A  RDR
00068 F81A 02              NOP
00069 F81B 02              NOP
00070 F81C 02              NOP
00071 F81D 02              NOP
00072 F81E 02              NOP
00073 F81F 96 11    LKC3   LDA A  TRCSR
00074 F821 48              ASL A
00075 F822 24 FB           BCC    LKC3
00076 F824 96 12           LDA A  RDR       RECEIVE THE CHARACTER FROM THE NORTH STAR.
00077 F826 81 26           CMP A  #$26      CHECK FOR AN "&".
00078 F828 26 05           BNE    CKQM
00079 F82A BD F900         JSR    ODAT      GO TO ASCII TO BCD TO BINARY SUBROUTINE.
00080 F82D 20 F0           BRA    LKC3
00081 F82F 81 3F    CKQM   CMP A  #$3F      CHECK FOR A "?".
00082 F831 26 EC           BNE    LKC3
00083 F833 BD FA00         JSR    TOAT      GO TO BINARY TO BCD TO ASCII SUBROUTINE.
00084 F836 96 11    LKC1   LDA A  TRCSR
00085 F838 48              ASL A
00086 F839 48              ASL A
00087 F83A 48              ASL A
00088 F83B 24 F9           BCC    LKC1
00089 F83D 86 0D           LDA A  #$0D      SEND A CARRIAGE RETURN ($0D) CHARACTER.
00090 F83F 97 13           STA A  TDR
00091 F841 20 DC           BRA    LKC3
00092 F843 02              NOP
00093 F844 02              NOP
00094 F845 02              NOP
00095                *
00096                *
00097                *  ASCII TO BCD TO BINARY CONVERSION SUBROUTINE.
00098                *
00099 F900                 ORG    $F900
00100 F900 86 FB           LDA A  #$FB      PORT1 (EXCEPT BIT2) IS CONFIGURED AS AN OUTPUT
00101 F902 97 C0           STA A  DDR1      BIT2 IS AN INPUT.
00102 F904 86 0A           LDA A  #$0A      SEND A LOW SIGNAL TO THE TRISTATE, MAKE THE GATE
00103 F906 97 02           STA A  DR1       HIGH, MAKE CLOCK2 INPUT OF 7495 HIGH.
00104 F908 86 FF           LDA A  #$FF      CONFIGURE PORTS 3 & 4 AS OUTPUT PORTS.
00105 F90A 97 04           STA A  DDR3
```

```
00106 F90C 97 05          STA A  DDR4
00107 F90E 02             NOP
00108 F90F 02             NOP
00109 F910 02             NOP
00110 F911 02             NOP
00111 F912 86 00          LDA A  #$00
00112 F914 97 92          STA A  CP
00113 F916 96 11    LKC2  LDA A  TRCSR
00114 F918 48             ASL A
00115 F919 24 FB          BCC    LKC2
00116 F91B 96 12          LDA A  RDR
00117 F91D 81 24          CMP A  #$24
00118 F91F 27 C8          BEQ    CHECK
00119 F921 80 30          SUB A  #$30
00120 F923 36             PSH A
00121 F924 7C 0092        INC    CP
00122 F927 20 ED          BRA    LKC2
00123 F929 96 92    CHECK LDA A  CP
00124 F92B 27 6A          BEQ    LAST
00125 F92D 86 00          LDA A  #$00
00126 F92F 97 AE          STA A  VALL
00127 F931 97 AF          STA A  VALH
00128 F933 97 AC          STA A  TMPL
00129 F935 97 AD          STA A  TMPH
00130 F937 97 AB          STA A  KNT
00131 F939 4C             INC A
00132 F93A 97 AA          STA A  MARK
00133 F93C 32       NEXT  PUL A
00134 F93D D6 AB          LDA B  KNT
00135 F93F 26 0D          BNE    KPOS
00136 F941 97 AE          STA A  VALL
00137 F943 7C 00AB        INC    KNT
00138 F946 96 92          LDA A  CP
00139 F948 91 AB          CMP A  KNT
00140 F94A 27 2B          BEQ    VAL1
00141 F94C 20 EE          BRA    NEXT
00142 F94E 5F       KPOS  CLR B
00143 F94F 0C       X10   CLC
00144 F950 48             ASL A
00145 F951 59             ROL B
00146 F952 97 AC          STA A  TMPL
00147 F954 D7 AD          STA B  TMPH
00148 F956 48             ASL A
00149 F957 59             ROL B
00150 F958 48             ASL A
00151 F959 59             ROL B
00152 F95A 0C             CLC
00153 F95B 9B AC          ADD A  TMPL
00154 F95D D9 AD          ADC B  TMPH
00155 F95F 7A 00AB        DEC    KNT
00156 F962 26 EB          BNE    X10
00157 F964 9B AE          ADD A  VALL
00158 F966 D9 AF          ADC B  VALH
00159 F968 97 AE          STA A  VALL
00160 F96A D7 AF          STA B  VALH
```

```
00161 F96C 7C 0CAA          INC     MARK
0C162 F96F 96 AA            LDA A   MARK
00163 F971 97 AB            STA A   KNT
00164 F973 91 92            CMP A   CP
00165 F975 26 C5            BNE     NEXT
00166 F977 96 AE    VALL    LDA A   VALL        STORE THE 16-BIT BINARY DATA ON THE
0C167 F979 97 06            STA A   DR3         MULTIPROGRAMMER INTERFACE LINES.
00168 F97B 96 AF            LDA A   VALH
00169 F97D 97 C7            STA A   DR4
00170 F97F 86 02            LDA A   #$02        PULL THE CLOCK2 INPUT OF 7495 LOW.
00171 F981 97 02            STA A   DR1
0C172 F983 86 0A            LDA A   #$0A        PULL IT BACK HIGH.
00173 F985 97 02            STA A   DR1
00174 F987 02               NOP                 NOPS CREATES SUFFICIENT TIME DELAY FOR THE
0C175 F988 02               NOP                 DATA TO STABILIZE ON THE INTERFACE LINES.
0C176 F989 02               NOP
0C177 F98A 86 C8            LDA A   #$08        PULL THE GATE OF MULTIPROGRAMMER LOW.
00178 F98C 96 02    BACK    LDA A   DR1         CHECK FOR FLAG OF MULTIPROGRAMMER.
0C179 F98E 84 04            AND A   #$04
0C180 F990 26 FA            BNE     BACK        IF HIGH, GO BACK.
00181 F992 86 0A            LDA A   #$0A        PULL THE GATE HIGH.
0C182 F994 97 02            STA A   DR1
00183 F996 02               NOP
0C184 F997 39       LAST    RTS
0C185                   *
0C186                   *
0C187                   *  BINARY TO BCD TO ASCII CONVERSION SUBROUTINE.
0C188                   *
0C189 FA00                      ORG     $FA00
0C190                   *
0C191 FA00 86 0A            LDA A   #$0A        ENABLE THE RECEIVER & TRANSMITTER OF SCI.
0C192 FA02 97 11            STA A   TRCSR
00193 FA04 02               NOP
00194 FA05 02               NOP
0C195 FA06 02               NOP
0C196 FA07 02               NOP
0C197 FA08 02               NOP
0C198 FA09 96 11            LDA A   TRCSR       DUMMY READ
00199 FA0B 02               NOP
00200 FA0C 02               NOP
0C201 FA0D 86 0C            LDA A   #$00        CONFIGURE PORTS 3 & 4 AS INPUT PORTS.
0C202 FA0F 97 04            STA A   DDR3
0C203 FA11 97 05            STA A   DDR4
00204 FA13 86 FB            LDA A   #$FB
0C205 FA15 97 0C            STA A   DDR1
00206 FA17 96 08            LDA A   #$08        SEND A HIGH SIGNAL TO TRISTATE
0C207 FA19 97 02            STA A   DR1         MAKE THE GATE HIGH.
00208 FA1B 02               NOP
0C209 FA1C 02               NOP
00210 FA1D 02               NOP
00211 FA1E 02               NOP
0C212 FA1F CE C0A6          LDX     #LSD3
0C213 FA22 86 03            LDA A   #$03
0C214 FA24 C6 00            LDA B   #$00
00215 FA26 E7 CC    INC     STA B   0,X
```

```
00216 FA28 08              INX
00217 FA29 4A              DEC A
00218 FA2A 2A FA           BPL     INC
00219 FA2C 96 01           LDA A   DR4
00220 FA2E 84 0F           AND A   #$0F
00221 FA30 97 A1           STA A   SUBHI
00222 FA32 96 06           LDA A   DR3
00223 FA34 97 A0           STA A   SUBLO
00224 FA36 86 E8           LDA A   #$E8
00225 FA38 97 A2           STA A   MINLO
00226 FA3A 86 03           LDA A   #$03
00227 FA3C 97 A3           STA A   MINHI
00228 FA3E BD FB00 POS1    JSR     SUB16
00229 FA41 96 A4           LDA A   NEG
00230 FA43 81 00           CMP A   #$00
00231 FA45 26 05           BNE     NEG1
00232 FA47 7C 00A6         INC     LSD3
00233 FA4A 20 F2           BRA     POS1
00234 FA4C 96 A0    NEG1   LDA A   SUBLO
00235 FA4E 8B E8           ADD A   #$E8
00236 FA50 97 A0           STA A   SUBLO
00237 FA52 96 A1           LDA A   SUBHI
00238 FA54 89 03           ADC A   #$03
00239 FA56 97 A1           STA A   SUBHI
00240 FA58 01 64           STA A   #$64
00241 FA5A 97 A2           STA A   MINLO
00242 FA5C 86 00           LDA A   #$00
00243 FA5E 97 A3           STA A   MINHI
00244 FA60 BD FB00 POS2    JSR     SUB16
00245 FA63 96 A4           LDA A   NEG
00246 FA65 81 00           CMP A   #$00
00247 FA67 26 05           BNE     NEG2
00248 FA69 7C 00A7         INC     LSD2
00249 FA6C 20 F2           BRA     POS2
00250 FA6E 86 64    NEG2   LDA A   #$64
00251 FA70 9B A0           ADD A   SUBLO
00252 FA72 97 A0           STA A   SUBLO
00253 FA74 96 A1           LDA A   SUBHI
00254 FA76 89 00           ADC A   #$00
00255 FA78 97 A1           STA A   SUBHI
00256 FA7A 86 0A           LDA A   #$0A
00257 FA7C 97 A2           STA A   MINLO
00258 FA7E 7F 00A3         CLR     MINHI
00259 FA81 BD FB00 POS3    JSR     SUB16
00260 FA84 96 A4           LDA A   NEG
00261 FA86 81 00           CMP A   #$00
00262 FA88 26 05           BNE     NEG3
00263 FA8A 7C 00A8         INC     LSD1
00264 FA8D 20 F2           BRA     POS3
00265 FA8F 96 A0    NEG3   LDA A   SUBLO
00266 FA91 8B 0A           ADD A   #$0A
00267 FA93 97 A5           STA A   LSD
00268 FA95 C6 04           LDA B   #$04
00269 FA97 09       ASCII  DEX
00270 FA98 86 30           LDA A   #$30
```

```
00271 FA9A AB C0          ADD A  0,X
00272 FA9C A7 00          STA A  0,X
00273 FA9E 5A             DEC B
00274 FA9F 26 F6          BNE    ASCII
00275 FAA1 CE 00A6        LDX    #LSD3
00276 FAA4 C6 04          LDA B  #$04
00277 FAA6 D7 AB          STA B  KNT
00278 FAA8 A6 00    NEXT1 LDA A  0,X
00279 FAAA 81 30          CMP A  #$30
00280 FAAC 27 10          BEQ    ZERO
00281 FAAE D6 11    LKC4  LDA B  TRCSR
00282 FAB0 58             ASL B
00283 FAB1 58             ASL B
00284 FAB2 58             ASL B
00285 FAB3 24 F9          BCC    LKC4
00286 FAB5 97 13          STA A  TDR       TRANSMIT CHARACTER TO NORTH STAR.
00287 FAB7 08             INX
00288 FAB8 7A 00AB        DEC    KNT
00289 FABB 26 11          BNE    PATCH
00290 FABD 39             RTS
00291 FABE D6 AB    ZERO  LDA B  KNT
00292 FAC0 C1 01          CMP B  #$01
00293 FAC2 26 04          BNE    NOT
00294 FAC4 86 30          LDA A  #$30
00295 FAC6 20 E6          BRA    LKC4
00296 FAC8 7A 00AB  NOT   DEC    KNT
00297 FACB 08             INX
00298 FACC 20 DA          BRA    NEXT1
00299 FACE A6 00    PATCH LDA A  0,X
00300 FAD0 20 DC          BRA    LKC4
00301               *
00302               *
00303               *   16 BIT SUBTRACTION ROUTINE
00304               *
00305 FB00                ORG    $FB00
00306               *
00307 FB00 96 A3          LDA A  MINHI
00308 FB02 36             PSH A
00309 FB03 96 A2          LDA A  MINLO
00310 FB05 36             PSH A
00311 FB06 7F 00A5        CLR    CARRY
00312 FB09 7F 00A4        CLR    NEG
00313 FB0C 43             COM A
00314 FB0D 0C             CLC
00315 FB0E 8B 01          ADD A  #$01
00316 FB10 24 03          BCC    L1
00317 FB12 7C 00A5        INC    CARRY
00318 FB15 D6 A3    L1    LDA B  MINHI
00319 FB17 53             COM B
00320 FB18 DB A5          ADD B  CARRY
00321 FB1A 97 A2          STA A  MINLO
00322 FB1C D7 A3          STA B  MINHI
00323 FB1E 7F 00A5        CLR    CARRY
00324 FB21 0C             CLC
00325 FB22 96 A0          LDA A  SUBLO
```

```
00326 FB24 9B A2          ADD A  MINLO
00327 FB26 24 03          BCC    L2
00328 FB28 7C CCA5        INC    CARRY
00329 FB2B D6 A1    L2    LDA B  SUBHI
00330 FB2D DB A3          ADD B  MINHI
00331 FB2F DB A5          ADD B  CARRY
00332 FB31 97 AC          STA A  SUBLO
00333 FB33 D7 A1          STA B  SUBHI
00334 FB35 2A 03          BPL    L3
00335 FB37 73 00A4        COM    NEG
00336 FB3A 32       L3    PUL A
00337 FB3B 97 A2          STA A  MINLO
00338 FB3D 32             PUL A
00339 FB3E 97 A3          STA A  MINHI
00340 FB40 39             RTS
00341                *
00342                *
00343                *
00344 FFFE                ORG    $FFFE
00345 FFFE F8             FCB    $F8,$00   STORE THE RESET VECTOR.
      FFFF 00
00346                *
00347                *
00348                     END
```

SYMBOL TABLE

```
STAK   00FF  0CAT  F900  1CAT  FA00  SUB16  FB00  DDR1   0000  DDR2   0001  DR1    0002  DR2    C003  DDR3   0004
DDR4   0005  DR3   0006  DR4   0007  RMCR   0010  TRCSR  0011  RDR    0012  TDR    0013  CP     CC92  VALL   00AE
VALH   00AF  TMPL  00AC  TMPH  00AD  KNT    0CAB  MARK   00AA  LSD3   00A6  LSD2   00A7  LSD1   CCA8  LSD    00A9
SUBHI  00A1  SUBLO 00A0  MINHI 00A3  MINLO  00A2  NEG    00A4  CARRY  00A5  LKC0   F810  LKC3   F81F  CKCM   F82F
LKC1   F836  LKC2  F916  CHECK F929  NEXT   F93C  KPOS   F94E  X10    F94F  VAL1   F977  BACK   F98C  LAST   F997
INC    FA26  POS1  FA3E  NEG1  FA4C  POS2   FA60  NEG2   FA6E  POS3   FA81  NEG3   FA8F  ASCII  FA97  NEXT1  FAA8
LKC4   FAAE  ZERO  FABE  NOT   FAC8  PATCH  FACE  L1     FB15  L2     FB2B  L3     FB3A
```

APPENDIX III.

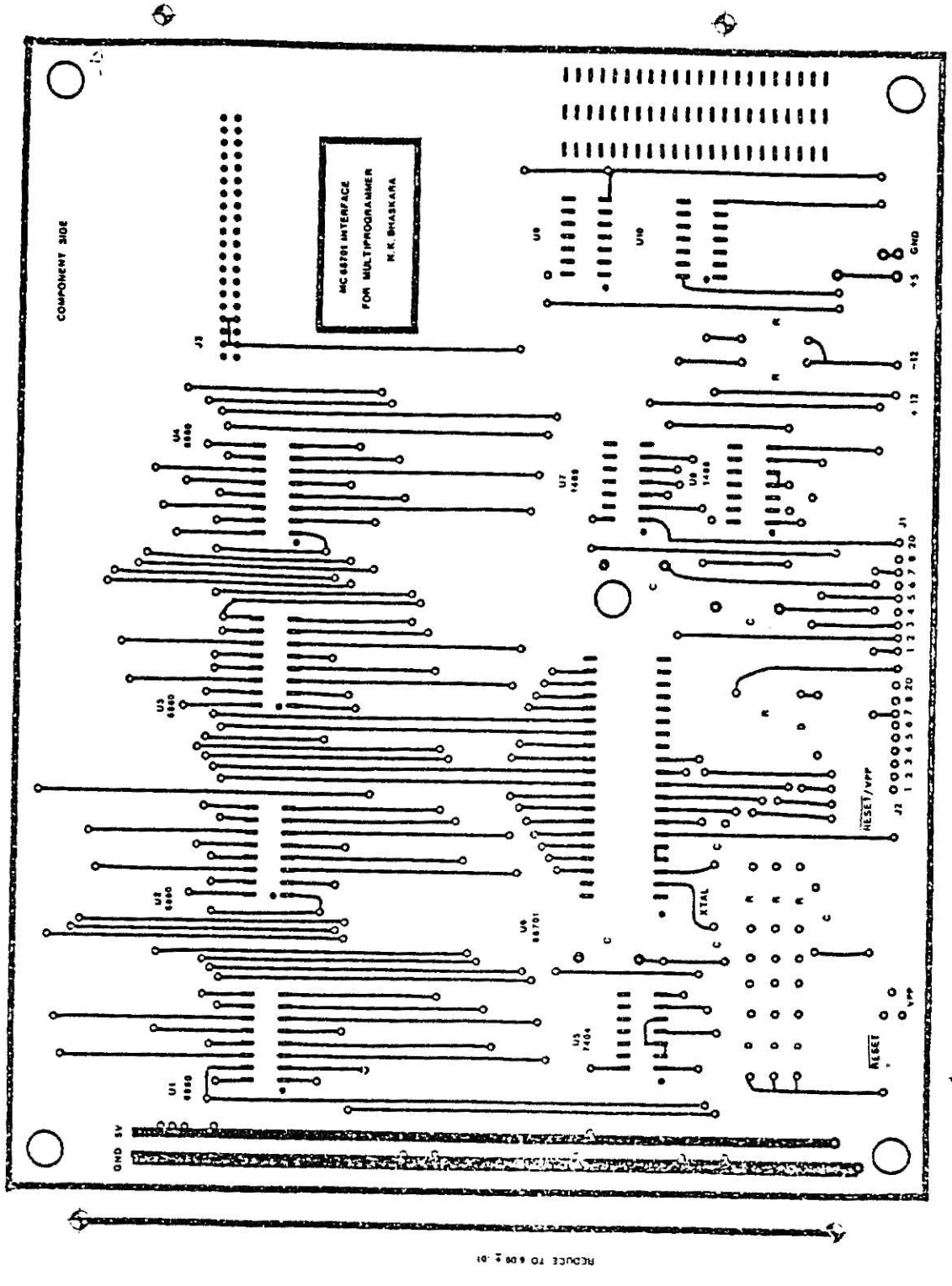
CLOCK Program (Test Program)

MOTOROLA M6800 CROSS ASSEMBLER, RELEASE 1.1

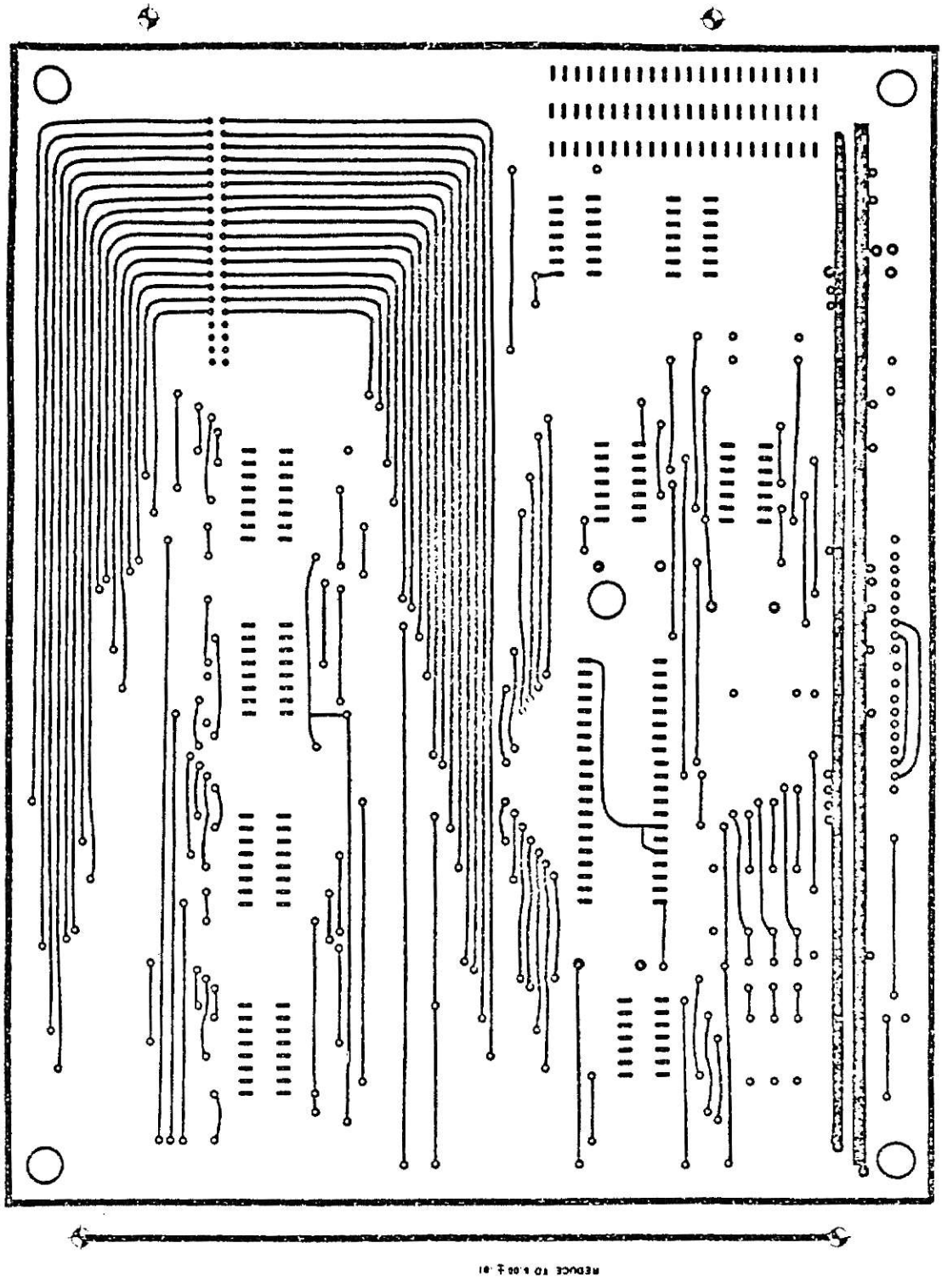
```
00001                           NAM     CLOCK
00002                   *                                                       *
00003                   *************************************************************
00004                   *                                                       *
00005                   *  THIS IS A SIMPLE PROGRAM TO TEST IF THE MC68701 PROGRAMMER IS     *
00006                   *  FUNCTIONING CORRECTLY. THE STEPS INVOLVED ARE:       *
00007                   *  1. ENTER THE PROGRAM GIVEN BELOW USING THE MC68701 PROGRAMMING     *
00008                   *     UNIT.                                             *
00009                   *  2. OPERATE THE MC68701 IN MODE 7.                    *
00010                   *  3. SCOPE ANY OF THE BITS ON PORT 1.                  *
00011                   *  IF A CLOCK IS DISPLAYED ON THE SCOPE, THEN THE MC68701 HAS BEEN    *
00012                   *  PROGRAMMED SUCCESSFULLY. IF NOT, IT HAS FAILED TO PROGRAM.         *
00013                   *                                                       *
00014                   *************************************************************
00015                   *                                                       *
00016        0000       DDR1    EQU     $00
00017        0002       DR1     EQU     $02
00018 F800                         ORG     $F800
00019                   *                                                       *
00020 F800 86 FF                   LDA A   #$FF        CONFIGURE PORT1 AS AN OUTPUT PORT.
00021 F802 97 00                   STA A   DDR1
00022 F804 86 FF        BACK       LDA A   #$FF        OUTPUT ALL ONES ON PORT1.
00023 F806 97 02                   STA A   DR1
00024 F808 02                      NOP
00025 F809 02                      NOP
00026 F80A 02                      NOP
00027 F80B 02                      NOP
00028 F80C 86 C0                   LDA A   #$00        OUTPUT ALL ZEROS ON PORT1.
00029 F80E 97 C2                   STA A   DR1
00030 F810 02                      NOP
00031 F811 02                      NOP
00032 F812 02                      NOP
00033 F813 02                      NOP
00034 F814 7E F804                 JMP     BACK
00035                   *
00036 FFFE                         ORG     $FFFE
00037 FFFE F8                      FCB     $F8,$00     STORE THE RESET VECTOR.
      FFFF 00
00038                   *
00039                              END
```

APPENDIX IV.

Printed Circuit Board Layout of the Interface



PRINTED CIRCUIT BOARD LAYOUT
(Component Side)



PRINTED CIRCUIT BOARD LAYOUT
(Circuit Side)

MOTOROLA MC68701 MICROCOMPUTER INTERFACE FOR THE HP6940B MULTIPROGRAMMER

by

NAGABUSHAN K. BHASKARA

B.E., Bangalore University (India), 1980

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1983

ABSTRACT

An interface using the Motorola MC68701 microcomputer has been designed to provide communication between the Hewlett Packard Multiprogrammer and any computer using the RS-232C standard. The computer that is presently being used for this purpose is the North Star Horizon II. In operation the computer transmits and receives characters serially in ASCII form using the RS-232C Standard. The interface receives the ASCII characters, performs serial to parallel conversion and provides parallel data using TTL logic to the Multiprogrammer. Also the interface receives parallel binary data from the Multiprogrammer, performs parallel to serial conversion and provides serial data to the computer. The MC68701 interface provides a minimum hardware system with an almost 50% reduction in total hardware, as compared to the Motorola 6802 microprocessor interface for the Multiprogrammer, that is currently being used in the Instrumentation Laboratory.