# UNIT I

# INTRODUCTION

# INTRODUCTION

RB5X was designed with user experimentation and expansion in mind. This makes it a truly personal robot, to be used in the home and in the classroom to learn about robotics, microcomputers, physics, electronics, mechanics, and sensor technology. This robot provides exceptional versatility and ease of operation, particularly for control and monitoring applications. While most microcomputers manipulate data and text, RB5X emphasizes control and sensing of relays, switches, light transducers, and a large variety of "real world" capabilities.

These Instructional Materials take you through a sequence that allows you to become proficient in the use of the electronics that comprise RB5X. This proficiency then allows you to expand RB5X's capabilities to the limits of your imagination. We hope you find the same elements of excitement in discovery that we did.

## OBJECTIVES

Upon completion of this course of study, the student will be able to:

1. Identify all chips used in the RB5X system by location and function;
2. Identify all registers and their functions used in RB5X's INS8073 microprocessor;
3. Identify RB5X's system bus by jack and pin, and the signals they each carry;
4. Use ASCII Express "The Professional" as an operating system to control RB5X using an Apple computer;
5. Use the high-level language Tiny BASIC to communicate with RB5X;
6. Identify and use RB5X system- and user-defined ports;
7. Convert light, heat, and sound inputs into data RB5X can use;
8. Use sonar to allow RB5X to sense and respond to its environment; and
9. Add new capabilities to the robot.

## PREREQUISITES

1. A demonstrated proficiency in electronics fundamentals with an emphasis on digital and microprocessor electronics.
2. A familiarity with the computer language BASIC.
3. At least one RB5X mobile robot
4. At least one RobotLab

## REFERENCES

1. The RB5X Reference Manual.
2. ASCII Express "The Professional" instruction manual.

3. NSC Tiny BASIC user's manual.

## TECHNICAL INFORMATION

RB5X can be programmed using any computer that has an RS-232 serial communications interface. The experiments in these Instructional Materials make use of the Apple II and the Apple II+. You program RB5X's microprocessor directly via the RS-232 interface by using the Apple computer as a "dumb terminal."

RB5X's heart is the INS8073 microprocessor from National Semiconductor. It is an 8-bit, 4.0 MHz device with 2.5K ROM (read-only memory) containing a Tiny BASIC interpreter and 64 bits of internal RAM (random-access memory). The 8073 microprocessor has seven, 16-bit registers and three, 8-bit registers. These registers function in the same way as those in most other microprocessor architectures.

One important feature of the INS8073 is that it allows you to program it directly in Tiny BASIC without having to add circuitry. Nothing surpasses BASIC as an almost universal programming language, used by engineers, scientists, hobbyists, business people, teachers, and students.

Writing programs for RB5X in Tiny BASIC offers many advantages over assembly language or machine code, including fast and simple debugging of programs and an English-like simplicity allowing for fast and easy program writing. Programs can be loaded and executed anywhere in memory without modification. Tiny BASIC is ideally suited for control and monitoring applications, which typically demand millisecond response times. When faster responses are required, Tiny BASIC easily accommodates links to machine-language routines.

RB5X's electronics circuitry is composed of commonly available integrated circuits similar to those found in most expandable personal computers. RB5X has a 16-bit address bus and an 8-bit data bus along with assorted control lines that form its system bus structure. All system bus lines are accessible via card slots on the mother board.

RB5X has nine I/O (input/output) ports, three of which are system-defined (for the LEDs, motors, bumpers, horn, and sonar). The remaining six are user-defined and can be used for the transfer of data between external expansion circuits and the rest of the system.

The sonar circuitry, which adds "sight" to the robot, is an example of a capability that uses a system-defined port. The Environmental

Sense Kit option (with its analog-to-digital board) is an example of circuitry that uses a user-defined port.

We suggest that each class have at least one RB5X robot available for student use and experimentation, and at least one RobotLab per class. RobotLab is General Robotics Corporation's experimentation and test tool: it consists of all of RB5X's components, except motors and wheels, laid out on a bread board. All the lab experiments in this course, up to and including Experiment 6.3, can be done on either the complete, mobile robot (RB5X) or on the bread board (RobotLab). Keep in mind, however, that most of the experiments are easier to execute on RobotLab because of its easy accessibility.

## SUMMARY

RB5X is an environmentally interacting, sophisticated, expandable personal robot. Your involvement with the robot will be both challenging and rewarding. Welcome to the field of robotics, a field whose time has come for development. You are on the pioneering edge of that development. Get out your sextant and prepare for the future.

# UNIT II

# R B 5 X  F A M I L I A R I Z A T I O N

# CHAPTER 1

# CHIPS USED ON THE RB5X CPU BOARD

## OVERVIEW

Before you start experimenting with any electronic system, you should first analyze the circuits to thoroughly understand their function. In this section, you will look at each chip on RB5X's CPU (central processing unit) board, and will study their function and location.

**NOTE:** Exercise caution when handling the CPU board, as most of the chips on the board are of the CMOS family and can be damaged by static electricity. Before removing the CPU from RB5X's mother board, ground yourself by touching any unpainted surface on the robot, such as the vertical struts holding the LED board.

## OBJECTIVES

Upon completion of this text material, you will be able to identify all the chips used on RB5X's CPU board by their:

1. Function,
2. Identifier, and
3. Part number.

## PREREQUISITES

1. Review Appendix A, "CPU Board Electronics Parts List."
2. Review Appendix B, "CPU Board Chip Data Sheets," to become familiar with the pin diagrams of the chips found on the CPU board.

## REFERENCES

1. Appendix B, "CPU Board Chip Data Sheets."

## TECHNICAL INFORMATION

RB5X's CPU board contains the following integrated circuits, or chips. The following information includes the chip number, the chip designator, and the general function of the chip on the CPU board. For details on each chip, consult Appendix A and Appendix B.

**U1**
SN74LS139--Decoder chip
Selectively enables the I/O ports

**U4, U20, and U23**

INS8255--Programmable peripheral interface (PPI)
Buffers data into and out of the CPU board

**U5**
MC1413--Driver
Drive the LEDs ported at jack J7

**U6**
SN74LS14--Inverter
Handles simple, logic-level inversion

**U7**
SN74LS121--Dual AND gate
Performs logical AND function on CPU address lines

**U8**
INS8073--Microprocessor
Performs all mathematical operations and functions as heart of system

**U9**
ICL7660--Voltage converter
Establishes reference voltages

**U11**
LM392--Comparator
Compares two voltages

**U12**
ICL8211--Voltage sensor
Detects low voltage level of batteries

**U13**
CD4503--Hexadecimal buffer
Tri-state level control for switch SW1

**U14**
2716--EPROM
Socket for chip that holds fixed programs in read-only memory (ROM)

**U15**
74LS154--Decoder
Enables memory and I/O

**U16, U17, U18, and U19**
HM6116--RAM memory (2K by 8)
Handles temporary program storage

Figure 1, "CPU Board Chips," shows the layout of the chips on the CPU board, the chip numbering sequence, and the part number. You can see that the chips are number from bottom to top, and from left to right when the CPU board is held with the horn in the lower left corner. You should also note that some of the numbers in the sequence are not used (U2, U3, U10, U21, and U22).

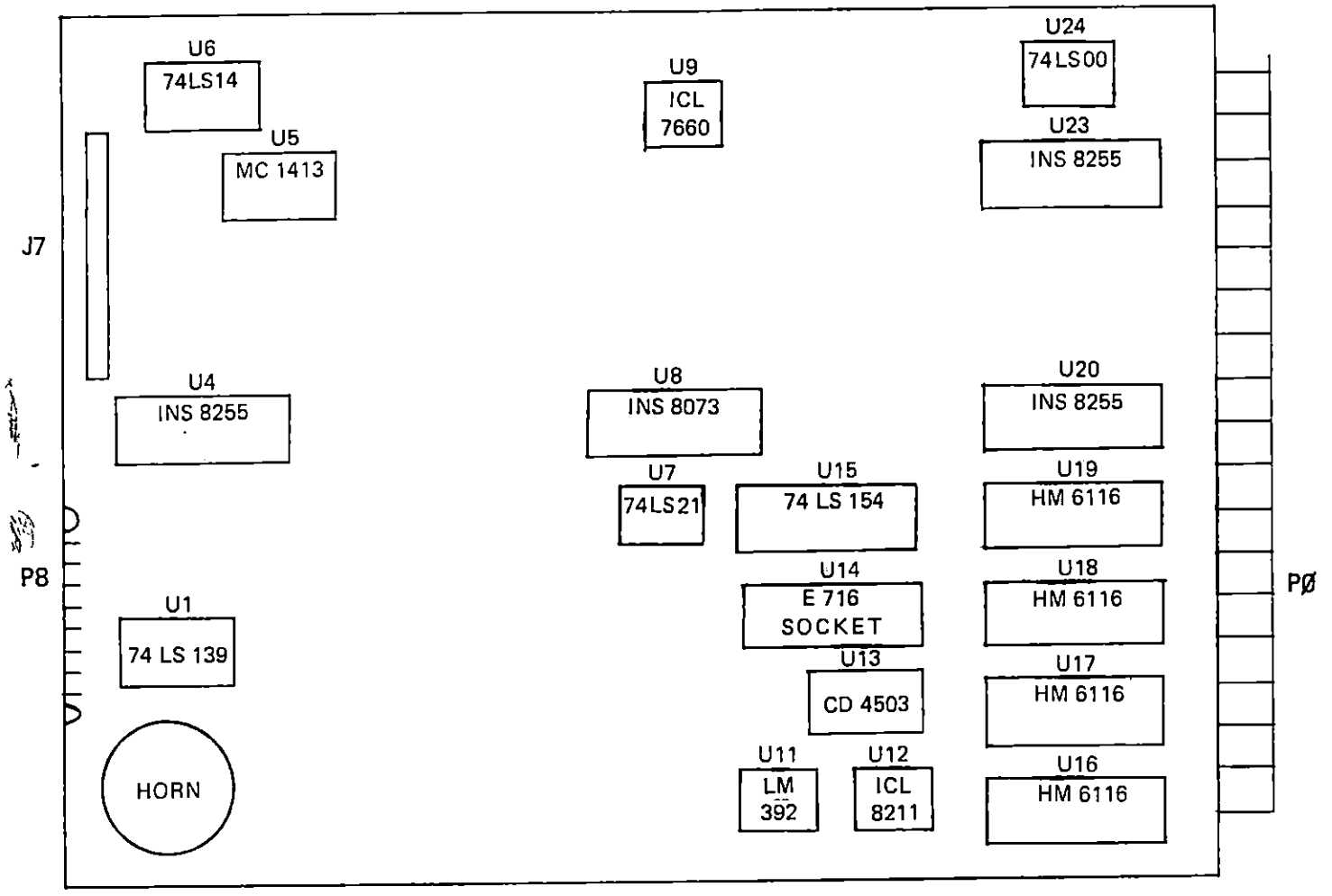



**Figure 1. CPU Board Chips**

In the experiment that follows, we ask you to identify these chips by part number, location, reference identifier, and function. Careful study of the information contained in this section greatly simplifies your work on this experiment.

## SUMMARY

You are now familiar with the chips used on the CPU board of the RB5X robot. You should be able not only to identify a specific chip by its part number, but also by its identifier and function. You should now also understand the numbering sequence of the chips on the CPU board.

## QUESTIONS

1. Since the MC14503B is the Motorola equivalent of the National Semiconductor CD4503B, what logic level is needed on the enable input to disable this tri-state buffer and to force all outputs to go to the high-impendence state?

---

2. Which pins of the INS8255 MOS programmable peripheral interface (PPI) chip are attached to the CPU data bus?

---

3. What is the function of U1 on the CPU board?

---

4. What is the reference number and function of the chip closest to the horn on the CPU board?

---

NOTES:

# Experiment 1.1

## Chip Location and Identification

### Overview

In Chapter 1, you studied the chips used on RB5X's CPU board, including their function and their schematic identifier. In this experiment, we ask you to cross-reference the function identifier, the part number, and the physical location of the chips on the CPU board.

### Objectives

Upon completion of this experiment, you will be able to physically located a chip on the CPU board when given:

1. A chip description (function),
2. A chip designator (identifier), and
3. A chip type (part number).

### Prerequisites

1. Review Appendix A, "CPU Board Electronics Parts List."

2. CAREFULLY remove the CPU board from the **POWERED-DOWN** RB5X.

3. You should have an understanding of the precautions to be used when handling CMOS circuitry.

### References

1. Appendix B, "CPU Board Chip Data Sheets."

### Technical Information

During this experiment, you need to refer to Figure 2, which shows the physical layout of the chips used on the CPU board. Note that the CPU board is positioned with the component side up and the horn in the lower left corner. Also note that the chips are designated sequentially from bottom to top, and from left to right (some chip designators in the sequence are not used).

### Procedure

1. Remove the CPU board from the robot, carefully disconnecting the two cables that attach the board to the interface panel and to J6. Position the CPU board in front of you with the component side up and the horn in the lower left corner.

2. Locate U9, and fill in the blanks below:

   A. Description (function):_____

   B. Type (part number):_____

3. Locate U17, and fill in the blanks below:

   A. Description (function):_____

   B. Type (part number):_____

4. Locate the 74LS00, and fill in the blanks below:

   A. Designator (identifier):_____

   B. Description (function):_____

5. Locate the INS8255 and fill in the blanks below:

   A. Designator (identifier):_____

   B. Description (function):_____

6. Locate the microprocessor and fill in the blanks below:

   A. Designator (identifier):_____

   B. Type (part number):_____

7. Locate the decoder and fill in the blanks below:

   A. Designator (identifier):_____

   B. Type (part number):_____

8. Refer to Figure 2 and label all chips locations by designator. (Mark your answers directly on the illustration.)

9. Locate the chip marked K in Figure 2. What is this chip's designator, description, and type?

   A. Designator (identifier):_____

   B. Description (function):_____

   C. Type (part number):_____

10. Locate the chip marked I in Figure 2. What is this chip's designator, description, and type?

   A. Designator (identifier):_____

   B. Description (function):_____

   C. Type (part number):_____

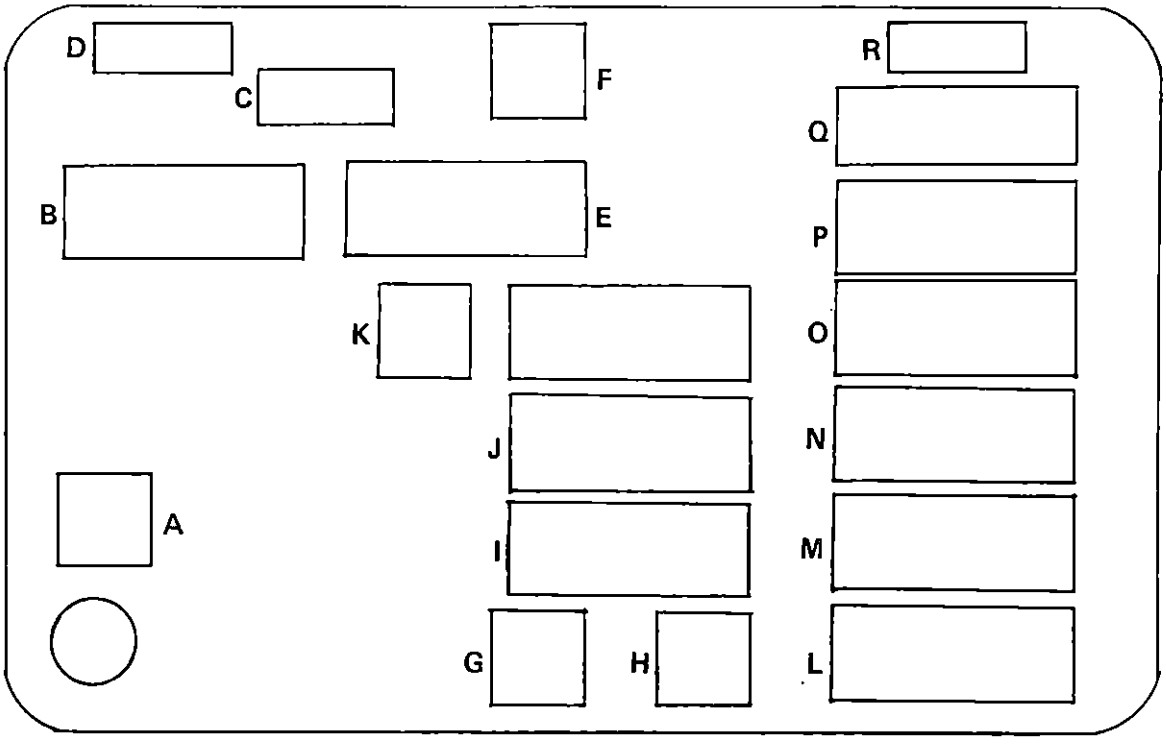



**Figure 2. CPU Board Chip Locations**

## Summary

During this experiment, you cross-referenced the function, identifier, part number, and physical location of the chips on the CPU board. You also learned that the designators are from bottom to top, and from left to right; some chips designators are missing in the sequence. You should now be able to identify the designator, the description, the type, and the location of all chips used on RB5X's CPU board.

**Questions**

1. Which chip designators are omitted from the count sequence?

2. What is the type and designator of the chip located to the right of the SW1 designator?

3. What is the type and description of the chip located directly above resistor R43?

4. Which chips could be interchanged without disrupting the basic function of the CPU board?

5. Why is there no chip plugged into the U14 socket?

NOTES:

## Experiment 1.2

# Initialzing RB5X with the Utility Software Module

### Overview

In Chapter 1, you identified all the chips on RB5X's CPU board. You should have noticed an empty socket at U14, which can hold EPROM (erasable, programmable, read-only memory) chips. In fact, this socket is remotely wired to the software module socket on RB5X's interface panel, enabling the robot to respond to instructions from preprogrammed routines (like those contained on the utility module that comes standard with RB5X) as soon as you switch the robot on. Other preprogrammed software modules are available from General Robotics Corporation or your RB5X dealer.

The utility software module contains an assortment of program routines, accessible through bumper, or switch, combinations. One of the key routines is the power on/reset self-diagnostic function. This program checks the battery source bit and if it is 0, it sounds the horn in one long blast. (If the source bit is 0, the battery is too low for safe operation of the robot. If the source bit is 1, the battery is fully charged.)

If the robot's charge is good, the program instructs the robot to sound three short toots of the horn and then to test its motor functions by executing:

- A forward roll,
- A clockwise spin,
- A counterclockwise spin, and
- A backward roll.

For further information, please refer to the machine-language listing of the power on/reset self-diagnostic function contained in Appendix 2 of the RB5X Reference Manual.

In this experiment, you observe the operation of this initialization routine. You also connect the CPU board to the system.

## Objectives

Upon completion of this experiment, you will be able to:

1. Initialize the robot on start-up,
2. Initialize the robot on software reset,
3. Properly orient the CPU board in the J0 socket on RB5X's mother board,
4. Properly connect the ribbon cable from the interface panel to the U14 socket, and
5. Properly connect the ribbon cable from J6 to the CPU board.

## Prerequisites

1. You should be familiar with Section B, "Getting Started," in the RB5X Reference Manual.
2. You should have the most recent version of the RB5X utility software module.
3. You should have an understanding of the precautions to be observed when handling CMOS circuitry.

### References

1. Section B, "Getting Started," from the RB5X Reference Manual.

2. Appendix C, "Mother Board Interconnect List."

### Technical Information

You can find the location of J0 and J6 by examining Appendix C, "Mother Board Interconnect List." You also must locate the 24-pin and the 50-pin ribbon cables (which you disconnected in Experiment 1.1); you will use these cables to reconnect the CPU board to the interface panel and J6.

**NOTE:** The utility software module, which has 2K or 4K of address locations (depending on the version), is mounted in a chip holder. Be sure to handle the module only by the end tabs on the holder. Never insert or remove this module, or any other software module, while the RB5X is switched on. The module only fits one way in the software socket, so do not force it.

### Procedure

1. Position RB5X or RobotLab in front of you with the interface panel either facing you (RB5X) or to your right (RobotLab).

2. Seat the CPU board in J0 on the mother board, with the component side away from you.

3. Connect the 24-pin ribbon cable to the U14 socket on the CPU board. The red identifier goes to pin 24 in the socket. Connect the other end of the 24-pin ribbon cable to the plug at the top left edge of the interface panel. The red identifier must be to the right.

**NOTE:** IMPROPER CONNECTION OF THIS CABLE COULD CAUSE A SYSTEM FAILURE.

4. Connect the 50-pin ribbon cable to the P0 connect at the top left of the CPU board. The red identifier goes to the left side of the connector. The other end of the 50-pin ribbon cable connects to J6 on the mother board, with the red identifier to the front.

5. Insert the utility software module into the socket on the interface panel.

6. Make sure the switch to the right of the socket is set properly, depending on whether your module is 2K or 4K. (The module's label identifies it as either 2K or 4K.)

7. Switch the power on.

8. RB5X responds by:

o Sounding three short beeps of its horn, followed by five flashing LEDs on the LED board.

**NOTE:** If you hear only one long beep, the battery charge level is low. Switch the power off and plug the battery charger into a 110 VAC source (RobotLab) or push RB5X into its charger-nest, which should be connected to a 110 VAC source. After the robot is fully charged, return to Step 7 and resume the experiment.

o RB5X then rolls forward for one second, and the two front, green LEDs light for one second. (If you are using RobotLab, the two, front green LEDs simply light for one second and a motor relay is activated.)

o RB5X then spins clockwise, and the green LEDs at clock positions 10:00 and 5:00 light for one second. (If you are using RobotLab, the LEDs simply light and a motor relay is activated.)

o RB5X then spins counterclockwise, and the green LEDs at clock positions 7:00 and 1:00 light for one second. (If you are using RobotLab, the LEDs simply light and a motor relay is activated.)

o RB5X then rolls backward for one second, and the two rear, green LEDs light for one second. (If you are using RobotLab, the LEDs simply light and a motor relay is activated.)

o RB5X (or RobotLab) then flashes the four, clear LEDs, with the center amber LED remaining on.

**NOTE:** You may have to power the robot up several times to catch all of this sequence.

9. To demonstrate RB5X's speech capabilities, insert the voice/sound synthesis option, with the PC board in slot J1 or J2 on the mother board and the component side away from you. Next,

connect the speaker wire to J1 on the voice/sound synthesis board. With the voice/sound synthesis option installed, RB5X (or RobotLab) says "Hello, I am the RB5X Intelligent Robot" during the final stage of initialization.

**NOTE:** The potentiometer on the top right corner of the voice/sound synthesis board controls voice and sound volume, while the potentiometer on the left side controls the pitch of the voice.

10. Press the software reset button. Compare the sequence of events with those observed at power up. They should be the same.

**Summary**

During this experiment, you observed the initialization sequence started by power up and software reset. You should now know how to properly connect the CPU and its cables. Also, you should be able to insert software modules into the socket on the interface panel and to properly set the software switch.

**Questions**

1. If the front-most, bumper switch is number 1 and the bumpers are connected clockwise, what happens when bumpers 2 and 5 are pressed simultaneously?

_____

_____

_____

_____

2. Following the initialization sequence, what happens when bumpers 2 and 4 are pressed simultaneously?

_____

_____

3. Following an initialization sequence, what happens when bumpers 1 and 4 are pressed simultaneously?

_____

_____

_____

4. We know that the green LEDs are indicators for when the motor relays are activated. Using Figure 3, match the green LEDs to their corresponding motor relay.

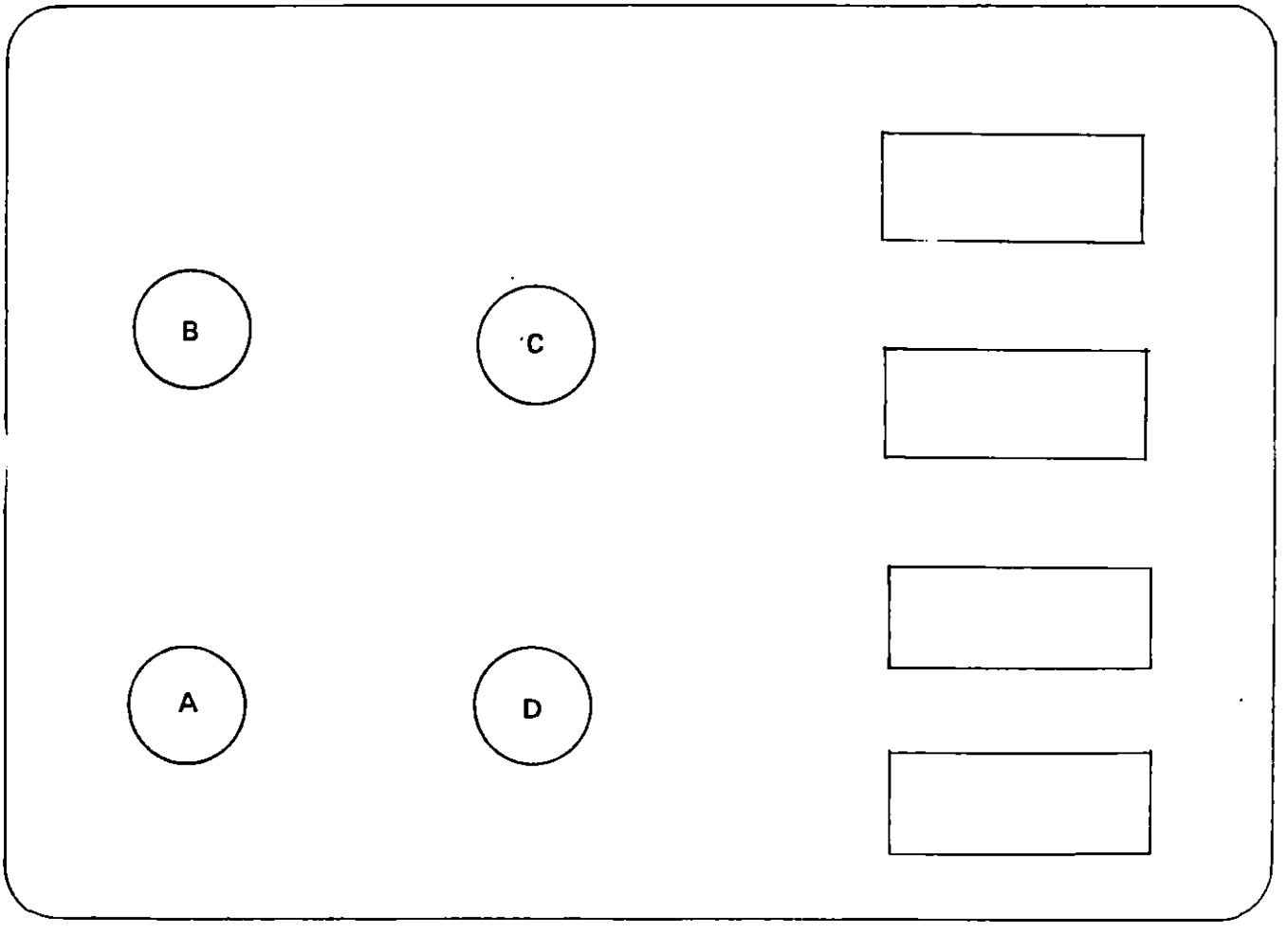



**Figure 3. LEDs and Motor Relays**

# CHAPTER 2

# THE INS8073 MICROPROCESSOR

## OVERVIEW

In this section, we begin discussing RB5X's INS8073 microprocessor. As you will see, this microprocessor was selected for use on the RB5X robot because it has Tiny BASIC in ROM. This allows you an easy way to program the robot using any computer with RS-232 interface capability.

RB5X's CPU board is centered around the 8073. In this section, we will discuss this processor in an overview. The 8073 has seven, 16-bit registers and five, 8-bit registers. The detailed function of these registers is covered in Appendix 4 of the RB5X Reference Manual. For purposes of brevity, we'll only discuss a summary of these.

Figure 4 contains a simplified block diagram of the INS8073. Refer to this figure during the following discussion of the INS8073.

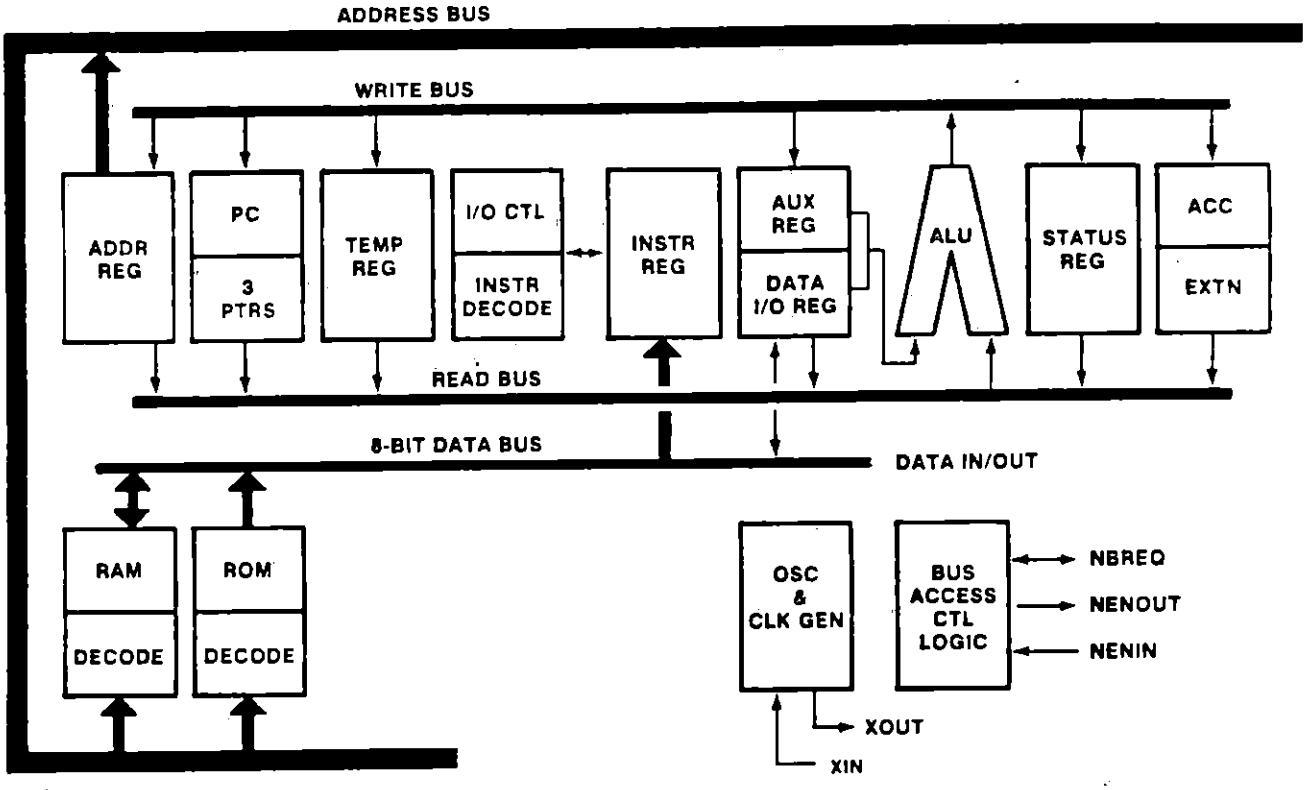



**Figure 4. 70-Series Family Simplified Block Diagram**

## OBJECTIVES

Upon completion of this material, you will be able to identify the different registers used in the INS8073 microprocessor and the function of each.

## PREREQUISITES

1. You should be familiar with the basic microprocessor architecture.

2. You should understand the function of microprocessor registers, such as:
   - o Accumulator
   - o Program counter
   - o Status flags
   - o Stack pointer
   - o Address register

## REFERENCES

1. Appendix 4, "INS8070-Series Family Microprocessor Data Sheets," of the RB5X Reference Manual.

## TECHNICAL INFORMATION

Figure 5 shows the 70-series microprocessor pin diagram. Notice that the 8073 has 16 address lines ($A_0$ through $A_{15}$) and eight data lines ($D_0$ through $D_7$).

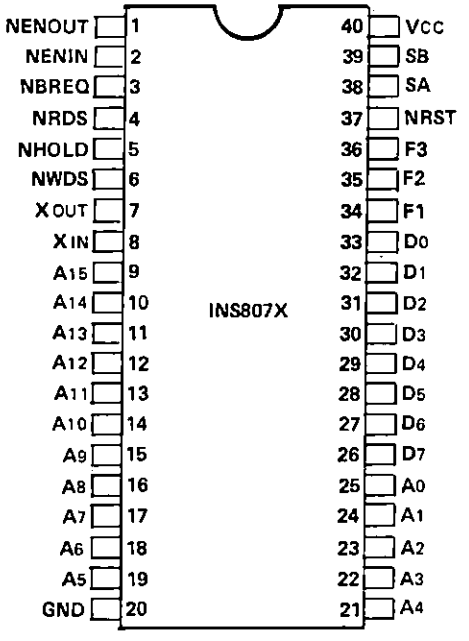



**Figure 5. 70-Series Pin Configuration**

The 8073 is a self-contained, general-purpose microprocessor designed for ease of operation. The timing circuit necessary to operate the 8073 can be a resistor-capacitor (R-C) network where timing is not particularly critical. If the application requires precision clock control, the R-C network can be replaced with a crystal. No external timing circuits (clock drivers, phase splitters, buffers, and so forth) are required in either case.

### 16-Bit Registers

The program-counter register (PC) is designed to contain the address of the next instruction to be executed. The PC register is incremented before each instruction is fetched.

The 8073 has three pointer registers, designated as follows:

o Stack pointer (SP)

o Pointer 2 (P2)

o Pointer 3 (P3)

The SP register is a general-purpose pointer designed to always show the top of the stack. It is incremented and decremented by PUSH and POP instructions, respectively. The P2 and P3 registers allow the programmer access to more than one stack at a time. Access to these two stacks is accomplished by the auto-indexe. addressing mode.

The address register is not directly accessible to the programmer. It contains the memory address of the information currently being operated upon by the microprocessor.

The temporary register is used for temporary storage of double-byte words. It also is used to hold a 16-bit multiplier or a divisor during these arithmetic operations.

The final 16-bit register in the 8073 is the extended (double-precision) accumulator register (EA). This register is formed by combining two, 8-bit registers -- the accumulator register and the extension register. The results of all double-precision arithmetic operations are found here. This register is also used to transfer double-byte words to and from memory.

### 8-Bit Registers

The accumulator register (AC) is a primary working register. The results of all 8-bit arithmetic and logic operations are stored here.

The extension register (E) is a temporary storage register. When combined with the AC register, it contains the high-order byte of 16-bit, multiple-precision words. The E/AC register combination is referred to as the extended accumulator register (EA).

The status register (S) stores the flag bits. These eight bits are as follows:

- o Carry/link (CY/L)
- o 2' complement overflow (OV)
- o Sense A (SA)

- Sense B (SB)
- Flag F1
- Flag F2
- Flag F3
- Interrupt enable (IE)

Both the SA and SB flags are interrupt lines that can be used for RS-232 signals and the F1 flag is an input that can be used for outputting RS-232 signals.

Figure 6 shows the arrangement of these eight flags in the eight registers:

| CY/L | OV | SB | SA | $F_3$ | $F_2$ | $F_1$ | IE |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**Figure 6. INS8073 Flags and Registers**

The instruction register is another register that is not directly accessible to the programmer, but is set according to program control and can only be changed again by more program control. The instruction register holds the operation code while this code is being decoded by the instruction decoder.

The auxiliary register and data I/O register are used for temporary storage of all I/O bytes as they are received or transmitted over the 8-bit, bidirectional data bus. This register is not directly accessible to the programmer.

The instruction set for the 8073 is shown in Appendix 4 of the RB5X Reference Manual. This instruction set consists of 192 instructions, which are grouped in the following nine addressing modes:

- Implied
- PC relative
- Pointer relative (via SP, P2, or P3)
- Single-byte immediate and double-byte immediate
- Direct
- Auto-indexed (via P2 or P3)
- Indexed (via P2 or P3)
- Absolute
- Indirect

## SUMMARY

In this section, you read about the different 8-bit and 16-bit registers found in the INS8073 microprocessor. You should now know the different status flags found in this microprocessor, and the ones that can be used to input and output data to an RS-232 connector. You also studied the different addressing modes that can be used when programming the INS8073 in machine language.

## QUESTIONS

1. How many 16-bit registers are found in the INS8073?

   _____

2. How many 8-bit registers are found in the INS8073?

   _____

3. What two 8-bit registers can be combined to form one 16-bit register?

   _____

4. Name four of the addressing modes in which the INS8073 can be programmed.

   _____

# CHAPTER 3

# THE RB5X BUS SYSTEM

## OVERVIEW

In Chapter 2, we discussed the INS8073 microprocessor and its internal registers. Now we'll look at the "highway" that allows this microprocessor to communicate with its support circuitry and with the outside world.

This highway, called a bus structure, is frequently a two-way street allowing the microprocessor to respond to the external environment and allowing the external environment to respond to the microprocessor. Along with this communication is a power-distribution network and a system for synchronizing these communications (timing/control).

A portion of these lines are address lines (unidirectional). These address lines are similar in function to the notation we put on the outside of an envelope to be mailed. It has no relationship to the contents of the envelope, but does contain information about where the contents are to be delivered. Our address bus lines allow us to send "contents" to and receive from RAMs, ROMs, ports, and the world.

Another key aspect of the highway "bus structure" is the data lines. These are analogous to the contents of an envelope. Just as you send and receive letters within envelopes, the data bus lines are

bidirectional. They contain information in the form of binary data that can originate in the microprocessor and can be sent to the outside world, or can originate in the outside world and be received by the microprocessor.

Synchronization (control) lines are used to determine in which direction the information flows, and which device in the outside world can send or receive this information.

## OBJECTIVES

Upon completion of this material, you will be able to:

1. Trace each jack pin to its termination point on the CPU board;
2. Identify which jack pins contain power (+5 VDC) and ground signals;
3. Determine which jack pins are address lines;
4. Determine which jack pins are data lines;
5. Determine which jack pins are used for communications external to the CPU board (I/O lines);
6. Identify which jack pins are the I/O lines on the RS-232, Port 1 connector; and
7. Determine which jack pins allow connection between the CPU board and the motor control board, the LED board, the sonar card, the power board, and the interface panel.

## PREREQUISITES

None.

## REFERENCES

1. CPU board schematic in Appendix D, "RB5X Schematics."
2. Mother board and wire diagram schematic in Appendix D, "RB5X Schematics."
3. Motor control board schematic in Appendix D, "RB5X Schematics."

4. LED board schematic in Appendix D, "RB5X Schematics."

5. Sonar card schematic in Appendix D, "RB5X Schematics."

6. Interface board schematic in Appendix D, "RB5X Schematics."

7. Power board schematic in Appendix D, "RB5X Schematics."

## TECHNICAL INFORMATION

Connector J0 on the mother board is the main connector to the CPU board. It handles the majority of all communication to and from the CPU board.

Connectors J1 and J2 are wired in parallel, and are used to provide communication to and from U20 on the CPU board to the outside world.

Connectors J3 and J4 are wired in parallel, and are used to provide communication to and from U23 on the CPU board to the outside world.

Connector J5 is used primarily for memory expansion and for communication to and from the voice/sound synthesis option.

Figure 7 in Section C, "Electronics," in the RB5X Reference Manual illustrates in block-diagram form these connections. Table 3.1 shows how the CPU board is wired to various I/O devices on the robot, such as bumper switches, programmable LEDs, and sonar.

| CPU Board I/O Chip | Port | I/O Device |
|---|---|---|
| U4 | A | Bumpers, programmable LEDs, horn, sonar on/off |
| U4 | B | Motors, sonar information, tape sense (battery recharge) |
| U4 | C | Charge and battery sense |

**Table 3.1 - CPU / I/O Device Wiring**

The RS-232 serial data link connects to microprocessor pins 38 (SA) and 34 (F1) through connector J7 on pins 4 and 1, respectively. You use this connector later in this chapter to connect an Apple computer to program the robot.

Table 3.2 addresses jacks J1 and J2 on the mother board. It identifies, by pin number, the line function as well as the chips and pins on the CPU board connected to that line.

Table 3.3 addresses jacks J3 and J4. It identifies, by pin number, the line function as well as the chips and the pins on the CPU board connected to that line.

Table 3.4 addresses jack J5. It identifies, by pin number, the line function as well as the chips and pins on the CPU board connected to that line.

| Origin | Termination | Identifier |
|---|---|---|
| Pin 1 | Ground | |
| Pin 2 | +5 VDC | |
| Pin 3 | No connection | |
| Pin 4 | No connection | |
| Pin 5 | No connection | |
| Pin 6 | No connection | |
| Pin 7 | No connection | |
| Pin 8 | No connection | |
| Pin 9 | No connection | |
| Pin 10 | No connection | |
| Pin 11 | U20, pin 3 | Port 2A1 |
| Pin 12 | U20, pin 1 | Port 2A3 |
| Pin 13 | U20, pin 39 | Port 2A5 |
| Pin 14 | U20, pin 37 | Port 2A7 |
| Pin 15 | U20, pin 15 | Port 2C1 |
| Pin 16 | U20, pin 17 | Port 2C3 |
| Pin 17 | U20, pin 12 | Port 2C5 |
| Pin 18 | U20, pin 10 | Port 2C7 |
| Pin 19 | U20, pin 19 | Port 2B1 |
| Pin 20 | U20, pin 21 | Port 2B3 |
| Pin 21 | U20, pin 23 | Port 2B5 |
| Pin 22 | U20, pin 24 | Port 2B6 |
| Pin A | Ground | |
| Pin B | +5 VDC | |
| Pin C | No connection | |
| Pin D | No connection | |
| Pin E | No connection | |
| Pin F | No connection | |
| Pin H | No connection | |
| Pin J | U8, pin 2 | NENIN |
| Pin K | U8, pin 3 | NBREQ |
| Pin L | U8, pin 7 | X O |
| Pin M | U20, pin 4 | Port 2A0 |
| Pin N | U20, pin 2 | Port 2A2 |
| Pin P | U20, pin 40 | Port 2A4 |
| Pin R | U20, pin 38 | Port 2A6 |
| Pin S | U20, pin 14 | Port 2C0 |
| Pin T | U20, pin 16 | Port 2C2 |
| Pin U | U20, pin 13 | Port 2C4 |
| Pin V | U20, pin 11 | Port 2C6 |
| Pin W | U20, pin 18 | Port 2B0 |
| Pin X | U20, pin 20 | Port 2B2 |
| Pin Y | U20, pin 22 | Port 2B4 |
| Pin Z | U20, pin 25 | Port 2B7 |

**Table 3.2 - Jacks J1 and J2**

| Origin | Termination | Identifier |
|---|---|---|
| Pin 1 | Ground | |
| Pin 2 | +5 VDC | |
| Pin 3 | No connection | |
| Pin 4 | No connection | |
| Pin 5 | No connection | |
| Pin 6 | No connection | |
| Pin 7 | No connection | |
| Pin 8 | No connection | |
| Pin 9 | U8, pin 37 | NRST |
| Pin 10 | U24, pin 4, and U24, pin 10 | BDCIS |
| Pin 11 | U23, pin 3 | Port 1A1 |
| Pin 12 | U23, pin 1 | Port 1A3 |
| Pin 13 | U23, pin 39 | Port 1A5 |
| Pin 14 | U23, pin 37 | Port 1A7 |
| Pin 15 | U23, pin 15 | Port 1C1 |
| Pin 16 | U23, pin 17 | Port 1C3 |
| Pin 17 | U23, pin 12 | Port 1C5 |
| Pin 18 | U23, pin 10 | Port 1C7 |
| Pin 19 | U23, pin 19 | Port 1B1 |
| Pin 20 | U23, pin 21 | Port 1B3 |
| Pin 21 | U23, pin 23 | Port 1B5 |
| Pin 22 | U23, pin 24 | Port 1B6 |
| Pin A | Ground | |
| Pin B | +5 VDC | |
| Pin C | No connection | |
| Pin D | No connection | |
| Pin E | No connection | |
| Pin F | No connection | |
| Pin H | No connection | |
| Pin J | U8, pin 2 | NENIN |
| Pin K | U8, pin 3 | NBREQ |
| Pin L | U8, pin 7 | X O |
| Pin M | U23, pin 4 | Port 1A0 |
| Pin N | U23, pin 2 | Port 1A2 |
| Pin P | U23, pin 40 | Port 1A4 |
| Pin R | U23, pin 38 | Port 1A6 |
| Pin S | U23, pin 14 | Port 1C0 |
| Pin T | U23, pin 16 | Port 1C2 |
| Pin U | U23, pin 13 | Port 1C4 |
| Pin V | U23, pin 11 | Port 1C6 |
| Pin W | U23, pin 18 | Port 1B0 |
| Pin X | U23, pin 20 | Port 1B2 |
| Pin Y | U23, pin 22 | Port 1B4 |
| Pin Z | U23, pin 25 | Port 1B7 |

**Table 3.3 - Jacks J3 and J4**

| Origin | Termination | Identifier |
|---|---|---|
| Pin 1 | Ground | |
| Pin 2 | +5 VDC | |
| Pin 3 | U15, pin 23, and U8, pin 13 | A11 |
| Pin 4 | U15, pin 21, and U8, pin 11 | A13 |
| Pin 5 | U15, pin 22, and U8, pin 12 | A12 |
| Pin 6 | U14; U16; U17; U18; U19, pin 13; U8, pins 30,44 | D3 |
| Pin 7 | U14; U16; U17; U18; U19, pin 17; U8, pins 26,48 | D7 |
| Pin 8 | U14; U16; U17; U18; U19, pin 14; U8, pins 29,45 | D4 |
| Pin 9 | U14; U16; U17; 18; U19, pin 9; U8, pins 33,41 | D0 |
| Pin 10 | U14; U16; U17; U18; U19, pin 23; U8, pins 16,33 | A8 |
| Pin 11 | U14; U16; U17; U18; U19, pin 2; U8, pins 18,31 | A6 |
| Pin 12 | U14; U16; U17; U18; U19, pin 4; U8, pins 4,29 | A4 |
| Pin 13 | U14; U16; U17; U18; U19, pin 6; U8, pins 23,27 | A2 |
| Pin 14 | U14; U16; U17; U18; U19, pin 8; U8, pin 25 | A0 |
| Pin 15 | U16; U17; U18; U19, pin 20; U14, pin 18; U8, pins 4, 39 | READ |
| Pin 16 | U14; U16; U17; U18; U19, pin 19; U8, pins 14,35 | A10 |
| Pin 17 | U15, pin 8 | CS5 |
| Pin 18 | U15, pin 10 | CS7 |
| Pin 19 | U15, pin 13 | CS9 |
| Pin 20 | U15, pin 15 | CS11 |
| Pin 21 | U15, pin 20 and U8, pin 10 | A14 |
| Pin 22 | U8, pin 2, and U8, pin 15 | NENIN |
| | | |
| Pin A | Ground | |
| Pin B | +5 VDC | |
| Pin C | U8, pins 3,20 | NBREQ |
| Pin D | U24, pins 5,10 | BDIS |
| Pin E | U8, pin 7 | X O |
| Pin F | U14; U16; U17; U18; U19, pin 10; U8, pins 32,42 | D1 |
| Pin H | U14; U16; U17; U18; U19, pin 15; U8, pins 28,46 | D5 |
| Pin J | U14; U16; U17; U18; U19, pin 16; U8, pins 27,47 | D6 |
| Pin K | U14; U16; U17; U18; U19, pin 11; U8, pins 31,43 | D2 |
| Pin L | U14; U16; U17; U18; U19, pin 1; U8, pins 17,32 | A7 |
| Pin M | U14; U16; U17; U18; U19, pin 3; U8, pins 19,30 | A5 |
| Pin N | U14; U16; U17; U18; U19, pin 5; U8, pins 22,28 | A3 |
| Pin P | U14; U16; U17; U18; U19, pin 8; U8, pin 25 | A1 |
| Pin R | U16; U17; U18; U19, pin 21; U8, pins 6,38; U20,23, pin 36 | WRITE |
| Pin S | U14; U16; U17; U18; U19, pin 22; U8, pins 15,34 | A9 |
| Pin T | U15, pin 7 | CS4 |
| Pin U | U15, pin 9 | CS6 |
| Pin V | U15, pin 11 | CS8 |
| Pin W | U15, pin 14 | CS10 |
| Pin X | U15, pin 16 | CS12 |
| Pin Y | U15, pin 18, and U8, pin 9 | A15 |
| Pin Z | U8, pin 37, and J7, pin 2 | NRST |

**Table 3.4 - Jack J5**

## SUMMARY

With the tables in this chapter, you can now easily tell the origin and termination of any line from any pin of any jack to its chip pin. This is a great help during the upcoming experiment, when you trace lines throughout the RB5X system.

You should now have a better understanding of the RB5X system bus. With this data, you are armed with the information to allow you to trace all signal and power paths from the peripherals to the CPU, and vice versa.

## QUESTIONS

1. What is the function of the address bus line?

2. What is the function of the port lines?

3. What peripherals are connected to the CPU board via J7?

## Experiment 3.1

## Tracing the Bus

### Overview

In Chapter 3, we discussed which pins on the mother board set of jacks were connected to which chip pins on RB5X's CPU board. This experiment gives you experience in tracing the bus structure for the whole system. You will trace lines from various boards back to the CPU board. The external boards to be traced include:

- o Motor control board,
- o LED board,
- o Sonar card,
- o Power board, and
- o Interface board.

### Objectives

Upon completion of this experiment, you will be able to identify I/O signal paths from the CPU board to the external boards listed above.

## Prerequisites

All previous experiments in this workbook.

## References

1. CPU board schematic in Appendix D, "RB5X Schematics."
2. Mother board and wire diagram schematic in Appendix D, "RB5X Schematics."
3. Motor control board schematic in Appendix D, "RB5X Schematics."
4. LED board schematic in Appendix D, "RB5X Schematics."
5. Sonar card schematic in Appendix D, "RB5X Schematics."
6. Interface board schematic in Appendix D, "RB5X Schematics."
7. Power board schematic in Appendix D, "RB5X Schematics."

## Technical Information

In this experiment, you identify continuity between various points on the system with an ohmmeter. All power to the system must be switched off. (If you are using RobotLab, this includes removing the 110 VAC from the wall and the positive lead connector from the battery.) Each jack (J) number has a corresponding plug (P) number.

## Procedure

1. Trace pin 1 of the MC1413 on the motor control board to U14 on the CPU board. Which pin does it connect to on U14?

_____

Verify the continuity with an ohmmeter.

---

2. Trace the anode of the photodiode on the motor control board to U11 on the CPU board. Which pin on U11 is connected to the anode side of the photodiode?

---

What is the ohmmeter reading?

---

3. Trace the cathode of the diode wired to pin 5 on the soldered connector on the LED board to U14 on the CPU board. What pin of U14 is connected to this LED?

---

What is the ohmmeter reading between these points?

---

4. Trace from the LED on pin 24 of the soldered connector on the LED board to U5 on the CPU board. To what pin of U5 does the LED connect?

---

Why is there a resistor on the anode side on this LED, while none of the others have one there?

---

---

5. Trace the signal path from the XLG line of the sonar card to U11 on the CPU board. To what pin of U11 does it connect?

---

What would be the difficulties in verifying this with an ohmmeter?

---

6. Which pin of J0 on the CPU board is pulled to ground with the software reset switch? (HINT: The software reset switch is SW1 on the power control board.)

_____

7. SA and F1 are signal I/O lines for the microprocessor. Pins 2 and 3 are the signal I/O pins for the RS-232 connector on the interface panel. Which pins on the Port 1 RS-232 connector on the interface panel go to SA and F1 on the microprocessor chip?

_____

8. Verify that pin 1 on the software module socket is wired to pin 1 on the U14 socket. **CAUTION:** Remove the software module from the socket on the interface panel before taking the measurement.

## Summary

You have gained experience in tracing signal paths from the outlying boards to the CPU board, using both the schematic reading technique and an ohmmeter verification procedure. The circuitry is now yours to explore.

## Questions

1. Why are all power lines to the motors at zero volts potential when the relays are deactivated?

_____

_____

_____

2. J5 is a 44-pin connector numbered 1 through 22 and A through Z. Which pin designators are not used? Why?

3. Which jacks on the mother board are wired in parallel?

4. Which jack on the mother board connects to the battery recharger circuit?

5. On the CPU board schematic, directly below the horn, are 12, .01 micro fared capacitors. Why is this used type of capacitor used instead of a single capacitor of equal value?

# CHAPTER 4

# GETTING STARTED WITH ASCII PRO

## OVERVIEW

ASCII Express "The Professional" (ASCII Pro) is a communications software package that can be used as an operating system for any Apple II computer, and that allows RB5X to use your Apple as a "dumb terminal." ASCII Pro allows you to communicate with RB5X in Tiny BASIC, and contains procedures for saving on diskette and retrieving from diskette those programs that you write for RB5X. ASCII Pro also determines the proper baud rate for transferring data to and from RB5X, and gives you access to disk commands such as CATALOG and DELETE.

## OBJECTIVES

Upon completion of this text material you will be able to:

1. Have RB5X turn control over to your Apple computer,
2. Set the desired baud rate for data transfer,
3. Save Tiny BASIC programs on the diskette,
4. Delete Tiny BASIC programs from Apple diskette,

5. Catalog the program files on Apple diskette, and

6. Retrieve and run Tiny BASIC programs from the Apple diskette.

## PREREQUISITES

1. Review Section B, "Getting Started," in the RB5X Reference Manual.

## REFERENCES

None.

## TECHNICAL INFORMATION

Due to the detailed nature of ASCII Pro, we have broken this section of Chapter 4 into subsections. Each contains specific, step-by-step procedures to facilitate your use of ASCII Pro.

In the following instructions, the symbol <RET> indicates a carriage return on the Apple keyboard.

### Setting Up the Robot

The following steps illustrate how to power up the computer and RB5X, and how to set the baud rate to transfer data to and from the robot.

1. Boot the ASCII Pro diskette in your Apple. The computer returns the ASCII Pro prompt -- AE:TERM--> -- on the CRT.

2. Press the Control key and the Q key simultaneously. (This combination of key presses is from now on shown as ^Q.) Now press the B key.

3. The computer asks you to enter the baud rate. Press the 4 key to set the computer's baud rate to 300.

4. Locate the robot's baud rate switches on the top center of the CPU board. Set them to 0110 to configure the robot to communicate at 300 baud.

5. Make sure the utility software module is properly plugged into the socket on RB5X's interface panel. Then switch the robot on and allow it to run its self-diagnostic routine.

6. Press bumpers 2 and 4 on the robot simultaneously to enter Tiny BASIC command mode. The computer displays "STOP AT 30" followed by |-.

7. Type in:

   **LIST<RET>**

   The following program then displays:

   ```
   10 CLEAR
   20 LINK @#85A1
   30 STOP
   ```

   This is the Tiny BASIC program that accesses the initialization routine located on the utility software module.

8. Type in:

   **NEW #1000<RET>**

   Wait for the computer to return the Tiny BASIC prompt |.

9. Type in:

   **NEW<RET>**

   This clears all program variables. You are now ready to enter your own programs.

### ASCII Pro Menus

The following is a list of the menu option available to you while using ASCII Pro with RB5X.

Press ^Q 1 (Control/Q followed by the 1 key) to view the first ASCII Pro menu. The menu lists 18 options, but we'll only address those important when using RB5X.

#### C=CLEAR BUFFER

This option clears the contents of the copy buffer that is used to

load and save programs to and from diskette. This is the area of computer memory through which ALL data must pass when moving to and from diskette.

**I=DISK COMMANDS**

This option allows you to CATALOG or DELETE the diskette program files.

**R=COPY BUFFER ON-OFF**

This option enables and disables the copy buffer.

**S=SEND A FILE**

This option allows you to send a program file from diskette to the computer via the copy buffer.

**W=WRITE BUFFER TO DISK**

This option allows you to store a program, via the copy buffer, on diskette from the computer.

Press ^Q 2 to display menu #2. This menu displays the status of 19 menu options. Notice that the baud rate is 300. The Y option allows you access to the EDIT functions. You may or may not want to use these functions during program modification operations. Refer to the ASCII Pro manual for more information on this option.

Press ^Q followed by one tap of the space bar to return to command mode. Type in LIST<RET> and you'll notice that the computer returns the >- prompt. This signifies that all programs in the computer have been erased.

### Saving a Program

Let's assume that the following program has already been entered into the computer.

```
10 PRINT"THIS IS A TEST TO SAVE A PROGRAM"
20 STOP
```

The following steps are needed to save this program on the computer diskette.

1. Press ^Q R to turn on the copy buffer.

2. Type LIST<RET> to load the program into the copy buffer.

3. Press ^Q R to turn off the copy buffer.

4. Press ^Q W to store the contents of the copy buffer on the computer diskette. The computer responds by asking FILENAME?

5. Give the program a name -- for example, let's call this one JUNK. Enter:

   **JUNK<RET>**

   The diskette activates and saves your program. When the save operation is complete, the computer displays the prompt AE:TERM->.

6. Type ^Q I to enter the Disk command option. Press C<RET> to CATALOG the files on diskette.

   The diskette catalog displays, but because of its length, you must press the space bar to view the complete catalog. The last line of the catalog listing informs you of the number of sectors and memory locations still available for use on this diskette.

### Loading Programs from Diskette

To load a program into the computer from diskette, you must do the following:

1. Type NEW<RET> to clear the computer of all existing programs.

2. Press ^Q I followed by C<RET>. This allows you to view the catalog and to get the correct spelling of the program name.

3. Press ^Q C to clear the copy buffer.

4. Press ^Q R to run the copy buffer on.

5. Press ^Q S to send a file from diskette into the computer's memory. The computer asks FILENAME?.

6. Type in the name of the program to be loaded and then press the Return key.

7. The computer asks S)TANDARD OR P)ROTOCOL?. Press Return to select the default of S.

8. The computer now asks L)INE OR C)HARACTER?. Press Return to select the default of L.

9. The computer now asks PROMPT? Press > and the proper program is loaded and listed for you. When the listing is complete, the computer beeps and displays AE:TERM-->.

10. To run this program, type RUN<RET>. The computer gives you an ERROR 4 message. ERROR 4 always appears after you load a program; you may ignore it.

11. Retype RUN<RET> and the program executes.

### Error Messages

There are 12 error messages that ASCII Pro and Tiny BASIC may return.

| Error | Meaning |
|---|---|
| 1 | Out of memory |
| 2 | Statement used improperly |
| 3 | Unexpected character (after legal command) |
| 4 | Syntax error (used when no other applies) |
| 5 | Value (format) error |
| 6 | Ending quote missing from string |
| 7 | Go target line does not exist |
| 8 | Return without previous GOSUB |
| 9 | FOR/NEXT or DO/UNTIL nested too deep |
| 10 | NEXT without matching FOR |
| 11 | UNTIL without previous DO |
| 12 | Division by zero |

### Deleting a File from Diskette

Use the following steps to delete a file from the computer diskette.

1. Press ^Q I followed by C<RET> to list the programs on the diskette. Check for the exact spelling of the program you wish to delete.

2. Press ^Q I followed by D. The system then prompts you for FILENAME?.

3. Type in the name of the program you wish to delete, followed by a Return.

4. The disk drive activates, erasing the named program and computer returns AE:TERM--> when the process is complete.

5. You may wish to reconsult the diskette catalog to ensure that the correct program was erased.

## SUMMARY

You should now have a working knowledge of the following procedures:

1. Turning control of RB5X over to the computer,

2. Setting the baud rate of both RB5X and the computer to allow for data transfer,

3. Loading and saving programs to and from diskette, and

4. Cataloging and deleting programs from diskette.

These procedures allow you to use RB5X with the ASCII Pro diskette. We encourage you to return to this section as often as needed until the procedures become second nature.

## QUESTIONS

1. List the procedures necessary to set the baud rates of both RB5X and the computer to 300.

_____

_____

_____

_____

_____

2. List the procedures necessary to save a program named TEST on diskette.

3. What would be the probable cause for the computer to issue an ERROR 1 message?

4. What are the consequences of not having both RB5X and the computer set to 300 baud?

## Experiment 4.1

## Initializing with ASCII Pro

### Overview

In Experiment 1.2, you learned how to initialize RB5X using its utility software module and how to properly set the interface panel switches. As you remember, the initialization sequence is as follows (keeping in mind that RB5X moves when the green LEDs light; RobotLab does not, having no motors):

1. The robot sounds three short beeps of its horn, followed by the flashing of five LEDs on the LED board.

2. The two front, green LEDs light for one second and RB5X rolls forward.

3. The green LEDs at clock positions 10:00 and 5:00 light for one second, and RB5X spins clockwise.

4. The green LEDs at clock positions 7:00 and 1:00 light for one second, and RB5X spins counterclockwise.

5. The two rear, green LEDs light for one second and RB5X rolls backward for one second.

6. The four clear LEDs flash and the center, amber LED remains on.

This routine causes the robot to go to the top of the memory on the utility software module and to execute a series of machine-language commands to perform the power on/reset routine.

During this experiment, you execute the initialization routine using the ASCII Pro diskette and your Apple computer.

### Objectives

Upon completion of this experiment, you will be able to:

1. Use the computer to run the initialization routine on RB5X or RobotLab.

### Prerequisites

1. Knowing how to set up the robot with ASCII Pro.

### References

1. Chapter 4, "Getting Started with ASCII Pro."

### Technical Information

The Tiny BASIC program to begin the initialization routine consists of the following lines:

```
10 CLEAR
20 LINK #85A1
30 STOP
```

The CLEAR command zeroes out all variables from previous programs. This is a standard BASIC command.

The LINK #85A1 command allows you to execute a machine-language program by giving, in Tiny BASIC, the hexadecimal starting address of the machine-language routine.

The STOP command is self-explanatory.

This technique of initializing at the beginning of a Tiny BASIC program can be very useful. It allows the robot to initialize, and to wait for bumpers 2 and 4 to be pressed before continuing execution of

the Tiny BASIC program.

If you use RB5X (instead of RobotLab) during this experiment, remember to disconnect the RS-232 cable from the robot before the main body of the program begins execution.

**Procedure**

1. Boot the ASCII Pro diskette on the Apple computer.

2. Press ^Q B followed by the 4 key to set the baud rate of the computer to 300.

3. With the utility software module in the socket on the interface panel and the RS-232 cable connecting the robot and the computer, power up RB5X (or RobotLab) and allow it to run through its self-diagnostic routine.

4. After this routine finishes executing, press bumpers 2 and 4 simultaneously to enter command mode. The computer returns the message STOP AT 30.

5. Type:

   **NEW #1000<RET>**

6. Wait for the Tiny BASIC prompt | and then type:

   **NEW<RET>**

   to clear RAM of all programs.

7. Enter the following program:

   **10 CLEAR<RET>**
   **20 LINK #85A1<RET>**
   **30 STOP<RET>**

8. Type:

   **LIST<RET>**

   to check that you properly entered the program.

9. Type:

   RUN<RET>

   to perform the initialization routine.

10. Modify the program as follows:

```
30 PRINT"TINY BASIC NOW HAS CONTROL"<RET>
40 STOP<RET>
```

11. Check the program for errors by typing:

    LIST<RET>

12. Type:

    RUN<RET>

    to run this new program. After initialization is complete, press bumpers 2 and 4 simultaneously.

Notice that the robot has returned control to the computer and has completed its Tiny BASIC program by printing TINY BASIC NOW HAS CONTROL on the the CRT.

**Summary**

This experiment has instructed you how to initialize the robot using the computer. You have also learned how to delay execution of the main body of a Tiny BASIC program. To resume execution of the remainder of the program, you simply press bumpers 2 and 4.

**Questions**

1. In the following program, what would happen after bumpers 2 and 4 are pressed?

```
10 CLEAR
20 LINK #85A1
30 LINK #85A1
40 STOP
```

2. At what hexadecimal address does the initialization routine begin?

3. What is the purpose of the CLEAR command?

NOTES:

## Experiment 4.2

## Saving and Loading Programs

### Overview

Two important aspects of ASCII Pro allow you to SAVE and LOAD Tiny BASIC programs on the Apple diskette. This saves you time, since you don't have to re-enter programs each time you want the robot to execute a routine. During this experiment, you enter simple Tiny BASIC programs, save these programs on diskette, clear the computer's memory of all programs, reload and execute a program, and catalog and delete the same program from the diskette.

### Objectives

Upon completion of this experiment, you will be able to:

1. Enter simple Tiny BASIC programs into the computer,
2. Save Tiny BASIC programs on diskette,
3. Clear the computer of all programs,
4. Reload and execute Tiny BASIC programs from diskette,
5. Catalog the program files on diskette, and

6. Delete program files from diskette.

### Prerequisites

1. Completion of Chapter 4, "Getting Started with ASCII Pro."

2. Completion of Experiment 4.1, "Initializing with ASCII Pro."

### References

1. Chapter 4, "Getting Started with ASCII Pro."

### Technical Information

It is very cumbersome to have to re-enter Tiny BASIC programs into the computer every time you need them. The following steps save Tiny BASIC programs on diskette. (These steps were outlined earlier and are repeated here to remind you of the process.)

1. Write the program.

2. Press ^Q R. This turns the copy buffer on.

3. Type LIST<RET>. This loads the program into the copy buffer.

4. Press ^Q R. This turns the copy buffer off.

5. Press ^Q W. The computer then displays the question FILENAME?.

6. Type in the program, or file, name followed by a Return.

The following steps are needed to load a Tiny BASIC program from diskette:

1. Type NEW #1000<RET>.

2. Type NEW<RET> after the computer displays the Tiny BASIC prompt >.

3. Press ^Q R. This turns on the copy buffer.

4. Press ^Q S followed by the name of the program to be loaded from diskette.

5. The computer asks S)TANDARD OR P)ROTOCOL? Press Return to select the default of S.

6. the computer asks L)INE OR C)HARACTER? Press Return to select the default of L.

7. The computer asks PROMP? Press the > key.

8. The diskette activates and the program is loaded and listed on the CRT. Upon return of the AE:TERM--> prompt, the program is ready to be run.

9. Enter RUN<RET>. The first RUN statement after loading returns an ERROR 4 message on the CRT. Just ignore this error message and retype RUN<RET>.

You now have the list of steps to load and save your Tiny BASIC programs on the computer diskette. We will now demonstrate the use of these steps in the following procedure.

**Procedure**

1. Boot the ASCII Pro diskette.

2. Set the baud rate to 300 on both the robot and the computer.

3. Initialize the robot, and press bumpers 2 and 4 simultaneously once RB5X has completed its self-diagnostic routine.

4. Type in the following Tiny BASIC program, remembering to end each line with a Return.

```
 10 PRINT"THIS IS USED TO TEST LOADING"
 20 PRINT" AND SAVING OF PROGRAMS ON THE"
 30 PRINT" DISKETTE."
 40 PRINT"":PRINT"":PRINT"":PRINT""
 50 PRINT"IT WILL ALSO BE USED TO TEST"
 60 PRINT" THE CATALOG AND DELETE COMMANDS."
 70 PRINT"":PRINT"":PRINT"":PRINT""
 80 PRINT"THIS PROGRAM IS CALLED"
 90 PRINT" LOAD AND SAVE TEST.
100 STOP
```

5. Run the program by typing:

RUN<RET>.

6. Press:

   ^Q R

   to turn on the copy buffer.

7. Type:

   LIST<RET>

   to load the program into the copy buffer.

8. Press:

   ^Q R

   to turn the copy buffer off. **NOTE:** If you omit this step, every keystroke that follows is added to the copy buffer along with the program and is saved along with the program.

9. Press:

   ^Q W

   and type in:

   LOAD AND SAVE TEST<RET>

   in response to the question FILENAME?

**NOTE:** If a file by this name already exists on the diskette, the computer responds with the following:

FILE EXISTS" OVERWRITE? (DEF=N)

Answer this question with Y (for Yes) if you wish to write over the existing file and the system saves the new program.

Answer this question with N (for No) if you wish to change the name of the new program. The system then asks FILENAME? and waits for you to enter a new file name, followed by a Return.

10. Check to see that the saved procedure worked by typing:

^Q I

The computer returns the question CATALOG OR DELETE?

11. Enter:

    C<RET>

    to select the catalog option. In some cases, the catalog is more than one page long. When the listing stops, press the space bar to view successive pages. The last line of the catalog contains a statement showing the number of free sectors and available memory on the diskette.

12. Verify that LOAD AND SAVE TEST is in the catalog listing. **NOTE:** It may be anywhere in the listing.

13. If the LOAD AND SAVE TEST program is not shown in the listing, repeat steps 5 through 12.

14. Type:

    NEW<RET>

    to erase all programs from the computer's memory.

15. Type:

    LIST<RET>

    to verify that all programs have been erased.

16. Load the LOAD AND SAVE TEST program from diskette by typing:

    ^Q R

    to turn on the copy buffer.

17. Type:

    ^Q S

    and, at the request for FILENAME?,

    LOAD AND SAVE TEST<RET>

18. Press the Return key when the computer asks you S)TANDARD OR P)ROTOCOL? to select the default of S.

19. Press the Return key when the computer asks you L)INE OR C)HARACTER? to select the default of L.

20. Press > in answer to the question PROMPT? The diskette activates, and the program loads and lists on the CRT. When the load procedure is complete, the computer beeps and displays the familiar AE:TERM--> prompt.

21. Type:

    **RUN<RET>**

    and the computer returns an ERROR 4 message. Ignore this message and retype RUN<RET>.

22. To delete this program from diskette, type:

    **^Q I**
    **D**

23. In response to the question FILENAME?, type:

    **LOAD AND SAVE TEST<RET>**

24. Call up the diskette's catalog and note that LOAD AND SAVE TEST no longer exists on the diskette. Remember, however, that this program still resides in the computer.

25. To verify this, type:

    **LIST<RET>**

26. Type:

    **NEW<RET>**

    to erase this program from the computer's memory.

    Use the DELETE command carefully to avoid erasing important program files on your diskette.

## Summary

This experiment has given you practice in using the computer diskette to load and save programs. It has also given you experience in viewing the contents of a diskette using the CATALOG command and in removing unwanted programs from diskette using the DELETE command.

## Questions

1. How many free sectors exist on your diskette?

2. Is the copy buffer on or off after you load a program from diskette?

3. List the procedures to save on diskette a program named TEST.

4. List the procedures to load from diskette a program named NEW TEST.

5. List the procedures to erase from diskette a program named ANOTHER TEST.

_____

_____

_____

_____

_____

_____

# CHAPTER 5

# A COMMAND OF TINY BASIC

## OVERVIEW

In previous chapters and experiments, you learned to initialize RB5X using ASCII Pro, to load and save Tiny BASIC programs from and to diskette, and to catalog and delete files from the computer diskette. In this section, you use ASCII Pro and Tiny BASIC to write programs to control the actions of the robot.

## OBJECTIVES

Upon completion of this text material, you will understand how to uses the following Tiny BASIC commands:

1. LINK

2. DELAY

3. CLEAR

4. DO/UNTIL

## PREREQUISITES

1. Chapter 4, "Getting Started with ASCII Pro."

## REFERENCES

1. Chapter 4, "Getting Started with ASCII Pro."
2. Section B, "Getting Started," in the RB5X Reference Manual.
3. Appendix E, "Utility Software Module Routine Listings."

## TECHNICAL INFORMATION

Being able to load and save programs, as well as to catalog and delete, necessitates that these programs first be written in Tiny BASIC. As you discovered in Chapter 4, "Getting Started with ASCII Pro," some of the Tiny BASIC commands are similar to standard BASIC commands. These common commands include, but are not limited to, the following:

1. PRINT
2. INPUT
3. GOTO
4. IF/THEN
5. FOR/NEXT
6. GOSUB/RETURN
7. LIST
8. RUN
9. NEW
10. Logic Operators
    A. AND
    B. OR
    C. NOT
11. Arithmetic Operators
    A. +
    B. -
    C. /
    D. *
    E. =
    F. |
    G. [
    H. [|

You should already have an understanding of these basic commands. The ones we concentrate on now are those that are unique

to Tiny BASIC:

1. LINK
2. DELAY
3. CLEAR
4. DO/UNTIL

There are other unique Tiny BASIC commands, but they are beyond the scope of this course. The following paragraphs discuss each of these Tiny BASIC commands and give examples of their application.

### LINK

The LINK command allows you to transfer control from any Tiny BASIC program to an 8073 microprocessor, machine-language subroutine, provided that the machine-language subroutine contains the RET (Return) instruction at the end of the subroutine.

**Example**

The following program was used to initialize RB5X using the ASCII Pro diskette:

```
10 CLEAR
20 LINK #85A1:REM INITIALIZATION ROUTINE
30 STOP
```

This exits from your Tiny BASIC program into the machine-language subroutine that begins at address #85A1, and returns you to your Tiny BASIC program at line #30. (The subroutine at address #85A1 has a RET at the end.) You find the program by going to the address listed (#85A1) and following it until you see a RET statement (address #8657). (A listing of the machine-language subroutines located on the utility software module can be found in Appendix E.)

### DELAY

The DELAY command allows you to insert pauses in the execution of a program, and is followed by a number. This number can be from 1 to 1040 with each number representing a one-millisecond pause. For example, a delay of one second is DELAY 1000. For longer delays than that, you can use the example shown below.

**Example**

```
10 FOR I=1 TO X: DELAY1000:NEXT I
```

This gives you a delay of one second for each count of I. Use X to represent the number of seconds you wish to have the execution paused.

## CLEAR

The CLEAR statement is used to zero all variables, and to terminate all pending interrupts and loops. Use the CLEAR statement carefully, as it may crash your program.

**Example**

The following is an example of how to properly use the CLEAR statement.

```
10 B=@#7800:REM PEEK COMMAND TO READ BUMPERS
20 PRINT B
30 FOR I=1 TO 3:DELAY 1000:NEXT I
40 CLEAR
50 PRINT B
```

This program reads signals to the robot's bumper switches and assigns their value to variable B. It displays the value of variable B on the CRT, clears all variables including B, and then prints the cleared value of variable B, which is, of course, zero.

## DO/UNTIL

This statement is used to program loops, helping keep the GOTO statements to a minimum. The overall effect is to greatly improve the readability and clarity of your Tiny BASIC programs.

**Example**

The following program segment illustrates the use of the DO/UNTIL statement.

```
10 N=1
20 DO
30 N=N+1
40 PRINT N
```

```
50 UNTIL (N>10)
60 STOP
```

The program prints the incremented value of N and stops at line 60 when the conditions in the parentheses are met. In other words, the program stops when N is greater than 10.

## SUMMARY

With a background in regular BASIC and this technical information on some of the commands unique to Tiny BASIC, you are now ready to create your own Tiny BASIC programs for RB5X. You will have plenty of opportunities to use these skills during the following experiments.

## QUESTIONS

1. How long a delay, in seconds, is created by the following program segment?

```
50 PRINT"THIS IS THE START OF THE DELAY."
60 DELAY1000
70 FOR I=1 TO 5:DELAY1000:DELAY1000
80 NEXT I
90 PRINT"THIS IS THE END OF THE DELAY."
```

---

2. Upon completion of the following program, what are the six values printed on the CRT?

```
70 B=1
80 C=2
90 D=3
100 PRINT B:PRINT C:PRINT D
110 CLEAR
120 PRINT B:PRINT C:PRINT D
130 STOP
```

---

3. The following program performs what machine-language subroutine? (HINT: Check Appendix E, "Utility Software Module Routine Listings.") At what address does the machine-language subroutine begin and end?

```
10 CLEAR
20 LINK#8069
30 STOP
```

4. What happens if line 20 of the preceding program is changed to the following?

```
20 LINK#80EF
```

5. In the following program, how many times does the DO/UNTIL loop execute?

```
10 A=10
20 B=100
30 C=50
40 I=1
50 DO
60 N=A+B-C
70 PRINT N
80 I=I+1
90 UNTIL (I=5)
100 STOP
```