



**EXPERIMENT ASSEMBLY
AND
REAL-TIME FIRMWARE
MANUAL**

SELECTED EXPERIMENTS

INTEGRATED COMPUTER SYSTEMS, INC.

REV. A/4/81

... O (.)



EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

SELECTED EXPERIMENTS

by Edward Dillingham

REV. A/4/81



INTEGRATED
COMPUTER
SYSTEMS™

THIS PAGE INTENTIONALLY LEFT BLANK.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

TABLE OF CONTENTS

	PAGE
1. INTRODUCTION	1-1
1.1 Microprocessor Training Laboratory	1-2
1.2 Receiving Inspection	1-3
1.3 Assembly	1-3
1.3.1 MTS and Power Supply	1-4
1.3.2 Keyboard Test	1-5
1.3.3 Installing PROM's	1-11
1.3.4 Testing PROM's	1-16
1.3.5 Connecting Interface Training System	1-18
1.3.6 Testing ITS	1-22
1.3.7 Integrated Experiment Assembly	1-22
2. EDUCATIONAL FIRMWARE	2-1
2.1 Moving Message Demonstration (Program 0)	2-3
2.2 Playing a Tune from Memory	2-5
2.3 Playing a Tune from the Keyboard	2-6
2.4 Binary Arithmetic (Program 1)	2-8
2.5 Hexadecimal Arithmetic (Program 2)	2-9
2.6 Thermometer (Program 3)	2-12
2.7 Cooling Thermostat (Program 3)	2-13
2.8 Motor Speed Control (Program 4)	2-14
2.9 Program Storage	2-16
3. BINARY NUMBERS	3-1
3.1 Demonstration Program "bin"	3-1
3.2 Hexadecimal Numbers	3-5
3.3 Review and Self-Test	3-8
3.4 Binary Addition	3-9
3.5 Binary Subtraction	3-11
3.6 Hexadecimal Arithmetic	3-12

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

TABLE OF CONTENTS (continued)

	PAGE
4. STARTING COURSE 525A	4-1
4.1 Further Reference to This Book	4-2
4.2 User Programs and the Monitor	4-3
5. COURSE 536A EXPERIMENTS	5-1
5.1 The PMTL Experiment Board	5-3
5.1.1 Loudspeaker Control	5-7
5.1.2 Thermostor Connection	5-9
5.1.3 Slot Sensor Connection	5-9
5.1.4 Motor Control Drive Circuit	5-11
5.2 Experiment Board Connections for Course 536A Experiments - VOLUME I	5-13
5.3 Experiment Board Connections for Course 536A Experiments - VOLUME II	5-24
6. THERMOMETER AND THERMOSTAT (Program 3)	6-1
7. MOTOR SPEED CONTROL (Program 4)	7-1
8. TUNE AND MESSAGE VARIATIONS (Program 0)	8-1

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

FIGURES AND TABLES

		PAGE
Figure 1-1	MTS Board Layout	1-12
Figure 1-2	PROM Locations	1-13
Figure 1-3	Installing PROM's	1-15
Figure 1-4	MTS/ITS Ribbon Cable	1-19
Figure 1-5	ITS and MTS Connected	1-21
Figure 1-6	ITS and Experiment Assembly	1-23
Figure 1-7	Control Wire and Batteries	1-25
Figure 1-8	Experiment Assembly Controls	1-27
Figure 5-1	Integrated Experiment Board	5-2
Figure 5-2	Loudspeaker Connections	5-6
Figure 5-3	Thermistor and EXT4 Input Connections	5-8
Figure 5-4	Motor Circuit Connections	5-10
Table 5-1	Effects of Potentiometer Adjustments on Experiment Board and ITS	5-4
Table 8-1	Note Pitch	8-5
Table 8-2	Note Duration	8-6

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

This page intentionally left blank.



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 1

INTRODUCTION



THIS PAGE INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

The use of microprocessors and microcomputers as control devices has created a revolution in the design of many products and systems. The equipment, firmware and test materials supplied by ICS teach the technical use of microprocessors in four ways:

- Microprocessor Training Laboratory provides a program development and testing tool with interfaces to experimental setups.
- Real Time Educational Firmware gives demonstrations that indicate some of the functions that a microcomputer can accomplish, and allow hands-on manipulation of the demonstrations.
- Course 525A teaches 8080 programming at the machine language level, programming techniques applicable to all microprocessors, and details of 8080 microcomputer hardware.
- Course 536A teaches interfacing and control system techniques with special emphasis on interrupt handling, real time programming and closed loop control.

This manual describes the assembly of the microprocessor training laboratory and the use of the demonstration programs. It uses some of these demonstrations to teach the fundamentals of binary and hexadecimal numbers. It also includes details on how to use the Integrated Experiment Assembly in the experiments of Course 536A.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

1.1 Microprocessor Training Laboratory

The four major components supplied with ICS Courses 525A and 536A are connected together to make a complete training laboratory. These components are:

- MTS - Microprocessor Training System
This is the circuit board with a transparent cover, 25 key keyboard, and numeric display. It includes the 8080 Microprocessor and its memory.

- ITS - Interface Training System
This is a second printed circuit board which includes input/output ports, analog to digital converter, timers, and a vectored priority interrupt control system.

- Integrated Experiment Assembly
This is the smaller printed circuit board on which are mounted a loudspeaker, thermistor, motor, and several switches and pots (variable resistors).

- Power Supply
This provides regulated DC power for all of the subsystems. It also serves as a mounting base for the MTS.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Additional components include various interconnecting cables and three Read Only Memory chips that must be installed in the MTS. The next sections of this introduction describe how to connect these components together to form a complete microprocessor training laboratory.

Other components available from ICS include a carrying case for compact and portable storage of the training laboratory, and an S-100 expansion option. These will not be discussed further here.

In addition to the materials supplied you will need three alkaline AA cells (to power the motor used in some experiments) and a voltmeter.

1.2 Receiving Inspection

Each of the major components of the training laboratory is shipped with a packing list, which should be checked for completeness. If any item is missing or damaged, telephone Integrated Computer Systems for advice.

1.3 Assembly

The components will be assembled in three stages, with a check after each stage to be sure that it has been completed properly. If any check is not completed successfully, please telephone ICS for advice before proceeding further.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

1.3.1 MTS and Power Supply

Place the power supply on a desk or table with the sloping face toward the user. Note the mounting brackets at the two lower corners of the sloping face; these will support the computer. On the MTS there are two black knobs with plastic posts extending through the circuit board; these posts fit into the holes of the mounting brackets.

Place the lower edge of the MTS on the table and rest its upper edge at the top of the sloping surface of the power supply. It is easier to make the power connection before firmly mounting the computer. The power should be off while making this connection.

Examine the multiconductor cable from the power supply. A group of five wires should be at the left, and two red wires should be at the right. Twist the cable if necessary, and plug its connector into the socket at the top left corner of the MTS.

Now reach under the plastic cover and lift the two black knobs to withdraw the mounting posts. Locate the MTS to mate the posts with the mounting bracket holes, and press down to insert the posts. The plastic cover can be flexed to push the knobs down from above.

Finally, plug the ac power cord into the socket at the back of the power supply and into a power outlet. Turn power on with the switch at the back of the power supply, just above the power cord. Press the red RST key of the MTS keyboard, and your display should

show:

8	2	0	0
---	---	---	---

		?	?
--	--	---	---

where the question marks represent any number 0 - 9 or A, B, C, D, E, or F. This is the way the hexadecimal numbers are presented on your display. If your display stays unlit, check the fuse just below the power switch and that the power cord is plugged in properly. Then switch on again and press the RST key.

1.3.2 Keyboard Test

Press the red RST key on the MTS keyboard. The numeric display above the keyboard should show 8200 in the four left digits, followed by two blank digits, and the two right hand digits may show any data.

Test the keyboard and display by pressing keys in the following sequence.

RST	8200	??
-----	------	----

(The question marks are not displayed; any data may appear here.)

MEM	8200	.??
-----	------	-----

A decimal point appears, indicating that data can be entered into memory at address 8200.

Memory is a vital part of the computer. It stores both programs and data. Each memory location has a unique "address", represented by the four digit number at the left. The content of a memory location is the information stored there, represented by the two characters displayed at the right.

Now press numeric keys in the sequence show below, observing the

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

display.

0	8200	.00
1	8200	.01
2	8200	.12
3	8200	.23
4	8200	.34

Continue up to 9, then A, B, C, D, E, F. Note that B is displayed as to avoid confusion with 6 or 8, and that D appears as d. When you have completed the sequence the display should show:

8200 .EF

The computer deals with binary numbers which are represented here as two digit values. Each digit can have a value not only from 0 through 9, but also ten through fifteen represented as A through F. This is discussed in Section 3 of this manual.

The foregoing test shows that the hexadecimal keys (0-9, A-F) function correctly. The following steps test the functions of the Command keys, without attempting to explain these functions. Press the keys as indicated below and check that the display agrees with that shown.

At the left edge of the MTS there is a toggle switch with its handle projecting slightly beyond the edge of the circuit board. This is the AUTO/SS switch. Turn it to the SS position (down).

Enter the following trivial program by pressing keys as indicated:

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

RST	8200	??
MEM	8200	.??
0	8200	.00
NEXT	8201	.??
A	8201	.0A
F	8201	.AF
NEXT	8202	.??
3	8202	.03
7	8202	.37
NEXT	8203	.??
3	8203	.03
C	8203	.3C
NEXT	8204	.??
0	8204	.00

The program has been entered. MEM and NEXT commands have been operated. Now we shall test the other commands, by executing this program one step at a time.

ADDR	8200	00
------	------	----

This shows the present program location and instruction.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

STEP 8201 AF

We have executed the first instruction, and are ready to execute the next one. To the left of the numeric display there are two red indicators (LED's). Until now they have been meaningless. The next instructions affect them.

STEP 8202 37

The lower indicator is on. The upper indicator is off.

STEP 8203 3C

Both LED indicators are on.

STEP 8204 00

The lower indicator is off and the upper indicator is on.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

To test the remaining command keys perform the following steps.

The display still shows:

	8204	00
ADDR	8204	00
BRK	8204	BP.
ADDR	8204	00
8	0008	
2	0082	
0	0820	
0	8200	00
RUN	8204	00
REG	8204	
A	8204	A-01
BRK	8204	BP.00
CLR		BP.

All keys have now been tested. We shall explain their functions as programs are developed in Course 525A.

This page intentionally left blank.

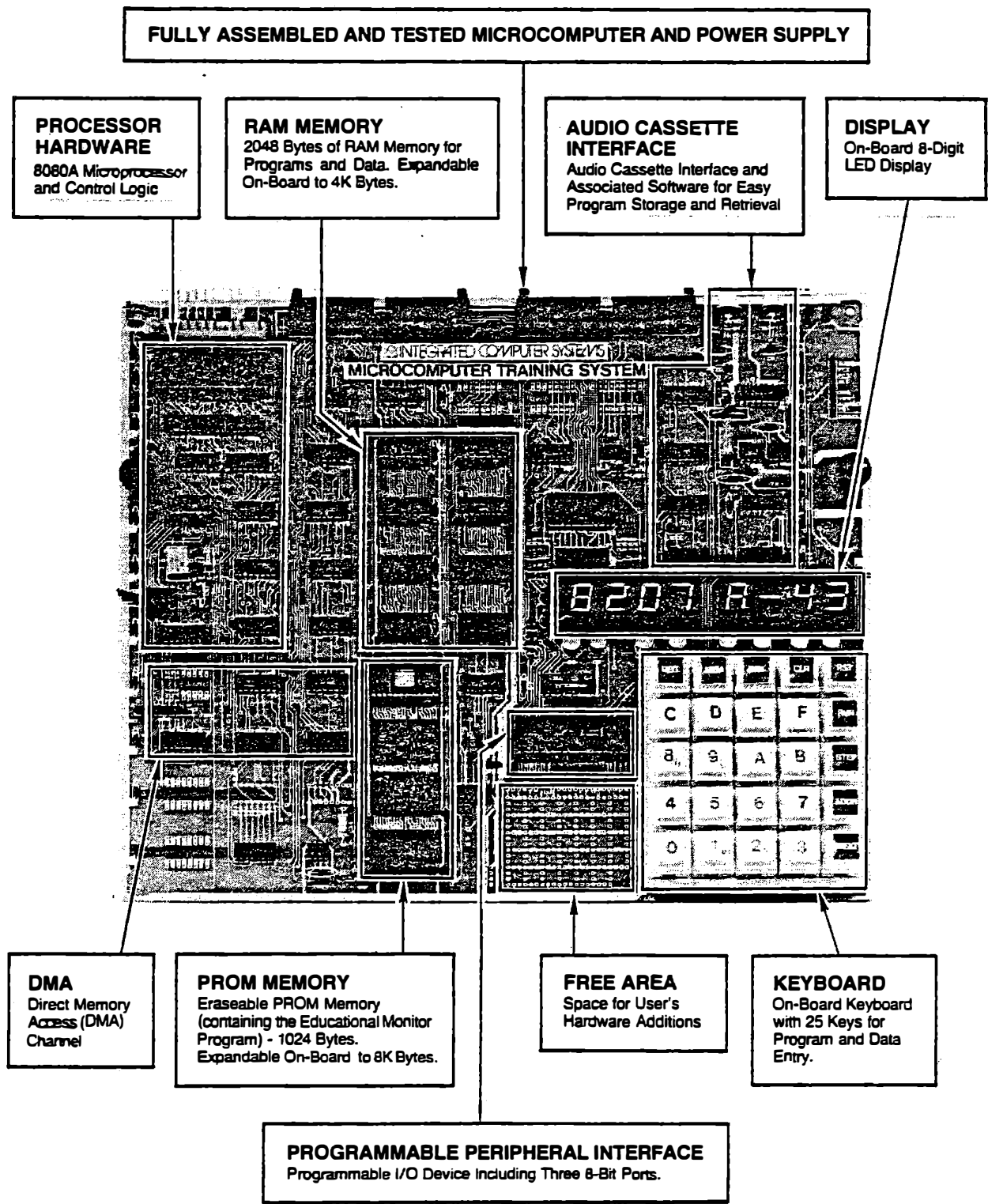
1.3.3 Installing PROM's

In the keyboard test we entered a program into memory and executed that program. Another kind of memory is used in the microcomputer for permanent programs, called Read Only Memory because its contents cannot be changed by the computer. All of the functions we have performed - entering numbers through the keyboard, displaying memory content, and executing the program you entered - were done under control of a permanent program called the "monitor." This program is stored in a single integrated circuit called a PROM (programmable read only memory.)

Figure 1-1 shows the location of the PROM memory area on the MTS. The monitor PROM is installed, and three empty sockets are provided for insertion of the three Educational Firmware PROM's. Figure 1-2 shows the positions for these. They must be inserted in the correct sockets. Note that the small notch on one end of each PROM must be to the right.

After each PROM is installed, it is advisable to test it, since if any one is inserted badly it may affect the operation of others. Following the procedure below, install the first PROM, labelled THCL, immediately below the monitor PROM.

INTEGRATED COMPUTER SYSTEMS, INC.

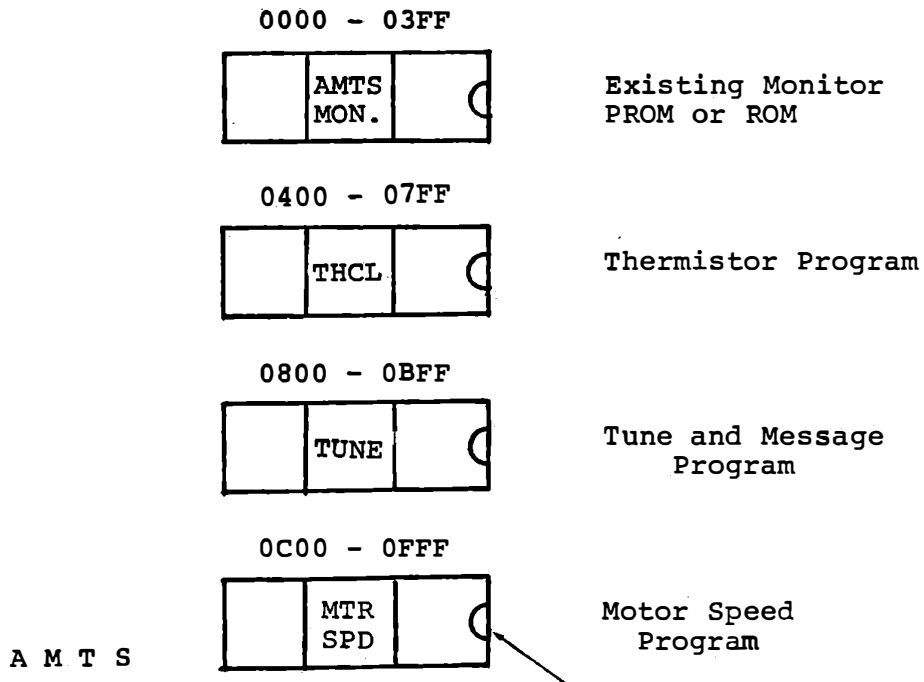


MTS Board Layout

Figure 1-1

INSTALLING PROMS

Three Programmable Read Only Memory chips are supplied with the Integrated Assembly Board. These have been loaded with demonstration programs. They are to be plugged into three empty sockets on the Microcomputer Training System, immediately below a similar chip already installed.



NOTE DIRECTION OF NOTCH!

PROM Locations

Figure 1-2

CAUTIONS

- TURN OFF THE POWER

- PROM'S can be erased or damaged by electrostatic shock. Before handling them, discharge any static electricity from your body by touching a grounded metal object (such as the MTS power supply) with both hands. While handling the PROM, try to avoid touching the pins.

- The pins are easily bent, so be careful when inserting the PROM's. A good procedure is to align one row of pins, with the others up a little, so that you can see both rows. (Figure 3a). Now tilt the PROM down, applying a little force to align the second row. (Figure 3b). Press firmly down to insert all pins at once. If you should bend a pin, carefully pry the PROM up, remove it and straighten the bent pin with thin nosed pliers.

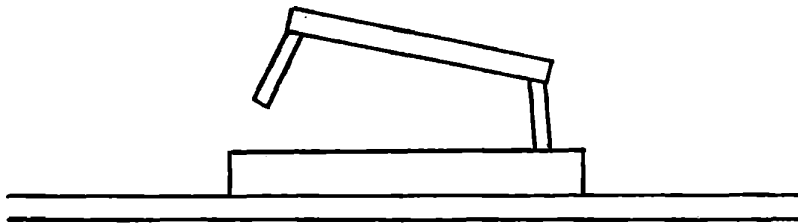


Figure 1-3a

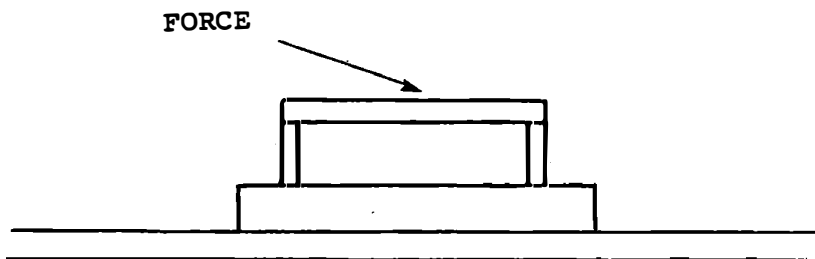


Figure 1-3b

Installing PROM's

Figure 1-3

1.3.4 Testing PROM's

Most of the programs in these PROM's require the Interface Training System and the Integrated Experiment Assembly, which have not yet been connected. We can test to make sure that the memory chips have been installed correctly, by the following procedure.

After intalled the first PROM (THCL):

- Turn power on
- Set AUTO/SS switch to AUTO
- Press these keys:

RST					8200	??
ADDR	0	6	0	0	0600	21
RUN						
RUN					25.0	
STEP					2.0	
BRK					0693	C9

If these displays appear correctly, the PROM has been inserted correctly.

Turn power off and insert the second PROM, "TUNE".

Turn power on and press these keys.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

RST

ADDR 0 4 0 0 0400 C3

RUN

Some data may appear briefly. Then after some delay a moving message will appear very slowly, starting with a series of decimal points. Speed it up by pressing

2 0002

ADDR

Now the message will appear at a readable speed.

If this is successful the second PROM has been installed correctly. Turn power off and install the third PROM, "MTRSPD."

Turn power on and press:

RST 8200 ??

ADDR 0 4 0 0 8200 C3

RUN

This repeats the previous test. Now press:

BRK bin

BRK Arit

BRK

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

RUN	25.0	
BRK	0000	SPEd

If all of these appear then all of the PROM'S have been correctly inserted.

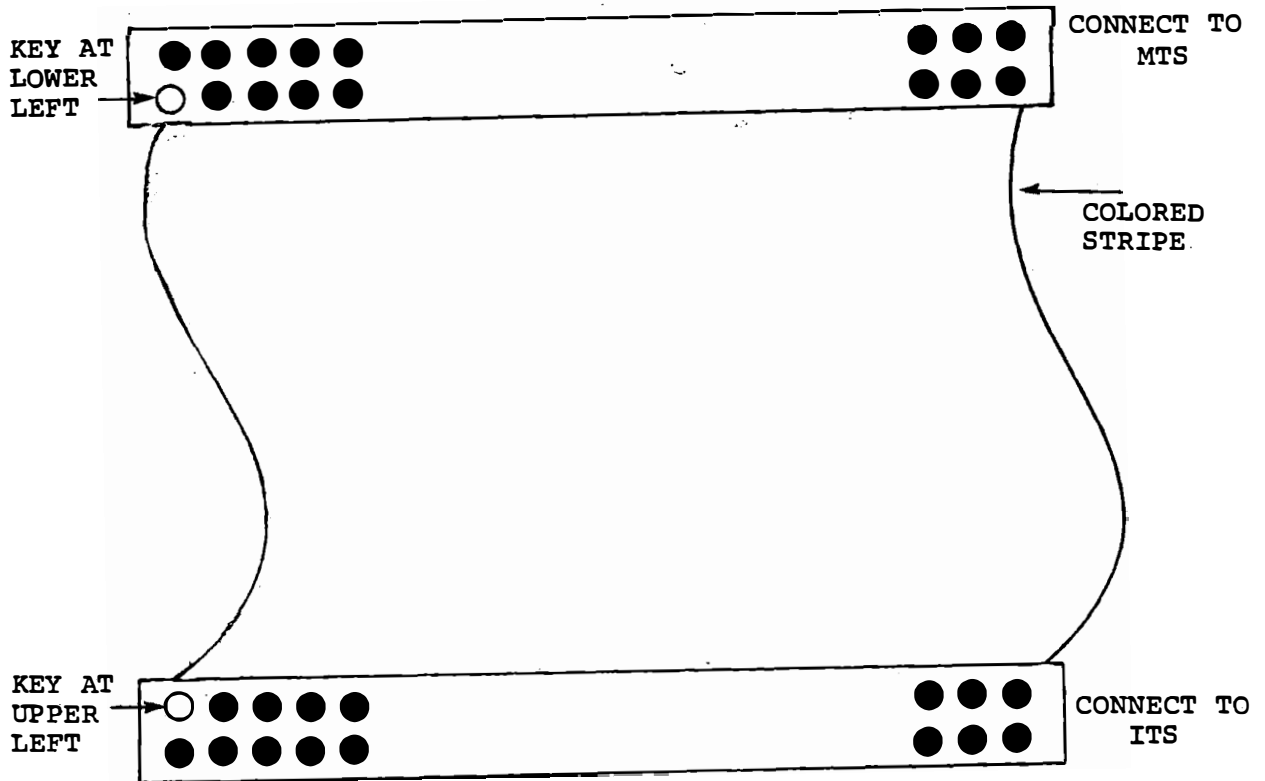
The microcomputer is now ready to operate. For convenience you may prefer to use it in this form without having the Interface Training System attached, since that is not used in Course 525A. To use some of the demonstration programs, described in Section 2 of this manual, you must connect the Interface Training system and the Integrated Experiment Assembly.

1.3.5 Connecting Interface Training System

The ITS provides for input and output with devices not directly connected to the computer. It is needed for Course 536A, and also for several of the demonstration programs.

A ribbon cable is supplied for connecting the MTS to the ITS. The two connectors are identical except for a keyed pin. Looking at the front of the connectors you should see the pattern shown in Figure 1-4. The connector with the key at the lower left corner plugs into the MTS from behind, with the cable going downward.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL



MTS/ITS Ribbon Cable

Figure 1-4

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

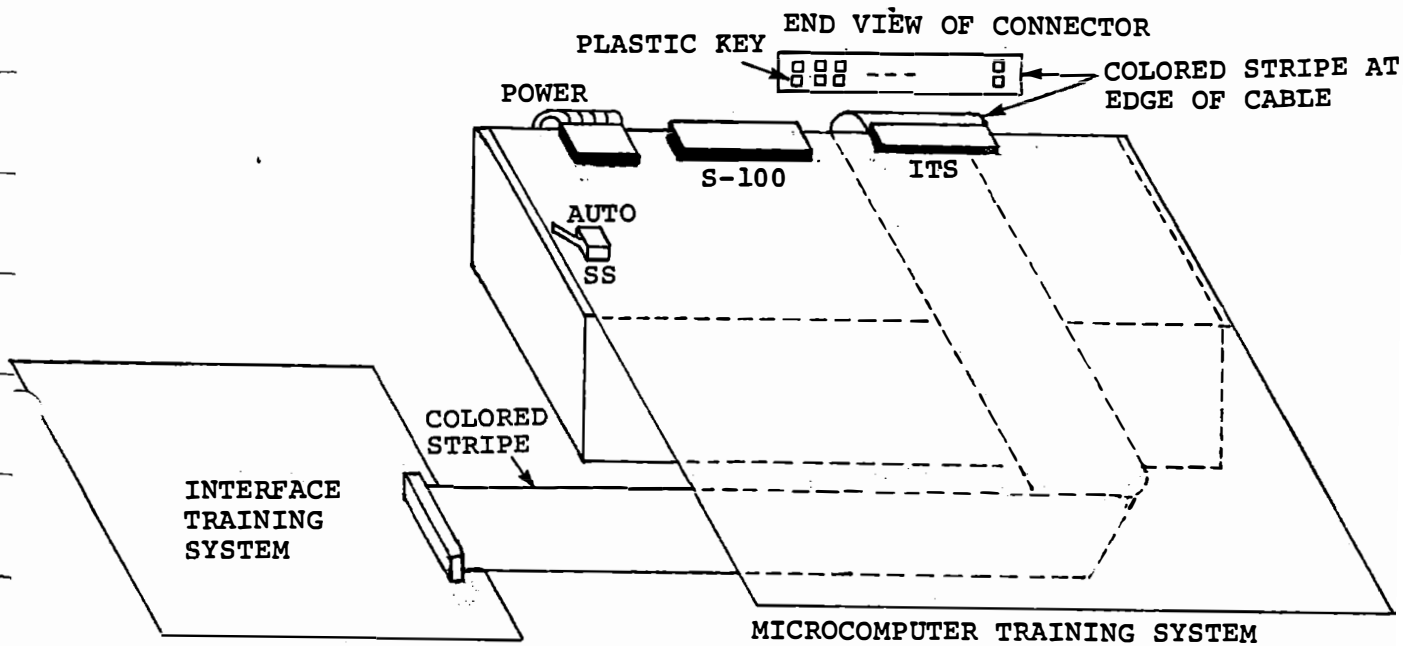
CAUTION - Before connecting ITS to MTS be sure that power is turned off.

Figure 1-5 shows the MTS and ITS connected with the ribbon cable. The ITS may be on the left (as shown) or behind the MTS, according to personal preference. To place the ITS at the left, release the MTS from the power supply by lifting the black knobs that hold it down. (It is not necessary to unplug the power cable.) Place the ribbon cable between the power supply and the MTS circuit board; plug it into the right hand connector on the MTS, and again attach the MTS to the power supply. Make a 90 degree fold in the cable and bring it out to the left, under the MTS.

If you prefer to place the ITS behind the MTS, simply plug the cable into the connector on the MTS from behind. Fold or twist the cable so that the colored stripe is away from the front edge of the ITS.

Be sure that power is turned off. Plug the second connector into the ITS with the key toward the front edge of the circuit board and the colored stripe toward the back edge.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL



ITS and MTS Connected

Figure 1-5

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

1.3.6 Testing the Interface Training System

The moving message demonstration makes a reasonable test of the ITS. Turn power on and press these keys.

```
RST                                8200    ??  
  
ADDR 0    4    0    0            8200    C3  
  
RUN
```

Now the moving message appears at a readable speed immediately.

A more complete test will be made with the Integrated Experiment Assembly attached.

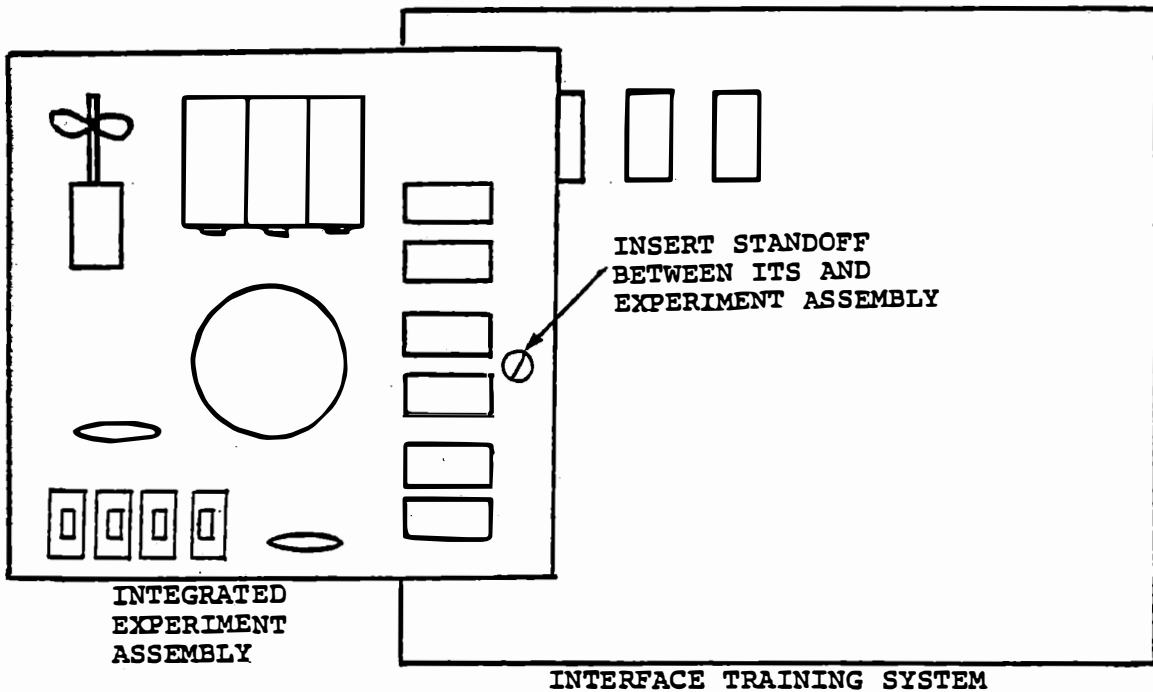
1.3.7 Integrated Experiment Assembly

External devices needed for the demonstration programs and the more advanced experiments of Course 536A are mounted on the Integrated Experiment Assembly circuit board. The devices include a motor with a propeller, and an optical disc for indicating speed; a "slot sensor" which detects the clear and black segments on the optical disc; a loudspeaker; a thermistor for temperature measurement (at the end of a coiled cable); four slide switches and two variable resistors. Six white tie blocks on the experiment board are positioned to correspond exactly to six similar blocks along the left edge of the interface training system circuit board. The pins

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

extending from the bottom of the experiment board will fit into the tie blocks on the interface training system. Make sure these pins are straight; if necessary they should be straightened with thin nosed pliers.

Figure 1-6 shows the configurations of the ITS and the experiment assembly when they are attached.



ITS and Experiment Assembly

Figure 1-6

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

The ITS is supported by five legs with rubber feet. One of these (as indicated in Figure 1-6) is used to mount a mechanical support for the experiment assembly. Replace the screw that holds the leg to the ITS with a standoff screw, supplied with the experiment assembly. The experiment assembly plastic bag also contains a plastic cable relief and two legs with rubber feet, which are used to support the protruding part of the experiment assembly circuit board. Place the cable relief around the coiled cable, near the lower edge of the board. Mount the cable relief on the top surface of the board and one leg on the bottom, using one screw through the hole near the lower left corner. Mount the other leg at the upper left.

Carefully locate the experiment assembly as shown, with its downward extending pins aligned with holes in the ITS tie blocks and a hole in the experiment assembly aligned with the standoff screw. (When it is correctly located, exactly half of the upper left tie block on the ITS will be hidden.) Press the boards together to insert the pins. Insert the screw that was removed from the ITS through the experiment board into the standoff.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

At the top right corner of the experiment board a wire is soldered into a hole. This wire is to be plugged into the top left hole of the middle tie block on the experiment board.

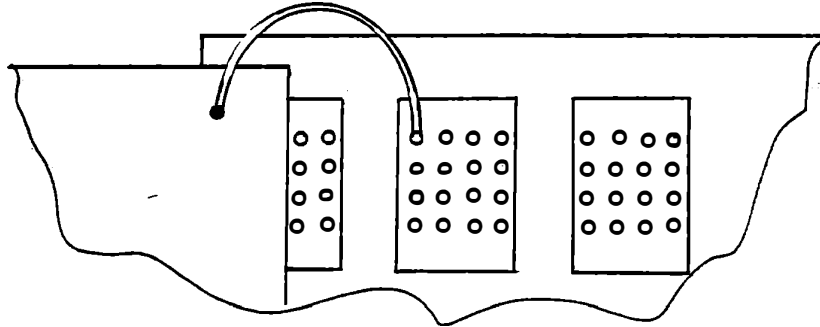


Figure 1-7a

A battery clip is provided to hold three AA cells to power the motor. (alkaline cells are recommended.) All three cells must be mounted with the positive ends down (toward the loudspeaker).

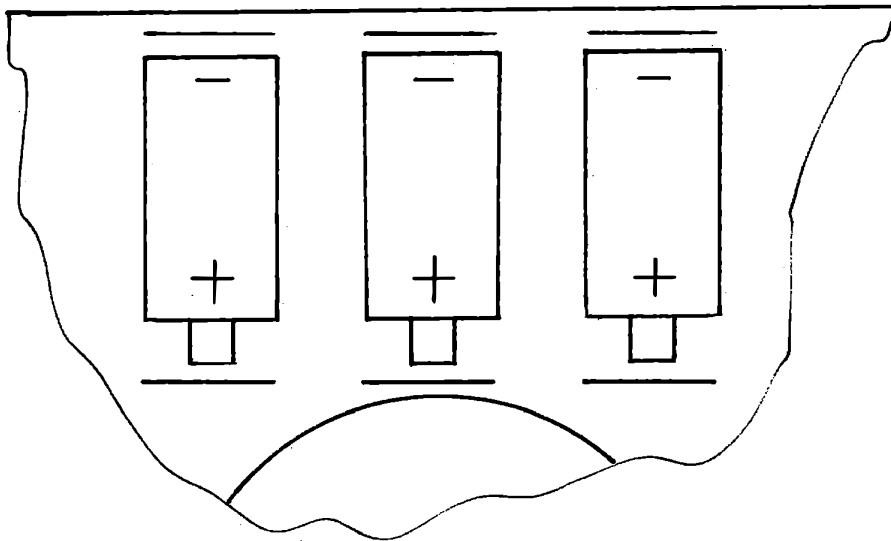
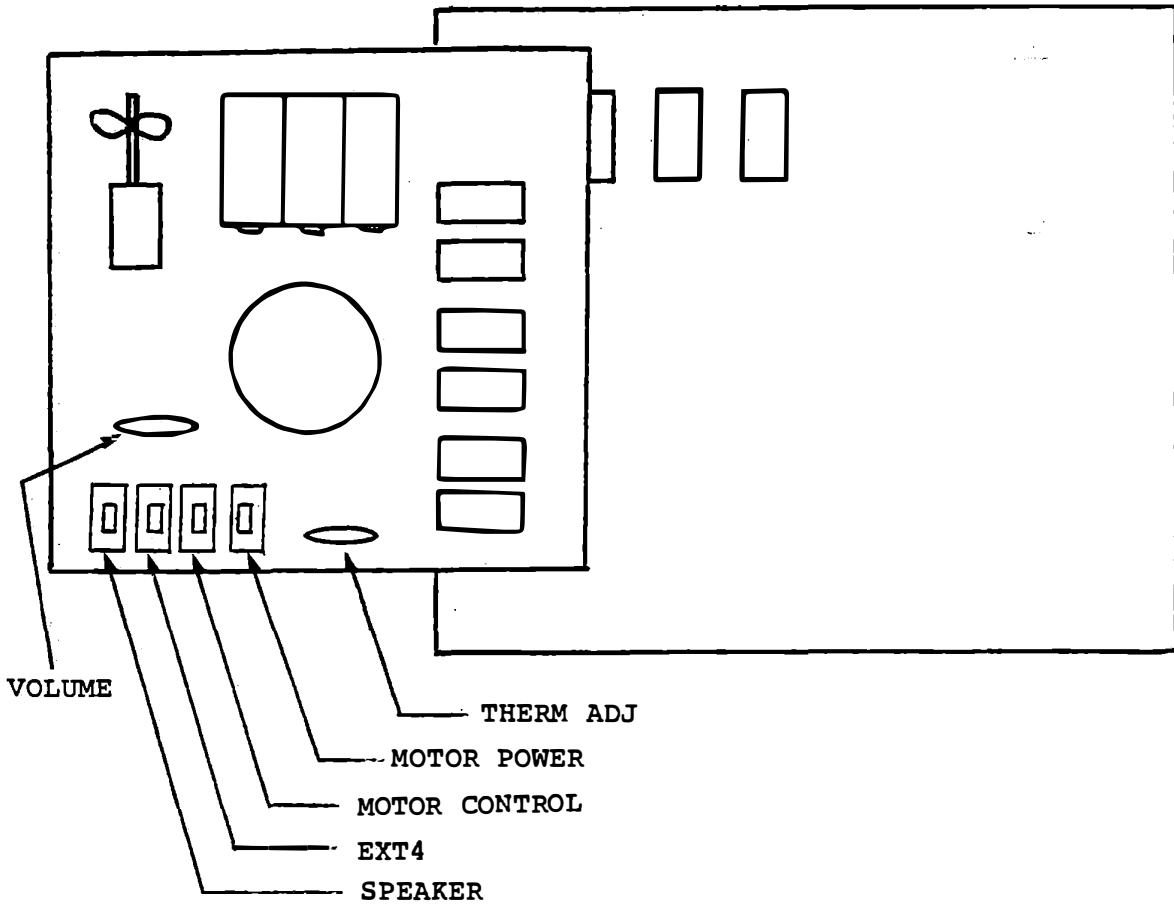


Figure 1-7b

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

This completes the assembly of the Microprocessor Training Laboratory. The next section briefly introduces the demonstration programs, which will fully test the system. Note the six controls on the Integrated Experiment Assembly shown in Figure 1-8. For the demonstration programs all four switches should be down. The two potentiometers will be adjusted later.



Experiment Assembly Controls

Figure 1-8

This page intentionally left blank.



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 2
EDUCATIONAL FIRMWARE

THIS PAGE INTENTIONALLY LEFT BLANK.

2. EDUCATIONAL FIRMWARE

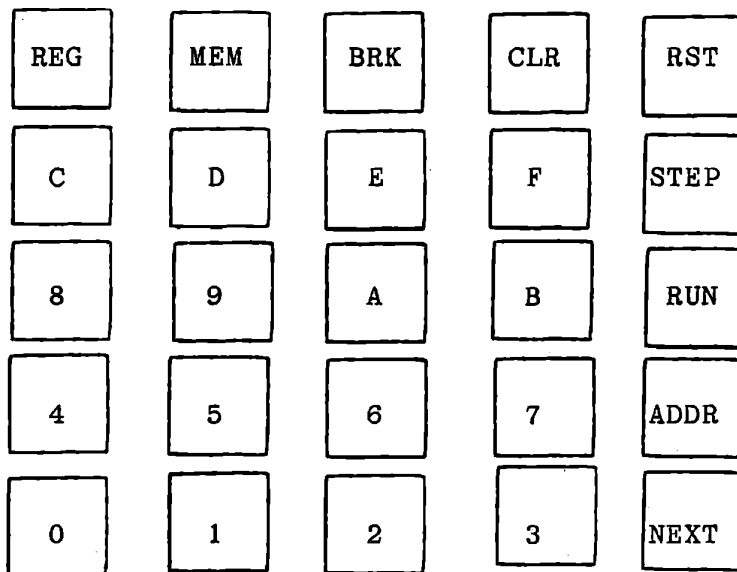
Five programs are supplied in read only memory, and eight experiments are described on the following pages.

1. Moving Message (Program 0)
2. Playing a Tune from Memory (Program 0)
3. Keying a Tune (Program 0)
4. Binary Arithmetic (Program 1)
5. Hexadecimal Arithmetic (Program 2)
6. Thermometer (Program 3)
7. Cooling Thermostat (Program 3)
8. Motor Speed Control (Program 4)

With the training laboratory set up and operating, we will very briefly try each of the demonstrations to indicate the scope of the training courses. Then two of the demonstration programs will be used to introduce binary and hexadecimal number systems. From there, you will proceed into Course 525A.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Look at the keyboard and the display. You will use these immediately. There are 16 white keys and 9 colored keys.



The red RST key always stops whatever the computer is doing and starts the computer at a known place. It cannot be used in any other way.

The white keys (0 - 9 and A - F) have the purpose of entering numbers into programs. The meaning and use of a number depends on the program and the commands given to the program either before or after the number. You will write programs that use these keys, and you will use built-in programs that accept numbers from these keys.

The command keys (REG, MEM, BRK, CLR, STEP, RUN, ADDR, and NEXT) are used to enter commands to the built-in monitor and demonstration programs. You will also learn to redefine these

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

keys in your own programs. The labels on these keys refer to their functions in the monitor program, where they are principally used.

2.1 Moving Message Demonstration (Program 0)

To become familiar with the keys, and to start the first demonstration press these keys and observe the display as you do so.

RST	8200	??
-----	------	----

The question marks do not appear, but rather some characters (0 - 9, A - F) that cannot be predicted at present. (Don't worry if the characters look peculiar.)

8200 should always appear at the left when you press RST. This is the "address" of a program, but now we want to use a program that is located at a different address, namely 0400.

ADDR	8200	??
------	------	----

The display does not change.

0	0000	
4	0004	
0	0040	
0	0400	C3

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

You have entered the desired address: 0400. When the fourth digit is entered, the right-hand digits display C3, which is an instruction. You will learn the meaning of this instruction early in Course 525A.

RUN

This command starts operation of the program at address 0400. Up to here you have been using the "monitor" program. Now the first demonstration program is running. The meaning of the command keys is completely changed, because a different computer program is receiving them.

The display shows a moving message, with a mixture of upper and lower case alphabetic characters. The display devices have only seven segments (plus a decimal point), being intended really for decimal numbers. Some letters (A, C, E, F, H, etc.) are easily created, but others are rather forced. Try to read all of the message -- it will repeat indefinitely.

2.2 Playing a Tune from Memory

In Section 2.1 we started the first demonstration program by pressing:

```
RST   ADDR   0 4 0 0   RUN
```

With this program running we can also play a tune. Be sure that all four switches on the experiment board are in the DOWN position. Locate the VOLUME control on the experiment board. You can adjust this by rotating the blue disc with your fingers or with a small screwdriver inserted into a slot inside the disc.

While the moving message is displayed, press RUN. A tune should be heard on the loudspeaker. Adjust the volume control if necessary. Press RUN again to start the tune again if it ends before the adjustment is satisfactory.

A different tune can be played by pressing NEXT, RUN. The selected tune can be repeated by pressing RUN and the other tune can be selected by pressing NEXT.

Moving Message and Tune are two parts of the same program. This program can always be started by:

```
RST   ADDR   0 4 0 0   RUN
```

It can be reached from any other demonstration program by pressing 0, BRK.

2.3 Playing a Tune from the Keyboard

This experiment is also part of the Tune program. You can enter a new tune through the keyboard.

2.3.1 Procedure: MEM displays a memory location where the tune will be stored . Keys 0 - F select notes, which are played and recorded while the key is held down. The notes are the natural notes of the key of C, with "C" giving middle C.

After keying in the tune, press RUN to play the tune that you have recorded.

Hint: Make the notes fairly long and move quickly between notes, because the program also records the rests between notes.

2.3.2 Several tunes can be stored and played successively:

Procedure: MEM display a tune address
Key in tune.
RUN plays the tune.
MEM displays another tune address.
Key in tune.
RUN plays the tune.

2.3.3 Now the different tunes can be played by NEXT, RUN.

Alternately, any of the tunes can be selected by entering its number:

0 NEXT RUN plays Home on the Range.

1 NEXT RUN plays the Drunken Sailor.

2 NEXT RUN plays the first tune keyed in.

3 NEXT RUN plays the second tune keyed in, etc.

The tunes are being selected from a "directory". A tune can be removed from the directory after it has been played by pressing CLR.

2.3.4 What is taught by this kind of experiment:

- a) Accepting and storing input data.
- b) Using a hardware timer to control external equipment (the speaker).
- c) Using a hardware timer for interrupts, and using interrupts to control an output (the tone and duration of a note).
- d) A file management system with a directory (to select the different tunes). This is demonstrated very effectively by the procedure of Section 2.3.3.

2.4 Binary Arithmetic (Program 1)

With the moving message running, press BRK to enter the binary arithmetic program. This program is used in Section 3 to teach the fundamentals of binary numbers. It can be reached from any other demonstration program by pressing 1, BRK. The display shows "bin".

Notice here the difficulty of displaying the letter B. In the title of this program we have used a lower case "b", but this is too easily confused with the numeral 6. Our solution to the problem is shown in the next demonstration.

The binary arithmetic program is used in Section 3 to teach the fundamentals of binary and hexadecimal numbers. You can enter a binary number by pressing keys 0 and 1 only:

1 0 0 1 0000 1001

If you press one of the other numeric keys its binary equivalent will be displayed.

9 0000 1001

We will skip the addition and subtraction functions of the binary arithmetic program until we reach Section 3.

2.5 Hexadecimal Arithmetic (Program 2)

From the binary arithmetic program enter the hexadecimal program "Arit" by pressing BRK. We will use this only to demonstrate the displays given for the sixteen hexadecimal keys. (The name hexadecimal means six and ten , or sixteen.)

Press keys as follows and observe the display:

	Arit
0	0000
1	0001
2	0012
3	0123 1F

As the keys are pressed, each key's value appears in the right-hand digit of the left-hand section of the display. Initially, leading zeros appear, but previous digits are shifted to the left, as when you entered ADDR 0400 RUN to start the demonstrations. After four keys have been pressed, two digits appear at the right; they are meaningless in the arithmetic program.

Continue to press keys:

4	1234	??
---	------	----


EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

The oldest key (0) has been lost and the most recent four keys are displayed.



5	2345	??
6	3456	??
7	4567	??
8	5678	??
9	6789	??

We have displayed all the numeric keys from 0 through 9. There are six more white keys (A, B, C, D, E, and F). In Section 3 we will see that these also represent numbers

A	789A	??
B	89AB	??

Note that B is displayed as 

This symbol is used to represent B to avoid confusion with the numerals 8 and 6. It is common to use:

-  for letter B (as in bin)
-  for numeral 6

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

In teaching many students we have found that these two symbols are constantly confused, so ICS has adopted the unique symbol shown in your display to represent B.

C	9ABC	??
D	ABCd	??

The lower case d is used, as is conventional.

E	BCdE	??
F	CdEF	??

Once again we will skip the arithmetic functions of the program until we reach Section 3.

2.6 Thermometer (Program 3)

Enter this program from "Arit" by pressing BRK. Enter from any of the demonstration programs by pressing 3, BRK. After a brief pause the display will show a temperature measurement in degrees Celcius.

The value displayed may be far from the correct temperature. First, all four swiches on the experiment board should be DOWN. Second, the analog input circuits must be adjusted for a correct reading. In Course 536A (Section 5.3.6) you will learn to make a correct adjustment . For an initial demonstration adjust the "ANALOG IN" potentiometer on the Interface Training System to obtain a reasonably realistic value. Warm the thermistor with your fingers to see the change in temperature displayed. Let it cool again.

This demonstration program is similar to one you will develop in Chapter 5 of Course 536A. There you will learn the techniques of measuring a voltage and converting that measurement to a temperature.

2.7 Cooling Thermostat with Alarm (Program 3)

This experiment shows the microcomputer controlling a fan to cool the thermistor when it is warmer than a setpoint, and sounding an alarm if it is too hot. The speed of the fan depends on the temperature. With the thermometer program running, enable the control function by pressing RUN.

The temperature limit is displayed at the left; the measured temperature at the right. While the measured temperature is less than the limit nothing else happens. For convenience in the demonstration, the limit should be about three degrees greater than room temperature. To change the limit, key in the desired temperature followed by RUN.

Warm the thermistor between your fingers. When its temperature rises above the limit, the fan will be turned on. As it becomes warmer, the fan speed will increase. At two degrees above the limit, the fan will be at full power and the alarm will sound.

Now let the thermistor cool. When its temperature drops below the alarm limit (two degrees above the displayed limit) the alarm will go off. When the temperature drops more than two degrees below the displayed limit, the fan will stop. Now it will not turn on again until the temperature exceeds the displayed limit. See Section 6 for various other controls and displays available with this program.

2.8 Motor Speed Control (Program 4)

This experiment duplicates an exercise in the Interface Training Course. One course exercise develops the concepts of proportional and integral control and uses the microcomputer for measurement and control of a voltage controlled by pulse width modulation, with strong emphasis on observing and understanding the effects. Then the motor speed control exercise uses an optical sensor with a spinning disc to measure speed, and drives the motor to achieve a preset speed under varying load conditions.

Procedure: To start either enter BRK (from the thermostat demonstration) or 4 BRK (from any demonstration).

At entry the display shows ???? SPED.

Press RUN to start the motor. The average speed is displayed at the left. It will seek a speed of 50 revolutions per second, by varying the power duty cycle applied to the motor. Duty cycle is displayed as a decimal fraction at the right.

Push gently on the end of the motor shaft. The duty cycle will increase to restore the 50 rps speed. Release the pressure and observe the motor speed hunting for the desired value.

Request a different speed by keying in a number followed by RUN. Speeds from about 10 to 60 rps can be achieved.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Remove the closed loop control by setting the gains in the control equation to zero: 0 MEM. Show that the motor slows down when load is applied. Restore closed loop control by keying 802 MEM. This sets proportional gain to 8 and integral gain to 2. Control will be quicker than originally because both gains were initially set to unity.

2.9 Program Storage

Students who are familiar with computers may be interested in the storage requirements for these demonstration programs. These suggest the complexity of programming that can be done in a small microcomputer.

<u>Program</u>	<u>Locations</u>	<u>Length</u>
Monitor	0000H thru 03FFH	1024 bytes
Demonstration and Executive	0400H thru 0491H and 0753H thru 079FH	223 bytes
Thermometer and Thermostat	0492H thru 0752H	705 bytes
Tune and Message	07A0H thru 0BFFH	1120 bytes
Motor Speed	0C00H thru 0DFFH	512 bytes
Hexadecimal Arithmetic	0E00H thru 0F8FH	400 bytes
Binary Arithmetic	0F90H thru 0FFFH	<u>112 bytes</u>
TOTAL -		4096 bytes



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 3
BINARY NUMBERS



INTEGRATED
COMPUTER
SYSTEMS™

THIS PAGE INTENTIONALLY LEFT BLANK.

3. BINARY NUMBERS

All digital computers use binary numbers -- numbers that can be represented on paper as a succession of ones and zeros:

0 1 0 1 0 0 1 0

Inside the computer these are represented by the states of electrical circuits. Each such circuit has only two states -- which can be called True or False; Set or Reset; On or Off; One or Zero.

3.1 Demonstration Program "bin"

The Educational Firmware includes a built-in computer program that will help you to understand these numbers. To operate this program press the following keys:

RST ADDR 0 4 0 0 RUN BRK

If another demonstration program is running, press 1 BRK. The display will show the name of this program, "bin".

Enter a binary number by pressing only the keys 0 and 1, to see how it is displayed.

0 1 0 1 0 0 1 0 displays 0101 0010

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Press CLR to get rid of this number. Press 0 to display:

0000 0000

To see how other numbers are represented in binary, press the keys shown below, and observe the display.

1	0000	0001
2	0000	0010
3	0000	0011
4	0000	0100
5	0000	0101

To understand this representation, note that the right-hand digit has a value of either 0 or 1. The next digit, displayed as 0 or 1, represents a value of 0 or 2. The third digit from the right represents a value of 0 or 4. Thus the binary number 0000 0011 is equivalent to $2 + 1 = 3$, while the binary number 0000 0101 = $4 + 0 + 1 = 5$.

Think of the way we represent decimal numbers greater than 9:

37	=	30	+	7				
207	=	200	+	0	+	7		
3207	=	3000	+	200	+	0	+	7

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Each digit to the left of the rightmost digit has a value (30, 200, 3000) equal to the numeral (0, 2, 3) multiplied by a power of 10; i.e., 10, 100, 1000. The same rule applies in binary numbers except that the value of a digit is multiplied by a power of 2.

Press CLR	1	0000	0001
Press	2	0000	0010
Press	4	0000	0100
Press	8	0000	1000

Placing a zero to the right of a decimal number multiplies the number by 10.

$$30 = 3 \times 10$$

Placing a zero to the right of a binary number multiplies the number by 2.

Press CLR	1	0000	0001	(1)
Press	0	0000	0010	(2)
Press	0	0000	0100	(4)
Press	0	0000	1000	(8)

In the demonstration program the computer discards the old value if you press any key greater than 1. For 0 or 1 it shifts the previously entered value left and inserts the 0 or 1. Press CLR to discard the previous value.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Enter the binary number equivalent to nine by pressing only the 1 and 0 keys.

CLR	The display shows	bin	
0	The display shows	0000	0000
1 0 0 1		0000	1001

Note that the decimal value 9 requires four binary digits ("bits"), but that several greater values can also be represented by four bits. Decimal ten is represented by 0000 1010 -- that is, 8 + 2. What decimal value is equivalent to 0000 1111?

1	=	1
10	=	2
100	=	4
1000	=	8
<u>1111</u>	=	15 (decimal)

3.2 Hexadecimal Numbers

For convenience in writing and displaying binary values we have another number system called hexadecimal (six-ten system) in which each digit has sixteen possible values, 0 - 15. These are all the possible values that can be represented by four bits. The demonstration program allows you to enter any of these sixteen values by pressing any of the sixteen white keys, labelled 0 - 9 and A - F.

0	0000	0000
1	0000	0001
2	0000	0010
3	0000	0011
4	0000	0100
5	0000	0101
6	0000	0110
7	0000	0111
8	0000	1000
9	0000	1001
A	0000	1010
B	0000	1011
C	0000	1100
D	0000	1101
E	0000	1110
F	0000	1111

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

You can obtain this same sequence by counting. Press CLR, 0. Then count up by pressing REG repeatedly.

CLR	0	0000	0000
REG		0000	0001
REG		0000	0010

After the sequence from 0 through F, the next count will put a 1 into the next higher bit.

		0000	1111
REG		0001	0000

It is here that we meet some confusion between hexadecimal and decimal values. The hexadecimal equivalent of 0001 0000 is 10 -- a one in the higher order digit and zero in the lower order digit. This number has a decimal value of 16. In this course we deal almost entirely with hexadecimal numbers, and the number 10 will generally mean 0001 0000. If a decimal value is intended it will be so labelled.

Again discard the old value, and enter 1.

CLR	1	0000	0001
-----	---	------	------

Now multiply this number by 2, by entering a zero at the right.

	0	0000	0010
--	---	------	------

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

This is the binary equivalent of 2. Repeatedly enter zeros, to obtain greater values, to match this table:

<u>BINARY</u>	<u>HEXADECIMAL</u>	<u>DECIMAL</u>
0000 0001	01	1
0000 0010	02	2
0000 0100	04	4
0000 1000	08	8
0001 0000	10	16 (decimal)
0010 0000	20	32 (decimal)
0100 0000	40	64 (decimal)
1000 0000	80	128 (decimal)

Beyond this, the highest bit is lost, so entering another 0 will make the display become 0000 0000. This is the effect of "modulo" arithmetic. There is no limit to the value of a number unless the space available for storing it is limited. In a computer, however, the space is always limited, just as the space on our eight digit display is limited. The largest number we can display in binary with this demonstration is 1111 1111. By adding the numbers above you will see that this is equivalent to a decimal value of 255. Expressed in hexadecimal this is represented as FF.

3.3 Review and Self-Test

We have met the binary and hexadecimal number systems. For practice, translate each of the binary values below into its hexadecimal equivalent. Check by pressing the hexadecimal key corresponding to your answer.

1001	_____
0100	_____
1010	_____
0011	_____
1100	_____
1101	_____
1000	_____
1111	_____

3.4 Binary Addition

Numbers are added in binary according to simple rules:

$0 + 0 = 0$
 $0 + 1 = 1$
 $1 + 0 = 1$
 $1 + 1 = 10$ (=2 decimal)
 $10 + 0 = 10$
 $10 + 1 = 11$
 $10 + 10 = 100$

The demonstration program can perform binary addition. Enter a binary number and press MEM to store it. Enter another number and press ADDR to add it to the previous value.

0	MEM	0	ADDR	0000	0000
1	ADDR			0000	0001
1	ADDR			0000	0010
1	0			0000	0010
ADDR				0000	0100

In the last step we added 10 to the previously stored value, also 10, to obtain the sum 0100. What is the decimal equivalent of this value?

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

ADDR adds a new value to the previous sum. If no new number is entered, the last number entered is used again.

1	0	ADDR	0000	0110
ADDR			0000	1000
ADDR			0000	1010
1	0	1	ADDR	0000
1		ADDR	0001	0000

A number can also be entered by a hexadecimal key, as before, and added to the previous sum.

6			0000	0110
ADDR			0001	0110

Perform the following additions by hand and check your answers with the computer:

		0100	1100
+		<u>0001</u>	<u>0001</u>
		0101	1101
+		<u>1000</u>	<u>0010</u>
		1101	1111
+		<u>0000</u>	<u>0001</u>
		1110	0000
+		<u>1000</u>	<u>0000</u>

If you have any difficulty, make up more examples and try them.

3.5 Binary Subtraction

The demonstration program also performs binary subtraction.

7	MEM	0000	0111
3		0000	0011
STEP		0000	0100

When you study subtraction and "twos complement" arithmetic in Course 525A, Chapter 10, this demonstration program will be very helpful.

3.6 Hexadecimal Arithmetic

Remember that the hexadecimal number system is merely a shorthand representation of a binary number. The Educational Firmware also includes a built-in program to accept and display hexadecimal numbers and perform arithmetic. To go from the binary program to the hexadecimal program, just press BRK. To start from scratch, press:

```
RST   ADDR   0     4     0     0     RUN
  2    BRK
```

If any of the five demonstration programs is running, 2 BRK will enter the hexadecimal program, called "Arit".

The eight-bit numbers we used in the exercise of Section 3.4 can be translated into hexadecimal. For instance:

BINARY		HEXADECIMAL
0100	1100	4C
<u>+ 0001</u>	<u>0001</u>	<u>+ 11</u>
0101	1101	5D

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

To test this with the hexadecimal "Arit" program, enter keys as follows:

CLR		Arit
4	C	004C
MEM		4C.
1	1	0011
ADDR		5D.

Translate the rest of the numbers and binary sums and test them.

0101	1101	5D
+ 1000	0010	_____
1101	1111	
1101	1111	
+ 0000	0001	_____
1110	0000	
1110	0000	
+ 1000	0000	_____
0110	0000	

With the hexadecimal program you can enter four hexadecimal digits, and obtain eight digit results. When you enter a four digit number, (which appears at the left as you key it in) another meaningless number appears at the right. Ignore this; it has nothing to do with the arithmetic.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

The arithmetic program also does hexadecimal subtraction, multiplication, and division. These will be helpful when you study arithmetic in Chapter 10 of Course 525A. The command keys as used in "Arit" have the following functions:

BRK	Exit from Arit to the thermometer program.
CLR	Clear the previous entry and result.
MEM	Store the number just entered in place of the old result. The number entered is cleared.
REG	Increment (add 1 to) the old result.
ADDR	Add the number most recently entered to the previous result. The number entered is retained, so successive operations may be performed using a constant entry value.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

STEP Subtract the number most recently entered from the previous result.

The number entered is retained.

NEXT Multiply the previous result by the number most recently entered.

The number entered is retained.

RUN Divide the previous result by the number most recently entered.

Display the result as XXXX.XXXX.

In further multiplications and divisions the entry value is treated as an integer, e.g.:

2.0000

2 NEXT 4.0000

For addition and subtraction the entry value is treated as a fraction.

2 ADDR 4.0002

This page intentionally left blank.



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 4

STARTING COURSE 525A



THIS PAGE INTENTIONALLY LEFT BLANK.

4. STARTING COURSE 525A

When you have worked through Sections 1 - 3 of this book you are ready to proceed with the material of the Microcomputer Software/Hardware Training Course. You can omit the material of the introductory chapter, "System Setup and Test Procedure." Some of the material in Chapter 1, "Hardware and Software Fundamentals" will now be familiar, but the basic concepts in that chapter are vital to your further understanding. Use the binary and hexadecimal demonstration programs as you study Chapter 1, and again in Chapter 10.

The work in Course 525A does not use the Interface Training System and experiment board. To avoid any interference, or surprises if you make programming mistakes, set all four switches on the experiment board to the UP position. This also relieves the battery of any power drain from transistor leakage current.

4.1 Further Reference to This Book

The remaining sections of this book will be useful in your later work.

- Section 5. Course 536A Experiments
- Section 6. Thermometer and Thermostat Program
- Section 7. Motor Speed Control Program
- Section 8. Adding Tunes and Messages

When you start work on the Interface Training Course, refer to Section 5 of this book. Some of the electrical setups for experiments in Course 536A are made easier by use of the experiment board; Section 5 of this book gives the necessary information. When you have finished Chapter 6 of 536A, the thermometer and thermostat demonstration programs will be much more meaningful; refer to Section 6 of this book. Similarly, Chapter 7 of 536A teaches about the design of the motor speed demonstration. Here you will want to refer to Section 7 of this book.

4.2 User Programs and the Monitor

In the demonstration programs you have been entering data and commands to built-in programs. The most important program supplied with the Microcomputer Training Systems is the "monitor". You will use this constantly as you carry out the work of the training courses. It provides the facilities for entering, testing and debugging your own programs. When you reach Section 1.5 of Course 525A the monitor will be described as you start to enter your own programs.

PROCEED NOW TO CHAPTER 1 OF COURSE 525A.

WHEN YOU START COURSE 536A, REFER TO

SECTION 5 OF THIS BOOK.

This page intentionally left blank.



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 5

COURSE 536A EXPERIMENTS



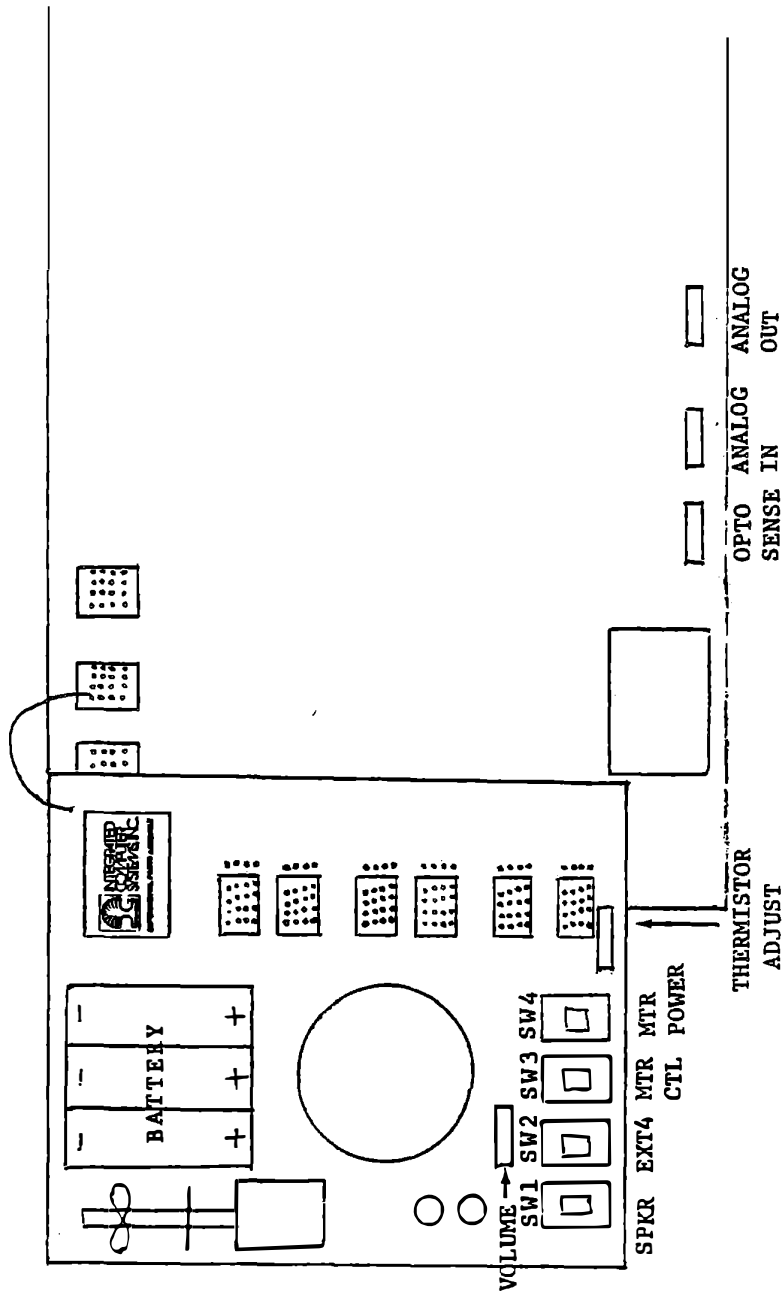
INTEGRATED
COMPUTER
SYSTEMS™

THIS PAGE INTENTIONALLY LEFT BLANK.

5. COURSE 536A EXPERIMENTS

The integrated experiment board is plugged into the Interface Training System. This board provides the external devices and interconnections for the demonstration programs and for the advanced experiments in the Interface Training Course (ICS Course 536A). The experiment board includes a battery holder for three AA cells to drive the motor mounted on the board, and four slide switches to permit various connections. When the experiment board is not in use these switches should all be in the UP position. This prevents any drain on the batteries and also avoids surprises that might result from programming errors while carrying out the exercises of Course 525A. No harm will result if the switches are inadvertently left in the DOWN position.

The Real Time Educational Firmware includes a number of demonstration programs in Read Only Memory. To use these demonstration programs all of the switches should be in the DOWN position, and batteries should be installed. The demonstrations were described briefly in Section 2; more detail is provided in Sections 3, 6, 7, and 8 of this book.



ITS and Integrated Experiment Assembly

Figure 5-1

5.1 The Integrated Experiment Board

Various experiments in Course 536A require different switch settings on the experiment board, and sometimes additional connections to be made by wires or clip leads. Course 536A does not describe the use of the experiment board. Whenever that course describes connections to be made for experiments, refer to the notes below for information on how to make the required setup with the experiment board.

Figure 5-1 shows the physical arrangement of the Interface Training System with the Integrated Assembly Board. Note the three adjustment potentiometers at the lower edge of the Interface Training System -- OPTO SENSE, ANALOG OUT, ANALOG IN. These are round devices with blue edges that can be rotated to adjust the resistance in analog circuits of the Interface Training System.

The experiment board also includes two pots, VOLUME and THERM ADJ. In temperature measuring experiments you will use THERM ADJ instead of OPTO SENSE.

Table 5-1 indicates the effects of rotating these pots.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

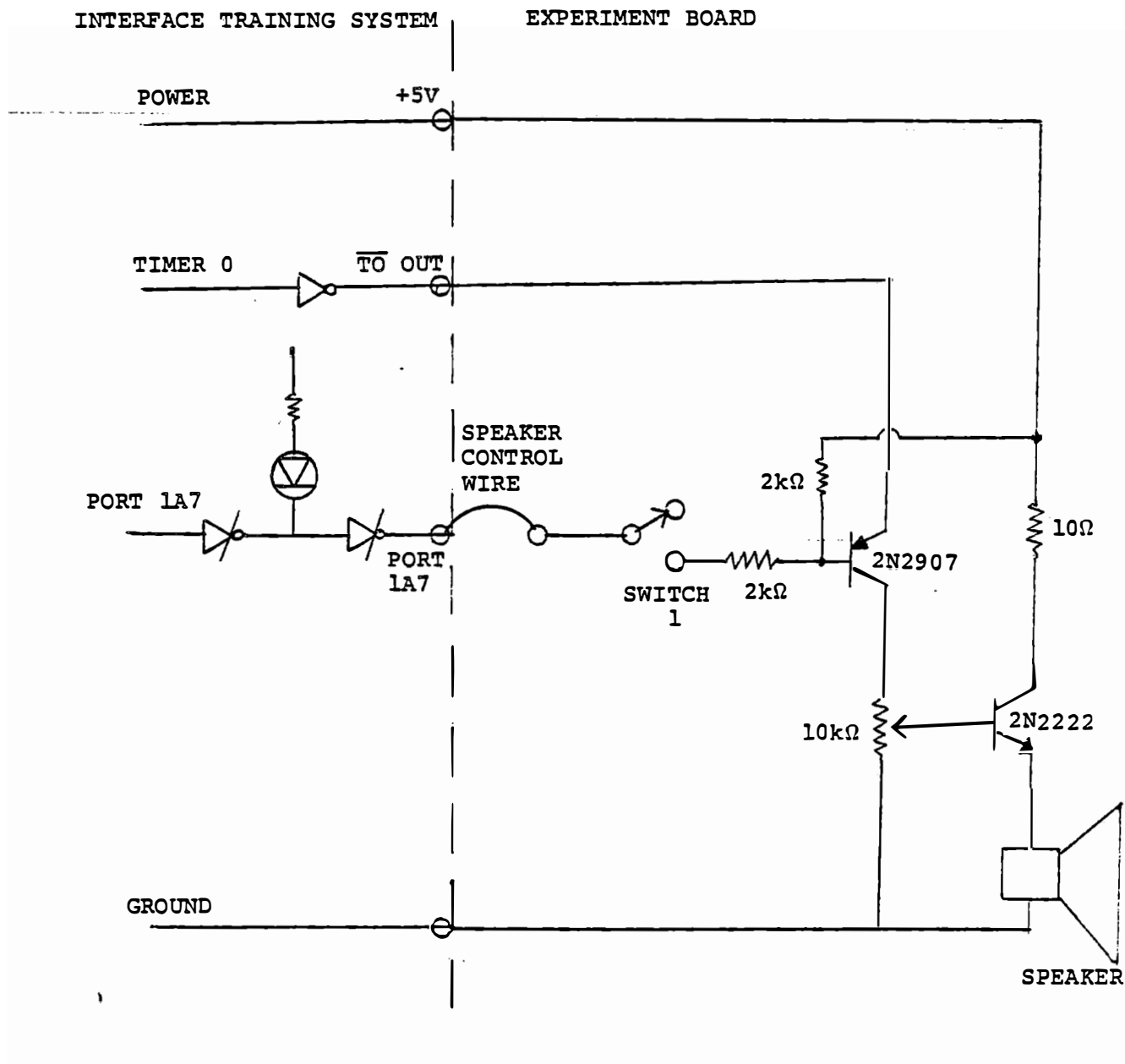
<u>POTENTIOMETER</u>	<u>RIGHT</u>	<u>LEFT</u>
VOLUME	Louder	Quieter
THERM ADJ	Lower Resistance or Lower Temperature	Higher Resistance or Higher Temperature
OPTO SENSE	Higher Resistance or Lower Voltage	Lower Resistance or Higher Voltage
ANALOG OUT	Higher Output Voltage	Lower Output Voltage
ANALOG IN	Lower Gain	Higher Gain

Effects of Potentiometer Adjustments

Table 5-1

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

The circuits of the Interface Training System are fully described in Course 536A. The following sections describe the circuits of the experiment board. Refer to these when you perform experiments involving the experiment board. When all of the switches are in the UP position, the experiment board is effectively disconnected for almost all functions.



Loudspeaker Connections

Figure 5-2

5.1.1 Loudspeaker Control

The loudspeaker on the experiment board is driven by a timer signal, TO OUT, from the Interface Training System. (See Figure 5-2.) A control input, normally connected by a wire to Port 1A7 output, turns the loudspeaker on when that signal is low. Switch number 1 (at the left of the experiment board) opens this connection when the loudspeaker is not wanted. In this configuration the $\overline{\text{TO}}$ OUT signal is effectively disconnected by a transistor switch. When the control input (Port 1A7) is low and the switch is on (DOWN) the output of Timer 0 drives the loudspeaker. A volume control is provided. To adjust the loudspeaker volume, use the demonstration program by pressing the following keys.

```

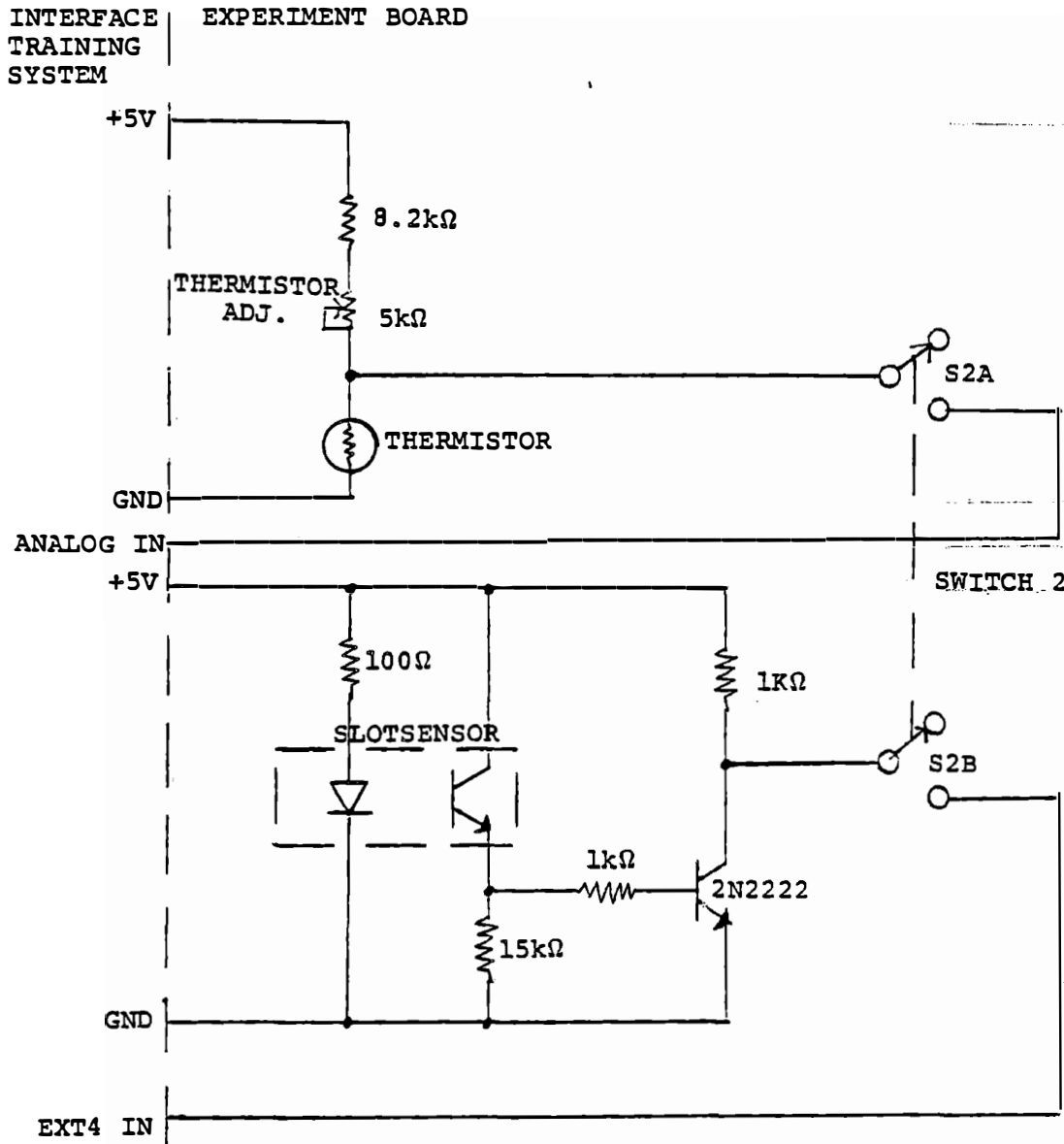
RESET
ADDR  0      4      0      0      RUN
    
```

The moving message appears.

```

RUN
    
```

A tune is played. Set Switch 1 down and adjust the volume control for comfort. If the tune ends before the adjustment is satisfactory, press RUN again.



Thermistor and EXT4 Input Connections

Figure 5-3

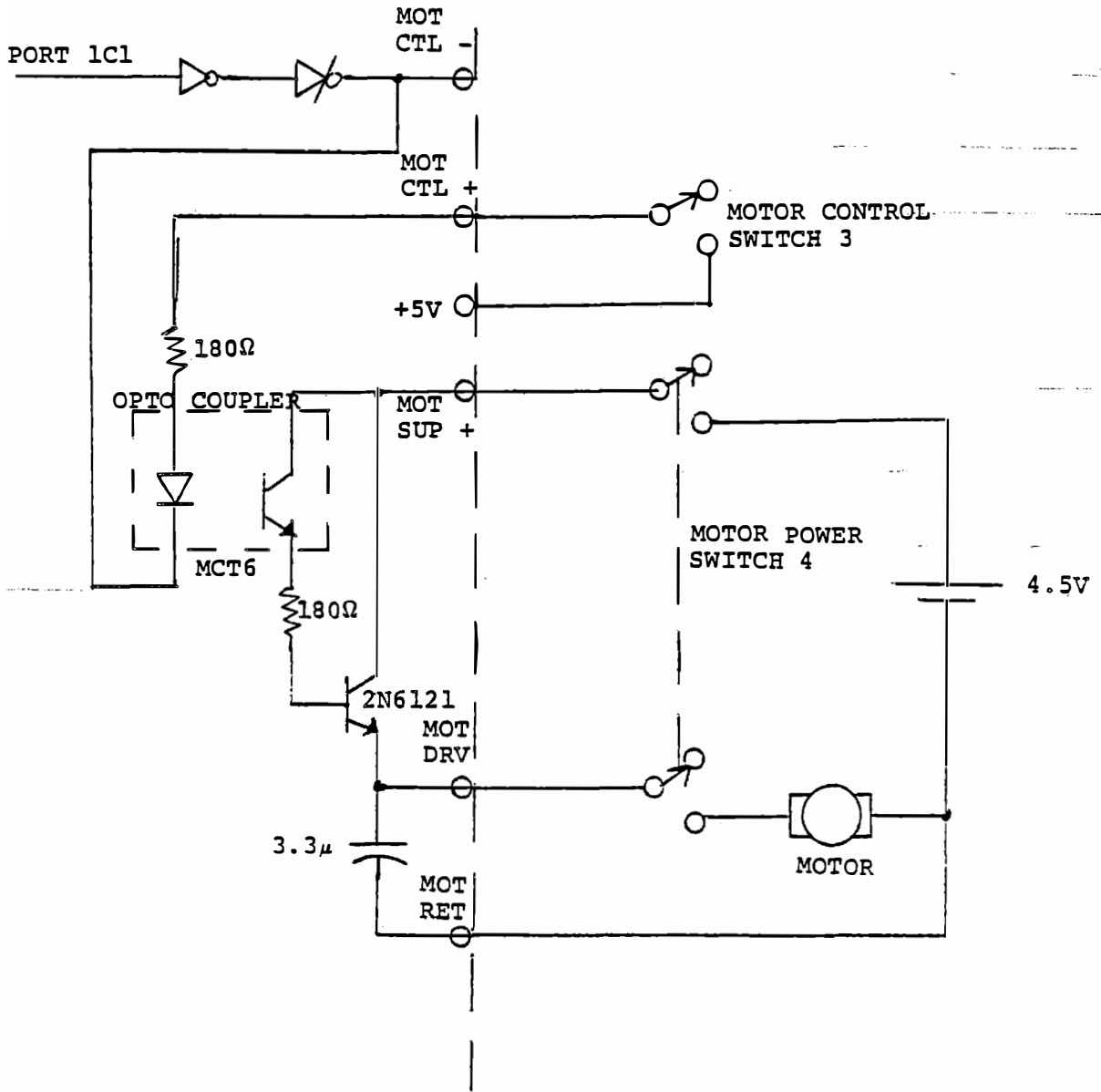
5.1.2 Thermistor Connection

A thermistor (temperature measuring device) is mounted on the experiment board. It is normally held in a spring clip in front of a fan blade on the motor, but can be removed for other experiments such as immersing it in water. Prolonged immersion should be avoided, as there is not a hermetic seal. The thermistor is connected to ground and through a resistor and potentiometer to +5 volts. (See Figure 5-3.) The connection point is connected, through Switch 2, to the ANALOG INPUT of the Interface Training System to permit temperature measurements with Switch 2 DOWN. The Analog Input can be used for other experiments with Switch 2 UP. This switch is shared with the Slot Sensor.

5.1.3 Slot Sensor Connection

The motor shaft has an optical disk with 16 pairs of alternating opaque and transparent segments. This disk passes between the LED and phototransistor of an MCT-8 slot sensor, to permit measurement of motor speed. The slot sensor signal is amplified (See Figure 5-3) and when Switch 2 is DOWN the output is connected to EXT4 IN, permitting the slot sensor signal to be read under program control or to generate an interrupt. With Switch 2 UP the EXT4 input can be used for other experiments.

INTERFACE TRAINING SYSTEM EXPERIMENT BOARD



Motor Circuit Connections

Figure 5-4

5.1.4 Motor Control and Drive Circuit

Figure 5-4 shows the motor control and motor drive connections of the Interface Training System. To avoid any motor noise from entering the computer a battery of three AA cells provides power for the motor. (See the lower part of Figure 5-4.) When Switch 4 is UP, the battery and motor are disconnected. The power transistor can be used for other purposes. When Switch 4 is DOWN, battery power is applied to the phototransistor in the MCT6 optical coupler and to the 2N6121 power transistor. If current flows in the light emitting diode (LED) of the optical coupler, the phototransistor is turned on and supplies base current to the 2N6121, which is thereby also turned on so that battery power is also applied to the motor. Note that the battery and motor are electrically isolated from the computer; the only interface is through the optical coupler.

Current in the LED of the optical coupler can be controlled by either of two sources, depending on the setting of Switch 3. In the DOWN position the anode of the LED is connected through a resistor to +5 volts. The cathode is connected (on the Interface Training System) to an open collector driver controlled by Port 1C1. If this output bit is set low, power is applied to the motor. If Port 1C1 is set high, no power is applied unless MOT CTL - is externally connected to ground.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

When Switch 3 is in the UP position, the LED anode is disconnected but is accessible at MOT CTL +. An external device wired to MOT CTL + can control the motor. A positive signal turns the motor on. The voltage applied must not exceed +5 volts, or damage to the optical coupler may result. As above, Port 1C1 still controls current flow in the LED, and must be set low, or MOT CTL - must be connected to ground.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

5.2 Experiment Board Connections for
Course 536A Experiments
VOLUME I

At each place in Course 536A where an electrical setup for an experiment is described, refer to these notes to see whether a different setup is to be used with the experiment board.

Section 2.2 (page 2-6)	No connections required. All switches UP.
Section 2.3 (page 2-11)	Connect as shown. All switches UP.
Section 2.6 (page 2-17)	Connect as described. All switches UP.
Section 2.7.4 (page 2-27)	Same connections as 2.6. All switches UP.
Section 2.9 (page 2-37)	Same connections as 2.6. All switches UP.
Additional Experiment:	Using the program of Section 2.9, set Switch 2 DOWN. Actuate the EXT4 input by rotating the fan by hand instead of grounding a clip lead.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 3.4 (page 3-15)	No connections. All switches UP.
Section 3.5 (page 3-21)	No connections. All switches UP.
Section 3.6.1 (page 3-25)	Connect as described. All switches UP.
Section 3.6.2.1 (page 3-34)	Connect as described. All switches UP.
Section 3.6.2.2 (page 3-36)	Same connections as 3.6.1. All switches UP.
Section 3.6.3 (page 3-38)	Same connections as 3.6.1. All switches UP.
Section 3.7.2 (page 3-43)	No connections. All switches UP.
Section 3.8 (page 3-51)	Connect as described. All switches UP.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 3.9.1 Connect TO OUT to EXT4 IN.
(page 3-61) Connect Port 1A7 OUT to GND.
 Switch 1 DOWN.
 Switches 2, 3, and 4 UP.

Timer 0 drives the loudspeaker when Switch 1 is DOWN,
provided that Port 1A7 is low. When operating in Mode
3, a buzz is audible. In Mode 2, the loudspeaker does
not respond to the narrow input pulses and no sound is
heard.

Section 3.9.2 No connections.
(page 3-62) All switches UP.

Section 4.2.1 Connect as shown.
(page 4-6) All switches UP.

Section 4.2.2 Connections same as 4.2.1.
(page 4-23) All switches UP.

Section 4.3.1 Speaker Control Wire
(page 4-26) to Port 1A7.
 Switch 1 DOWN to turn
 loudspeaker on.
 Switches 2, 3, and 4 UP.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 4.3.2 Same connections and
(page 4-29) switch settings as 4.3.1.

Section 4.3.3 Same connections and
(page 4-33) switch settings as 4.3.1.

NOTE: The tone table for Section 4.3.3 is available
in ROM at address 07A0. Instead of entering
the table in your program, use the table in
ROM. In the interrupt service routine address
the ROM table:

```
825D    21                    LXI H,07A0
825E    A0
825F    07
```

The coding of the tunes in the demonstration
is different than that used in Section 4.3.3,
so these cannot be used.

Section 4.3.4 Same connections and
(page 4-48) switch settings as 4.3.1.

Use ROM tone table as in 4.3.3.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 4.4 Connect speaker control
(page 4-51) wire to ground.

 Switch 1 DOWN.

 Switches 2, 3, and 4 UP.

After this experiment, reconnect the speaker
control wire to Port 1A7 output.

Section 4.6.3 Connect as shown.
(page 4-71) All switches UP.

 Adjust ANALOG OUT pot
 as described.

Section 4.7.1.1 Connect as described.
(page 4-77) All switches UP.

Section 4.7.1.2 Same connections as 4.7.1.1.
(page 4-78) All switches UP.

Section 4.7.2 Same connections as 4.7.1.1.
(page 4-80) All switches UP.

Section 4.7.2.8 Connect as described.
(page 4-108) All switches UP.

Section 4.7.3.1 Connections same as 4.7.2.8.
(page 4-116) All switches UP.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 5.1
(page 5-2)

Connect test point on AMTS
labelled AUDIO OUT to ITS
input OPTO IN.

Connect OPTO OUT to EXT4 IN.
All switches UP.

Section 5.1.2
(page 5-6)

Same connections as 5.1.
All switches UP.

Section 5.1.3
(page 5-24)

Connect test point on AMTS
labelled AUDIO IN to ITS
input OPTO IN.

Connect OPTO OUT to EXT4 IN.
All switches UP.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 5.2.1 Same connections as 5.1.
(page 5-24) All switches UP.

Section 5.2.1 ALTERNATIVE EXPERIMENT
(page 5-24) Measure Motor Speed
 No external connections.
 Switch 1 UP.
 Switches 2, 3, and 4 DOWN.

Modify Program as follows:

page 5-27 Insert into initialization
 MVI A,80 Program 8255 1
 OUT CNT1 A out, B out, C out
 Change in initialization
 MVI A,32 Load Timer 0 for
 OUT TIMO 6.25 millisecond interrupt

The program will count dark segments of the optical disk on the motor shaft. Since the disk has 16 dark segments counting for 5/8 second will give ten times the motor speed in revolutions per second.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

- Section 5.2.2
(page 5-30)
- Connect 100k ohm resistor
from ANALOG IN to GND.
- Connect lead from modem
output jack on MTS (jack
labelled with outward arrow)
to ANALOG IN.
- All switches UP.
- Section 5.3
(page 5-40)
- Connect as shown.
- Connect voltmeter to ANALOG OUT.
- All switches UP.
- Section 5.3.1
(page 5-43)
- Same connections as 5.3.
- All switches UP.
- Adjust ANALOG OUT pot as described.
- Section 5.3.2
(page 5-53)
- Same connections as 5.3.
- All switches UP.
- Section 5.3.3
(page 5-63)
- Same connections as 5.3.
- All switches UP.
- Section 5.3.4
(page 5-66)
- Same connections as 5.3.
- All switches UP.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

- Section 5.4.1 Same connections as 5.3.
(page 5-77) All switches UP.
- Section 5.4.2 Same connections as 5.3.
(page 5-83) All switches UP.
- Section 5.5 Same connections as 5.3.
(page 5-88) All switches UP.
- Section 5.5 Remove capacitor as described.
(page 5-96) All switches UP.
- Section 5.6.3 Connect voltmeter from ANALOG IN
(page 5-110, 111) to GND.
- Do not connect OPTO SENSE.
Switch 2 DOWN to connect thermistor.
Switches 1, 3, and 4 UP.

The ANALOG IN pot must be adjusted to divide the input voltage by 2. Use a voltmeter to measure the actual input voltage. Use a digital voltmeter program such as Figure 5-26, 5-28, or 5-30 to display the voltage measured by the A/D converter. Adjust the ANALOG IN pot to make the digital output equal to half of the actual voltage.

The experiment board has a pot labelled THERM ADJ. Use this (instead of OPTO SENSE) to adjust the thermistor input as described on page 5-111.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 5.6.5 Connections and calibration
(page 5-115) as in 5.6.3.

Section 5.6.6 Connections and calibration
(page 5-126) as in 5.6.3.

Section 5.6.7 Connections and calibration
(page 5-139) as in 5.6.3.

NOTE: The ROM contains a voltage to temperature
 conversion subroutine VTEMP at address 0418.
 This may be used in temperature experiments
 similar to the above, and in subsequent
 temperature control experiments. The temperature
 is displayed at the right with one decimal place.

Enter (A) = Voltage
Returns (A) = Voltage
 (BC) = Preserved
 (D) = Preserved
 (HL) = Temperature

Section 5.6.8 Connect as described.
(page 5-149) All switches UP.

NOTE: This experiment requires an external thermistor (not supplied). If you choose to perform this experiment it is suggested that a low resistance thermistor (1k or lower) be used to make self-heating more apparent.

Section 5.6.9 Connect as described.
(page 5-160) All switches UP.

NOTE: This experiment requires an external thermistor and a set of resistors, which are not supplied.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

5.3 Experiment Board Connections for
Course 536A Experiments
VOLUME II

Section 6.1 Connect external power supply
(page 6-3) to MOT SUP + and MOT RET.
 Connect 20 ohm, 10 watt
 resistor (not supplied)
 to MOT DRV and MOT RET.
 Switches 2 and 3 DOWN.
 Switches 1 and 4 UP.

 Caibrate thermistor as described for
 Section 5.6.3.

Alternate Connect as shown.
(page 6-5) All switches UP.

NOTE: The ROM contains FILTR and VTEMP which
may be used instead of the programs you
have developed. In the given solution
(Figure 6-4b):

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

8239	CD	CALL FILTR
823A	24	
823B	04	
823C	CD	CALL VTEMP
823D	18	
823E	04	
823F	EB	XCHG

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

the XCHG instruction must be added because VTEMP in ROM returns the temperature in (HL) rather than (DE). The remainder of the program solution in Figure 6-4b must be moved down by one byte.

This experiment may be modified to use the motor and fan to cool the thermistor as in the demonstrations. To do this, no external connections are necessary. Set Switch 1 UP and Switches 2, 3, and 4 DOWN. Modify the program to turn power on when the temperature is high and off when the temperature is low. Note, however, that the fan really has very little cooling effect. This is useful only as a demonstration, not as a real control system. Enter a setpoint one tenth degree greater than the measured temperature, so that the fan is off. Warm the thermistor momentarily by touching it with one finger. The fan will turn on; then as it cools through the setpoint it will switch on and off rapidly.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Section 6.1.2 Same connections and
(page 6-21) calibration as 6.1.

Again the program may be modified to use the ROM versions of FILTR and VTEMP, with XCHG after the return from VTEMP. If you want to use the fan, the same program modification described above must be made.

Section 6.1.3 Same connections and
(page 6-26) calibration as 6.1.

Section 6.1.4 Connect as shown.
(page 6-29) Switch 3 DOWN.
 Switches 1, 2, and 4 UP.

Section 6.2.1 Connect as shown.
(page 6-41) All switches UP.

These connections are used through the remainder of Chapter 6.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Chapter 7

Connect $\overline{\text{TO}} \text{ OUT}$ to G1 IN.

Switch 1 UP.

Switches 2, 3, and 4 DOWN.

The motor, optical disc, and slot sensor are mounted on the experiment board. An amplifier is provided for the slot sensor to ensure reliable operation. Setting Switch 2 DOWN connects the amplifier to the EXT4 input. No adjustment is necessary. Read Sections 7.1.2 and 7.1.3 for information, but do not make any of the connections except for $\overline{\text{TO}} \text{ OUT}$ to G1 IN. Insert three AA cells in the battery holder of the experiment board, with the positive ends down. The motor connections shown in Figure 7-6 are made by the switches.

These connections are used throughout Sections 7.1 through 7.3.

Section 7.4

(page 7-83)

Connect ANALOG OUT to MOT CTL +

Switches 1 and 3 UP.

Switches 2 and 4 DOWN.



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 6
THERMOMETER AND THERMOSTAT
(PROGRAM 3)



INTEGRATED
COMPUTER
SYSTEMS™

THIS PAGE INTENTIONALLY LEFT BLANK.

6. THERMOMETER AND THERMISTOR (Program 3)

Cooling Thermostat with Variable Speed Fan and Alarm

Here are given further details describing the demonstration programs.

This was originally a thermostat with simple on-off control, but to make a more interesting demonstration the fan speed was made variable. The result is actually a proportional control system within a restricted range.

Temperature is measured using the analog to digital converter to sense voltage across a thermistor. The measured value is filtered by a digital noise filter. At intervals of about one-half second the filtered voltage is converted to decimal degrees Celsius and displayed.

At initialization the control function is disabled and only the measured temperature is displayed. This provides a thermometer demonstration. When RUN is pressed, the control limits are calculated for preset temperature limits and the upper limit is displayed. The control function is now enabled.

A temperature limit is set initially to 25.0 degrees C. This can be changed by entering a decimal number followed by RUN. This value represents the upper limit of the desired temperature range. When the temperature exceeds this value the fan is turned on.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

A deadband value is set initially to 2.0 degrees C. This can be changed by entering a decimal number followed by STEP.

One decimal place can be entered in either of the above values by using key D as a decimal point. For example:.

2D5 STEP sets the deadband to 2.5 degrees C.

26D7 RUN enters a setpoint of 26.7 degrees C.

Either value can be displayed by pressing RUN or STEP without entering any data.

The alarm limit is equal to the upper limit plus the deadband. When the temperature rises above this value, the fan is at full full power and the alarm is turned on.

The lower limit is equal to the upper limit minus the deadband. When the temperature falls below this value the fan is turned off. It then stays off until the temperature again rises above the upper limit.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

When the fan is running, its speed is proportional to the difference between the temperature and the lower limit. At the upper limit half power is applied and at the alarm limit full power is applied. There are two exceptions: when the fan is first turned on, full power is applied briefly to overcome starting friction; and, the power is never reduced below 25% until the fan is turned off.

The fan power control can be displayed by pressing ADDR. The left digits show the duty cycle as a fraction of hexadecimal 10. 04 represents 25%; 08 represents 50%. The next two digits show 03 if the motor is off, 02 if it is on. This display is primarily intended for debugging rather than demonstrations. Press RUN to restore the normal display.

The CLR key turns off the alarm. Heat the thermistor well above the alarm limit. (Immerse it in hot water or coffee, if possible, or enter an alarm limit barely above room temperature.) The alarm will sound. Turn it off by pressing CLR. It will stay off while the thermistor cools, even though still above the alarm limit. Reheat the thermistor and the alarm will sound again. This demonstrates a realistic control system requirement, where an audible alarm is turned off when remedial action has been taken, but sounds again if the remedial action fails to solve the problem.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Control is exercised by the interrupt service routines. The measured thermistor voltage is entered into a digital noise filter. The filtered voltage is compared with binary values calculated when a temperature or deadband is entered. Three values are stored: the lower temperature limit, the deadband, and the alarm limit. Since the voltage decreases with rising temperature, the alarm limit voltage is lower than the upper and lower limits. Because the thermistor is non-linear, all of these must be recalculated when either the temperature limit or the deadband is changed.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

The binary deadband and lower limit can be displayed by pressing REG.

Display Upper Limit	RUN	<input type="text" value="25.0"/>
Display Binary Values	REG	<input type="text" value="0683"/>
Deadband = 06		
Lower Limit = 83		
Upper Limit = 83 - 06 = 7D		
Enter Limit	50 RUN	<input type="text" value="50.0"/>
Display Binary Values	REG	<input type="text" value="0441"/>
Deadband = 04		
Lower Limit = 41		
Upper Limit = 41 - 04 = 3D		
Enter Deadband	10 STEP	<input type="text" value="10.0"/>
Display Binary Values	REG	<input type="text" value="114E"/>
Deadband = 11		
Lower Limit = 4E		
Upper Limit = 4E - 11 = 3D		

This display is intended primarily for debugging, but may be of interest to some engineers.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

SUMMARY OF CONTROLS FOR THMCL

RUN	Start control function and display the upper temperature limit.
XX RUN	Enter an upper temperature limit.
XXDX RUN	Enter an upper temperature limit with one decimal place.
STEP	Display the deadband.
XX STEP	Enter the deadband.
XDX STEP	Enter a deadband with one decimal place.
CLR	Turn off the alarm. The alarm remains off until the temperature rises and is greater than the alarm limit.
ADDR	Display power control.
REG	Display binary deadband and lower limit.
MEM	Not defined.
BRK	Go to Motor Speed demonstration.
n BRK	Go to program number n.



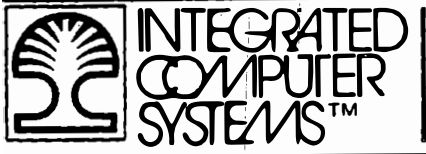
INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 7

MOTOR SPEED CONTROL

(PROGRAM 4)



THIS PAGE INTENTIONALLY LEFT BLANK.

7. MOTOR SPEED CONTROL (Program 4)

Proportional Plus Integral Control with PWM

This demonstrates a closed loop speed control system. Motor speed is measured by sensing the optical disc on the motor shaft which interrupts the slot sensor light path.

The program measures average speed by counting segments in decimal during 10/16 second. This gives speed in revolutions per second, displayed in four digits at the left, with one decimal place.

Speed is controlled by a feedback loop with proportional and integral gain, based on an instantaneous speed measurement. The time for one segment to pass through the sensor is measured and converted to speed by:

$$\text{Speed} = K / \text{Time}.$$

The calculated speed is compared with the desired speed (entered with RUN) and the new power pulse width is calculated by subroutine WIDTH from:

$$\text{Width} = G_p E + G_i \int E$$

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Motor power is controlled by Timer 0 and Timer 1. Timer 1 generates 256 interrupts in 10/16 of a second. If the motor has been commanded to run, power is applied. Timer 1 starts Timer 0, which interrupts after a time equal to "width" and turns power off. The ratio of the Timer 1 interval to the Timer 0 interval is the "power duty cycle". This is converted to decimal and displayed at the right as a percentage.

The instantaneous speed is output to the D/A converter. A voltmeter connected here displays the speed. Alternately, the voltmeter may be connected to the motor to show average power.

SUMMARY OF CONTROLS FOR MTRSPD

RUN Starts the motor. The previously requested speed is resumed.

XX RUN Enters a new desired speed in revolutions per second. The speed is entered in decimal. Any value from 0 to 99 may be entered, but in general the motor will only run from about 10 to 60 rps.

STEP Stops the motor. The average speed display is replaced by a count of the segments that pass the sensor while the motor coasts to a stop.

or

CLR While the motor is stopped, the propeller can be turned by hand. The count will change but it will not reliably count segments. The sensor may count several times at each segment.

MEM While the key is pressed, the preset gain settings are displayed. Proportional gain is shown in the two left-hand digits and integral gain in the following two digits.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

XXXX MEM

Enter new gains. For instance, 801 MEM sets proportional gain to 8 and integral gain to 1. These seem to be good values for speed control.

Try setting zero proportional gain and high integral gain by 8 MEM. Slow the motor by pressing the propeller hub and then releasing it. The motor speed will hunt much more noticeably before settling. It may continue to hunt indefinitely.

0 MEM

Set both gains to zero, giving open loop control.

XX NEXT

Set duty cycle. With open loop control the duty cycle will not change. Press RUN to start the motor if it is stopped.

ADDR

Undefined.

REG

Undefined.

BRK

Go to Tune and Message program.

n BRK

Go to program number n.



INTEGRATED
COMPUTER
SYSTEMS™

EXPERIMENT ASSEMBLY AND
REAL-TIME FIRMWARE MANUAL

SECTION 8

TUNE AND MESSAGE VARIATIONS

(PROGRAM 0)



INTEGRATED
COMPUTER
SYSTEMS™

THIS PAGE INTENTIONALLY LEFT BLANK.

8. TUNE AND MESSAGE VARIATIONS (Program 0)

This program shows a moving message and simultaneously plays a tune. Many features are available, including:

Select either of two tunes in ROM.

Select any tune stored in RAM.

Key in a tune, playing each note, and storing it in RAM.

Change the tempo of a tune.

Select a different moving message stored in RAM.

The moving message is timed to the tune that is playing or was last played. In the absence of an Interface Training System, the moving message can still be displayed.

Hardware Timer 1 is programmed to interrupt at an interval entered as the tempo for the tune. The usual tempo is set at 20 (decimal) giving an interrupt every 2000 clock pulses. When a tune is playing each note code taken from memory gives a count representing the note duration. This is counted down at each Timer 1 interrupt and the note ends when the count reaches zero.

When each note of the tune starts, the note code addresses a table giving a count representing the inverse of the frequency for that note. This value is loaded to Timer 0, whose output is then the required frequency to sound the note.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Tunes can be entered using the hexadecimal keys as a keyboard. Only the natural notes of the key of C are provided. C gives middle C. Other notes are shown below:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E	F	G	A	B	C	D	E	F	G	A	B	C	D	E	F

Several tunes can be stored in RAM and played successively or selectively by this procedure.

MEM represents storage space for a tune, and by default sets the tempo to 20 (decimal). Follow with hex keys to play notes.

RUN plays the tune.

MEM requests storage space for another tune. This procedure can be repeated as long as desired. The storage space allocated automatically starts at 8000. This same space is used by the Motor Speed Program, so the tune will be destroyed when that demonstration is used. If you want to preserve a tune, specify a location at 8400 or above, by:

ADDR	8	4	0	0	MEM
------	---	---	---	---	-----

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

When a tune is keyed in, its address is automatically entered into the tune directory so that it can be selected again by using the NEXT key, which addresses each tune in the directory in turn. The directory is cleared each time the TUNMSG program is restarted. Then a tune that has been saved in memory can be placed into the directory by:

```
ADDR   XXXX   RUN
```

Instead of using the keyboard as a musical instrument, a tune may be precoded and entered into memory using the monitor. To play such a tune, after it has been entered, do this:

```
RST
ADDR   0       4       0       0       RUN
ADDR   X       X       X       X       RUN
```

where XXXX is the starting location of the precoded tune. This procedure plays the tune, and also stores its address in the directory so that it can be selected again by using the NEXT key.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

A pre-encoded tune must follow the format used by the TUNMSG program, which is shown below. Byte 0 specifies the tempo, byte 1 contains FF as a marker. Following byte pairs give note pitch and duration for each note. A rest equivalent to a 1/64 note is automatically entered after each note is played; longer rests can be programmed by setting pitch as 30H. Slurs are not available. The last two bytes in the tune code are set to FF to mark the end of the tune.

TUNE CODE

BYTE	CODE
0	Tempo
1	FF
2	Note Pitch (See Table 8-1)
3	Note Duration (See Table 8-2)
Last two bytes	FFFF

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Tone	4 Octaves			
C	00	0C	18	24
C (Db)	01	0D	19	25
D	01	0E	1A	26
D (Eb)	03	0F	1B	27
E	04	10	1C	28
F	05	11	1D	29
F (Gb)	06	12	1E	2A
G	07	13	1F	2B
G (Ab)	08	14	20	2C
A	09	15	21	2D
A (Bb)	0A	16	22	2E
B	0B	17	23	2F
REST	30			
END	FF			

Note Pitch

Table 8-1

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

Note Duration	Code (high byte of note code)
1/64	01
1/32	02
1/16	04
1/8	08
1/4	10
1/2	20
Whole	40
2	80
4	00

(Add values in hexadecimal for intermediate durations, e.g., 18 for 3/8.)

Note Duration

Table 8-2

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

You can examine the coding for either of the tunes in ROM by using the monitor. These tunes are found at OAA0 (Home on the Range) and OAF4 (Drunken Sailor). You can also play a tune one note at a time. Enter the TUNMSG program, and press STEP. One note will be played and the address and code for the next note will be displayed. The first "n" notes of a tune can be played by entering the number desired followed by STEP. You can stop a tune in the middle by pressing RUN, and then STEP when you want to stop it; play several notes one at a time using STEP; then finish playing the tune by pressing ADDR.

Note that when stepping through a tune any rests that have been included in the coding are also displayed as though they were notes. A tune entered by playing it on the keyboard always includes a rest after every note, so you must press STEP twice for each actual note.

Although every tune in memory must start with a tempo byte, this can be altered when the tune is played by entering a one or two digit decimal number before pressing RUN. This does not change the coding of the tune, but is stored in the directory and is used whenever the tune is replayed.

The following pages summarize the effects of the keys in the TUNMSG program.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

CONTROLS FOR TUNMSG PROGRAM

RUN Play the current tune

X RUN Enter a different tempo

XX RUN and play the current tune.

XXXX RUN Select a tune by its address
in memory. If this address
is not in the directory it is
entered into the directory.

REG Display the address of the
current tune.

X REG Enter a different tempo for the
current tune.

XXXX REG Enter a tune address into the
directory and make it the current
tune.

NEXT Display the address of the next
tune in the directory and make
it the current tune.

n NEXT Display the address of the nth
tune in the directory and make
it the current tune.

(Note that REG and NEXT do not start playing the tune, nor do they stop a tune that is already playing.)

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

- MEM Prepare to key in a tune.
 Selects and displays the next
 available address in RAM for
 storing the tune, and enters
 it into the directory. Sets a
 default tempo of 20.
- X MEM As above but enters a tempo
XX MEM into the tune memory.
- XXXX MEM As above but enters XXXX as
 the tune address.

After MEM has been pressed, key in the tune by pressing hexadecimal keys. The program times the duration of each note, and also rests between notes. The note is played while the key is held down. When the key is released the address and code for the notes are displayed.

When the tune has been keyed in, press RUN to play it. The tune is held in memory and its address is in the directory.

EXPERIMENT ASSEMBLY AND REAL-TIME FIRMWARE MANUAL

STEP Play the current tune one
 note at a time and display
 the note code being played.
 Also stop a tune that has been
 started by RUN.

n STEP Play n notes of the current
 tune, starting at the beginning.

CLR Remove the current tune from
 the directory. It is no
 longer available to be played.
 It can be restored by entering
 its address followed by REG or
 RUN.

ADDR Restart the moving message.
 Also finish playing a tune that
 has been stopped by STEP.

X ADDR Change the speed of the moving
XX ADDR message. (Higher numbers move
 faster.)

XXXX ADDR Select a moving message stored
 in RAM by entering its address.

BRK Go to the binary arithmetic
 program.

n BRK Go to program n.

1
1
(1
1
1
1
1
1
(1
1
1
1
1
1
(1

אין תהיה נחמה לך ואל תהיה נחמה לך ואל תהיה נחמה לך ואל תהיה נחמה לך