1986

Design of a Laboratory course in Embedded Computer Systems

Alexander Nikoloff
University of Central Florida

DESIGN OF A LABORATORY COURSE IN EMBEDDED COMPUTER SYSTEMS

BY

ALEXANDER NIKOLOFF
B.S.E, University of Central Florida, 1985

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the Degree of Master of Science in Engineering
in the Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Summer Term
1986

ABSTRACT

This research report discusses and presents the design of a university undergraduate level laboratory course introducing the topic of embedded computer systems. The course utilizes the Rainbow 100 computer and the Data Translation LDT2801 interface board to illustrate this concept.

Lab problems in Digital to Analog conversions, Analog to Digital conversions, Digital input/output, serial communication, motor drivers and parallel communication are presented, as are fully documented solutions. Suggested lecture material appropriate to the course is reviewed.

ACKNOWLEDGEMENTS

THANKS:

MOM

DAD

KATINKA

NATHALIE

Dr. BAUER

Dr. LINTON

Dr. KLEE

TABLE OF CONTENTS

LIST OF TABLES v

LIST OF FIGURES. vi

INTRODUCTION 1

CHAPTER ONE - COURSE OVERVIEW. 3
The Basics of Interfacing 3
Laboratory Set-up 3
Design of the Course. 5

CHAPTER TWO - THE LDT2801 BOARD. 8
LDT2801's Driver: The Rainbow 100 8
LDT2801 Description 8

CHAPTER THREE - LAB COURSE LIMITATIONS AND
POSSIBLE SOLUTIONS 22

CONCLUSION 25

APPENDIX 26

REFERENCES 84

LIST OF TABLES

1. LDT2801 Operating Code
(Bits 0 to 3 in Command Register) 13

2. LDT2801 REad Error Bits and Explanation 14

3. LDT2801 Command Sequence. 15

4. Explanation of Symbols used in Table 3. 16

LIST OF FIGURES

1. Rainbow 100 System Block Diagram. 9

2. Rainbow 100 Communication and Printer Port Signals. . 10

3. Diagram of Automated Factory Set-up 38

4. Intersil Display Data Sheet 48

5. Pertinent Information About Votrax. 66

INTRODUCTION

We have entered a new age: an age where man needs very little human interaction to accomplish great mathematical and engineering feats. It is an age where man and computer have become inseparable units. But as intimate as this relationship may initially appear, it is actually rather superficial for the majority of users. In general, computers are used for the purpose of establishing two-way communication between man and metal. And for most, this relationship will never mature past the point described above.

However, some will enter into a realm of coexistence with the computer experienced by very few. They will discover that man and machine need not be the sole interactors in this world of flesh and metal. Metal can control other metal just as simply as humans can control a single machine. A human can make a computer control another computer or piece of machinery. In short, computers can be interfaced to the real world.

Most university courses succeed in teaching students to "interface" with a single terminal. However, few are exposed to the latter relationship; one where man programs one computer to control another. This research report aims to set up a lab course providing exposure to embedded

computer systems. Such a system consists of a computer, along with its related software. Together, this system monitors the process using sensors and controls the process with actuators. In the lab course, the student will gain hands-on experience in interfacing a computer with various machinery that could easily be transferred to real-world situations.

CHAPTER ONE - COURSE OVERVIEW

The Basics of Interfacing

The applications for embedded computer systems are limited only by the boundaries of human imagination. With a sensor, and an actuator, interfacing becomes trivial. For example, the area of robotics extensively utilizes the computer's interfacing capabilities. More specifically, the automobile industry is an excellent case to address. In this industry, one would be astounded by the number of parts controlled by microprocessors. Today's cars come equipped with electronic carburetors, fuel injection controllers, ignition controllers, transmission system controllers, pollution controllers, temperature sensors and cruise controllers, just to mention a few. The list of examples is endless, but it is obvious that computers are embedded within a great many formerly purely mechanical devices. It is for this reason that a course in interfacing is a vital link in understanding how one machine can control one or more other machines.

Laboratory Set-up

Designing a laboratory supplement to a class centered around embedded computer systems is no minor task. Due to the nature of the subject, and to the limited knowledge of

the average student in the area of interfacing, one must approach the subject in such a way as to avoid the snowballing effect. (One must keep in mind that the only prerequisite for this class is one course in FORTRAN or any higher-level language and a first course in digital circuits.) Yet the labs must be challenging enough to interest the group as well as to serve as a learning tool. With this strategy in mind, the labs are designed as follows: The first two laboratory exercises are trivial operations aimed at familiarizing the student with the equipment. Concentration is placed on the Data Translation LDT2801 board and how it interfaces with the Rainbow 100 computer. The third laboratory introduces the idea of polling. In this lab, the student has the opportunity to interface the Rainbow 100 computer to an automated factory model. Also introduced in this lab is the 8086 microprocessor. As the course continues, the student is exposed to the LDT2801's A/D conversion feature. Then, the communication ports of the Rainbow are explored and the student is given the chance to write an 8086 assembly program sending a string of characters to the Votrax, the speech synthesizer. As a grand finale, two Rainbows are made to communicate via an RS232 cable and then via the LDT2801 board.

Design of the Course

Since a lab is highly dependent upon its accompanying course, a few words are in order concerning the course. The book selected to serve as the foundation for the course in embedded computer systems is called Real Time Programming: Neglected Topics written by Caxton C. Foster (1981). This book offers an overview of the basic concepts of interfacing. It includes the study of peripheral interface adapters (PIAs), multiplexors, semaphores and interrupts. It also explains some of the problems that arise when a physical connection is made between a digital computer and the external world.

The book approaches the subject of interfacing in an extremely informal manner. The material presented is both interesting and applicable to everyday life. The author does not spend a lot of time on a single subject. Instead, he chooses to cover much ground with few words. The book is very successful in giving the reader a peak at various aspects of interfacing. However, it is difficult to use as a classroom text because it only introduces the student to a wide variety of subjects. It whets the appetite, yet it does not elaborate on any subject, thereby leaving the serious student famished for additional information. Therefore, the lecture sessions of the course should encourage the critical thinking of the student by including

supplemental material.

A university level course in embedded computer systems should include the vital subject of hardware/software interaction. It is just as important to be aware of how to connect two terminals as it is to know how the software works. The term "RS232" should not sound foreign to anybody even slightly familiar with interfacing. After all, it is the standard term, set in 1962, used to describe one of the most widely used interfacing mechanisms between computers and peripherals. It is equally frustrating to electrically interface two computers, and be unsuccessful due to insufficient knowledge of the associate software environments.

We live in an analog world. We measure our environment in terms of analog parameters: temperature, pressure, strain, air flow, etc. But if we are going to have any kind of rapport at all with computers, we have to speak their language. They won't learn ours. It is therefore fundamental to have a knowledge of analog to digital (A/D) as well as digital to analog (D/A) converters. The LDT2801 board, used in the labs, and described in Chapter Two of this report, provides exposure to this topic. Last, but not least, this course is not complete without a discussion of interface components. Bruce A. Artwick , in his book entitled, Microcomputer Interfacing (1980), does a fine job describing driver circuits, receiver circuits, input/output

integrated circuits, and high powered interface circuits, to mention a few. In addition, the labs included in the appendix, give a feel for the versatility of the computer, and serve to integrate the topics presented in class to situations found in the real world.

CHAPTER TWO - THE LDT2801 BOARD

LDT2801's Driver: The Rainbow 100

The Rainbow 100 is a microcomputer manufactured by Digital Equipment Corporation capable of driving the LDT2801 board. It features a dual processor system. Included within the Rainbow are 8086 and Z-80 microprocessors.

Figure 1 presents a block diagram of the hardware. The LDT2801 is connected to the EXT COMM port of the eight bit data bus. The Rainbow used for this research report is equipped with 256k bytes of memory running under CPM/86. Figure 2 shows a diagram of the communications and printer port signals. This figure should be of interest in the solution of the later labs where communication through the printer port and the communication port is going to take place.

LDT2801 Description

The LDT2801 I/O board, manufactured by Data Translation, Inc., is designed to operate with a Digital Rainbow Computer. The board is capable of Analog to Digital (A/D), Digital to Analog (D/A), and Digital input/output (I/O) conversions.

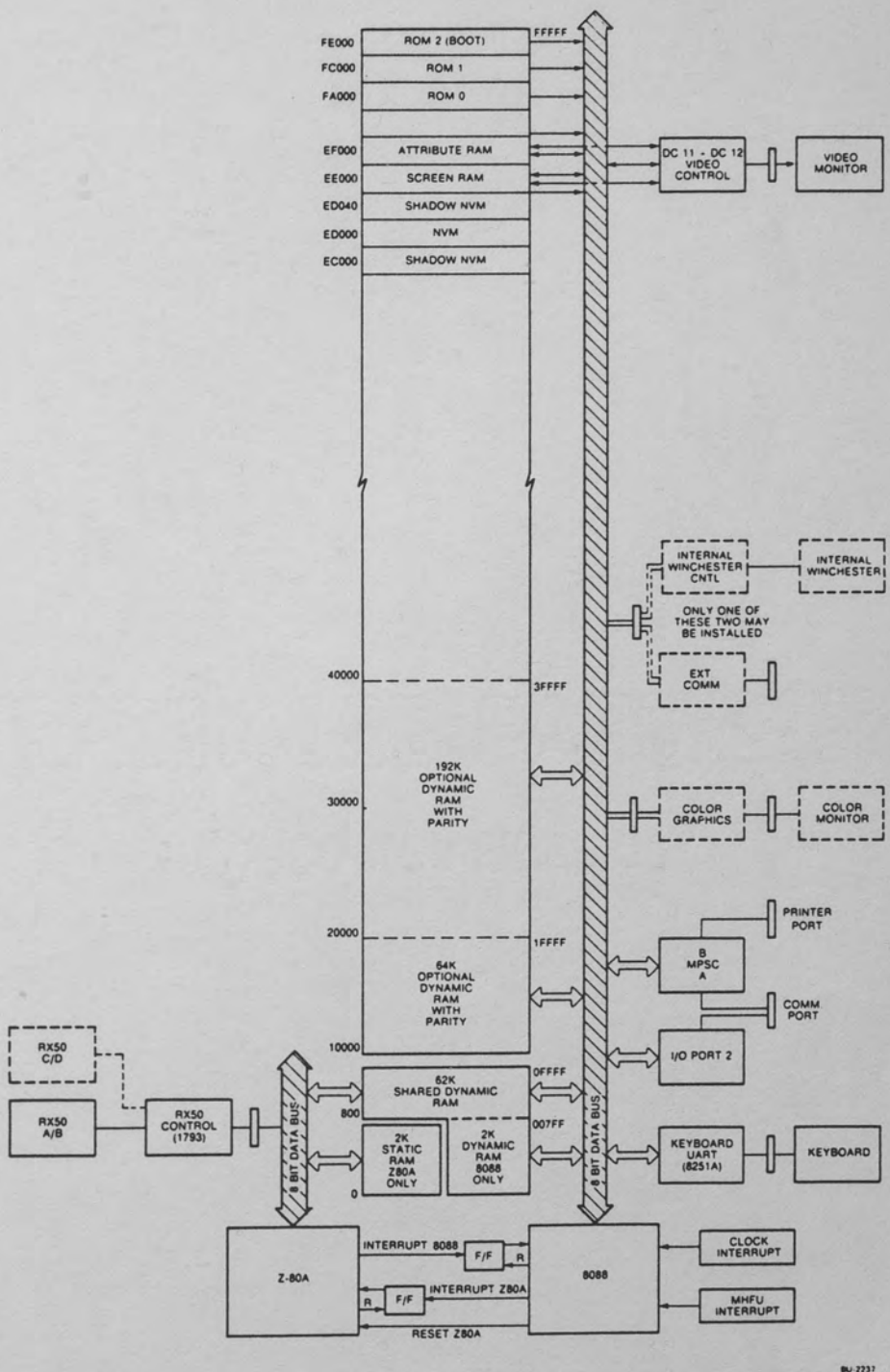


Figure 1. Rainbow 100 System Block Diagram

CP/M-86/80 V2.0 Technical Documentation, (1984)

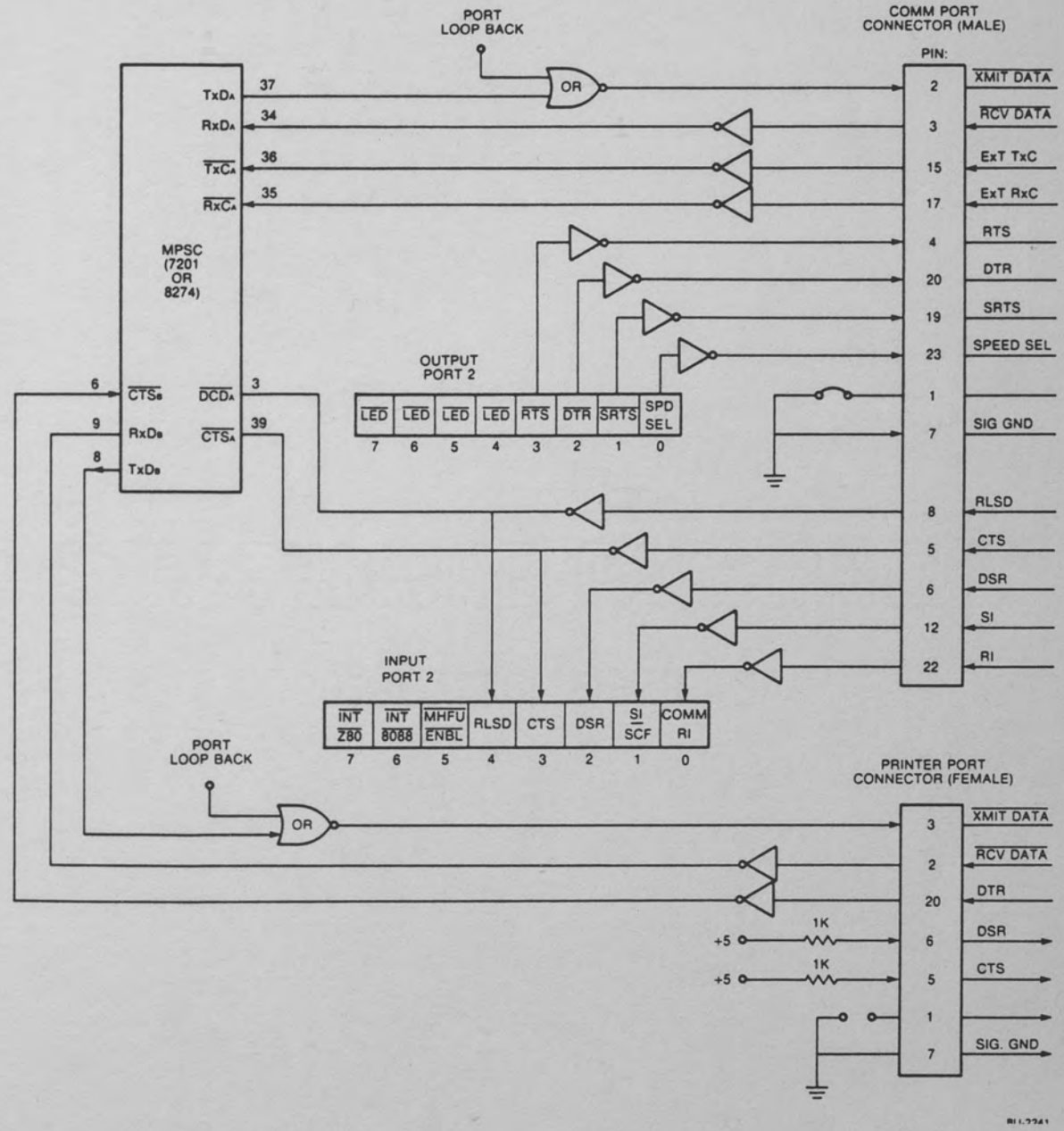


Figure 2. Rainbow 100 Communication and Printer Port Signals

CP/M-86/80 V2.0 Technical Documentation, (1984)

The LDT2801 can be set up in numerous ways. For the purposes of this report, the board is set up as follows:

- o Eight differential input channels for A/D conversions (12 bit resolution)
- o Two output D/A channels (12 bit resolution)
- o Two ports for digital input/output with eight bits each

A more detailed description will be presented within this chapter. For further information pertaining to the configuration of the LDT2801, please refer to the User Manual for the LDT2801/707(1984).

Registers

The LDT2801 is equipped with a series of commands that control the basic routines. The programming languages used are Assembly and Basic.

Four registers are used by the LDT2801.

1. DATA IN:
   - o Receives data from Rainbow
   - o Receives command parameters from Rainbow
   - o Data received originates from D/A, A/D and digital (I/O) converters
   - o Write Only register
   - o Address: Base (where base = 24 Hex for the factory-set boards)

2. DATA OUT:
   - o Contains data to be retrieved from Rainbow
   - o Used to retrieve information pertaining to error bit pattern
   - o Data comes from A/D, D/A, and digital I/O converters
   - o Read Only register
   - o Address: Base

3. COMMAND REGISTER:
   - o Receives command information from Rainbow
   - o LDT2801 has sixteen predefined commands
   - o Lower four bits of register identify commands
   - o Write Only register
   - o Address: Base + 1

4. STATUS REGISTER:
   - o Contains information pertaining to current status of LDT2801
   - o Bits in Status Register are used as flags to communicate error status to the Rainbow
   - o Read Only register
   - o Address: Base + 1

TABLE 1

LDT2801 OPERATING CODE (BITS 0 TO 3 IN COMMAND REGISTER)

Command	Opcode	Modifiers
Reset	0000	
Clear Error	0001	
Read Error Reg.	0010	
Set Internal Clock Period	0011	
Set Digital Port For Input	0100	Ext. Trig*
Set Digital Port For Output	0101	Ext. Trig
Read Digital Input Immediate	0110	Ext. Trig
Write Digital Output Immediate	0111	Ext. Trig
Write D/A Immediate	1000	Ext. Trig
Set D/A Parameters	1001	
Write D/A	1010	Ext. Trig, Ext. Clk.**, Cont.***
Test	1011	Ext. Trig
Read A/D Immediate	1100	Ext.Trig
Set A/D Parameters	1101	
Read A/D	1110	Ext. Trig, Ext. Clk, Cont.
Stop Operation	1111	

* External Trigger: A pulse from an outside source that is sent to the LDT2801 and serves as an instruction set commanding the board to enact a specified set(s) of command(s).

** Clock: An electrical pulse initiating repetitive data conversions in Block commands.

*** Continuous Mode: Upon issuing a command, the board will continue to generate data in the absence of any additional commands.

TABLE 2

LDT2801 READ ERROR BITS AND EXPLANATION

Bit	Name and explanation
0*-	Reserved: Not used.
1 -	Command overwrite: A new command was issued before completing execution of the previous command.
2 -	Clock set: During a Set Internal Clock Period command a value of 0 or 1 was attempted to be written to the Data In Register.
3 -	Digital Port Select: Only valid entries are 0 (port 0),1 (port 1),or 2 (both ports).
4 -	Digital Port Select: A read was attempted on a port set for output or a write was attempted on a port set for input.
5 -	DAC Select: Only legal parameters are 0 (DAC0), 1 (DAC1), and 2 (Both DAC0 and DAC1).
6 -	DAC clock: The clock rate is too high or too low.
7 -	DAC #Conversions Value: Legal values are 3 to 1000.
8 -	A/D Channel: Legal parameters are 0 to 8 for differential operation and 0 to 15 for single-ended operation.
9 -	A/D Gain: Legal parameters are 0, 1, 2, or 3.
10-	A/D clock: The clock rate is too high or too low.
11-	A/D Multiplexer: The clock rate is too high and the A/D channel multiplexer dose not have enough settling time.
12-	A/D #Conversion: Legal values are 3 to 1000.
13-	Data: Dat was written to the Data in Register, but no previous command was issued.

*READ ERROR REGISTER DATA BITS

Data Out Low Byte:

7	6	5	4	3	2	1	0

Data Out High Byte:

15	14	13	12	11	10	9	8

TABLE 3

LDT2801 COMMAND SEQUENCE

Command	Data 1	Data 2	Data 3	Data 4
Reset				
Clear Error				
Read Error Reg.			Data L	Data H
Set Internal Clock Period				
Set Digital Port For Input	Port#			
Set Digital Port For Output	Port#			
Read Digital Input Immediate	Port#		Data	[Data P1]
Write Digital Output Immediate	Port#		Data	Data P1
Write D/A Immediate	Gain	Dac#	Data L	Data H
Set D/A Parameters	Gain Con# L	Dac# Con# H		
Write D/A	Dac# Data H [Data D1L]	Data L [Data D1L]		
Test				
Read A/D Immediate			Data L	Data H
Set A/D Parameters	Gain End# Con# H	Start# Con# L		
Read A/D			Data L	Data H
Stop Operation				

TABLE 4

EXPLANATION OF SYMBOLS USED IN TABLE 3

Symbol	Explanation
Data 1	Data in register
Data 2	Data in register
Data 3	Data out register
Data 4	Data out register
Data L	Low byte of the data
Data H	High byte of the data (A/D or D/A: 4 lower bits)
Port#	Digital port 0 or 1. If a 2 is specified both are selected.
Data	Data read from the Data Out Register
[Data P1]	Optional. If both ports have been selected, then the second data read from the Data Out Register corresponds to port 1.
Gain	The two bit descriptor corresponds to the following gains: byte value:gain, 0:1, 1:2, 2:4, 3:8.
Dac#	Selects the output Dac 0 or 1. If a 2 is specified both are selected.
Con# L	Lower byte of the number of conversions to be performed.
Con# H	High byte of the number of conversions to be performed.
[Data D1L]	Optional. If both ports have been selected, then the second set of data read from the Data Out Register corresponds to Dac 1 (low byte).
[Data D1H]	Optional. If both ports have been selected, then the second set of data read from the Data Out Register corresponds to Dac 1 (high byte).
Start#	The channel number to begin the multiplexing with.
End#	The channel number to end the multiplexing with.

* Note: For a complete documentation of the commands and a more indepth examination, please refer to DT311/LDT2801 User Manual.

Bit Explanations of the Registers

Command Register

Bit Configuration:

7	6	5	4	3	2	1	0

Bits 0 to 3: Opcode. These four bits are reserved for commands. A list of the sixteen commands can be found in Table 1.

Bit 4. Always "0". Not used for any command but should always be written as a zero to insure error-free performance.

Bit 5. Continuous. Can only be a "1" when used in conjunction with READ A/D and WRITE D/A. This parameter, when set to "1," will cause the previous two commands to execute in a continuous matter until a STOP command has been issued. For all other commands, excluding the above-mentioned one, this bit should be a "0." Failure to set this bit as specified may cause the LDT2801 to enter an undefined state.

Bit 6. External clock. Can only be a "1" when used in conjunction with READ A/D and WRITE D/A. This parameter when set to "1" will cause the previous commands to synchronize with an external clock pulse.

Bit 7. External Trigger. When set, this bit enables the synchronization of the command to an external trigger. The LDT2801 merely sets up the command. However, it

will not execute the command unless it is instructed to do so via the external trigger.

Status register:

Bit Configuration:

7	6	5	4	3	2	1	0

Bit 0. Data Out Ready. Flags the Rainbow that valid data is present at the data register. This data can come from a digital source or an analog one. After the data register is read, the bit is cleared. This bit should always be checked before reading the data register. The data register should be read after a power up in order to empty the register and be sure that no irrelevant data is present.

Bit 1. Data In Full. This bit indicates that the LDT2801 has not yet processed the available information at the data register. It can also indicate that a command byte is present in the command register. This byte should always be checked before writing to the Data In Register. If new data is written to the register while Bit 1 is still set, the previous information will be deleted.

Bit 2. Ready. Used as a flag to indicate when the LDT2801 is ready to accept a new command. In the event a new command is given before a previous one has been carried out, a Command Overwrite Error will result.

Bit 3. Command. Indicates to which register the last byte was written. A "1" implies the last byte was written to the Command Register, while a "0" indicates the last byte was written to the Data Register.

Bits 4-6. Not used. These bits are always read back as zeroes.

Bit 7. Composite error. Flags the occurrence of an error. Executing the Read Error Command will determine the nature of the error. Bit 7 will remain set until a Reset or Clear Error command has been issued.

Data In Register and Data Out Register

Data Out Low Byte configuration:

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Data Out High Byte configuration:

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

Bits 0 to 7 (data byte 1, low byte). Contains the low bits in a twelve bit or sixteen bit data transfer. It is the complete byte in a eight bit transfer. This is the first byte read or written from/to the Data Registers.

Bits 8 to 15 (data byte 2, high byte). Contains the high bites in a twelve bit or sixteen bit data transfer. If it is a twelve bit transfer, bits 4 to 7 are "0." This is the second byte read or written from/to the Data Registers.

Command Sequencing

Tables 3 and 3A provide a summary of the necessary parameters needed to perform a given command. Below is a short outline explaining how to set up the commands in the LDT2801.

1. Check Status Register until Ready bit (bit 2) is set.
2. Write the command to the Command Register.

   If no parameters are needed go to step 6.

3. Check Status Register until Data In Full (bit 1) is not set.
4. Write the parameter needed to the Data In Register.
5. If more parameters are needed to set up the command repeat steps 2 and 3 until exhaustion of the parameters.
6. If no output is expected then go to step 10.
7. Check Status Register until Data Out Ready bit (bit 0) is set.
8. Read the data from the Data Out Register.
9. If more data is needed to be read then repeat steps 7 and 8 until exhaustion of the data.

10. Command executed in completion.

If the command is set in continuous mode, it will be necessary to issue a Stop command. In this case, the LDT2801 reads or writes continuously and will not stop unless an error occurs or a Stop command is issued.

CHAPTER THREE -LAB COURSE LIMITATIONS AND POSSIBLE SOLUTIONS

The laboratory material presented in this paper has a few inherent limitations. For one, it assumes that one is pretty familiar with the Rainbow 100 computer. For those few who have been exposed to the Rainbow before, this laboratory does much to expand their horizons. Having already secured a strong foundation, this user can more readily appreciate the concept of interfacing the Rainbow to an LDT2801 board. However, a newcomer to both the Rainbow 100 and the LDT2801 is faced with the task of not only learning about the board, but of becoming familiar with the Rainbow as well. Although this does place the more inexperienced students at a slight disadvantage, the benefits derived from this course will more than compensate for the extra time and effort that some had to endure. In essence, both groups, the experienced as well as the novices, should leave this course with a pretty good understanding of the interfacing process.

Another problem is that the student is not given the opportunity to build the interface hardware. All the peripherals and interface mechanisms are set up prior to the laboratory sessions. This limits the individual by forcing him/her to accept the interface as it is built. One cannot alter the way it functions. A course that would

stress designing drivers for motors and other devices would remedy this deficiency.

Another limitation is one that is inherent in the LDT2801 board. The LDT2801 does not feature interrupts. For the case addressing independent and simultaneous motion of various motors, this poses a problem because it ties up the processor and does not allow it to do multiple tasks. In the automated factory model, for example, the only means of telling when a motor has reached its destination is by polling a switch.

Fortunately, there are a few options inherent in the LDT2801 that can serve as alternatives to interrupts. An external trigger is built into the system. This places the processor in a wait state. However, it does not free the processor to perform other tasks. An alternative would be to connect all the switches to the external trigger. This operation would set up the motion of the next motor and then wait until the current motor triggers the board by depressing the corresponding switch. While the present motor is turned off, the next one would be activated. With this method, one could control up to sixteen motors. Yet another way to increase the capacity of the LDT2801 is to multiplex the input and decode the output of the board. This would give the board 512 control outlets for the input and output combined or 256 outlets each. But this last option does have one serious limitation. It would

require the LDT2801 to drive a large TTL (transistor-transistor logic) load. This cannot be done as the maximum allowable load is 24 milliamps per output port or an equivalent of 30 standard TTL loads. This merely illustrates that one can get around the requirement for interrupts. However, this detour may be a little rough and a bit bumpy. But it will work.

Another problem placing constraints on the lab course centers around the problem of documentation. Obtaining basic information from the Rainbow manual is, at times, next to impossible. These manuals were not written with the average student in mind. They were written for someone that has experience using the Rainbow. Many assumptions are made, the biggest being that nobody but an experienced Rainbow needs to use the manual. This is a problem with documentation in general. It is written by an expert who takes for granted what most of us would appreciate seeing on paper. Unfortunately, there is not a quick and easy solution to this problem.

In general, even though the course does have its limitations, they are not so great as to inhibit progress. Roadblocks are there; however, it is up to the student to be merely aware of them, and know how to avoid running into them.

CONCLUSION

This research report presented both the attributes and the limitations of the laboratory course designed. It grew out of a need for a laboratory class that would accompany and complement a class in embedded computer systems. This need was not merely a local one limited to the University of Central Florida. The need for personnel familiar with interfacing is one that exists in the real world as well. Automated processes are all too common in the real world and unfortunately, few people are ever exposed to interfacing computers to other computers and to various peripherals. The limited exposure obtained in the lab hopefully taught the students the basics of interfacing. Perhaps, a handful will be inspired to continue their exploration of the vast world of interfacing.

APPENDIX

LAB # 1

DIGITAL INPUT/OUTPUT

OBJECTIVE

- Introduction to the LDT2801 board digital input/output capacities.
- Familiarization with the CP/M-86 operating system, the source code and the LDT2801 commands
- Review of programming in Basic

EQUIPMENT

- Rainbow 100 computer
- LDT2801 Interface board
- Heath Kit Digital experimenter
- Necessary wire

SETUP

Connect the hardware as follows

- Port 0 of the LDT2801 to the Led lights on the Heath Kit Digital Experimenter board
- Port 1 of the LDT2801 to the switches on the Heath Kit Digital Experimenter board.

* Make sure that the connection order of the switches correspond to the connection order of the LEDs.

PROCEDURE

Write a program in Basic to monitor the status of the four input switches and to display any changes on the LED.

The program should perform as follows:

1. Stop, clear and reset the LDT2801
2. Set the digital output/input ports on the LDT2801
3. Read the input port
4. Write to the output port the information read from the input port
5. Repeat steps 3 and 4 in an endless loop

LAB # 1 SOLUTION

```
100 '' Program designed by: Alexander Nikoloff
110 '' Date created: 19 may 1986
120 ''
130 DEFINT A-Z
140 BASE.ADD       =&H20
150 COMMAND.REG     =BASE.ADD + 1
160 STATUS.REG      =BASE.ADD + 1
170 DATA.REG        =BASE.ADD
180 COMMAND.WAIT    =&H4
190 WRITE.WAIT      =&H2
200 READ.WAIT       =&H5
210 ''
220 PORT.0          =&H0
230 PORT.1          =&H1
240 CCLEAR          =&H1
250 CERROR          =&H2
260 CSOUT           =&H5
270 CSIN            =&H4
280 CSTOP           =&HF
290 CDOUT           =&H7
300 CDIN            =&H6
310 ''
320 '' STOP AND CLEAR THE BOARD
330 OUT COMMAND.REG, CSTOP
340 TEMP = INP(DATA.REG)
350 WAIT STATUS.REG, COMMAND.WAIT
360 OUT COMMAND.REG, CCLEAR
370 PRINT "DT2801 IS STOPPED AND CLEARED
380 ''
390 ''SET PORT 1 FOR OUTPUT
400 ''
410 WAIT STATUS.REG, COMMAND.WAIT
420 OUT COMMAND.REG, CSOUT
430 WAIT STATUS.REG, WRITE.WAIT, WRITE.WAIT
440 OUT DATA.REG, PORT.1
450 ''
460 ''SET PORT 0 FOR INPUT
470 ''
480 WAIT STATUS.REG, COMMAND.WAIT
490 OUT COMMAND.REG, CSIN
500 WAIT STATUS.REG, WRITE.WAIT, WRITE.WAIT
510 OUT DATA.REG, PORT.0
520 ''
530 ''READ PORT 0
540 ''
550 WAIT STATUS.REG, COMMAND.WAIT
560 OUT COMMAND.REG, CDIN
570 WAIT STATUS.REG, WRITE.WAIT, WRITE.WAIT
580 OUT DATA.REG, PORT.0
590 WAIT STATUS.REG, READ.WAIT
```

```
600 BYTES = INP(DATA.REG)
610 ''
620 ''WRITE PORT 1
630 ''
640 WAIT STATUS.REG, COMMAND.WAIT
650 OUT COMMAND.REG, CDOUT
660 WAIT STATUS.REG,WRITE.WAIT, WRITE.WAIT
670 OUT DATA.REG, PORT.1
680 WAIT STATUS.REG, WRITE.WAIT, WRITE.WAIT
690 OUT DATA.REG, BYTES
700 GOTO 530
710 END
```

LAB # 2

DIGITAL INPUT/OUTPUT

OBJECTIVE

- Continued familiarization with the CP/M-86 operating system
- Introduction to the 8086 Assembler
- Operating the LDT2801 in Assembly language
- Demonstrate the use of an external trigger

EQUIPMENT

- Rainbow 100 computer
- LDT2801 Interface board
- Heath Kit Digital experimenter
- Necessary wire

SETUP

Connect the hardware as follows:

- Connect external trigger to one of the momentary switches on the Heath Kit Digital Experimenter board
- Connect Port 0 of the LDT2801 to the LEDs (lights) on the Heath Kit Digital Experimenter board
- Connect Port 1 of the LDT2801 to the switches on the Heath KIt Digital Experimenter board. Be sure to note that the order of switch connections corresponds to the order of LED connections.

PROCEDURE

Write a program to monitor the status of the four input switches and to display any changes on the LEDs. The user should be able to operate with or without a trigger. If trigger mode is selected the LEDs should only change after the external trigger is enabled. If the trigger is not enabled, there should be no change in the status of the LEDs.

Write an Assembly program that does the following:

1. Stop, clear and reset the LDT2801
2. Set the digital output/input ports on the LDT2801
3. Ask user if he/she wants to wait for an external trigger
4. Read the input port (wait for trigger if set)

5. Write to the output port the information read from the input port
6. Repeat steps 4 and 5 in a endless loop

LAB # 2 SOLUTION

```
1:
2: ; LAB #2
3: ; DESIGNED BY: ALEXANDER NIKOLOFF
4: ;
5: ; MEETS THE SPECIFIED REQUIREMENTS FOR LAB #2
6: ;
7:          CSEG
8:          ORG     100H
9: ALEX:    MOV     AL,CSTOP        ;STOP THE LDT2810
10:         OUT     CREG,AL
11:         IN      AL,DREG         ;CLEAR THE DATA REGISTER
12:         MOV     BX,CWAIT
13:         MOV     DX,SREG
14:         CALL    WAITT
15:         MOV     AL,CCLEAR       ;CLEAR THE LDT2810
16:         OUT     CREG,AL
17:
18:         MOV     BX,CWAIT
19:         MOV     DX,SREG
20:         CALL    WAITT
21:         MOV     AL,CSOUT        ;SET THE DIGITAL OUTPUT PORT
22:         OUT     CREG,AL
23:         MOV     BL,WWAIT
24:         MOV     BH,WWAIT
25:         MOV     DX,SREG
26:         CALL    WAITT
27:         MOV     AL,PORT1        ;PORT # 1 SET FOR OUTPUT
28:         OUT     DREG,AL
29:
30:         MOV     BX,CWAIT
31:         MOV     DX,SREG
32:         MOV     DX,SREG
33:         CALL    WAITT
34:         MOV     AL,CSIN         ;SET DIGITAL PORT FOR INPUT
35:         OUT     CREG,AL
36:         MOV     BL,WWAIT
37:         MOV     BH,WWAIT
38:         MOV     DX,SREG
39:         CALL    WAITT
40:         MOV     AL,PORT0        ;PORT # 0 SET FOR INPUT
41:         OUT     DREG,AL
42:
43:         MOV     CL,9            ;DISPLAY PROMPT TO CRT
44:         MOV     DX,300H
45:         INT     224
46:         MOV     CL,1            ;GET ANSWER FROM USER
47:         INT     224
48:         MOV     TEMP,CDIN
49:         XOR     AL,'Y'
50:         JE      TRIGGER
```

```
51:         XOR     AL,'y'
52:         JNE     AGAIN
53: TRIGGER:MOV     TEMP,TRIG       ;SET UP LDT2810N COMMAND IN TEMP
54: AGAIN:
55:         MOV     BX,CWAIT
56:         MOV     DX,SREG
57:         CALL    WAITT
58:         MOV     AL,TEMP         ;READ PORT 0 WAIT FOR
59:         OUT     CREG,AL         ;TRIGGER IF APPLICABLE
60:         MOV     BL,WWAIT
61:         MOV     BH,WWAIT
62:         MOV     DX,SREG
63:         CALL    WAITT
64:         MOV     AL,PORT0
65:         OUT     DREG,AL
66:         MOV     BX,RWAIT
67:         MOV     DX,SREG
68:         CALL    WAITT
69:         IN      AL,DREG
70:         PUSH    AX
71:
72:         MOV     BX,CWAIT
73:         MOV     DX,SREG
74:         CALL    WAITT
75:         MOV     AL,CDOUT        ;OUTPUT THE READ INFORMATION
76:         OUT     CREG,AL         ;TO PORT 1
77:         MOV     BL,WWAIT
78:         MOV     BH,WWAIT
79:         MOV     DX,SREG
80:         CALL    WAITT
81:         MOV     AL,PORT1
82:         OUT     DREG,AL
83:         MOV     BL,WWAIT
84:         MOV     BH,WWAIT
85:         MOV     DX,SREG
86:         CALL    WAITT
87:         POP     AX
88:         OUT     DREG,AL
89:         MOV     CL,11           ;CHECK FOR CONSOLE INPUT
90:         INT     224             ;EXIT IF PRESENT
91:         TEST    AL,0FFH         ;IF TRIGGER IS PRESENT,
92:         JE      AGAIN           ;EXECUTION WILL TERMINATE
93:         MOV     CL,0            ;AFTER A TRIGGER
94:         MOV     DL,0
95:         INT     224
96:
97:
98:
99: ;SUBROUTINE WAITT
100:
```

```
;     PARAMETERS: DX=PORT
;                 BL=BIT PATTERN (AND)
;                 BH=BIT PATTERN (XOR)

;     DESCRIPTION:READS THE SPECIFIED OUTPUT PORT IN DX
;                 UNTILL THE SPECIFIED BIT IN BL IS OBTAINED
;                 BH IS XOR'ED WITH THE PORT DATA AND THEN
;                 AND'ED WITH BL. IF THE RESULT IS NON ZERO
;                 WAIT IS COMPLETED.
;
;     RETURNS:    ALL REGISTERS IN ORIGINAL FORM
;
WAITT:  PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSHF
        MOV     CX,BX
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        SHR     CX,1
        AND     BX,00FFH
WAIT1:  IN      AL,DX
        XOR     AL,CL
        AND     AL,BL
        JE      WAIT1
        POPF
        POP     CX
        POP     BX
        POP     AX
        RET

        DSEG
        ORG     300H
BASE    EQU     20H
CREG    EQU     BASE+1
SREG    EQU     BASE+1
DREG    EQU     BASE
CWAIT   EQU     4H
WWAIT   EQU     2H
RWAIT   EQU     5H
PORT0   EQU     0H
PORT1   EQU     1H
CRESET  EQU     0H
CCLEAR  EQU     1H
CERROR  EQU     2H
```

```
151: CSOUT   EQU     5H
152: CSIN    EQU     4H
153: CSTOP   EQU     0FH
154: CDOUT   EQU     7H
155: CDIN    EQU     6H
156: TRIG    EQU     86H
157: PROMPT  DB      'DO YOU WANT EXTERNAL TRIGGER? Y/N $'
158: TEMP    RB      1
159:         END
```

LAB # 3

AUTOMATED FACTORY

OBJECTIVE

- Familiarization with the idea of polling
- Exposure to the operation of a simple assembly line

EQUIPMENT

- Rainbow 100 computer
- LDT2801 Interface board
- Heath Kit Digital experimenter
- Automated factory model (Fisher Technics)
- Interface from the model to the Rainbow
- Necessary wire

SETUP

The model will already be set up upon arrival to lab. It was built with Fisher Technics components. The model consists of a series of switches used to determine the position of the motors. Switches one and eight are photocells. For more information, please see attached diagrams.

The connections are as follows:

LDT2801		MODEL
Digital I/O Port 0		
BIT 0	----------	SW1
BIT 1	----------	SW2
BIT 2	----------	SW3
BIT 3	----------	SW4
BIT 4	----------	SW5
BIT 5	----------	SW6
BIT 6	----------	SW7
BIT 7	----------	SW8
DIGITAL I/O Port 1		
BIT 0	----------	M1F
BIT 1	----------	M1B
BIT 2	----------	M2F
BIT 3	----------	M2B

BIT 4 ---------- M3F
BIT 5 ---------- M3B
BIT 6 ---------- M4
BIT 7 ---------- VIOLET LIGHTS

Note* A "1" denotes that the switch has been depressed. The photocells work the opposite way.

PROCEDURE

Write a program using 8086 Assembly to control the motion of the material-handling model.

The program should perform and control the following operations:

1. Initialize the model. (Set all motors and arms to starting position.)
2. Wait for Start key to be pressed.
3. If the material is not on the carrier, display a message indicating this. (ex., "Material is not ready."). Go to step 2.
4. If the material is on the carrier, the carrier should be advanced to the transfer point.
5. Transfer the material from the carrier to the conveyor.
6. Move the material to the end of the line.
7. Turn all motors off and go to step 1.
8. Hazardous motion should be interrupted by pressing the "space bar," on the keyboard.
9. "Space bar" routine should perform the following operation:
    a. Turn off all the motors.
    b. Turn on the Violet lights.
    c. Display a message (ex., "System is not in normal condition").
    d. Go to step 1.

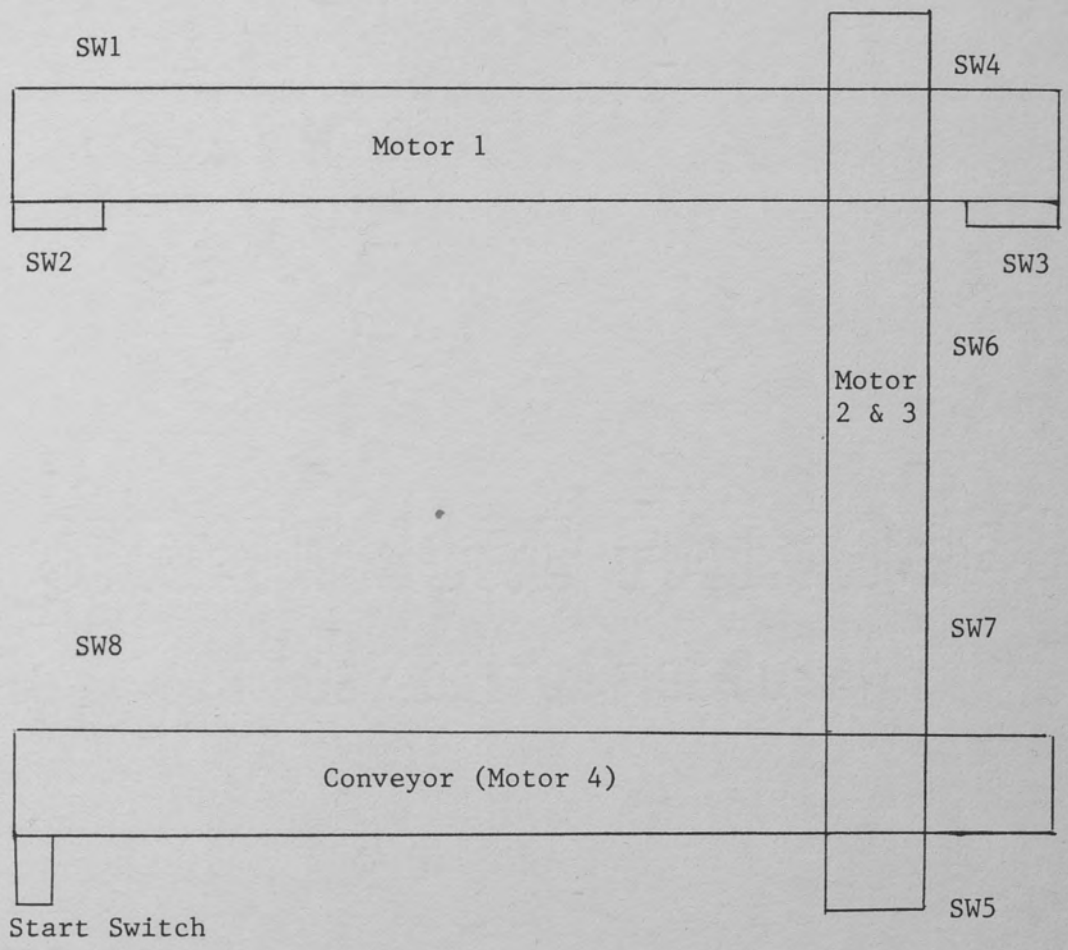


Figure 3. Diagram of Automated Factory Setup.

LAB # 3 SOLUTION

```
1: ;LAB#3
2: ;DESIGNED BY: ALEXANDER NIKOLOFF
3: ;
4: ;MEETS THE REQUIREMENTS FOR LAB#3
5: ;MATERIAL HANDLING MODEL
6:
7:
8:
9:          CSEG
10:         ORG     100H
11:
12: ;SET UP THE LDT2801
13:
14: ALEX:   MOV     AL,CSTOP        ;STOP THE LDT2801
15:         OUT     CREG,AL
16:         IN      AL,DREG         ;CLEAR THE DATA REGISTER
17:         CALL    COMMANDWAIT
18:         MOV     AL,CCLEAR       ;CLEAR THE LDT2810
19:         OUT     CREG,AL
20:
21:         CALL    COMMANDWAIT
22:         MOV     AL,CSIN         ;SET THE DIGITAL INPUT  PORT
23:         OUT     CREG,AL
24:         CALL    WRITEWAIT
25:         MOV     AL,PORT0        ;PORT # 0 SET FOR INPUT
26:         OUT     DREG,AL
27:
28:         CALL    COMMANDWAIT
29:         MOV     AL,CSOUT        ;SET THE DIGITAL OUTPUT PORT
30:         OUT     CREG,AL
31:         CALL    WRITEWAIT
32:         MOV     AL,PORT1        ;PORT #  1 SET FOR OUTPUT
33:         OUT     DREG,AL
34:
35:
36:
37: MOVEMATERIAL:
38:
39: INITIALIZATION:         ;INITIALIXE THE POSITION OF ALL THE MOTORS
40:
41:         MOV     BH,M1B
42:         MOV     BL,SW2
43:         CALL    MM
44:
45:         MOV     BH,M2B
46:         MOV     BL,SW4
47:         CALL    MM
48:
49:         MOV     BH,M3B
50:         MOV     BL,SW7
```

```
51:          CALL    MM
52:
53:          MOV     CL,9            ;DISPLAY PROMPT TO CRT
54:          MOV     DX,OFFSET MESS1 ;"SYSTEM READY"
55:          INT     224
56:
57: NOGOOD:
58:
59:          CALL    COMMANDWAIT
60:          MOV     AL,CDIN         ;SET UP COMMAND AND WAIT FOR TRIGGER
61:          OR      AL,TRIG         ; START SWITCH
62:          OUT     CREG,AL
63:          CALL    WRITEWAIT
64:          MOV     AL,PORT0
65:          OUT     DREG,AL
66:          CALL    READWAIT
67:          IN      AL,DREG
68:          AND     AL,SW1
69:          JE      GOODM
70:          MOV     CL,9            ;DISPLAY PROMPT TO CRT
71:          MOV     DX,OFFSET MESS2 ;"NO MATERIAL PRESENT:"
72:          INT     224
73:          JMP     NOGOOD
74:
75: GOODM:   MOV     BH,M1F          ;START MOTION
76:          MOV     BL,SW3
77:          CALL    MM
78:
79:          MOV     BH,M3F
80:          MOV     BL,SW6
81:          CALL    MM
82:
83:          MOV     BH,M2F
84:          MOV     BL,SW5
85:          CALL    MM
86:
87:          MOV     BH,M3B
88:          MOV     BL,SW7
89:          CALL    MM
90:
91:          MOV     BH,M4           ;SEPARATE RUTINE NEEDED
92:          MOV     BL,SW8          ;BECAUSE OF INVERSE FUNCTIONING
93:          CALL    COMMANDWAIT     ;OF THE PHOTOCELS
94:          MOV     AL,CDOUT
95:          OUT     CREG,AL
96:          CALL    WRITEWAIT
97:          MOV     AL,PORT1
98:          OUT     DREG,AL
99:          CALL    WRITEWAIT
100:         MOV     AL,BH
```

```
101:          OUT     DREG,AL
102:
103: XX1:     CALL    TESTKB
104:          CALL    COMMANDWAIT
105:          MOV     AL,CDIN         ;READ INPUT PORT
106:          OUT     CREG,AL         ;AND POLE UNTILL SWITCH
107:          CALL    WRITEWAIT       ;IS DEPPRESED
108:          MOV     AL,PORT0
109:          OUT     DREG,AL
110:          CALL    READWAIT
111:          IN      AL,DREG
112:          XOR     AL,0FFH
113:          AND     AL,BL
114:          JE      XX1
115:
116:          JMP     INITIALIZATION
117:
118:
119: MOTORMOTION:
120: MM:
121: ;        PARAMETERS BH:MOTOR#AND DIRECTION
122: ;                   BL:SWICH# TO POLE
123: ;        EXIT CONDITION: IF KEBOARD IS DEPRESED
124:
125:
126:          CALL    COMMANDWAIT
127:          MOV     AL,CDOUT        ;MOVE MOTOR
128:          OUT     CREG,AL         ;
129:          CALL    WRITEWAIT
130:          MOV     AL,PORT1
131:          OUT     DREG,AL
132:          CALL    WRITEWAIT
133:          MOV     AL,BH
134:          OUT     DREG,AL
135:
136: X1:      CALL    TESTKB
137:          CALL    COMMANDWAIT
138:          MOV     AL,CDIN         ;READ INPUT PORT
139:          OUT     CREG,AL         ;AND POLE UNTILL SWITCH
140:          CALL    WRITEWAIT       ;IS DEPPRESED
141:          MOV     AL,PORT0
142:          OUT     DREG,AL
143:          CALL    READWAIT
144:          IN      AL,DREG
145:          AND     AL,BL
146:          JE      X1
147:
148: KILLIT:  CALL    COMMANDWAIT     ;SEQUENSE TO STOP ALL MOTORS
149:          MOV     AL,CDOUT
150:          OUT     CREG,AL
```

```
151:          CALL    WRITEWAIT
152:          MOV     AL,PORT1
153:          OUT     DREG,AL
154:          CALL    WRITEWAIT
155:          MOV     AL,KILL
156:          OUT     DREG,AL
157:
158:          RET
159:
160: ;RUTINE THAT POLES THE KEYBOARD
161:
162: TESTKB:
163:          PUSH    BX              ;AN  ENTRY THEN EXIT
164:          PUSH    CX
165:          PUSHF
166:          MOV     CL,11
167:          INT     224
168:          TEST    AL,0FFH
169:          JNE     HOME1
170:          POPF
171:          POP     CX
172:          POP     BX
173:          RET
174: HOME1:   POP     CX
175:          POP     BX
176:          POP     AX
177:
178:          CALL    COMMANDWAIT     ;SEQUENSE TO STOP ALL MOTORS
179:          MOV     AL,CDOUT        ;AND TURN ON THE LIGHTS
180:          OUT     CREG,AL
181:          CALL    WRITEWAIT
182:          MOV     AL,PORT1
183:          OUT     DREG,AL
184:          CALL    WRITEWAIT
185:          MOV     AL,KILL
186:          OR      AL,80H
187:          OUT     DREG,AL
188:
189:          MOV     CL,01H          ;CLEANUP THE BUFFER
190:          INT     224
191:
192:          MOV     CL,9            ;DISPLAY PROMPT TO CRT
193:          MOV     DX,OFFSET MESS3
194:          INT     224
195:
196: HOME2:   MOV     CL,01           ;GET RESPONSE FROM USER
197:          INT     224
198:          TEST    AL,'Y'
199:          JE      HOME3
200:          TEST    AL,'y'
```

```
201:            JE      HOME3
202:
203: HOME:
204:            CALL    KILLIT          ;KILL ALL MOTORS AND LIGHTS
205:            MOV     CL,0            ;EXIT TO CPM
206:            MOV     DL,0
207:            INT     224
208:
209: HOME3:     JMP     INITIALIZATION
210:
211:
212: ;SUBROUTINES TO CONTROLL THE LDT2801
213:
214:
215: COMMANDWAIT:      ;POLES THE STATUS OF THE LDT2810, AND RETURNS
216:                    ;WHEN IT IS READY TO ACCEPT A COMMAND
217:            PUSH    AX
218:            PUSHF
219: WAITC:     IN      AL,SREG
220:            AND     AL,CWAIT
221:            JE      WAITC
222:            POPF
223:            POP     AX
224:            RET
225:
226: WRITEWAIT:       ;POLES THE STATUS OF THE LDT2810, AND RETURNS
227:                    ;WHEN ONE CAN WRITE TO THE DATA REGISTER
228:
229:            PUSH    AX
230:            PUSHF
231: WAITW:     IN      AL,SREG
232:            XOR     AL,WWAIT
233:            AND     AL,WWAIT
234:            JE      WAITW
235:            POPF
236:            POP     AX
237:            RET
238:
239: READWAIT:        ;POLES THE STATUS OF THE LDT2810, AND RETURNS
240:                    ;WHEN ONE CAN READ THE DATA REGISTER
241:
242:            PUSH    AX
243:            PUSHF
244: WAITR:     IN      AL,SREG
245:            AND     AL,RWAIT
246:            JE      WAITR
247:            POPF
248:            POP     AX
249:            RET
250:
```

```
WAIT1:  IN      AL,DX           ;EXTRA RUTINE JUST IN CASE
        XOR     AL,CL
        AND     AL,BL
        JE      WAIT1
        RET

        DSEG
        ORG     300H
BASE    EQU     20H
CREG    EQU     BASE+1
SREG    EQU     BASE+1
DREG    EQU     BASE
CWAIT   EQU     4H
WWAIT   EQU     2H
RWAIT   EQU     5H
PORT0   EQU     0H
PORT1   EQU     1H
PORT2   EQU     2H
CRESET  EQU     0H
CCLEAR  EQU     1H
CERROR  EQU     2H
CSOUT   EQU     5H
CSIN    EQU     4H
CSTOP   EQU     0FH
CDOUT   EQU     7H
CDIN    EQU     6H
TRIG    EQU     80H
SW1     EQU     00000001B       ;PHOTOSENSOR
SW2     EQU     00000010B
SW3     EQU     00000100B
SW4     EQU     00001000B
SW5     EQU     00010000B
SW6     EQU     00100000B
SW7     EQU     01000000B
SW8     EQU     10000000B       ;PHOTOSENSOR
M1F     EQU     00000001B
M1B     EQU     00000010B
M2F     EQU     00000100B
M2B     EQU     00001000B
M3F     EQU     00010000B
M3B     EQU     00100000B
M4      EQU     01000000B
LIGTHS  EQU     10000000B
KILL    EQU     00
MESS2   DB      'NO MATERIAL PRESENT, PLEASE FIX IT      '
        DB      0DH,0AH,0AH,'$'
MESS3   DB      'TRANSPORT INTERRUPTED, ABNORMAL PROCEDURE    '
        DB      0DH,0AH
        DB      'PLEASE PRESS KEYBOARD TO BEGIN EXECUTION AGAIN'
```

```
301:          DB      0DH,0AH
302:          DB      'OR "Y" TO EXIT THE PROGRAM                    '
303:          DB      0DH,0AH,0AH,'$'
304: MESS1    DB      'SYSTEM READY... TO EXIT PROGRAM HIT KEYBOARD'
305:          DB      0DH,0AH,0AH,'$'
306:
307:          END
```

LAB # 4

LCD DISPLAY DRIVER

OBJECTIVE

- To use the acquired knowledge of 8086 Assembler in a more complex situation
- To develop a driver for an LCD display

EQUIPMENT

- Rainbow 100 computer
- LDT2801 Interface board
- Intersil ICM7231/32/33/34 Proto Board
- Five volt power supply
- Necessary connection wire

SETUP

Connect the following on the LDT2801 and INTERSIL:

LDT2801		INTERSIL
Digital I/O Port 1		
BIT 0	----------	D0
BIT 1	----------	D1
BIT 2	----------	D2
BIT 3	----------	D3
BIT 4	----------	D4
BIT 5	----------	D5
DIGITAL I/O Port 0		
BIT 0	----------	A0
BIT 1	----------	A1
BIT 2	----------	CS1,CS2 for ICM 7233AF IC#2
BIT 3	----------	CS1,CS2 for ICM 7233AF IC#1
GRN	----------	GROUND

PROCEDURE

Write a program to control the Intersil Display proto-board. The user should enter a string of valid data and then display this data in a rotating manner. The display board is an eight character display system. The string should be displayed as follows: First

display a blank and then display the characters by rotating them in from the right one-by-one. The string should be displayed continuously in an infinite loop. Set up an assembly program providing a "suave" exit from this loop.

Please read the Specs on the attached page.

The Intersil display works as follows:

1. Set up character on D5,D4,D3,D2,D1,D0
2. Set up the position of the character on A1 and on A0 (See Table 5. Note that each chip handles four positions.) At the same time, set BIT 2 and BIT 3 to a logical zero.
3. Enable the chip that corresponds to the final position by setting BIT 2 or BIT 3 to a logical one.

ICM7231/32/33/34

INTERSIL

ICM7233 PARALLEL INPUT ALPHA DISPLAY

TERMINAL	PIN NO.	DESCRIPTION		FUNCTION	
D0 D1 D2 D3 D4 D5	30 31 32 33 34 35	Least Significant Most Significant	6 Bit (ASCII) Data Inputs	Input Data See Table 4	HIGH = Logical One 1 LOW = Logical Zero 0
A0 A1	37 38	Least Significant Most Significant	Address Inputs	Input Add. See Table 5	
$\overline{CS1}$ $\overline{CS2}$	39 1	Chip Select Inputs		Both inputs LOW load data into input latches. Rising edge of either input causes data to be latched, decoded and sent out to addressed character.	

Table 4

CODE INPUT				DISPLAY OUTPUT D5, D4			
D3	D2	D1	D0	0, 0	0, 1	1, 0	1, 1
0	0	0	0	P	P		0
0	0	0	1	A	Q	!	1
0	0	1	0	B	R	"	2
0	0	1	1	C	S	#	3
0	1	0	0	D	T	$	4
0	1	0	1	E	U	%	5
0	1	1	0	F	V	&	6
0	1	1	1	G	W	'	7
1	0	0	0	H	X	(	8
1	0	0	1	I	Y	)	9
1	0	1	0	J	Z	*	:
1	0	1	1	K	[	+	;
1	1	0	0	L	\	,	<
1	1	0	1	M	]	-	=
1	1	1	0	N	↑	.	>
1	1	1	1	O	←	/	?

DATA DECODING
6 - BIT ASCII—18 SEGMENT
(ICM7233/34)

Table 5

CODE INPUT			DIGIT SELECTED
ICM 7234 ONLY			
A2	A1	A0	
0	0	0	D1
0	0	1	D2
0	1	0	D3
0	1	1	D4
1	0	0	D5
1	0	1	NONE
1	1	0	NONE
1	1	1	NONE

ADDRESS
DECODING
(ICM7233/34)

Figure 4. Intersil Display Data Sheet.

LAB # 4 SOLUTION

```
 1: ;LAB#4
 2: ;DESIGNED BY: ALEXANDER NIKOLOFF
 3: ;
 4: ;MEETS THE REQUIREMENTS FOR LAB#5
 5:
 6:          CSEG
 7:          ORG     100H
 8: ALEX:    MOV     AL,CSTOP        ;STOP THE LDT2810
 9:          OUT     CREG,AL
10:          IN      AL,DREG         ;CLEAR THE DATA REGISTER
11:          CALL    COMMANDWAIT
12:          MOV     AL,CCLEAR       ;CLEAR THE LDT2810
13:          OUT     CREG,AL
14:
15:          CALL    COMMANDWAIT
16:          MOV     AL,CSOUT        ;SET THE DIGITAL OUTPUT PORT
17:          OUT     CREG,AL
18:          CALL    WRITEWAIT
19:          MOV     AL,PORT2        ;PORT # 0 AND 1 SET FOR OUTPUT
20:          OUT     DREG,AL
21:
22:          MOV     CL,9            ;DISPLAY PROMPT TO CRT
23:          MOV     DX,OFFSET MESS1
24:          INT     224
25:          MOV     CL,10           ;GET STRING FROM USER
26:          MOV     DX,OFFSET BUFFER
27:          INT     224
28:
29:          MOV     AL,20H
30:          MOV     SI,1            ;INITIAL INDEX
31:          MOV     BX,OFFSET BUFFER
32:          MOV     [BX],AL
33:          MOV     CH,0
34:          MOV     CL,[BX+SI]
35:          MOV     COUNT,CX        ;COUNT=#OF CHARACTERS+8
36:          ADD     COUNT,8H
37:
38:          MOV     DOUBLE,CX
39:          MOV     [BX+SI],AL
40:          MOV     CX,8H
41:          ADD     DOUBLE,OFFSET BUFSTA-2
42:          INC     SI
43:
44: DUPLI:   MOV     AL,[BX+SI]      ;DUPLICATE THE FIRST 8 CHARACTERS
45:          XCHG    DOUBLE,BX       ;THE END OF THE STRING
46:          MOV     [BX+SI],AL
47:          XCHG    DOUBLE,BX
48:          INC     SI
49:          LOOP    DUPLI
50:
```

```
51:
52: DOITAGAIN:
53:         MOV     BX,OFFSET INIT-1
54:         MOV     CX,COUNT
55:         MOV     SI,0
56: DOIT:
57:         CALL    DISPLAYBUFER
58:         CALL    DELAY
59:         INC     SI
60:
61:         PUSH    SI              ;CKECK THE KEBOARD STATUS. IF
62:         PUSH    BX              ;AN  ENTRY THEN EXIT
63:         PUSH    CX
64:         PUSHF
65:         MOV     CL,11
66:         INT     224
67:         TEST    AL,0FFH
68:         JNE     HOME1
69:         POPF
70:         POP     CX
71:         POP     BX
72:         POP     SI
73:
74:         LOOP    DOIT
75:
76:         MOV     BX,OFFSET BUFSTA
77:         MOV     CX,COUNT        ;LOAD NEW START CONDITION. NO LEADING
78:         SUB     CX,8H           ;SPACES
79:         MOV     SI,0
80:         JMP     DOIT
81: HOME1:  POPF
82:         POP     CX
83:         POP     BX
84:         POP     SI
85:         JMP     HOME
86:
87:
88: DISPLAYBUFER:                   ;DISPLAY THE WINDOW
89:
90:         MOV     TEMP,3H
91:         MOV     CHIP,CHIP2
92:         PUSH    SI
93: NEXT:
94:         INC     SI
95:         CALL    COMMANDWAIT
96:         MOV     AL,CDOUT        ;OUTPUT THE CHARACTER
97:         OUT     CREG,AL         ;TO PORT 1
98:         CALL    WRITEWAIT
99:         MOV     AL,PORT1
100:        OUT     DREG,AL
```

```
101:            CALL    WRITEWAIT
102:            MOV     AL,[BX+SI]
103:            OUT     DREG,AL
104:
105:            CALL    COMMANDWAIT
106:            MOV     AL,CDOUT        ;OUTPUT THE POSITION OF THE
107:            OUT     CREG,AL         ;CHARACTER TO PORT 0
108:            CALL    WRITEWAIT
109:            MOV     AL,PORT0
110:            OUT     DREG,AL
111:            CALL    WRITEWAIT
112:            MOV     AL,TEMP
113:            OUT     DREG,AL
114:
115:            CALL    COMMANDWAIT
116:            MOV     AL,CDOUT        ;OUTPUT THE LACH TRIGGER TO
117:            OUT     CREG,AL         ;PORT 0 AND DISPLAY THE INFORMATION
118:            CALL    WRITEWAIT
119:            MOV     AL,PORT0
120:            OUT     DREG,AL
121:            CALL    WRITEWAIT
122:            MOV     AL,CHIP
123:            OUT     DREG,AL
124:
125:            DEC     TEMP            ;DETERMINE IF ALL 8 CHARACTERS
126:            JGE     NEXT            ;HAVE BEEN DISPLAYED
127:            MOV     TEMP,3H
128:            AND     CHIP,CHIP1
129:            JNE     EXIT
130:            MOV     CHIP,CHIP1
131:            JMP     NEXT
132: EXIT:
133:            POP     SI
134:            RET
135:
136: HOME:
137:            MOV     CL,0            ;EXIT TO CPM
138:            MOV     DL,0
139:            INT     224
140:
141:
142:
143: COMMANDWAIT:       ;POLES THE STATUS OF THE LDT2810, AND RETURNS
144:                    ;WHEN IT IS READY TO ACCEPT A COMMAND
145:            PUSH    AX
146:            PUSHF
147: WAITC:     IN      AL,SREG
148:            AND     AL,CWAIT
149:            JE      WAITC
150:            POPF
```

```
151:          POP     AX
152:          RET
153:
154: WRITEWAIT:       ;POLES THE STATUS OF THE LDT2810, AND RETURNS
155:                  ;WHEN ONE CAN WRITE TO THE DATA REGISTER
156:
157:          PUSH    AX
158:          PUSHF
159: WAITW:   IN      AL,SREG
160:          XOR     AL,WWAIT
161:          AND     AL,WWAIT
162:          JE      WAITW
163:          POPF
164:          POP     AX
165:          RET
166:
167: READWAIT:        ;POLES THE STATUS OF THE LDT2810, AND RETURNS
168:                  ;WHEN ONE CAN READ THE DATA REGISTER
169:
170:          PUSH    AX
171:          PUSHF
172: WAITR:   IN      AL,SREG
173:          AND     AL,RWAIT
174:          JE      WAITR
175:          POPF
176:          POP     AX
177:          RET
178:
179: WAIT1:   IN      AL,DX           ;EXTRA RUTINE JUST IN CASE
180:          XOR     AL,CL
181:          AND     AL,BL
182:          JE      WAIT1
183:          RET
184:
185: DELAY:           ;DELAYS SO ONE CAN READ THE WINDOW
186:
187:          PUSH    CX
188:          PUSHF
189:          MOV     CX,001FFH
190: GO:      PUSH    CX
191:
192: GO1:     MOV     CX,CX
193:          LOOP    GO1
194:          POP     CX
195:          LOOP    GO
196:          POPF
197:          POP     CX
198:          RET
199:
200:          DSEG
```

```
201:          ORG     300H
202: BASE     EQU     20H
203: CREG     EQU     BASE+1
204: SREG     EQU     BASE+1
205: DREG     EQU     BASE
206: CWAIT    EQU     4H
207: WWAIT    EQU     2H
208: RWAIT    EQU     5H
209: PORT0    EQU     0H
210: PORT1    EQU     1H
211: PORT2    EQU     2H
212: CRESET   EQU     0H
213: CCLEAR   EQU     1H
214: CERROR   EQU     2H
215: CSOUT    EQU     5H
216: CSIN     EQU     4H
217: CSTOP    EQU     0FH
218: CDOUT    EQU     7H
219: CDIN     EQU     6H
220: TRIG     EQU     86H
221: CHIP1    EQU     00000100B
222: CHIP2    EQU     00001000B
223: MESS1    DB      'PLEASE INPUT A STRING UP TO 80 CHARACTERS'
224:          DB      0DH,0AH
225:          DB      'INPUT HAS TO BE IN CAPITALS AND A LAGING SPACE'
226:          DB      0DH,0AH
227:          DB      'TO EXIT HIT ANY KEY'
228:          DB      0DH,0AH
229:          DB      '$'
230: INIT     DB      '     '
231: BUFFER   DB      80
232:          RB      1
233: BUFSTA   RW      88
234: TEMP     DB      4
235: DOUBLE   DW      0
236: COUNT    DW      0
237: CHIP     DB      2
238:          END
```

LAB # 5

GENERATE A SINE CURVE FROM DIGITAL DATA

OBJECTIVE

- Familiarization with the D/A feature of the LDT2801
- Generation of a 100-point sine wave on the oscilloscope

EQUIPMENT

- Rainbow 100 computer
- LDT2801 Interface board
- Oscilloscope
- Function generator
- Necessary connection wire

SETUP

- Connect the oscilloscope to the DAC1 screw connection on the LDT2801.
- Connect the function generator to the external clock connection.
- Using the function generator, generate a unipolar square wave with an amplitude of no more than five volts. Any more than this could damage the LDT2801!

PROCEDURE

- Write a program in BASIC that generates an array of 100 points of a sine wave.
- The program should ask the user if an external clock should be used.
- Be sure to adjust the amplitude into the twelve bits of resolution that the LDT2801 can handle. Also remember that since the twelve bits are separated into two bytes, the sine wave needs an upper and lower value.
- Display the sine wave on the oscilloscope.
- Output the sine wave using the continuous modifier on the WRITE A/D command.
- Now, change the external clock frequency until an error occurs in the LDT2801.
- Include an error-checking routine and a diagnosis of the type of error encountered.

LAB # 5 SOLUTION

```
10 '' WRITTEN BY ROBERT C. PACE
20 '' MODIFIED BY ALEXANDER NIKOLOFF
30 '' TO MEET LAB#5 REQUIREMENTS
100     PRINT CHR$(27);"[?31" 'CLEAR SCREEN
110     PRINT
120     PRINT "                  THIS PROGRAM GENERATES"
130     PRINT "       A SINE WAVE AND WRITES IT TO DAC PORT 1     "
140     PRINT
150     PRINT
160 ''
170    DEFINT A-Z
180    BASE.ADDRESS      = &H20
190    COMMAND.REGISTER  = BASE.ADDRESS + 1
200    STATUS.REGISTER   = BASE.ADDRESS + 1
210    DATA.REGISTER     = BASE.ADDRESS
220    COMMAND.WAIT      = &H4
230    WRITE.WAIT        = &H2
240    READ.WAIT         = &H5
250  ''
260    CCLEAR            = &H1
270    CERROR            = &H2
280    CSTOP             = &HF
290    CCLOCK            = &H3
300    CSDA              = &H9
310    CWDA              = &HA
320    CCONTINUOUS       = &H20
330    EXT.TRIGGER       = &H80
340    EXT.CLOCK         = &H40
350    PERIOD#           = 60000!
360    FACTOR.12         = 4096
370    FACTOR.8          = 256
380    DAC1              = 1
386    DACSELECT         = 2
390    DUMMY             = 5
400 ''
410 '' Check for legal Status Register.
420 ''
430    STATUS = INP(STATUS.REGISTER)
440    IF NOT((STATUS AND &H70) = 0) THEN GOTO 1830
450 ''
460 '' Stop and clear the DT2801.
470 ''
480    OUT  COMMAND.REGISTER, CSTOP
490    TEMP = INP(DATA.REGISTER)
500    WAIT STATUS.REGISTER,  COMMAND.WAIT
510    OUT  COMMAND.REGISTER, CCLEAR
520 ''
530 ''
540    FACTOR = FACTOR.12
550 ''
```

```
560 '' Calculate data values for 100 point sine waves.
570 ''
580    PRINT : PRINT "         Calculating sine wave values."
590    PRINT
600 ''
610    DIM DA0LOW(100),DA0HIGH(100),DA1LOW(100),DA1HIGH(100)
620 ''
630    FOR LOOP = 0 TO 99
650    ANGLE#   = (2 * 3.1416 * LOOP)/100
660    DA1VALUE = (FACTOR/2 - 1) * SIN(ANGLE#) + FACTOR/2
710    DA1HIGH(LOOP) = INT(DA1VALUE/256)
720    DA1LOW(LOOP)  = INT(DA1VALUE - DA1HIGH(LOOP) * 256)
740    NEXT LOOP
750 ''
760 '' Set clock frequency to 6.667 (or 13.333 Hz.)
770 ''
780 '' Write SET CLOCK PERIOD command.
790 ''
800    WAIT STATUS.REGISTER,  COMMAND.WAIT
810    OUT  COMMAND.REGISTER, CCLOCK
820 ''
830 '' Write high and low bytes of PERIOD#.
840 ''
850    PERIODH = INT(PERIOD#/256)
860    PERIODL = PERIOD# - PERIODH * 256
870    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
880    OUT  DATA.REGISTER,    PERIODL
890    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
900    OUT  DATA.REGISTER,    PERIODH
910 ''
920 '' Set-up DAC parameters.
930 ''
940 '' Write SET DAC PARAMETERS command.
950 ''
960    WAIT STATUS.REGISTER,  COMMAND.WAIT
970    OUT  COMMAND.REGISTER, CSDA
980 ''
990 '' Write the DAC SELECT byte.
1000 ''
1010    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1020    OUT  DATA.REGISTER,    DAC1
1030 ''
1040 '' Write two bytes of a dummy number of conversions word.
1050 ''
1060    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1070    OUT  DATA.REGISTER,    DUMMY
1080    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1090    OUT  DATA.REGISTER,    DUMMY
1100 ''
1110    INPUT "        Use EXTERNAL CLOCK (Y/N)";Y$
```

```
1120 ''
1130    IF Y$ = "Y" OR Y$ = "y" THEN COMMAND = EXT.CLOCK
1140    IF Y$ = "N" OR Y$ = "n" THEN COMMAND = 0
1150    IF Y$ = "Y" OR Y$ = "y" THEN GOTO 1200
1160    IF Y$ = "N" OR Y$ = "n" THEN GOTO 1200
1170 ''
1180    PRINT : PRINT "        Please respond 'Y' or 'N' only."
1190    GOTO 1100
1200 ''
1210    PRINT "        Writing sine waves to D/A outputs."
1260    PRINT "        Type any character to stop."
1270    PRINT
1280 ''
1290 '' Start DAC conversions using DAC select 1, continuous modifier set.
1300 '' Write WRITE D/A command.
1310 ''
1320    WAIT STATUS.REGISTER,  COMMAND.WAIT
1330    OUT  COMMAND.REGISTER, (CWDA + CCONTINUOUS + COMMAND)
1340 ''
1350 '' Continuously write output data to D/A's at clock rate.
1360 ''
1370    Y$ = INKEY$
1380    WW = WRITE.WAIT : SR = STATUS.REGISTER : DR = DATA.REGISTER
1390  ''
1400    FOR LOOP = 0 TO 99 : WAIT SR, WW, WW
1420    OUT  DR, DA1LOW(LOOP) : WAIT SR, WW, WW : OUT  DR, DA1HIGH(LOOP)
1430    Y$ = INKEY$ : IF NOT(Y$ = "") THEN GOTO 1460
1440    STATUS = INP(SR) : IF (STATUS AND &H80) THEN GOTO 1590
1450    NEXT LOOP : GOTO 1390
1460 ''
1470 '' Stop DT2801.
1480 ''
1490    OUT COMMAND.REGISTER, CSTOP
1500 ''
1510 '' Check for ERROR.
1520 ''
1530    WAIT STATUS.REGISTER,  COMMAND.WAIT
1540    STATUS = INP(STATUS.REGISTER)
1550    IF (STATUS AND &H80) THEN GOTO 1590
1560 ''
1570    PRINT : PRINT "         WRITE SINE WAVES TO D/A Operation Complete"
1580    GOTO 1910
1590 ''
1600 '' Fatal board error.
1610 ''
1620    PRINT
1630    PRINT "FATAL BOARD ERROR"
1640    PRINT "STATUS REGISTER VALUE IS ";HEX$(STATUS);" HEXIDECIMAL"
1650    PRINT : PRINT CHR$(7) : PRINT CHR$(7) : GOSUB 1700
1660    PRINT "ERROR REGISTER VALUES ARE:"
```

```
1670    PRINT "      BYTE 1 -   ";HEX$(ERROR1);"  HEXIDECIMAL"
1680    PRINT "      BYTE 2 -   ";HEX$(ERROR2);"  HEXIDECIMAL"
1690    PRINT : GOTO 1910
1700 ''
1710 '' Read the Error Register.
1720 ''
1730    OUT  COMMAND.REGISTER, CSTOP : TEMP = INP(DATA.REGISTER)
1740 ''
1750    WAIT STATUS.REGISTER, COMMAND.WAIT
1760    OUT COMMAND.REGISTER, CERROR
1770 ''
1780    WAIT STATUS.REGISTER, READ.WAIT
1790    ERROR1 = INP(DATA.REGISTER)
1800    WAIT STATUS.REGISTER, READ.WAIT
1810    ERROR2 = INP(DATA.REGISTER)
1820    RETURN
1830 ''
1840 '' Illegal Status Register.
1850 ''
1860    PRINT
1870    PRINT "FATAL ERROR - ILLEGAL STATUS REGISTER VALUE"
1880    PRINT "STATUS REGISTER VALUE IS ";HEX$(STATUS);" HEXIDECIMAL"
1890    PRINT CHR$(7) : PRINT CHR$(7)
1900 ''
1910    PRINT : PRINT
1920 ''
1930   INPUT "         Run program again (Y/N)";Y$
1940   IF Y$ = "Y" OR Y$ = "y" THEN RUN
1950   IF Y$ = "N" OR Y$ = "n" THEN SYSTEM
1960 ''
1970   PRINT : PRINT "         Please respond with 'Y' or 'N'."
1980   GOTO 1920
2060   END
```

LAB # 6

SAMPLE A FUNCTION AND TRANSLATE THE DATA INTO VOLTAGES

OBJECTIVE

- Familiarization with the A/D capacity of the LDT2801
- Sampling a function

EQUIPMENT

- Rainbow 100 computer
- LDT2801 Interface board
- Oscilloscope
- Function generator
- Necessary connection wire

SETUP

Connect the function generator and the oscilloscope to CHANNEL 0 on the LDT2801.

PROCEDURE

- Write a basic program that generates an series of samples from a function on CHANNEL 0.
- Print all the values to the printer.
- Plot the data obtained.
- Change the number of samples and repeat the procedure for a couple of different values.
- Include an error-checking routine that also diagnoses the type of error encountered.

Remember that the twelve bit data are separated into two bytes. Therefore you need a low and a high part to the sampled value. Use the continuous modifier on the READ D to A command.

LAB # 6 SOLUTION

```
100  '' written by  Robert C. Pace. 5/30/84
110  '' modified by Alexander Nikoloff. 6/17/86
120     PRINT CHR$(27);"[?3l" 'clear screen : PRINT
130     PRINT "          Program samples channel 00"
140     PRINT "          And takes the inputed number of samples"
150     PRINT : PRINT
160 ''
170    DEFINT A-Z
180    BASE.ADDRESS      = &H20
190    COMMAND.REGISTER  = BASE.ADDRESS + 1
200    STATUS.REGISTER   = BASE.ADDRESS + 1
210    DATA.REGISTER     = BASE.ADDRESS
220    COMMAND.WAIT      = &H4
230    WRITE.WAIT        = &H2
240    READ.WAIT         = &H5
250 ''
260    CSTOP             = &HF
270    CCLEAR            = &H1
280    CERROR            = &H2
290    CCLOCK            = &H3
300    CSAD              = &HD
310    CRAD              = &HE
320    EXT.CLOCK         = &H40
330    EXT.TRIG          = &H80
340    PERIOD#           = 40000!
350    MIN.CONV          = 3
360    MAX.CONV          = 1000
370 ''
380 '' Dimension arrays to hold high and low bytes of A/D Data.
390 ''
400    DIM ADL(MAX.CONV), ADH(MAX.CONV)
410 ''
420 '' A/D parameter constants.
430 ''
440    PGH(0) = 1 : PGH(1) = 2  : PGH(2) = 4   : PGH(3) = 8
450    PGL(0) = 1 : PGL(1) = 10 : PGL(2) = 100 : PGL(3) = 500
460    PGX(0) = 1 : PGX(1) = 1  : PGX(2) = 1   : PGX(3) = 1
470 ''
480    SE.CHANNELS = 16         : DI.CHANNELS = 8
490    DT2818.CHANNELS = 1      : EXP.CHANNELS = 64
500 ''
510    FACTOR.10# = 1024        : FACTOR.12# = 4096
520    FACTOR.16# = 32768!
530 ''
540    UNI.RANGE = 10           : UNI.OFFSET = 0
550    BIP.RANGE = 20           : BIP.OFFSET = 10
560    BIP16.RANGE = 10         : BIP16.OFFSET = 0
570    UNI8.RANGE = 5           : UNI8.OFFSET = 0
580 ''
590 '' Check for legal Status Register.
```

```
600 ''
610    STATUS = INP(STATUS.REGISTER)
620    IF NOT((STATUS AND &H70) = 0) THEN GOTO 2400
630 ''
640 '' Stop and clear the DT2801.
650 ''
660    OUT  COMMAND.REGISTER, CSTOP
670    TEMP = INP(DATA.REGISTER)
680    WAIT STATUS.REGISTER,  COMMAND.WAIT
690    OUT  COMMAND.REGISTER, CCLEAR
700 ''
710 '' Set internal clock rate to 10 Hz (20 Hz DT2801-A, DT2818)
720 ''
730 '' Write SET CLOCK PERIOD command.
740 ''
750    WAIT STATUS.REGISTER,  COMMAND.WAIT
760    OUT  COMMAND.REGISTER, CCLOCK
770 ''
780 '' Write high and low bytes of PERIOD#.
790 ''
800    PERIODH = INT(PERIOD#/256)  154
810    PERIODL = PERIOD# - PERIODH * 256  64
820    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
830    OUT  DATA.REGISTER,    PERIODL
840    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
850    OUT  DATA.REGISTER,    PERIODH
860 ''
870 ''Set factors to correspond with the LDT2810 board
880 ''
890    FACTOR# = FACTOR.12#  4096
900    GAIN(0) = PGH(0) : GAIN(1) = PGH(1)
910    GAIN(2) = PGH(2) : GAIN(3) = PGH(3)
920 ''
930 '' Respond to query with 'Y' or 'N'.
940 ''
950    PRINT : PRINT "          Please respond with 'Y' or 'N' only."
960 ''
970 '' Bipolar range and offset.
980 ''
990    RANGE = BIP.RANGE : OFFSET = BIP.OFFSET
1000 '' Differential number of channels.
1010 ''
1020    NUMBER.CHANNELS = DI.CHANNELS
1030 ''
1040 '' Get A/D gain.
1050 ''
1060    PRINT
1070    PRINT "          Set gain, start channel, end channel and number of"
1080    PRINT "          conversions values to be used for A/D parameters."
1090    PRINT : PRINT "          ";
```

```
1100    PRINT "Legal values for gain are ";GAIN(0);", ";GAIN(1);
1110    PRINT ", ";GAIN(2);", and ";GAIN(3);"."
1120    INPUT "          Gain value = ";Y
1130 ''
1140    FOR GAIN.CODE = 0 TO 3 : IF GAIN(GAIN.CODE) = Y  THEN GOTO 1190
1150    NEXT GAIN.CODE
1160 ''
1170    PRINT : PRINT "          Please use legal gain value."
1180    GOTO 1090
1190 ''
1200 '' Set the channel selection for the input signal
1210    START.CHANNEL=0
1220    END.CHANNEL=0
1230 ''
1240 ''
1250 '' Get number of conversions to do.
1260 ''
1270    PRINT : PRINT : PRINT "          ";
1280    PRINT "Legal values for number of conversions are ";MIN.CONV;
1290    PRINT " through ";MAX.CONV;"."
1300    INPUT "          Number of conversions value = ";NUM.CONV
1310 ''
1320    IF (NUM.CONV >= MIN.CONV AND NUM.CONV =< MAX.CONV) THEN GOTO 1360
1330 ''
1340    PRINT : PRINT "          Please use legal number of conversions value."
1350    GOTO 1240
1360 ''
1370 '' Do a SET A/D PARAMETERS command to set up the A/D converter.
1380 '' Write SET A/D PARAMETERS command.
1390 ''
1400    WAIT STATUS.REGISTER,  COMMAND.WAIT
1410    OUT  COMMAND.REGISTER, CSAD
1420 ''
1430 '' Write A/D gain byte.
1440 ''
1450    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1460    OUT  DATA.REGISTER,    GAIN.CODE
1470 ''
1480 '' Write A/D start channel byte.
1490 ''
1500    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1510    OUT  DATA.REGISTER,    START.CHANNEL
1520 ''
1530 '' Write A/D end channel byte.
1540 ''
1550    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1560    OUT  DATA.REGISTER,    END.CHANNEL
1570 ''
1580 '' Write high and low bytes of NCONVERSIONS%.
1590 ''
```

```
1600    NUMBERH = INT(NUM.CONV/256)
1610    NUMBERL = NUM.CONV - NUMBERH * 256
1620    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1630    OUT  DATA.REGISTER,    NUMBERL
1640    WAIT STATUS.REGISTER,  WRITE.WAIT, WRITE.WAIT
1650    OUT  DATA.REGISTER,    NUMBERH
1660 ''
1670    PRINT
1680 ''
1690    WAIT STATUS.REGISTER,  COMMAND.WAIT
1700    OUT  COMMAND.REGISTER, CRAD
1710 ''
1720 ''
1730    FOR LOOP = 1 TO NUM.CONV       : WAIT STATUS.REGISTER,  READ.WAIT
1740    ADL(LOOP) = INP(DATA.REGISTER) : WAIT STATUS.REGISTER,  READ.WAIT
1750    ADH(LOOP) = INP(DATA.REGISTER) : NEXT LOOP
1760 ''
1770 ''
1780 '' Check for ERROR.
1790 ''
1800    WAIT STATUS.REGISTER,  COMMAND.WAIT : STATUS = INP(STATUS.REGISTER)
1810    IF (STATUS AND &H80) THEN GOTO 2160
1820 ''
1830 '' Calculate and print the A/D readings in volts.
1840 ''
1850    NCHAN = END.CHANNEL - START.CHANNEL + 1
1860    IF NCHAN =< 0 THEN NCHAN = NCHAN + NUMBER.CHANNELS
1870    PRINT
1880 ''
1890    FOR LOOP = 1 TO NUM.CONV
1900    DATA.VALUE# = ADH(LOOP) * 256 + ADL(LOOP)
1910    IF DATA.VALUE# > 32767 THEN DATA.VALUE# = DATA.VALUE# -65536!
1920 ''
1930    VOLTS# = ((RANGE * DATA.VALUE#/FACTOR#) - OFFSET)/GAIN(GAIN.CODE)
1940    CHANNEL = START.CHANNEL + ((LOOP - 1) MOD NCHAN)
1950    IF CHANNEL >= NUMBER.CHANNELS THEN CHANNEL = CHANNEL - NUMBER.CHANNELS
1960 ''
1970    PRINT "         CHANNEL "; : PRINT USING "##";CHANNEL;
1980    PRINT " = "; : PRINT USING "###.######";VOLTS#;
1990 ''
2000    IF CHANNEL = END.CHANNEL THEN PRINT
2010    IF CHANNEL = END.CHANNEL THEN PRINT
2020    NEXT LOOP
2030 ''
2040 '' Ask if more conversions are desired.
2050 ''
2060    PRINT : PRINT
2070    INPUT "          Do you want to do more conversions (Y/N)";Y$
2080 ''
2090    IF Y$ = "N" OR Y$ = "n" THEN GOTO 2130
```

```
2100    IF Y$ = "Y" OR Y$ = "y" THEN GOTO 1030
2110 ''
2120    GOSUB 8000 : GOTO 2030
2130 ''
2140    PRINT : PRINT : PRINT "        READ A/D Operation Complete"
2150    GOTO 2480
2160 ''
2170 '' Fatal board error.
2180 ''
2190    PRINT
2200    PRINT "FATAL BOARD ERROR"
2210    PRINT "STATUS REGISTER VALUE IS ";HEX$(STATUS);" HEXIDECIMAL"
2220    PRINT : PRINT CHR$(7) : PRINT CHR$(7) : GOSUB 2270
2230    PRINT "ERROR REGISTER VALUES ARE:"
2240    PRINT "      BYTE 1 -   ";HEX$(ERROR1);"  HEXIDECIMAL"
2250    PRINT "      BYTE 2 -   ";HEX$(ERROR2);"  HEXIDECIMAL"
2260    PRINT : GOTO 2480
2270 ''
2280 '' Read the Error Register.
2290 ''
2300    OUT  COMMAND.REGISTER, CSTOP : TEMP = INP(DATA.REGISTER)
2310 ''
2320    WAIT STATUS.REGISTER, COMMAND.WAIT
2330    OUT COMMAND.REGISTER, CERROR
2340 ''
2350    WAIT STATUS.REGISTER, READ.WAIT
2360    ERROR1 = INP(DATA.REGISTER)
2370    WAIT STATUS.REGISTER, READ.WAIT
2380    ERROR2 = INP(DATA.REGISTER)
2390    RETURN
2400 ''
2410 '' Illegal Status Register.
2420 ''
2430    PRINT
2440    PRINT "FATAL ERROR - ILLEGAL STATUS REGISTER VALUE"
2450    PRINT "STATUS REGISTER VALUE IS ";HEX$(STATUS);" HEXIDECIMAL"
2460    PRINT CHR$(7) : PRINT CHR$(7)
2470 ''
2480    PRINT : PRINT
2490 ''
2500   INPUT "         Run program again (Y/N)";Y$
2510   IF Y$ = "Y" OR Y$ = "y" THEN RUN
2520   SYSTEM
2530   END
```

LAB # 7

USING THE PRINTER PORT TO DRIVE A SPEECH SYNTHESIZER

OBJECTIVE

- Familiarization with the printer and communication ports of the Rainbow

EQUIPMENT

- Rainbow 100 computer
- RS-232c cable
- Votrax (speech synthesizer)

SETUP

Connect Votrax to the printer port via an RS-232c cable.

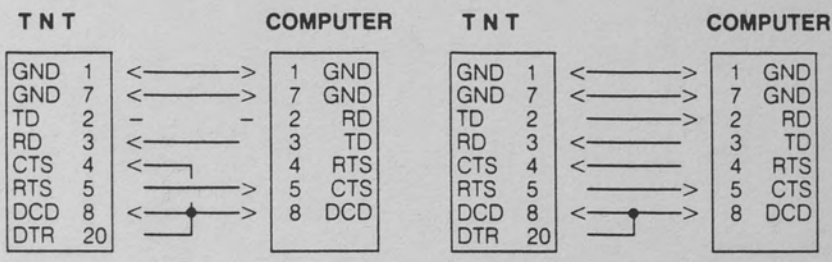
PROCEDURE

Write a 8086 assembly program to send a string of character to Votrax. It will be necessary to set up the following parameters on the printer port:

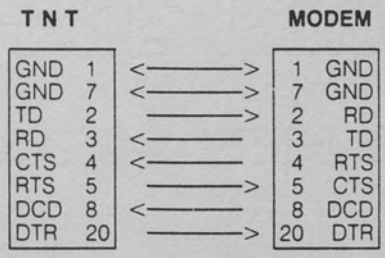
```
Baud rate: 300
Bits     : 7
Parity   : even
```

The set up must be handled within the program and a "suave" exit condition is a requirement. After the completion of the program reset the computer to the original printer port values. The decision of how to handle the communication between the Rainbow and Votrax is left to the creative mind of the programmer. See appended page for more information on the operation of Votrax.

Figure 4. RS-232 CABLE CONFIGURATIONS



A. TNT RECEIVE ONLY **B. TNT SEND-RECEIVE**



A. TNT WITH MODEM ONLY

GND = GROUND
TD = TRANSMIT DATA
RD = RECEIVE DATA

CTS = CLEAR TO SEND
RTS = REQUEST TO SEND
DCD = DATA CARRIER DETECT
DTR = DATA TERMINAL READY

Figure 5. Pertinent Information About Votrax

LAB # 7 SOLUTION

```
1: ;LAB#7
2: ;DESIGNED BY: ALEXANDER NIKOLOFF
3: ;
4: ;MEETS THE REQUIREMENTS FOR VOTRAX
5: ;VOTRAX IS CONNECTED TROUGH THE PRINTER PORT
6: ;THE PORT PARAMETERS ARE SET UP INTERNALY IN THE
7: ;PROGRAM TO BE COMPATABLE WITH VOTRAX. IT WILL BE NECESSARY TO
8: ;TO SET VOTRAX TO 300 BAUD BEFORE EXECUTING THE PROGRAM
9:
10:
11:         CSEG
12:         ORG     100H
13: ALEX:                           ;SET UP THE CORRESPONDING
14:         MOV     D1,DS           ;SEGMENTS IN THE DATA
15:         MOV     D2,DS
16:         MOV     X1,CS
17:
18:         MOV     CX,00
19:
20:
21:         MOV     CL,32H          ;BIOS CALL TO SET UP THE
22:         MOV     DX,OFFSET SETUP ;PRINTER PORT
23:         INT     224
24:         AND     AX,0FFH         ;MAKE SURE THE TASK IS
25:         JE      HOME            ;PERFORMED, OTHERWISE EXIT
26:
27: ;       MOV     CL,32H
28: ;       MOV     DX,OFFSET XBUFFER
29: ;       INT     224
30:
31:         MOV     CL,9            ;DISPLAY PROMPT TO CRT
32:         MOV     DX,OFFSET MESS1
33:         INT     224
34:
35:         JMP     T1
36:
37: NEXTONE:
38: DONE:   MOV     CL,10           ;GET STRING FROM USER
39:         MOV     DX,OFFSET BUFFER
40:         INT     224
41:
42: T1:
43:
44:         MOV     SI,01H
45:         MOV     BX,OFFSET BUFFER+1
46:         MOV     CH,0H
47:         MOV     CL,[BX]         ;SET UP A CHARACTER COUNTER IN CX
48:         AND     CX,0FFFFH       ;IF NO STRING EXIT
49:         JE      HOME
50:
```

```
NEXT:   MOV     AL,[BX+SI]
        MOV     CHARO,AL
        INC     SI
        PUSH    BX
        PUSH    CX
        PUSH    SI

N1:     MOV     CL,32H          ;BIOS CALL TO TRANSMIT A CHARACTER
        MOV     DX,OFFSET OUTCHAR
        INT     224
        POP     SI
        POP     CX
        POP     BX
        LOOP    NEXT            ;REPEAT UNTILL ALL CHARACTERS ARE
                                ;OUTPUTED
N2:     MOV     CHARO,0DH       ;OUTPUT A CR SO VOTRAX WILL SPEAK
        MOV     CL,32H
        MOV     DX,OFFSET OUTCHAR
        INT     224

        CALL    DELAY           ;DELAY TO ALLOW VOTRAX TO SPEAK


        MOV     CL,9            ;DISPLAY INPUT PROMPT TO CRT
        MOV     DX,OFFSET MESS2
        INT     224

        JMP     NEXTONE


HOME:

;       MOV     CL,32H
;       MOV     DX,OFFSET CANXBUFFER
;       INT     224

        MOV     CL,32H          ;SET DEFAULT "LIST"
        MOV     DX,OFFSET UPSET
        INT     224

        MOV     CL,0H
        MOV     DL,00H
        INT     224


DELAY:
        PUSH    CX
        PUSHF
        MOV     CX,000AFH
GO:     PUSH    CX
```

```
101:
102: GO1:    MOV     CX,CX
103:         LOOP    GO1
104:         POP     CX
105:         LOOP    GO
106:         POPF
107:         POP     CX
108:         RET
109:
110:
111:         DSEG
112:         ORG     300H
113: MESS1   DB      'PLEASE INPUT A STRING UP TO 80 CHARACTERS'
114:         DB      0DH,0AH
115:         DB      'THE STRING WILL BE SPOKEN BY VOTRAX          '
116:         DB      0DH,0AH
117:         DB      'INPUT A NEW STRING AFTER THE PROMPT "!" APPEARS'
118:         DB      0DH,0AH
119:         DB      'OR "CARRIAGE RETURN" TO EXIT THE PROGRAM'
120: MESS2   DB      0DH,0AH,'! $'
121:
122: OUTCHAR:
123:         DB      8CH             ;OUTPUT A CHAR TO TRANMIT BUFFER
124:         DW      0200H           ;DEV="LIST"
125: CHARO   DB      32H             ;CHAR TO OUTPUT
126:         DB      00H
127:
128: XBUFFER DB      94H
129:         DW      OFFSET XBUF
130: D1      DW      0000H
131: XBUF    DB      02H
132: X1      DW      0000H
133:         DW      OFFSET DONE
134:
135: CANXBUFFER:
136:         DB      95H
137:         DW      0200H
138:         DW      0000H
139:
140: UPSET   DB      83H             ;SET DEVICE TO ORIGINAL STATE
141:         DW      0200H           ;DEV="LIST"
142:         DW      0000H
143:
144: SETUP   DB      81H             ;PROGRAM DEVICE (PRINTER PORT)
145:         DW      OFFSET INIT
146: D2      DW      0000H
147: INIT    DB      02H             ;DEVICE NUMBER "PRINTER"
148:         DB      01H             ;MODE "DATA TALKS   "
149:         DB      01H             ;STOP BITS
150:         DB      03H             ;DATA BITS "7 "
```

```
151:          DB      01H             ;TRANSMIT PARITY "EVEN"
152:          DB      07H             ;BAUD RATE RCV "0300"
153:          DB      07H             ;BAUD RATE XMT "0300"
154:          DB      11H             ;XON CHAR
155:          DB      13H             ;XOFF CHAR
156:          DB      02H             ;RCV XON/XOFF
157:          DB      02H             ;XMT XON/XOFF
158:          RW      0050H           ;BUFFER SIZE(16 BIT VAL)
159:          DW      0000H           ;OFFSET OF BUFFER START
160:          DW      0000H           ;SEGMENT OF BUFFER START
161:          RW      0002H
162: BUFFER   DB      80
163:          DB      79
164:          DB      'MY NAME IS VOTRAX AND I AM REDY TO SERVE YOU'
165:          DB      '       PLEASE MAKE ME TALK                  '
166:          RB      80
167:          END
```

LAB # 8

SERIAL DATA COMMUNICATION

OBJECTIVE

- To write a simple communication program for serial communication between two Rainbow computers

EQUIPMENT

- Two Rainbow 100 computers
- RS-232c cable

SETUP

Connect the two computers through the communication ports with the RS-232c cable wired in Non Modem Mode.

PROCEDURE

Write a simple communication program that uses serial communication. The program should be able to send a message from one computer to the other in an interactive manner. The constraints of the solution are not limited. Any solution is acceptable.

LAB # 8 SOLUTION

```
AB#8
     2: ;DESIGNED BY: ALEXANDER NIKOLOFF
     3: ;
     4: ;MEETS THE REQUIREMENTS FOR THE SERIAL
     5: ;TRANSMITION OF DATA TROUGH THE COMMUNICATION
     6: ;PORT. THE TRANSMITION IS "DATA TALKS"
     7:
     8:         CSEG
     9:         ORG     100H
    10: ALEX:                           ;SET UP THE CORRESPONDING
    11:         MOV     D2,DS
    12:         MOV     D3,DS
    13:
    14:
    15:         MOV     CL,32H          ;BIOS CALL TO SET UP THE
    16:         MOV     DX,OFFSET SETUP ;COMM PORT
    17:         INT     224
    18:
    19:         MOV     CL,32H          ;SET ALL MODEM SIGNAL HIGH
    20:         MOV     DX,OFFSET SMODEM
    21:         MOV     MODEM,MODEOFF
    22:         INT     224
    23:
    24:         MOV     CL,32H          ;ENABLE THE RECIEVER
    25:         MOV     DX,OFFSET ERECIEV
    26:         INT     224
    27:
    28:         MOV     CL,32H
    29:         MOV     DX,OFFSET RMODEM
    30:         INT     224
    31:
    32: A1:
    33:         MOV     CL,9            ;DISPLAY PROMPT TO CRT
    34:         MOV     DX,OFFSET MESS1
    35:         INT     224
    36:
    37: NEXTONE:
    38: DONE:   MOV     CL,10           ;GET STRING FROM USER
    39:         MOV     BUFFER,78       ;MAX STRING VALUE
    40:         MOV     DX,OFFSET BUFFER
    41:         INT     224
    42:
    43:         MOV     SI,00H
    44:         MOV     BX,OFFSET BUFFER+1
    45:         MOV     CH,0H
    46:         MOV     CL,[BX]         ;SET UP A CHARACTER COUNTER IN CX
    47:         INC     BX
    48:         AND     CX,0FFFFH       ;IF NO STRING EXIT
    49:         JE      HOME
    50:
```

```
NEXT:   MOV     AL,[BX+SI]
        MOV     CHARO,AL
        INC     SI
        PUSH    BX
        PUSH    CX
        PUSH    SI

N1:     MOV     CL,32H          ;BIOS CALL TO TRANSMIT A CHARACTER
        MOV     DX,OFFSET OUTCHAR
        INT     224
        POP     SI
        POP     CX
        POP     BX
        LOOP    NEXT            ;REPEAT UNTILL ALL CHARACTERS ARE
                                ;OUTPUTED
        MOV     CHARO,0AH
        MOV     CL,32H
        MOV     DX,OFFSET OUTCHAR
        INT     224

        MOV     CHARO,0DH
        MOV     CL,32H
        MOV     DX,OFFSET OUTCHAR
        INT     224

        MOV     CL,32H          ;READ THE INPUT STATUS
        MOV     DX,OFFSET READIS
        INT     224
        AND     AL,0FFH
        JE      NOCHAR

        MOV     CL,9            ;DISPLAY INPUT PROMPT TO CRT
        MOV     DX,OFFSET MESS3
        INT     224


NEXTR:
        MOV     CL,32H          ;BIOS CALL TO TRANSMIT A CHARACTER
        MOV     DX,OFFSET READCHR
        INT     224
        AND     AL,0FFH         ; FF=CHAR IS PRESENT
        JE      NOCHAR

        MOV     DL,CL           ;ECHO TO THE SCREEN
        MOV     CL,02H
        INT     224

        JMP     NEXTR           ;REPEAT UNTILL ALL CHARACTERS ARE

NOCHAR: MOV     CL,9            ;DISPLAY INPUT PROMPT TO CRT
```

```
101:          MOV     DX,OFFSET MESS2
102:          INT     224
103:
104:          JMP     NEXTONE
105:
106:
107: HOME:
108:
109:
110:          MOV     CL,32H          ;SET DEFAULT "comm"
111:          MOV     DX,OFFSET UPSET
112:          INT     224
113:
114:          MOV     CL,0H
115:          MOV     DL,00H
116:          INT     224
117:
118:
119:
120:          DSEG
121:          ORG     300H
122: MESS1    DB      'THIS IS A SIMPLE COMMUNICATION PROGRAM   '
123:          DB      0DH,0AH
124:          DB      'THE "!" PROMPT MEANS READY TO INPUT          '
125:          DB      0DH,0AH
126:          DB      'THE "#" PROMPT MEANS MESSAGE FROM THE OTHER COMPUTER'
127:          DB      0DH,0AH
128:          DB      '"CARRIAGE RETURN" TO EXIT THE PROGRAM'
129: MESS2    DB      0DH,0AH,'! $'
130: MESS3    DB      0DH,0AH,'# $'
131:
132: OUTCHAR:
133:          DB      8CH             ;OUTPUT A CHAR TO TRANMIT BUFFER
134:          DW      0100H           ;DEV="COMM"
135: CHARO    DB      32H             ;CHAR TO OUTPUT
136:          DB      00H
137:
138:
139: SMODEM   DB      8FH
140:          DW      0100H
141: MODEM    DB      00H
142:          DB      00H
143: MODEOFF  EQU     0FFH
144:
145: RMODEM   DB      8EH
146:          DW      0100H
147:          DW      0000H
148:
149: DRECIEV  DB      86H
150:          DW      0100H
```

```
151:          DW      0000H
152:
153: ERECIEV  DB      85H
154:          DW      0100H
155:          DW      0000H
156:
157: READIS   DB      87H
158:          DW      0100H
159:          DW      0000H
160:
161: GETCHAR  DB      89H
162:          DW      0100H
163:          DW      0000H
164:
165: READCHR  DB      88H
166:          DW      0100H
167:          DW      0000H
168:
169: UPSET    DB      83H             ;SET DEVICE TO ORIGINAL STATE
170:          DW      0100H           ;DEV="COMM"
171:          DW      0000H
172:
173: SETUP    DB      81H             ;PROGRAM DEVICE (COMM    PORT)
174:          DW      OFFSET INIT
175: D2       DW      0000H
176:
177:          DW      0FFFFH
178:
179: INIT     DB      01H             ;DEVICE NUMBER "COMM   "
180:          DB      01H             ;MODE "DATA TALKS   "
181:          DB      01H             ;STOP BITS
182:          DB      05H             ;DATA BITS "7S"
183:          DB      03H             ;TRANSMIT PARITY "NONE"
184:          DB      10H             ;BAUD RATE RCV "9600"
185:          DB      10H             ;BAUD RATE XMT "9600"
186:          DB      11H             ;XON CHAR
187:          DB      13H             ;XOFF CHAR
188:          DB      02H             ;RCV XON/XOFF
189:          DB      02H             ;XMT XON/XOFF
190:          DW      1400H           ;BUFFER SIZE(16 BIT VAL)
191:          DW      OFFSET BUFF     ;OFFSET OF BUFFER START
192: D3       DW      0000H           ;SEGMENT OF BUFFER START
193: BUFF     RW      1400H
194:          RW      0002H
195: BUFFER   DB      78
196:          DB      78
197:          DB      'THIS CALL HAS SOME SORT OF BUG I DONT KNOW'
198:          DB      'WHY IT WORKS IF YOU HAVE SOME DATA IN THE'
199:          DB      'BUFFER BEFORE CALLING THE ROUTINE       '
200:          END
```

LAB # 9

PARALLEL DATA COMMUNICATION

OBJECTIVE

- To write a simple communication program for communication between two Rainbow computers, using the LDT2801 for parallel transmission

EQUIPMENT

- Two Rainbow 100 computers
- Two LDT2801 boards
- Necessary connection cable

SETUP

Connect the two computers through the LDT2801 boards with the supplied cable.

PROCEDURE

Write a simple communication program that uses parallel communication. The program should mimic the solution to Lab #8 in every possible way. The communication protocol is left to the imagination of the programmer. The constraints of the solution are not limited. Any solution is acceptable.

LAB # 9 SOLUTION

```
1: ;LAB#9
2: ;DESIGNED BY: ALEXANDER NIKOLOFF
3: ;
4: ;MEETS THE REQUIREMENTS FOR THE PARALLEL
5: ;TRANSMITION OF DATA TROUGH THE COMMUNICATION
6: ;PORT. THE TRANSMITION HAS A PROTOCOLL
7:
8:          CSEG
9:          ORG     100H
10:
11: ;SET UP THE LDT2801
12:
13: ALEX:   MOV     AL,CSTOP        ;STOP THE LDT2801
14:         OUT     CREG,AL
15:         IN      AL,DREG         ;CLEAR THE DATA REGISTER
16:         CALL    COMMANDWAIT
17:         MOV     AL,CCLEAR       ;CLEAR THE LDT2810
18:         OUT     CREG,AL
19:
20:         CALL    COMMANDWAIT
21:         MOV     AL,CSIN         ;SET THE DIGITAL INPUT  PORT
22:         OUT     CREG,AL
23:         CALL    WRITEWAIT
24:         MOV     AL,PORT0        ;PORT # 0 SET FOR INPUT
25:         OUT     DREG,AL
26:
27:         CALL    COMMANDWAIT
28:         MOV     AL,CSOUT        ;SET THE DIGITAL OUTPUT PORT
29:         OUT     CREG,AL
30:         CALL    WRITEWAIT
31:         MOV     AL,PORT1        ;PORT #  1 SET FOR OUTPUT
32:         OUT     DREG,AL
33:
34:
35:
36:         CALL    COMMANDWAIT
37:         MOV     AL,CDOUT        ;SET RTS (READY TO SEND)
38:         OUT     CREG,AL         ;
39:         CALL    WRITEWAIT
40:         MOV     AL,PORT1
41:         OUT     DREG,AL
42:         CALL    WRITEWAIT
43:         MOV     AL,RTS
44:         OUT     DREG,AL
45:
46:         MOV     CL,9            ;DISPLAY INTRODUCTION PROMPT TO CRT
47:         MOV     DX,OFFSET MESS1
48:         INT     224
49:
50:         CALL    COMMANDWAIT
```

```
51:          MOV     AL,CDIN         ;READ INPUT PORT
52:          OUT     CREG,AL         ;AND SEE IF RTS
53:          CALL    WRITEWAIT       ;CHAR IS PRESENT
54:          MOV     AL,PORT0
55:          OUT     DREG,AL
56:          CALL    READWAIT
57:          IN      AL,DREG
58:          XOR     AL,RTS
59:          JE      X1
60:          JMP     SEND
61: X1:      JMP     RECIEVE         ;IF RTS IS PRESENT GOTO RECIEVE
62:
63: SEND:
64: NEXTONE:
65: DONE:    MOV     CL,10           ;GET STRING FROM USER
66:          MOV     BUFFER,78       ;MAX STRING VALUE
67:          MOV     DX,OFFSET BUFFER
68:          INT     224
69:
70:          MOV     RTR,RTRCON      ;SET UP PARITY FOR TRANSMISION
71:          MOV     PARITY,CHECK
72:
73:          MOV     SI,-1
74:          MOV     BX,OFFSET BUFFER+1
75:          MOV     CH,0H
76:          MOV     CL,[BX]         ;SET UP A CHARACTER COUNTER IN CX
77:          INC     BX
78:          AND     CX,0FFFFH       ;IF NO STRING EXIT
79:          JNE     DONE1
80:          JMP     HOME
81:
82: DONE1:   CALL    COMMANDWAIT
83:          MOV     AL,CDOUT        ;SENT RTS CHARACTER
84:          OUT     CREG,AL         ;
85:          CALL    WRITEWAIT
86:          MOV     AL,PORT1
87:          OUT     DREG,AL
88:          CALL    WRITEWAIT
89:          MOV     AL,RTS
90:          OUT     DREG,AL
91:
92:
93: NEXT:    INC     SI
94:          PUSH    CX
95:          PUSH    SI
96:
97: S1:      CALL    COMMANDWAIT
98:          MOV     AL,CDIN         ;READ INPUT PORT
99:          OUT     CREG,AL         ;AND WAIT UNTIL RTR (READY TO RECIEVE)
100:         CALL    WRITEWAIT       ;CHAR IS PRESENT
```

```
101:          MOV     AL,PORT0
102:          OUT     DREG,AL
103:          CALL    READWAIT
104:          IN      AL,DREG
105:          XOR     AL,RTR
106:          JNE     S1
107:
108:          CALL    COMMANDWAIT
109:          MOV     AL,CDOUT        ;SEND THE CHARACTERS
110:          OUT     CREG,AL         ;
111:          CALL    WRITEWAIT
112:          MOV     AL,PORT1
113:          OUT     DREG,AL
114:          CALL    WRITEWAIT
115:          MOV     AL,[BX+SI]      ;GET THE CHARACTER
116:          AND     AL,01111111B    ;STRIP PARITY BIT
117:          OR      AL,PARITY       ;SET THE CORRECT PARITY
118:          OUT     DREG,AL         ;SEND THE CHARACTER
119:
120:          XOR     RTR,0FFH        ;SET NEW PARITY
121:
122: SEND1:   CALL    COMMANDWAIT
123:          MOV     AL,CDIN         ;READ CHARACTER
124:          OUT     CREG,AL         ;AND POLE UNTILL NEW CHARACTER
125:          CALL    WRITEWAIT       ;IS PRESENT
126:          MOV     AL,PORT0
127:          OUT     DREG,AL
128:          CALL    READWAIT
129:          IN      AL,DREG
130:          XOR     AL,RTR
131:          JNE     SEND1
132:
133:          POP     SI
134:          POP     CX
135:          XOR     PARITY,CHECK    ;CHANGE PARITY
136:
137:          LOOP    NEXT
138:
139:          CALL    COMMANDWAIT
140:          MOV     AL,CDOUT        ;SEND THE EMPTY CHARACTER
141:          OUT     CREG,AL         ;IT IS THE END OF TRANSMISION
142:          CALL    WRITEWAIT
143:          MOV     AL,PORT1
144:          OUT     DREG,AL
145:          CALL    WRITEWAIT
146:          MOV     AL,EMPTY        ;GET THE EMPTY CHARACTER
147:          OR      AL,PARITY
148:          OUT     DREG,AL         ;SEND THE CHARACTER
149:
150:
```

```
151:                                    ;OUTPUTED
152: RECIEVE:
153:          MOV     RTR,0FFH          ;RESET THE PARITIES
154:          MOV     PARITY,CHECK
155:
156:          MOV     CL,9              ;DISPLAY OUTPUT PROMPT TO CRT
157:          MOV     DX,OFFSET MESS3
158:          INT     224
159:
160: REC3:    CALL    COMMANDWAIT
161:          MOV     AL,CDIN           ;READ INPUT PORT
162:          OUT     CREG,AL           ;AND WAIT UNTIL RTS
163:          CALL    WRITEWAIT         ;CHAR IS PRESENT
164:          MOV     AL,PORT0
165:          OUT     DREG,AL
166:          CALL    READWAIT
167:          IN      AL,DREG
168:          XOR     AL,RTS
169:          JNE     REC3
170:
171: REC2:    CALL    COMMANDWAIT
172:          MOV     AL,CDOUT          ;SEND THE RTR CHARACTER
173:          OUT     CREG,AL           ;
174:          CALL    WRITEWAIT
175:          MOV     AL,PORT1
176:          OUT     DREG,AL
177:          CALL    WRITEWAIT
178:          MOV     AL,RTR
179:          OUT     DREG,AL
180:
181:
182: REC1:    CALL    COMMANDWAIT
183:          MOV     AL,CDIN           ;READ CHARACTER
184:          OUT     CREG,AL           ;AND POLE UNTILL NEW CHARACTER
185:          CALL    WRITEWAIT         ;IS PRESENT. THIS IS DETERMINED
186:          MOV     AL,PORT0          ;BY ALTERNATING 7 BIT
187:          OUT     DREG,AL
188:          CALL    READWAIT
189:          IN      AL,DREG
190:
191:          MOV     DL,AL
192:          AND     DL,01111111B      ;STRIP THE PARITY BIT
193:
194:          AND     AL,STRIP          ;KEEP THE PARITY BIT
195:          XOR     AL,PARITY         ;CHECK FOR THE CORRECT PARITY
196:          JNE     REC1              ;IF NOT NEW CHARACTER GO AGAIN
197:          XOR     PARITY,CHECK      ;CHANGE THE PARITY
198:          MOV     CL,DL
199:          XOR     CL,EMPTY          ;CHECK FOR END OF TRANSMITION
200:          JNE     X2
```

```
201:
202: NOCHAR: MOV     CL,9            ;DISPLAY INPUT PROMPT TO CRT
203:         MOV     DX,OFFSET MESS2
204:         INT     224
205:
206:         JMP     SEND
207:
208: X2:
209:         XOR     RTR,0FFH        ;CHANGE THE PARITY
210:         MOV     CL,02H          ;ECHO TO SCREEN
211:         INT     224
212:         JMP     REC2
213:
214: HOME:
215:
216:         MOV     CL,0H
217:         MOV     DL,00H
218:         INT     224
219:
220: ;SUBROUTINES TO CONTROLL THE LDT2801
221:
222:
223: COMMANDWAIT:    ;POLES THE STATUS OF THE LDT2801, AND RETURNS
224:                 ;WHEN IT IS READY TO ACCEPT A COMMAND
225:         PUSH    AX
226:         PUSHF
227: WAITC:  IN      AL,SREG
228:         AND     AL,CWAIT
229:         JE      WAITC
230:         POPF
231:         POP     AX
232:         RET
233:
234: WRITEWAIT:      ;POLES THE STATUS OF THE LDT2801, AND RETURNS
235:                 ;WHEN ONE CAN WRITE TO THE DATA REGISTER
236:
237:         PUSH    AX
238:         PUSHF
239: WAITW:  IN      AL,SREG
240:         XOR     AL,WWAIT
241:         AND     AL,WWAIT
242:         JE      WAITW
243:         POPF
244:         POP     AX
245:         RET
246:
247: READWAIT:       ;POLES THE STATUS OF THE LDT2801, AND RETURNS
248:                 ;WHEN ONE CAN READ THE DATA REGISTER
249:
250:         PUSH    AX
```

```
251:          PUSHF
252: WAITR:   IN      AL,SREG
253:          AND     AL,RWAIT
254:          JE      WAITR
255:          POPF
256:          POP     AX
257:          RET
258:
259: WAIT1:   IN      AL,DX           ;EXTRA RUTINE JUST IN CASE
260:          XOR     AL,CL
261:          AND     AL,BL
262:          JE      WAIT1
263:          RET
264:
265:
266:
267:
268:          DSEG
269:          ORG     300H
270: BASE     EQU     20H
271: CREG     EQU     BASE+1
272: SREG     EQU     BASE+1
273: DREG     EQU     BASE
274: CWAIT    EQU     4H
275: WWAIT    EQU     2H
276: RWAIT    EQU     5H
277: PORT0    EQU     0H
278: PORT1    EQU     1H
279: PORT2    EQU     2H
280: CRESET   EQU     0H
281: CCLEAR   EQU     1H
282: CERROR   EQU     2H
283: CSOUT    EQU     5H
284: CSIN     EQU     4H
285: CSTOP    EQU     0FH
286: CDOUT    EQU     7H
287: CDIN     EQU     6H
288: TRIG     EQU     80H
289: CHECK    EQU     10000000B
290: PARITY   DB      10000000B
291: STRIP    EQU     10000000B
292: FLAG     DB      0FFH
293: RTS      EQU     13
294: RTR      DB      0FFH
295: RTRCON   EQU     0FFH
296: RTRCO    EQU     0FFH
297: EMPTY    EQU     11
298: MESS1    DB      'THIS IS A SIMPLE COMMUNICATION PROGRAM   '
299:          DB      0DH,0AH
300:          DB      'THE "!" PROMPT MEANS READY TO INPUT          '
```

```
301:          DB      0DH,0AH
302:          DB      'THE "$" PROMPT MEANS MESSAGE FROM THE OTHER COMPUTER'
303:          DB      0DH,0AH
304:          DB      '"CARRIAGE RETURN" TO EXIT THE PROGRAM'
305: MESS2    DB      0DH,0AH,'! $'
306: MESS3    DB      0DH,0AH,'$ $'
307:
308: BUFFER   DB      78
309:          DB      78
310:          DB      'THIS CALL HAS SOME SORT OF BUG I DONT KNOW'
311:          DB      'WHY IT WORKS IF YOU HAVE SOME DATA IN THE'
312:          DB      'BUFFER BEFORE CALLING THE ROUTINE       '
313:          END
```

REFERENCES

Andrews, Michael. Programing Microprocessor Interfaces for Control and Instrumentation. Englewood Cliffs, New Jersey: Prentice-Hall, 1982.

Artwick, Bruce A. Microcomputer Interfacing. Englewood Cliffs, New Jersey: Prentice-Hall, 1980.

P/M-86 Operating System: Programer's Guide. Pacific Grove, California: Digital Research, 1983.

CP/M-86 Operating System: System Guide. Pacific Grove, California: Digital Research, 1983.

CP/M-86 Operating System: User's Guide. Pacific Grove, California: Digital Research, 1983.

CP/M-86/80 V2.0 Technical Documentation.Pacific Grove, California: Digital Research, 1984.

Dessy, Raymond E. Microprocessors & Minicomputers Interfacing and Applications Using the LSI-11. Blacksburg, Virginia: Virginia Politechnic Institute & State University, 1978.

Finkel, Jules. Computer-Aided Experimentation: Interfacing to Minicomputers. New York: John Wiley & Sons, 1975.

Foster, Caxton C. Real Time Programming: Neglected Topics. Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

Lenk, John D. A Hobbyist's Guide to Computer Experimentation. Englewood Cliffs, New Jersey: Prentice-Hall, 1985.

Seyer, Martin D. RS-232 Made Easy: Connecting Computers, Printers, Terminals, and Modems. Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

Snyder, Wesley E. Industrial Robots: Computer Interfacing and Control. Englewood Cliffs, New Jersey: Prentice-Hall, 1985.

User Manual for DT311/LDT2801. Marlborough, Massachusetts: Data Translation, Inc., 1984.