Design of a Laboratory course in Embedded Computer Systems

1986

Alexander Nikoloff
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Computer Engineering Commons

STARS Citation

https://stars.library.ucf.edu/rtd/4978

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
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
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES.</td>
<td>vi</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER ONE - COURSE OVERVIEW.</td>
<td>3</td>
</tr>
<tr>
<td>The Basics of Interfacing</td>
<td>3</td>
</tr>
<tr>
<td>Laboratory Set-up</td>
<td>3</td>
</tr>
<tr>
<td>Design of the Course</td>
<td>5</td>
</tr>
<tr>
<td>CHAPTER TWO - THE LDT2801 BOARD.</td>
<td>8</td>
</tr>
<tr>
<td>LDT2801's Driver: The Rainbow 100</td>
<td>8</td>
</tr>
<tr>
<td>LDT2801 Description</td>
<td>8</td>
</tr>
<tr>
<td>CHAPTER THREE - LAB COURSE LIMITATIONS AND POSSIBLE SOLUTIONS</td>
<td>22</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>25</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>26</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>84</td>
</tr>
</tbody>
</table>
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

Figure 2. Rainbow 100 Communication and Printer Port Signals

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

- Eight differential input channels for A/D conversions (12 bit resolution)
- Two output D/A channels (12 bit resolution)
- 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:
   - Receives data from Rainbow
   - Receives command parameters from Rainbow
   - Data received originates from D/A, A/D and digital (I/O) converters
   - Write Only register
   - Address: Base (where base = 24 Hex for the factory-set boards)
2. DATA OUT:  
- Contains data to be retrieved from Rainbow  
- Used to retrieve information pertaining to error bit pattern  
- Data comes from A/D, D/A, and digital I/O converters  
- Read Only register  
- Address: Base

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

4. STATUS REGISTER:  
- Contains information pertaining to current status of LDT2801  
- Bits in Status Register are used as flags to communicate error status to the Rainbow  
- Read Only register  
- Address: Base + 1
TABLE 1
LDT2801 OPERATING CODE (BITS 0 TO 3 IN COMMAND REGISTER)

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

* 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

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

*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**

<table>
<thead>
<tr>
<th>Command</th>
<th>Data 1</th>
<th>Data 2</th>
<th>Data 3</th>
<th>Data 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear Error</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Error Reg.</td>
<td></td>
<td></td>
<td>Data L</td>
<td>Data H</td>
</tr>
<tr>
<td><strong>Set Internal Clock Period</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Digital Port For Input</td>
<td></td>
<td>Port#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set Digital Port For Output</td>
<td></td>
<td>Port#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read Digital Input Immediate</td>
<td></td>
<td>Port#</td>
<td>Data</td>
<td>[Data Pl]</td>
</tr>
<tr>
<td>Write Digital Output Immediate</td>
<td></td>
<td>Port#</td>
<td>Data</td>
<td>Data Pl</td>
</tr>
<tr>
<td>Write D/A Immediate Gain</td>
<td></td>
<td>Dac#</td>
<td>Data L</td>
<td>Data H</td>
</tr>
<tr>
<td>Set D/A Parameters</td>
<td>Gain</td>
<td>Dac#</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Con# L</td>
<td>Con# H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write D/A</td>
<td>Dac#</td>
<td>Data L</td>
<td>[Data D1L]</td>
<td>[Data D1L]</td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read A/D Immediate</td>
<td></td>
<td></td>
<td>Data L</td>
<td>Data H</td>
</tr>
<tr>
<td>Set A/D Parameters</td>
<td>Gain</td>
<td>Start#</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End#</td>
<td></td>
<td>Con# L</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Con# H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read A/D</td>
<td></td>
<td></td>
<td>Data L</td>
<td>Data H</td>
</tr>
<tr>
<td>Stop Operation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Explanation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 1</td>
<td>Data in register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 2</td>
<td>Data in register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 3</td>
<td>Data out register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data 4</td>
<td>Data out register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data L</td>
<td>Low byte of the data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data H</td>
<td>High byte of the data (A/D or D/A: 4 lower bits)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port#</td>
<td>Digital port 0 or 1. If a 2 is specified both are selected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>Data read from the Data Out Register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Data Pl]</td>
<td>Optional. If both ports have been selected, then the second data read from the Data Out Register corresponds to port 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gain</td>
<td>The two bit descriptor corresponds to the following gains: byte value:gain, 0:1, 1:2, 2:4, 3:8.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dac#</td>
<td>Selects the output Dac 0 or 1. If a 2 is specified both are selected.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Con# L</td>
<td>Lower byte of the number of conversions to be performed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Con# H</td>
<td>High byte of the number of conversions to be performed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Data D1L]</td>
<td>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).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Data D1H]</td>
<td>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).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start#</td>
<td>The channel number to begin the multiplexing with.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End#</td>
<td>The channel number to end the multiplexing with.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* 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.
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 COUT = &H7
300 DIN = &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

; LAB #2
; DESIGNED BY: ALEXANDER NIKOLOFF
; MEETS THE SPECIFIED REQUIREMENTS FOR LAB #2

CSEG ORG 100H

AL, CSTOP ; STOP THE LDT2810
OUT CREG, AL
IN AL, DREG ; CLEAR THE DATA REGISTER
MOV BK, WAITT
MOV DX, SREG
CALL WAITT
MOV AL, CCLEAR ; CLEAR THE LDT2810
OUT CREG, AL

MOV BK, WAITT
MOV DX, SREG
CALL WAITT
MOV AL, CSOUT ; SET THE DIGITAL OUTPUT PORT
OUT CREG, AL

MOV BL, WAITT
MOV DX, SREG
CALL WAITT
MOV AL, CSIN ; SET DIGITAL PORT FOR INPUT
OUT CREG, AL

MOV BL, WAITT
MOV DX, SREG
CALL WAITT
MOV AL, PORT0 ; PORT # 0 SET FOR INPUT
OUT DREG, AL

MOV CL, 9 ; DISPLAY PROMPT TO CRT
MOV DX, 300H
INT 224
MOV CL, 1 ; GET ANSWER FROM USER
INT 224
MOV TEMP, COIN
XOR AL, 'Y'
JE TRIGGER
51: XOR AL, 'y'
52: JNE AGAIN
53: TRIGGER: MOV TEMP, TRIG ; SET UP LOT2810N COMMAND IN TEMP
54: AGAIN:
55: MOV BX, WAIT
56: MOV DX, SREG
57: CALL WAIT
58: MOV AL, TEMP ; READ PORT 0 WAIT FOR
59: OUT CREG, AL ; TRIGGER IF APPLICABLE
60: MOV BL, WAIT
61: MOV BH, WAIT
62: MOV DX, SREG
63: CALL WAIT
64: MOV AL, PORT0
65: OUT DREG, AL
66: MOV BK, WAIT
67: MOV DX, SREG
68: CALL WAIT
69: IN AL, DREG
70: PUSH AX
71: MOV BX, WAIT
72: MOV DX, SREG
73: CALL WAIT
74: MOV AL, COUT ; OUTPUT THE READ INFORMATION
75: OUT CREG, AL ; TO PORT 1
76: MOV BL, WAIT
77: MOV BH, WAIT
78: MOV DX, SREG
79: CALL WAIT
80: MOV AL, PORT1
81: OUT DREG, AL
82: MOV BL, WAIT
83: MOV BH, WAIT
84: MOV DX, SREG
85: CALL WAIT
86: POP AX
87: OUT DREG, AL ; CHECK FOR CONSOLE INPUT
88: MOV CL, 11
89: INT 224 ; EXIT IF PRESENT
90: TEST AL, 0FFH ; IF TRIGGER IS PRESENT,
91: JE AGAIN ; EXECUTION WILL TERMINATE
92: MOV CL, 0 ; AFTER A TRIGGER
93: MOV DL, 0
94: INT 224
95: 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

UNTIL 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

113: ; WAIT:  PUSH AX
114: ; PUSH BX
115: ; PUSH CX
116: ; PUSHF
117: ; MOV CX, BX
118: ; SHR CX, 1
119: ; SHR CX, 1
120: ; SHR CX, 1
121: ; SHR CX, 1
122: ; SHR CX, 1
123: ; SHR CX, 1
124: ; SHR CX, 1
125: ; SHR CX, 1
126: ; AND BX, 00FFH
127: ; WAIT1: IN AL, DX
128: ; XOR AL, CL
129: ; AND AL, BL
130: ; JE WAIT1
131: ; POPF
132: ; POP CX
133: ; POP BX
134: ; POP AX
135: ; RET
136: ;
137: ; DSEG
138: ; ORG 300H
139: ; BASE EQU 20H
140: ; CREG EQU BASE+1
141: ; SREG EQU BASE+1
142: ; DREG EQU BASE
143: ; CWAIT EQU 4H
144: ; WMWAIT EQU 2H
145: ; WMWAIT EQU 5H
146: ; PORT0 EQU 0H
147: ; PORT1 EQU 1H
148: ; CRESET EQU 0H
149: ; CCLEAR EQU 1H
150: ; CERROR EQU 2H
CS OUT EQU 5H
CSIN EQU 4H
CSTOP EQU 0FH
CDOUT EQU 7H
CDIN EQU 6H
TRIG EQU 86H
PROMPT DB 'DO YOU WANT EXTERNAL TRIGGER? Y/N $'
TEMP RB 1
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:

<table>
<thead>
<tr>
<th>LDT2801</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital I/O</td>
<td></td>
</tr>
<tr>
<td>Port 0</td>
<td></td>
</tr>
<tr>
<td>BIT 0 -------------------</td>
<td>SW1</td>
</tr>
<tr>
<td>BIT 1 -------------------</td>
<td>SW2</td>
</tr>
<tr>
<td>BIT 2 -------------------</td>
<td>SW3</td>
</tr>
<tr>
<td>BIT 3 -------------------</td>
<td>SW4</td>
</tr>
<tr>
<td>BIT 4 -------------------</td>
<td>SW5</td>
</tr>
<tr>
<td>BIT 5 -------------------</td>
<td>SW6</td>
</tr>
<tr>
<td>BIT 6 -------------------</td>
<td>SW7</td>
</tr>
<tr>
<td>BIT 7 -------------------</td>
<td>SW8</td>
</tr>
<tr>
<td>DIGITAL I/O</td>
<td></td>
</tr>
<tr>
<td>Port 1</td>
<td></td>
</tr>
<tr>
<td>BIT 0 -------------------</td>
<td>M1F</td>
</tr>
<tr>
<td>BIT 1 -------------------</td>
<td>M1B</td>
</tr>
<tr>
<td>BIT 2 -------------------</td>
<td>M2F</td>
</tr>
<tr>
<td>BIT 3 -------------------</td>
<td>M2B</td>
</tr>
</tbody>
</table>
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: ;MOVEMATERIAL:
37: ;
38: ;INITIALIZATION: ;INITIALIZE THE POSITION OF ALL THE MOTORS
39: ;
40: ;
41: MOV BH,M1B
42: MOV BL,SH2
43: CALL MM
44: ;
45: MOV BH,M2B
46: MOV BL,SH4
47: CALL MM
48: ;
49: MOV BH,M3B
50: MOV BL,SH7
51: CALL MM
52: MOV CL,9 ;DISPLAY PROMPT TO CRT
53: MOV DX,OFFSET MESS1 ;"SYSTEM READY"
54: INT 224
55: NOGOOD:
56: CALL COMMANDWAIT
57: MOV AL,COMIN ;SET UP COMMAND AND WAIT FOR TRIGGER
58: OR AL,TRIG ; START SWITCH
59: OUT CREG,AL
60: CALL WRITEWAIT
61: MOV AL,PORT0
62: OUT DREG,AL
63: CALL READWAIT
64: IN AL,DREG
65: AND AL,SW1
66: JE GOODM
67: MOV CL,9 ;DISPLAY PROMPT TO CRT
68: MOV DX,OFFSET MESS2 ;"NO MATERIAL PRESENT:"
69: INT 224
70: JMP NOGOOD
71: GOODM: MOV BH,M1F ;START MOTION
72: MOV BL,SW3
73: CALL MM
74: MOV BH,M3F
75: MOV BL,SW6
76: CALL MM
77: MOV BH,M2F
78: MOV BL,SW5
79: CALL MM
80: MOV BH,M3B
81: MOV BL,SW7
82: CALL MM
83: MOV BH,M4 ;SEPARATE RUTINE NEEDED
84: MOV BL,SW8 ;BECAUSE OF INVERSE FUNCTIONING
85: CALL COMMANDWAIT ;OF THE PHOTOCELLS
86: MOV AL,COMOUT
87: OUT CREG,AL
88: CALL WRITEWAIT
89: MOV AL,PORT1
90: OUT DREG,AL
91: CALL WRITEWAIT
92: MOV AL,BH
101: OUT DREG,AL

103: CALL TESTKB

104: CALL COMMANDWAIT

105: MOV AL,CDIN ;READ INPUT PORT

106: OUT CREG,AL ;AND POLE UNTILL SWITCH

107: CALL WRITEWAIT ;IS DEPRESS

108: MOV AL,PORT0

109: OUT DREG,AL

110: CALL READWAIT

111: IN AL,DREG

112: XOR AL,OFFH

113: AND AL,BL

114: JE XX1

115: JMP INITIALIZATION

118: MOTORMOTION:

120: PARAMETERS BH:MOTOR AND DIRECTION

121; PARAMETERS BL:SWITCH TO POLE

122: EXIT CONDITION: IF KEBOARD IS DEPRESSED

124: CALL COMMANDWAIT

125: MOV AL,CDOUT ;MOVE MOTOR

126: OUT CREG,AL

127: CALL WRITEWAIT

128: MOV AL,PORT1

129: OUT DREG,AL

130: CALL WRITEWAIT

131: MOV AL,BH

132: OUT DREG,AL

135: CALL TESTKB

137: CALL COMMANDWAIT

138: MOV AL,CDIN ;READ INPUT PORT

139: OUT CREG,AL ;AND POLE UNTILL SWITCH

140: CALL WRITEWAIT ;IS DEPRESED

141: MOV AL,PORT0

142: OUT DREG,AL

143: CALL READWAIT

144: IN AL,DREG

145: AND AL,BL

146: JE XX1

147: CALL COMMANDWAIT ;SEQUENCE TO STOP ALL MOTORS

148: MOV AL,CDOUT

149: 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: RET
158: ;ROUTINE THAT POLES THE KEYBOARD
159: 160: TESTKB:
161: 162: PUSH BX       ;AN ENTRY THEN EXIT
163: PUSH CX
164: PUSHF
165: MOV CL,11
166: INT 224
167: TEST AL,OFFH
168: JNE HOME1
169: POPF
170: POP CX
171: POP BX
172: RET
173: HOME1: POP CX
174: POP BX
175: POP AX
176: 177: CALL COMMANDWAIT   ;SEQUENCE TO STOP ALL MOTORS
178: MOV AL,CDOUT       ;AND TURN ON THE LIGHTS
179: OUT CREG,AL
180: CALL WRITEWAIT
181: MOV AL,PORT1
182: OUT DREG,AL
183: CALL WRITEWAIT
184: MOV AL,KILL
185: OR AL,80H
186: OUT DREG,AL
187: 188: MOV CL,01H    ;CLEANUP THE BUFFER
189: INT 224
190: 191: MOV CL,9     ;DISPLAY PROMPT TO CRT
192: MOV DX,OFFSET MESS3
193: INT 224
194: 195: HOME2: MOV CL,01    ;GET RESPONSE FROM USER
196: INT 224
197: TEST AL,'Y'
198: JE HOME3
199: 200: TEST AL,'y'
201: JE HOME
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 INITIALIZE
210:
211:
212: ; SUBROUTINES TO CONTROL 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: WAIT: IN AL,SREG
220: AND AL,WAIT
221: JE WAIT
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: WAIT: IN AL,SREG
232: XOR AL,WAIT
233: AND AL,WAIT
234: JE WAIT
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,WAITR
246: JE WAITR
247: POPF
248: POP AX
249: RET
250:
251: WAIT1: IN AL,DX ;EXTRA RUTINE JUST IN CASE
252: XOR AL,CL
253: AND AL,BL
254: JE WAIT1
255: RET
256:
257:
258: DSEG
259: ORG 300H
260: BASE EQU 20H
261: CREG EQU BASE+1
262: SREG EQU BASE+1
263: DREG EQU BASE
264: CWAIT EQU 4H
265: WAIT EQU 2H
266: RWAIT EQU 5H
267: PORT0 EQU 0H
268: PORT1 EQU 1H
269: PORT2 EQU 2H
270: CRESET EQU 0H
271: CCLEAR EQU 1H
272: CERROR EQU 2H
273: CSOUT EQU 5H
274: CSIN EQU 4H
275: CSTOP EQU 0FH
276: CDOUT EQU 7H
277: CDIN EQU 6H
278: TRIG EQU 80H
279: SW1 EQU 00000001B ;PHOTOSENSOR
280: SW2 EQU 00000010B
281: SW3 EQU 00000100B
282: SW4 EQU 00001000B
283: SW5 EQU 00010000B
284: SW6 EQU 00100000B
285: SW7 EQU 01000000B
286: SW8 EQU 10000000B ;PHOTOSENSOR
287: M1F EQU 00000001B
288: M1B EQU 00000010B
289: M2F EQU 00000100B
290: M2B EQU 00001000B
291: M3F EQU 00010000B
292: M3B EQU 00100000B
293: M4 EQU 01000000B
294: LIGHTS EQU 10000000B
295: KILL EQU 00
296: MESS2 DB ’NO MATERIAL PRESENT, PLEASE FIX IT
297: DB 0DH,0AH,0AH,’$'
298: MESS3 DB ’TRANSPORT INTERRUPTED, ABNORMAL PROCEDURE
299: DB 0DH,0AH
300: DB ’PLEASE PRESS KEYBOARD TO BEGIN EXECUTION AGAIN’
301:   DB       ODH,0AH
302:   DB       'OR "Y" TO EXIT THE PROGRAM
303:   DB       ODH,0AH,0AH,'$'
304:   MESS1  DB       'SYSTEM READY... TO EXIT PROGRAM HIT KEYBOARD'
305:   DB       ODH,0AH,0AH,'$'
306:   DB
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:

<table>
<thead>
<tr>
<th>LDT2801</th>
<th>INTERSIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital I/O Port 1</td>
<td></td>
</tr>
<tr>
<td>BIT 0 -------------</td>
<td>D0</td>
</tr>
<tr>
<td>BIT 1 -------------</td>
<td>D1</td>
</tr>
<tr>
<td>BIT 2 -------------</td>
<td>D2</td>
</tr>
<tr>
<td>BIT 3 -------------</td>
<td>D3</td>
</tr>
<tr>
<td>BIT 4 -------------</td>
<td>D4</td>
</tr>
<tr>
<td>BIT 5 -------------</td>
<td>D5</td>
</tr>
</tbody>
</table>

| 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

**ICM7233 PARALLEL INPUT ALPHA DISPLAY**

<table>
<thead>
<tr>
<th>TERMINAL</th>
<th>PIN NO.</th>
<th>DESCRIPTION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>30</td>
<td>Least Significant 6 Bit (ASCII) Data Inputs</td>
<td>Input Data</td>
</tr>
<tr>
<td>D1</td>
<td>31</td>
<td></td>
<td>See Table 4</td>
</tr>
<tr>
<td>D2</td>
<td>32</td>
<td></td>
<td>LOW = Logical Zero 0</td>
</tr>
<tr>
<td>D3</td>
<td>33</td>
<td></td>
<td>HIGH = Logical One 1</td>
</tr>
<tr>
<td>D4</td>
<td>34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>37</td>
<td>Least Significant Most Significant Address Inputs</td>
<td>Input Add See Table 5</td>
</tr>
<tr>
<td>A1</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS1</td>
<td>39</td>
<td>Chip Select Inputs</td>
<td>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.</td>
</tr>
<tr>
<td>CS2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4

<table>
<thead>
<tr>
<th>CODE INPUT</th>
<th>DISPLAY OUTPUT</th>
<th>D5</th>
<th>D4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>B</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>C</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>D</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>E</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>F</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>G</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>H</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>I</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>J</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>K</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>L</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>M</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>N</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>O</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>P</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**DATA DECODING**

6 BIT ASCII—16 SEGMENT (ICM7233/34)

### Table 5

<table>
<thead>
<tr>
<th>CODE INPUT</th>
<th>DIGIT SELECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2 A1 A0</td>
<td></td>
</tr>
<tr>
<td>0 0 0</td>
<td>D1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>D2</td>
</tr>
<tr>
<td>0 1 0</td>
<td>D3</td>
</tr>
<tr>
<td>0 1 1</td>
<td>D4</td>
</tr>
<tr>
<td>1 0 0</td>
<td>D5</td>
</tr>
<tr>
<td>1 0 1</td>
<td>NONE</td>
</tr>
<tr>
<td>1 1 0</td>
<td>NONE</td>
</tr>
<tr>
<td>1 1 1</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**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 ;CLEAR THE DATA REGISTER
10: CALL COMMANDA1T
11: MOV AL,CCLEAR ;CLEAR THE LDT2810
12: OUT CREG,AL
13: CALL COMMANDA1T
14: MOV AL,CSTOUT ;SET THE DIGITAL OUTPUT PORT
15: OUT CREG,AL
16: CALL WRITEWAIT
17: MOV AL,PORT2 ;PORT # 0 AND 1 SET FOR OUTPUT
18: OUT DREG,AL
19: MOV CL,9 ;DISPLAY PROMPT TO CRT
20: MOV DX,OFFSET MESS1
21: INT 224
22: MOV CL,10 ;GET STRING FROM USER
23: MOV DX,OFFSET BUFFER
24: INT 224
25: MOV AL,20H ;INITIAL INDEX
26: MOV SI,1
27: MOV BX,OFFSET BUFFER
28: MOV [BX],AL
29: MOV CH,0
30: MOV CL,[BX+SI]
31: MOV COUNT,CX ;COUNT=#OF CHARACTERS+8
32: ADD COUNT,8H
33: MOV DOUBLE,CX
34: MOV [BX+SI],AL
35: MOV CX,8H
36: ADD DOUBLE,OFFSET BUFSTA-2
37: INC SI
38: DUPLI: MOV AL,[BX+SI] ;DUPLICATE THE FIRST 8 CHARACTERS
39: XCHG DOUBLE,BX ;THE END OF THE STRING
40: MOV [BX+SI],AL
41: XCHG DOUBLE,BX
42: INC SI
43: LOOP DUPLI
44:
51:  DO IT AGAIN:
52:  MOV  BX, OFFSET INIT-1
53:  MOV  CX, COUNT
54:  MOV  SI, 0
55:  DO IT:
56:  CALL  DISPLAYBUFER
57:  CALL  DELAY
58:  INC  SI
59:  CALL  DISPLAYBUFER
60:  CALL  DELAY
61:  PUSH  SI
62:  PUSH  BX
63:  PUSH  CX
64:  PUSHF
65:  MOV  CL, 11
66:  INT  224
67:  TEST  AL, OFFH
68:  JNE  HOME1
69:  POPF
70:  POP  CX
71:  POP  BX
72:  POP  SI
73:  LOOP  DO IT
74:  MOV  BX, OFFSET BUFS TA
75:  MOV  CX, COUNT
76:  SUB  CX, 8H
77:  Movi  TEMP, 3H
78:  MOV  CHiP, CHIP2
79:  PUSH  SI
80:  NEXT:
81:  INC  SI
82:  CALL  COMMANDWAIT
83:  MOV  AL, COUT
84:  OUT  CREG, AL
85:  CALL  WRITEWAIT
86:  MOV  AL, PORT1
87:  OUT  DREG, AL
88:  DISPLAYBUFER:
89:  DISPLAY THE WINDOW
90:  MOV  TEMP, 3H
91:  MOV  CHiP, CHIP2
92:  PUSH  SI
93:  NEXT:
94:  INC  SI
95:  CALL  COMMANDWAIT
96:  MOV  AL, COUT
97:  OUT  CREG, AL
98:  CALL  WRITEWAIT
99:  MOV  AL, PORT1
100: OUT  DREG, AL
CALL WRITEWAIT
MOV AL,[BX+SI]
OUT DREG,AL

CALL COMMANDWAIT
MOV AL,CDOUT ;OUTPUT THE POSITION OF THE CHARACTER TO PORT 0
OUT CREG,AL

CALL WRITEWAIT
MOV AL,PORT0
OUT DREG,AL

CALL WRITEWAIT
MOV AL,TEMP
OUT DREG,AL

CALL COMMANDWAIT
MOV AL,CDOUT ;OUTPUT THE LAST TRIGGER TO PORT 0 AND DISPLAY THE INFORMATION
OUT CREG,AL

CALL WRITEWAIT
MOV AL,PORT0
OUT DREG,AL

CALL WRITEWAIT
MOV AL,CHIP
OUT DREG,AL

DEC TEMP ;Determine if all 8 characters
JBE NEXT ;have been displayed

MOV TEMP,3H
AND CHIP,CHIP1
JNE EXIT
MOV CHIP,CHIP1
JMP NEXT

EXIT:
POP SI
RET

HOME:
MOV CL,0 ;Exit to CPM
MOV DL,0
INT 224

COMMANDWAIT: ;Poles the status of the LDT2810, and returns
WHEN IT IS READY TO ACCEPT A COMMAND
PUSH AX
PUSHF
WAITC: IN AL,SREG
AND AL,CHAIT
JE WAITC
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:   WAIT:  IN    AL,SREG
160:   XOR   AL,WAIT
161:   AND   AL,WAIT
162:   JE     WAIT
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:   WAIT:  IN    AL,SREG
173:   AND   AL,WAIT
174:   JE     WAIT
175:   POPF
176:   POP    AX
177:   RET
178:
179: WAIT1: IN    AL,DX ;EXTRA ROUTINE 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
ORG 300H
BASE EQU 20H
CREG EQU BASE+1
SREG EQU BASE+1
DREG EQU BASE
CHIT EQU 4H
AWAIT EQU 2H
RAWAIT EQU 5H
PORTO EQU 0H
PORT1 EQU 1H
PORT2 EQU 2H
CRES 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 86H
CHIP1 EQU 00000100B
CHIP2 EQU 00001000B
MESS1 DB 'PLEASE INPUT A STRING UP TO 80 CHARACTERS'
DB 0DH,0AH
DB 'INPUT HAS TO BE IN CAPITALS AND A LAGING SPACE'
DB 0DH,0AH
DB 'TO EXIT HIT ANY KEY'
DB 0DH,0AH
DB '$'
INIT DB
BUFFER DB 80
RB 1
BUFFA RAM 60
TEMP DB 4
DOUBLE DW 0
COUNT DW 0
CHIP DB 2
END
LAB # 5

GENERATE A SINE CURVE FROM DIGITAL DATA

OBJECTIVE

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

EQUIPMENT

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

SETUP

  o Connect the oscilloscope to the DAC1 screw connection on the LDT2801.
  o Connect the function generator to the external clock connection.
  o 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

  o Write a program in BASIC that generates an array of 100 points of a sine wave.
  o The program should ask the user if an external clock should be used.
  o 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.
  o Display the sine wave on the oscilloscope.
  o Output the sine wave using the continuous modifier on the WRITE A/D command.
  o Now, change the external clock frequency until an error occurs in the LDT2801.
  o 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 COMMANDREGISTER = BASE.ADDRESS + 1
200 STATUSREGISTER = BASE.ADDRESS + 1
210 DATAREGISTER = BASE.ADDRESS
220 COMMANDWAIT = &H4
230 WRITEWAIT = &H2
240 READEWAIT = &H5
250 "
260 CLEAR = &H1
270 ERROR = &H2
280 STOP = &HF
290 CLOCK = &H3
300 CSA = &H9
310 CHA = &HF
320 CONTINUOUS = &H20
330 EXTTRIGGER = &H80
340 EXT.CLOCK = &H40
350 PERIOD# = 60000!
360 FACTOR.12 = 4096
370 FACTOR.8 = 256
380 DAC1 = 1
390 DACSELECT = 2
390 DUMMY = 5
400 "
410 " Check for legal Status Register.
420 "
430 STATUS = INP(STATUSREGISTER)
440 IF NOT((STATUS AND &H70) = 0) THEN GOTO 1830
450 "
460 " Stop and clear the DT2801.
470 "
480 OUT COMMANDREGISTER, CSTOP
490 TEMP = INP(DATAREGISTER)
500 WAIT STATUSREGISTER, COMMANDWAIT
510 OUT COMMANDREGISTER, CLEAR
520 "
530 "
540 FACTOR = FACTOR.12
550 "
'' Calculate data values for 100 point sine waves.
580 PRINT : PRINT "Calculating sine wave values."
590 PRINT
600 ''
610 DIM DAOLOW(100),DASHIGH(100),DA1LOW(100),DA1HIGH(100)
620 ''
630 FOR LOOP = 0 TO 99
640 ANGLE* = (2 * 3.1416 * LOOP)/100
650 DA1VALUE = (FACTOR/2 - 1) * SIN(ANGLE*) + FACTOR/2
660 DA1HIGH(LOOP) = INT((DA1VALUE/256)
670 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."
1220 PRINT " Type any character to stop."
1230 PRINT
1240 "
1250 " Start DAC conversions using DAC select 1, continuous modifier set."
1260 " Write WRITE D/A command."
1270 "
1280 WAIT STATUS.REGISTER, COMMAND=WAIT
1290 OUT COMMAND.REGISTER, (DIOA + CONTINUOUS + COMMAND)
1300 "
1310 CONTINUOUSLY WRITE OUTPUT DATA TO D/A'S AT CLOCK RATE.
1320 "
1330 Y$ = INKEY$ ;
1340 WW = WRITE.WAIT : SR = STATUS.REGISTER : DR = DATA.REGISTER
1350 "
1360 FOR LOOP = 0 TO 99 : WAIT SR, WW, WW
1370 OUT DR, DA1LOW(LOOP) ; WAIT SR, WW, WW : OUT DR, DA1HIGH(LOOP)
1380 Y$ = INKEY$ : IF NOT(Y$ = ") THEN GOTO 1460
1390 STATUS = INP(SR) : IF (STATUS AND &H80) THEN GOTO 1590
1400 NEXT LOOP : GOTO 1390
1410 "
1420 STOP DT2801.
1430 "
1440 OUT COMMAND.REGISTER, CSTOP
1450 "
1460 " Check for ERROR.
1470 "
1480 WAIT STATUS.REGISTER, COMMAND=WAIT
1490 STATUS = INP(STATUS.REGISTER)
1500 IF (STATUS AND &H80) THEN GOTO 1590
1510 "
1520 PRINT ; PRINT " WRITE SINE WAVES TO D/A Operation Complete"
1530 GOTO 1910
1540 "
1550 " Fatal board error.
1560 "
1570 PRINT "FATAL BOARD ERROR"
1580 PRINT "STATUS REGISTER VALUE IS ",HEX$(STATUS)" HEXADECIMAL"
1590 PRINT ; PRINT CHR$(7) ; PRINT CHR$(7) ; GOSUB 1700
1600 "PRINT "ERROR REGISTER VALUES ARE:"
PRINT "BYTE 1 - \"HEX\$\{ERROR1\}\"; HEXIDECIMAL"
PRINT "BYTE 2 - \"HEX\$\{ERROR2\}\"; HEXIDECIMAL"
PRINT : GOTO 1910
"
"Read the Error Register.
""
OUT COMMAND.REGISTER, CSTOP : TEMP = INP(DATA.REGISTER)
""
WAIT STATUS.REGISTER, COMMAND.WAIT
OUT COMMAND.REGISTER, CERROR
""
WAIT STATUS.REGISTER, READ.WAIT
ERROR1 = INP(DATA.REGISTER)
WAIT STATUS.REGISTER, READ.WAIT
ERROR2 = INP(DATA.REGISTER)
RETURN
""
 Illegal Status Register.
""
PRINT "FATAL ERROR - ILLEGAL STATUS REGISTER VALUE"
PRINT "STATUS REGISTER VALUE IS \"HEX\$\{STATUS\}\"; HEXIDECIMAL"
PRINT Chr$(7) : PRINT Chr$(7)
""
PRINT : PRINT
""
INPUT " Run program again \(Y/N\)\"; Y$
IF Y$ = \"Y\" OR Y$ = \"y\" THEN RUN
IF Y$ = \"N\" OR Y$ = \"n\" THEN SYSTEM
""
PRINT : PRINT " Please respond with \'Y\' or \'N\'."
GOTO 1920
""
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 a 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.
PRINT CHR$(27);"?31" 'clear screen : PRINT
PRINT "Program samples channel 00"
PRINT "And takes the inputed number of samples"
PRINT : PRINT
DEFINT A-Z
BASE.ADDRESS = &H20
COMMAND.REGISTER = BASE.ADDRESS + 1
STATUS.REGISTER = BASE.ADDRESS + 1
DATA.REGISTER = BASE.ADDRESS
COMMAND.WAIT = &H4
WRITE.WAIT = &H2
READ.WAIT = &H5
CSTOP = &HF
CCLEAR = &H1
CERROR = &H2
CCLOCK = &H3
CSAD = &H0
CRAO = &HE
EXT.CLOCK = &H40
EXT.TRIG = &H80
PERIOD# = 40000!
MIN.CONV = 3
MAX.CONV = 1000
' Dimension arrays to hold high and low bytes of A/D Data.
DIM ADL(MAX.CONV), ADH(MAX.CONV)
' A/D parameter constants.
PGH(0) = 1 : PGH(1) = 2 : PGH(2) = 4 : PGH(3) = 8
PGL(0) = 1 : PGL(1) = 10 : PGL(2) = 100 : PGL(3) = 500
PSX(0) = 1 : PSX(1) = 1 : PSX(2) = 1 : PSX(3) = 1
SE.CHANNELS = 16 : DI.CHANNELS = 8
DT2818.CHANNELS = 1 : EXP.CHANNELS = 64
FACTOR.10# = 1024 : FACTOR.12# = 4096
FACTOR.16# = 256
UNI.RANGE = 10 : UNI.OFFSET = 0
BIP.RANGE = 20 : BIP.OFFSET = 10
BIP16.RANGE = 10 : BIP16.OFFSET = 0
UNI8.RANGE = 5 : UNI8.OFFSET = 0
' Check for legal Status Register.
600 "
610  STATUS = INP(STATUS.REGISTER)
620  IF NOT((STATUS AND 6H70) = 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, DT2813)
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) ;
810  PERIODL = PERIOD# - PERIODH * 256 ;
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  FACTORl# = FACTOR.12#
900  GAIN(O) = PGH(O) ; 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 
1210 START.CHANNEL=0
1220 END.CHANNEL=0
1230 
1240 
1250 
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 NCONVATIONS*.
1590"
1600 NUMBERH = INT(NUM.COMV/256)
1610 NUMBERL = NUM.COMV - NUMBERH * 256
1620 WAIT STATUS REGISTER, WRITEWAIT, WRITEWAIT
1630 OUT DATA REGISTER, NUMBERL
1640 WAIT STATUS REGISTER, WRITEWAIT, WRITEWAIT
1650 OUT DATA REGISTER, NUMBERH
1660 ''
1670 PRINT
1680 ''
1690 WAIT STATUS REGISTER, COMMANDWAIT
1700 OUT COMMAND REGISTER, CRAD
1710 ''
1720 ''
1730 FOR LOOP = 1 TO NUM.COMV: WAIT STATUS REGISTER, READWAIT
1740 ADL(LOOP) = INP(DATA REGISTER): WAIT STATUS REGISTER, READWAIT
1750 ADH(LOOP) = INP(DATA REGISTER): NEXT LOOP
1760 ''
1770 ''
1780 '' Check for ERROR.
1790 ''
1800 WAIT STATUS REGISTER, COMMANDWAIT: STATUS = INP (STATUS REGISTER)
1810 IF (STATUS AND $H80) THEN GOTO 2160
1820 ''
1830 '' Calculate and print the A/D readings in volts.
1840 ''
1850 NCHAN = END CHANNEL - START CHANNEL + 1
1860 IF NCHAN =< 0 THEN NCHAN = NCHAN + NUMBER CHANNELS
1870 PRINT
1880 ''
1890 FOR LOOP = 1 TO NUM.COMV
1900 DATA.VALUE* = ADH(LOOP) * 256 + ADL(LOOP)
1910 IF DATA.VALUE* > 32767 THEN DATA.VALUE* = DATA.VALUE* - 65536
1920 ''
1930 VOLTS* = ((RANGE * DATA.VALUE*/FACTOR*) - OFFSET)/GAIN (GAIN CODE)
1940 CHANNEL = START CHANNEL + ((LOOP - 1) MOD NCHAN)
1950 IF CHANNEL = NUMBER CHANNELS THEN CHANNEL = CHANNEL - NUMBER CHANNELS
1960 ''
1970 PRINT " CHANNEL "; PRINT USING "***"; CHANNEL;
1980 PRINT " "; PRINT USING "####.#####"; VOLTS*;
1990 ''
2000 IF CHANNEL = END CHANNEL THEN PRINT
2010 IF CHANNEL = END CHANNEL THEN PRINT
2020 NEXT LOOP
2030 ''
2040 '' Ask if more conversions are desired.
2050 ''
2060 PRINT: PRINT
2070 INPUT " Do you want to do more conversions (Y/N)"; Y$
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$(ERROR2);" HEXIDECIMAL"
2260 PRINT : GOTO 2480
2270 " Read the Error Register.
2280 "
2290 OUT COMMAND REGISTER, CSTOP : TEMP = INP(DATA REGISTER)
2300 "
2310 WAIT STATUS REGISTER, COMMAND WAIT
2320 OUT COMMAND REGISTER, CERROR
2330 "
2340 WAIT STATUS REGISTER, READ WAIT
2350 ERROR1 = INP(DATA REGISTER)
2360 "
2370 WAIT STATUS REGISTER, READ WAIT
2380 ERROR2 = INP(DATA REGISTER)
2390 RETURN
2400 " Illegal Status Register.
2410 "
2420 PRINT
2430 PRINT "FATAL ERROR - ILLEGAL STATUS REGISTER VALUE"
2440 PRINT "STATUS REGISTER VALUE IS ";HEX$(STATUS);" HEXIDECIMAL"
2450 PRINT CHR$(7) ; PRINT CHR$(7)
2460 "
2470 PRINT : PRINT
2480 "
2490 INPUT " Run program again (Y/N)";Y$
2500 IF Y$ = "Y" OR Y$ = "Y" THEN RUN
2510 SYSTEM
2520 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 setup 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

<table>
<thead>
<tr>
<th>TNT</th>
<th>COMPUTER</th>
<th>TNT</th>
<th>COMPUTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND 1</td>
<td>↑</td>
<td>1 GND</td>
<td>↑</td>
</tr>
<tr>
<td>GND 7</td>
<td>↑</td>
<td>7 GND</td>
<td>↑</td>
</tr>
<tr>
<td>TD 2</td>
<td>→</td>
<td>2 RD</td>
<td>→</td>
</tr>
<tr>
<td>RD 3</td>
<td>→</td>
<td>3 TD</td>
<td>→</td>
</tr>
<tr>
<td>CTS 4</td>
<td>→</td>
<td>4 RTS</td>
<td>→</td>
</tr>
<tr>
<td>RTS 5</td>
<td>→</td>
<td>5 CTS</td>
<td>→</td>
</tr>
<tr>
<td>DCD 8</td>
<td>→</td>
<td>8 DCD</td>
<td>→</td>
</tr>
<tr>
<td>DTR 20</td>
<td>→</td>
<td>20 DTR</td>
<td>→</td>
</tr>
</tbody>
</table>

A. TNT RECEIVE ONLY

B. TNT SEND-RECEIVE

<table>
<thead>
<tr>
<th>TNT</th>
<th>MODEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND 1</td>
<td>↑</td>
</tr>
<tr>
<td>GND 7</td>
<td>↑</td>
</tr>
<tr>
<td>TD 2</td>
<td>→</td>
</tr>
<tr>
<td>RD 3</td>
<td>→</td>
</tr>
<tr>
<td>CTS 4</td>
<td>→</td>
</tr>
<tr>
<td>RTS 5</td>
<td>→</td>
</tr>
<tr>
<td>DCD 8</td>
<td>→</td>
</tr>
<tr>
<td>DTR 20</td>
<td>→</td>
</tr>
</tbody>
</table>

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
1: LAB #7
2: ; DESIGNED BY: ALEXANDER NIKOLOFF
3: ;
4: ; MEETS THE REQUIREMENTS FOR VOTRAX
5: ; VOTRAX IS CONNECTED THROUGH THE PRINTER PORT
6: ; THE PORT PARAMETERS ARE SET UP INTERNALLY 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: CSEG
11: ORG 100H
12: ALEX: ; SET UP THE CORRESPONDING
13: MOV D1, DS ; SEGMENTS IN THE DATA
14: MOV D2, DS
15: MOV X1, CS
16: MOV CX, 00
17: MOV CL, 32H ; BIOS CALL TO SET UP THE
18: MOV DX, OFFSET SETUP ; PRINTER PORT
19: INT 224
20: AND AX, OFFH ; MAKE SURE THE TASK IS
21: JE HOME ; PERFORMED, OTHERWISE EXIT
22: MOV CL, 9 ; DISPLAY PROMPT TO CRT
23: MOV DX, OFFSET MESS1
24: INT 224
25: MOV CL, [BX] ; SET UP A CHARACTER COUNTER IN CX
26: AND CX, OFFFH ; IF NO STRING EXIT
27: JE HOME
28:
51: NEXT:  MOV AL,[BX+SI]
52:  MOV CHARO,AL
53:  INC SI
54:  PUSH BX
55:  PUSH CX
56:  PUSH SI
57:  
58: N1:  MOV CL,32H ; BIOS CALL TO TRANSMIT A CHARACTER
59:  MOV DX,OFFSET OUTCHAR
60:  INT 224
61:  POP SI
62:  POP CX
63:  POP BX
64:  LOOP NEXT ; REPEAT UNTIL ALL CHARACTERS ARE
65:  ; OUTPUTED
66: N2:  MOV CHAR0,ODH ; OUTPUT A CR SO VOTRAX WILL SPEAK
67:  MOV CL,32H
68:  MOV DX,OFFSET OUTCHAR
69:  INT 224
70:  
71: CALL DELAY ; DELAY TO ALLOW VOTRAX TO SPEAK
72:  
73: 
74: MOV CL,9 ; DISPLAY INPUT PROMPT TO CRT
75:  MOV DX,OFFSET MESS2
76:  INT 224
77:  
78: JMP NEXTONE
79:  
80: HOME:
81: ;  
82: ;  MOV CL,32H
83: ;  MOV DX,OFFSET CANNBUFFER
84: ;  INT 224
85: ;  
86: MOV CL,32H ; SET DEFAULT "LIST"
87:  MOV DX,OFFSET UPSET
88:  INT 224
89:  
90: 
91: MOV CL,OH
92:  MOV DL,00H
93:  INT 224
94:  
95: 
96: DELAY:
97:  PUSH CX
98:  PUSHF
99:  MOV DX,000AFH
100: GO:  PUSH CX
G01: MOV CX
104: POP CX
105: LOOP G0
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 ODH,0AH
115: DB 'THE STRING WILL BE SPOKEN BY VOTRAV'
116: DB ODH,0AH
117: DB 'INPUT A NEW STRING AFTER THE PROMPT "!" APPEARS'
118: DB ODH,0AH
119: DB 'OR "CARRIAGE RETURN" TO EXIT THE PROGRAM'
120: MESS2 DB ODH,0AH,'! $'
121:
122: CUTFCHAR:
123: DB 6CH ;OUTPUT A CHAR TO TRANSMIT BUFFER
124: DW 0200H ;DEV="LIST"
125: CHAR 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: CANBUFFER:
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"
70

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 88
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

2: ;DESIGNED BY: ALEXANDER NIKOLOFF
3: 
4: ;MEETS THE REQUIREMENTS FOR THE SERIAL
5: ;TRANSMISSION OF DATA THROUGH THE COMMUNICATION
6: ;PORT. THE TRANSMISSION 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 RECEIVER
25: MOV DX, OFFSET ERECEIV
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 MESSI
35: INT 224
36: 
37: NEXTONE: ;SET STRING FROM USER
38: DONE: MOV CL, 10
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 DX, 0FFFFH ;IF NO STRING EXIT
49: JE HOME
50: 

51: NEXT: MOV AL,[BX+SI]
52: MOV CHARO,AL
53: INC SI
54: PUSH BX
55: PUSH CX
56: PUSH SI
57: 
58: NL: MOV CL,32H ;BIOS CALL TO TRANSMIT A CHARACTER
59: MOV DX,OFFSET OUTCHAR
60: INT 224
61: POP SI
62: POP CX
63: POP BX
64: LOOP NEXT ;REPEAT UNTILL ALL CHARACTERS ARE
65: ;OUTPUTED
66: MOV CHARO,0AH
67: MOV CL,32H
68: MOV DX,OFFSET OUTCHAR
69: INT 224
70: 
71: MOV CHARO,0DH
72: MOV CL,32H
73: MOV DX,OFFSET OUTCHAR
74: INT 224
75: 
76: MOV CL,32H ;READ THE INPUT STATUS
77: MOV DX,OFFSET READIS
78: INT 224
79: AND AL,OFFH
80: JE NOCHAR
81: 
82: MOV CL,9 ;DISPLAY INPUT PROMPT TO CRT
83: MOV DX,OFFSET MESS3
84: INT 224
85: 
86: 
87: NEXTR:
88: MOV CL,32H ;BIOS CALL TO TRANSMIT A CHARACTER
89: MOV DX,OFFSET READCHR
90: INT 224
91: AND AL,OFFH ; FF=CHAR IS PRESENT
92: JE NOCHAR
93: 
94: MOV DL,CL ;ECHO TO THE SCREEN
95: MOV CL,02H
96: INT 224
97: 
98: JMP NEXTR ;REPEAT UNTILL ALL CHARACTERS ARE
99: 
100: NOCHAR: MOV CL,9 ;DISPLAY INPUT PROMPT TO CRT
MOV DX,OFFSET MESS2
INT 224
JMP NEXTONE
MOV CL,32H ;SET DEFAULT "comm"
MOV DX,OFFSET UPSET
INT 224
MOV CL,0H
MOV DL,00H
INT 224
DSEG
ORG 300H
MESS1 DB 'THIS IS A SIMPLE COMMUNICATION PROGRAM'
DB ODH,0AH
DB 'THE *!* PROMPT MEANS READY TO INPUT'
DB ODH,0AH
DB 'THE *#* PROMPT MEANS MESSAGE FROM THE OTHER COMPUTER'
DB ODH,0AH
DB '"CARRIAGE RETURN" TO EXIT THE PROGRAM'
MESS2 DB ODH,0AH,'! $'
MESS3 DB ODH,0AH,'* $'
OUTCHAR:
DB 8CH ;OUTPUT A CHAR TO TRANSMIT BUFFER
DN 0100H ;DEV="COMM"
CHAR DB 32H ;CHAR TO OUTPUT
DB 00H
SMODEM DB 8FH
DW 0100H
MODEM DB 00H
DB 00H
MODEOFF EQU 0FFH
AMODEM DB 8EH
DW 0100H
DW 0000H
DRECEI V DB 8EH
DW 0100H
151:    DW   0000H
152:
153:    ERECIEV DB  85H
154:    DW   0100H
155:    DW   0000H
156:    DW   0000H
157:    READIS DB  87H
158:    DW   0100H
159:    DW   0000H
160:    DW   0000H
161:    GETCHAR DB  89H
162:    DW   0100H
163:    DW   0000H
164:    DW   0000H
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:    DW   0000H
173:    SETUP DB  81H ;PROGRAM DEVICE (COMM PORT)
174:    DW   OFFSET INIT
175:    D2    DW   0000H
176:    DW   0000H
177:    DW   0FFFFFFH
178:    DW   0000H
179:    INIT DB  01H ;DEVICE NUMBER "COMM"
180:    DB   01H ;MODE "DATA TALKS"
181:    DB   01H ;STOP BITS
182:    DB   05H ;DATA BITS "76"
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:    D8    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 THROUGH THE COMMUNICATION
6: ;PORT. THE TRANSMITION HAS A PROTOCOL
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, CLEAR ;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, CSOUT ;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
53: CALL WRITEWAIT ;CHAR IS PRESENT
54: MOV AL,PORTO
55: OUT DREG,AL
56: CALL READWAIT
57: IN AL,DREG
58: XOR AL,RTS
59: JE X1
60: JMP SEND
61: X1: JMP RECEIVE ;IF RTS IS PRESENT GOTO RECEIVE
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 TRANSMISSION
71: MOV PARITY,CHECK
72:
73: MOV SI,-1
74: MOV BX,OFFSET BUFFER+1
75: MOV CH,OH
76: MOV CL,[BX] ;SET UP A CHARACTER COUNTER IN CX
77: INC BX
78: AND CX,OFFFH ;IF NO STRING EXIT
79: JNE DONE1 ;READ INPUT PORT
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: SI: CALL COMMANDWAIT ;READ INPUT PORT
98: MOV AL,CDIN ;AND WAIT UNTIL RTS (READY TO RECEIVE)
99: OUT CREG,AL
100: CALL WRITEWAIT ;CHAR IS PRESENT
79

101: MOV AL,PORT0
102: OUT DREG,AL
103: CALL READWAIT
104: IN AL,DREG
105: XOR AL,RTR
106: JNE SI
107: CALL COMMANDWAIT
108: MOV AL,CDOUT ;SEND THE CHARACTERS
109: OUT CREG,AL ;
110: CALL WRITEWAIT
111: MOV AL,PORT1
112: OUT DREG,AL
113: CALL WRITEWAIT
114: MOV AL,[BX+SI] ;GET THE CHARACTER
115: AND AL,01111111B ;STRIP PARITY BIT
116: OR AL,PARITY ;SET THE CORRECT PARITY
117: OUT DREG,AL ;SEND THE CHARACTER
118: XOR RTR,OFFH ;SET NEW PARITY
119: SEND1: CALL COMMANDWAIT
120: MOV AL,C0IN ;READ CHARACTER
121: OUT CREG,AL ;AND POLE UNTILL NEW CHARACTER
122: CALL WRITEWAIT ;IS PRESENT
123: MOV AL,PORT0
124: OUT DREG,AL
125: CALL READWAIT
126: IN AL,DREG
127: XOR AL,RTR
128: JNE SEND1
129: POP SI
130: POP CX
131: XOR PARITY,CHECK ;CHANGE PARITY
132: LOOP NEXT
133: CALL COMMANDWAIT
134: MOV AL,CDOUT ;SEND THE EMPTY CHARACTER
135: OUT CREG,AL ;IT IS THE END OF TRANSMISSION
136: CALL WRITEWAIT
137: MOV AL,PORT1
138: OUT DREG,AL
139: CALL WRITEWAIT
140: MOV AL,EMPTY ;GET THE EMPTY CHARACTER
141: OR AL,PARITY
142: OUT DREG,AL ;SEND THE CHARACTER
143: CALL WRITEWAIT
144: MOV AL,PORT0
145: OUT DREG,AL
146: CALL WRITEWAIT
147: MOV AL,EMPTY ;SEND THE EMPTY CHARACTER
148: OUT DREG,AL ;SEND THE CHARACTER
149: CALL WRITEWAIT
150: MOV AL,PORT1
151: OUT DREG,AL
151: RECEIVE: ;OUTPUTED
152: MOV RTR,0FFH ;RESET THE PARITIES
153: MOV PARITY,CHECK
154: MOV CL,9 ;DISPLAY OUTPUT PROMPT TO CRT
155: MOV DX,OFFSET MESS3
156: INT 224
157: REC2: CALL COMMANDWAIT
158: MOV AL,C Din ;READ INPUT PORT
159: OUT CREG,AL ;AND WAIT UNTIL RTS
160: CALL WRITEWAIT ;CHAR IS PRESENT
161: MOV AL,PORT0
162: OUT DREG,AL
163: CALL READWAIT
164: IN AL,DREG
165: XOR AL,RTS
166: JNE REC3
167: REC3: CALL COMMANDWAIT
168: MOV AL,CD OUT ;SEND THE RTR CHARACTER
169: OUT CREG, AL ;
170: CALL WRITEWAIT
171: MOV AL,PORT1
172: OUT DREG,AL
173: CALL WRITEWAIT
174: MOV AL,RTR
175: OUT DREG,AL
176: REC2: CALL COMMANDWAIT
177: MOV AL,C Din ;READ CHARACTER
178: OUT CREG,AL ;AND POLE UNTILL NEW CHARACTER
179: CALL WRITEWAIT ;IS PRESENT. THIS IS DETERMINED
180: MOV AL,PORT0 ;BY ALTERNATING 7 BIT
181: OUT DREG,AL
182: CALL READWAIT
183: IN AL,DREG
184: MOV DL,AL
185: AND DL,0111111B ;STRIP THE PARITY BIT
186: JNE RECl
187: MOV CL,DL
188: XOR CL,EMPTY ;CHECK FOR END OF TRANSMITION
189: XOR AL,PARITY ;CHECK FOR THE CORRECT PARITY
190: JNE RECl ;IF NOT NEW CHARACTER GO AGAIN
191: MOV CL, DL
192: XOR AL, PARITY, CHECK ;CHANGE THE PARITY
193: ADD AL,STRIP ;KEEP THE PARITY BIT
194: XOR AL,PARITY ;CHECK FOR THE CORRECT PARITY
195: JNE RECl ;IF NOT NEW CHARACTER GO AGAIN
196: MOV CL,DL
197: XOR CL,EMPTY ;CHECK FOR END OF TRANSMITION
198: JNE X2
201: NOCHAR: MOV CL,9 ;DISPLAY INPUT PROMPT TO CRT
202: MOV DX,OFFSET MESS2
203: INT 224
204: JMP SEND
205: X2:
206: XOR RTR,0FFH ;CHANGE THE PARITY
207: MOV CL,02H ;ECHO TO SCREEN
208: JMP REC2
209: HOME:
210: MOV CL,0H
211: MOV DL,00H
212: INT 224
213: SUBROUTINES TO CONTROL THE LDT2801
214: COMMANDWAIT: ;POLES THE STATUS OF THE LDT2801, AND RETURNS
215: WAIT: WHEN IT IS READY TO ACCEPT A COMMAND
216: PUSH AX
217: PUSHF
218: IN AL,SREG
219: AND AL,WAIT
220: JE WAIT
221: POPF
222: POP AX
223: RET
224: WRITEWAIT: ;POLES THE STATUS OF THE LDT2801, AND RETURNS
225: WHEN ONE CAN WRITE TO THE DATA REGISTER
226: PUSH AX
227: SUBROUTINES TO CONTROL THE LDT2801
228: WAIT: WHEN ONE CAN READ THE DATA REGISTER
229: READWAIT: ;POLES THE STATUS OF THE LDT2801, AND RETURNS
230: PUSH AX
```assembly
251:       PUSHF
252:       WAIT:  IN  AL,SREG
253:       AND  AL,RAIWT
254:       JE    WAITR
255:       POPF
256:       POP  AX
257:       RET
258:       WAIT1: IN  AL,DX    ;;EXTRA ROUTINE JUST IN CASE
259:       XOR  AL,CL
260:       AND  AL,EL
261:       JE    WAIT1
262:       RET
263:       RET
264:       RET
265:       RET
266:       RET
267:       DSEG
268:       ORG  300H
269:       BASE  EQU  20H
270:       CREG  EQU  BASE+1
271:       SREG  EQU  BASE+1
272:       DREG  EQU  BASE
273:       CHALT EQU  4H
274:       WAILT EQU  2H
275:       RAIWT EQU  5H
276:       PORT0  EQU  0H
277:       PORT1  EQU  1H
278:       PORT2  EQU  2H
279:       CRESET  EQU  0H
280:       CLEAR  EQU  1H
281:       CERROR  EQU  2H
282:       CSOUT  EQU  5H
283:       CSIN   EQU  4H
284:       CSTOP  EQU  0FH
285:       CSTOP  EQU  0FH
286:       CCOUT  EQU  7H
287:       COIN   EQU  6H
288:       TRIG   EQU  80H
289:       CHECK  EQU  1000000B
290:       PARITY  DB  1000000B
291:       STRIP  EQU  1000000B
292:       FLAG   DB  OFFH
293:       RTS   EQU  13
294:       RTR   DB  OFFH
295:       RTRCON  EQU  OFFH
296:       RTRCO  EQU  OFFH
297:       EMPTY  EQU  11
298:       MESS1  DB  "THIS IS A SIMPLE COMMUNICATION PROGRAM"
299:       DB    DDH,ODH
300:       DB    "THE "!" PROMPT MEANS READY TO INPUT"
```
301:   DB    ODH,0AH
302:   DB    'THE **' PROMPT MEANS MESSAGE FROM THE OTHER COMPUTER'
303:   DB    ODH,0AH
304:   DB    '"CARRIAGE RETURN' TO EXIT THE PROGRAM'
305:   MESS2 DB    ODH,0AH,!' $'
306:   MESS3 DB    ODH,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


