DESIGN OF A REAL-TIME MULTI-CHANNEL MICROPROCESSOR BASED DATA ACQUISITION AND CONTROL SYSTEM

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Muntasir J. Hadji
November, 1988
To my parents
who sacrificed the years
in endless love and support,
without which this job could not have been done.
## CONTENTS

### Chapter 1: Introduction
- Multi-channel systems 3
- Purpose of the thesis 4

### Chapter 2: A/D Converters
- Basic conversion definitions 6
- A/D types 7
  - V/F converters 8
  - Dual-slope integration 10
  - Successive-approximation 13
  - Tracking converters 15
  - Parallel converters 16
- Microprocessor interface 18
- Sample and Hold 20

### Chapter 3: The System Hardware
- System's block diagram 22
- The CPU 23
- Address decoding 25
- The acquisition unit 27
- Control unit 32
- Programming of the 8155 ports 33
  - Port A 33
  - Port B 34
  - Port C 36
- Serial Communication 36
- Baud rate generator 39
- The RS-232C interface 40
- RS-232C driver/receiver 41
- Memory buffer 42
- The system ROM 43
Chapter 4: **Software**

Monitor flowchart
System initialization
USART configuration
System commands
Monitor commands
  - Software reset
  - Control unit set/reset
  - Characters and data output
  - Continuous data output
Acquisition unit commands
Hexadecimal byte output
Decimal byte output
Output subroutine
Control unit commands

Chapter 5: **Conclusion**

Testing of the system
System's hardware limitations
Software characteristics
Range of applications
Expanding the system
Summary

Appendix A: **Operating system listing**
Appendix B: **Temperature monitor program listing**
Appendix C: **System schematic diagram**
Appendix D: **Data sheets**

References

**FIGURES**

1.1 Simple data acquisition system
1.2 Block diagram of data acquisition and control system
1.3 Basic Multi-channel parallel conversion method
2.1 Ideal conversion relationship
2.2 Linearity error
2.3 Voltage to frequency A/D converter
2.4 Dual-slope integrating A/D converter
2.5 Dual-slope integrating timing diagram
2.6 Block diagram of a successive-approximation converter
2.7 Example of successive-approximation conversion
2.8 Tracking A/D converter
2.9 Parallel A/D converter
2.10 Half-flash A/D converter
2.11 Basic microprocessor interface for an A/D converter
3.1 System block diagram
3.2 8085 Low order address decoding
3.3 8085 Basic address/data timing diagram
3.4 Address decoding circuit
3.5 Memory and I/O space allocation
3.6 ADC0820 hookup circuit and its timing diagram
3.7 Design of the acquisition unit
3.8 Interfacing of the I/O ports to the CPU
3.9 Port A hookup circuit to an optoisolator
3.10 Port B configuration
3.11 Port C configuration
3.12 USART interface to the system bus
3.13 Baud rate generator circuit
3.14 RS232C connection
3.15 RS232C receiver/transmitter
3.16 System memory buffer
3.17 The system ROM interface circuit
4.1 General Flowchart of the system monitor
4.2 System initialization flowchart
4.3 Flowchart of USART configuration
4.4 Commands classification flowchart
5.1 Organization of the hardware design

TABLES

3.1 Address ranges for the system’s devices
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>The system port ranges</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Port assignment of the analog input channels</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Port B bit assignments</td>
<td>35</td>
</tr>
<tr>
<td>3.5</td>
<td>Control signals for the 8251 UART</td>
<td>38</td>
</tr>
<tr>
<td>3.6</td>
<td>System baud rates</td>
<td>40</td>
</tr>
<tr>
<td>4.1a</td>
<td>Acquisition unit commands</td>
<td>52</td>
</tr>
<tr>
<td>4.1b</td>
<td>Control unit commands</td>
<td>53</td>
</tr>
<tr>
<td>4.1c</td>
<td>Monitor commands</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>Execution time of the system commands</td>
<td>68</td>
</tr>
</tbody>
</table>
Chapter I

INTRODUCTION

One of the interesting uses for computers other than being calculating and accounting machines is to have them measure data and control or guide systems. Many applications these days require real-world quantities including pressure, temperature, speed, position...etc.

The process of collecting data is called data acquisition. The above types of analog quantities or data must be converted as rapidly and as accurately as necessary to digital form—a process called digitization— in order to be processed by a computer. The basic tool for accomplishing this is the analog-to-digital converter (A/D) that translates analog measurements which are characteristic of most phenomena in the real world to digital codes. These codes are used in information processing, computing, data transmission and control systems. A/D converters output a series of binary numbers (usually in parallel form) corresponding to the value of the analog signal being sampled.

A typical Data Acquisition unit consists of the components shown in figure 1.1. In this system, a physical input (such as temperature, pressure,
etc,) is converted to a DC voltage through a transducer, then digitized by the A/D whose output is buffered before being transmitted to a computer or communication channel for processing. Data acquisition systems consist basically of the blocks shown in figure 1.1, with the exception that they may have more input channels, data storage capabilities, and a control unit able to control systems and devices according to the analog input. A block diagram of a Data Acquisition and Control System is shown in figure 1.2.

![Block diagram of data acquisition and control system](image)

**Figure 1.2**: Block diagram of data acquisition and control system

In this system, an analog input is sampled and digitized, then processed by a computer which would issue an appropriate command to the control unit according to the value of the sampled analog input. This can be explained better with a simple example of a room air-conditioner. Suppose a room's temperature is to be maintained at 25°C. It is constantly monitored by an A/D through a temperature probe or a transducer. When the temperature drops to a value below 25°C, the computer or the processing unit issues a command to the control unit to switch off the air-conditioner, and to switch it on when the temperature rises above 25°C. This application and others require a few samples each second. However, a high sampling
rate is needed in many other applications such as digital imaging and video devices, where the sampling frequency range is in the MHz. This rate can be achieved in today's technology with the use of fast A/D and S/H devices in a microprocessor based system.

**Multi-Channel Systems:**

In multi-channel conversion systems, elements of the acquisition system may be shared by two or more input sources. Although the conventional way to digitize data from many analog channels is to share a single A/D converter (multiplexed inputs), a parallel conversion approach is becoming more practical. Since the cost of A/Ds has been dropping continuously, it is now possible to build a multi-channel acquisition system just as economically as an analog multiplexed system. Figure 1.3 shows a block diagram of a multi-channel parallel conversion system.

![Figure 1.3: Basic multi-channel parallel conversion method](image-url)
There are a number of advantages to this conversion method. First of all, each converter runs at top speed, providing a greater flow of data into the digital interface. Also, with more channels each converter is looking at continuously changing data, rather than jumping from one input to another. This may allow the sample and holds to be eliminated, thus reducing the number of components and the cost. A further advantage to the parallel method is found in an industrial environment where data acquisition is required over a large geographical area. By digitizing the analog inputs at their source and transmitting serial digital data, rather than the original analog signal, to the processing center, the line frequency pickup and ground loop interference are eliminated. Further, the digital signals can be optically coupled for complete electrical isolation.

**Purpose of this thesis:**

The purpose of this work was to design and build a multi-channel microprocessor controlled data acquisition and control system that can read analog inputs from eight different channels using the parallel conversion method, and can control eight separate devices. The system communicates to any computer with an RS-232C port, and responds to a set of commands from a host computer.

In the following chapters, the design steps are discussed, both in hardware and software, accompanied with figures and diagrams. The A/D types and operation are discussed in chapter II, since a knowledge about their characteristics is necessary to understand the design requirement. Chapter III gives a complete description of the design and hardware implementation. The software part of the design and the system operation
are discussed in chapter IV. Finally, the results achieved from using the system and the complete listing of the operating system are given in chapter V.
Chapter II

A/D CONVERTERS

The basic concepts of analog-to-digital conversion have been known and used for a long time, but today's advances in integrated circuit technology have brought great improvements in performance, size, and cost. This chapter discusses the different types of A/D converters that are available, how they work and how to interface them to a microprocessor system.

**Basic conversion definitions:** [2]

Figure 2.1 shows the graph for an ideal 3-bit A/D converter. Since the analog input can assume any value, it must be quantized by partitioning the continuous input into 8 discrete ranges. All analog values within a given range are represented by the same digital code corresponding to the nominal midrange value. Therefore, there is an inherent quantization uncertainty of

![Figure 2.1: Ideal conversion relationship](image-url)
±1/2 LSB, and the only way to reduce it is to increase the number of bits. As can be seen from the graph, all analog inputs between 0 and 1/16 full scale (FS) are represented by an output code of 000. All input between 1/16FS and 3/16FS are represented by an output code of 001, etc.

Figure 2.2 shows a graph with linearity error. This error is defined as a deviation from the line between the end points (zero and full-scale). [2]

![Graph showing linearity error]  
*Figure 2.2: Linearity Error*

Nonlinearity can be expressed as a percentage of full-scale or in fractions of an LSB. ±1/2 LSB is a desirable specification. Linearity can not be externally adjusted.

**A/D types:**

These devices, which may range from tiny ICs to bulky instruments, convert analog input data -usually voltage- into its equivalent digital form. The important characteristics of the A/D converters are accuracy, linearity, resolution, conversion speed, and of course cost. Other less important
characteristics include input range, digital output code, and size. There are five types of A/D converters in general use. These are:

1- Voltage-to-Frequency (V/F).
2- Dual slope integrating.
3- Successive approximation.
4- Tracking.
5- Parallel or flash converters.

Each converter has characteristics that make it most useful for a specific class of applications, based on speed, accuracy, and cost.

**V/F Converters:** [7]

These converters are relatively slow, requiring from several milliseconds to a significant fraction of a second to perform a conversion. On the other hand, they are capable of a high resolution at moderate cost. Figure 2.3 shows a voltage-to-frequency converter. In that circuit, \( R_{IN}, C_{INT}, \) and the op-amp form an analog integrator. All of \( R_{IN} \)'s current flows into \( C_{INT} \) charging it. Therefore the integrator's output changes linearly in a negative direction at a rate proportional to the input. At the same time, \( C_{REF} \) is charged negatively by \( V_{REF} \). When the integrator's output goes negative, the comparator sends a high to the pulse generator which in turn activates switch \( S_1 \) and causes \( C_{REF} \) to discharge into the integrator and returns it to a positive level, causing an output increase equal to:
Figure 2.3: A voltage-to-frequency A/D converter

\[ \Delta V = V_{\text{REF}} \left( \frac{C_{\text{REF}}}{C_{\text{INT}}} \right) \] (1)

The time required for the output to return to zero comes from this eq:

\[ \Delta V = \frac{1}{R_{\text{IN}} C_{\text{INT}}} (V_{\text{IN}} T) \] (2)

Rearranging Eq. 2:

\[ T = \frac{R_{\text{IN}} C_{\text{INT}} \Delta V}{V_{\text{IN}}} \] (3)

Eq. 3 can be written as:

\[ f = \frac{V_{\text{IN}}}{R_{\text{IN}} C_{\text{INT}} \Delta V} \] (4)
Substituting Eq. 1 for $\Delta V$:

$$f = \frac{V_{IN}}{R_{IN} C_{INT}} \cdot \frac{1}{V_{REF}} \cdot \frac{C_{INT}}{C_{REF}}$$

(5)

Which reduces to:

$$f = V_{IN} \cdot \frac{1}{R_{IN} C_{REF}} \cdot \frac{1}{V_{REF}}$$

(6)

Equation 6 shows that the circuit's output frequency depends on the input voltage, reference voltage, $C_{REF}$, and $R_{IN}$.

The digital output code of the analog input frequency is obtained by counting pulses for a period of time. The length of time depends upon the resolution required; for 10 bit resolution, time should be long enough to count 1024 pulses at the full scale frequency. Typical frequencies are tens of kHz; a ten bit conversion at 10 kHz requires about 10 ms. Each additional bit doubles the time requirement. However, extremely high resolution is possible at low cost.

**Dual-slope integration:** [7]

Dual-slope integrating A/D converters are widely used in digital voltmeters and in other single input meters and instruments. Their conversion speed ranges from milliseconds to hundreds of milliseconds. The conversion accuracy of these converters depends on a single reference voltage. Like the V/F converters, a high resolution is possible if long conversion time is acceptable. A block diagram of an integrating converter is shown in figure 2.4, and its timing diagram is shown in figure 2.5.
At the start of conversion the input signal $V_{\text{in}}$ is applied to the integrator for a fixed time $t_1$, during which its output ramps negatively and is equal to:

$$V = \frac{V_{\text{in}} \cdot t_1}{R_{\text{in}} \cdot C_{\text{int}}} \quad (1)$$

A reference voltage $V_{\text{ref}}$ having opposite polarity is then applied to the integrator at the same time a counter is started. The integrator discharges (de-integrates) and its output ramps positively until it reaches zero at time $t_2$, at which the count is stopped and transferred into a latch. The change in voltage $\Delta V$ is given by:

$$\Delta V = \frac{V_{\text{ref}} \cdot t_2}{R_{\text{in}} \cdot C_{\text{int}}} \quad (2)$$
Rearranging Eq. 2:

$$t_2 = \frac{R_{IN} C_{INT} \Delta V}{V_{REF}}$$

(3)

From Eqs. 1 and 3:

$$t_2 = \frac{V_{IN} t_1}{V_{REF}}$$

(4)

Equation 4 shows that the output is proportional to $V_{IN}/V_{REF}$. The conversion accuracy of Dual-slope integration is independent of both the capacitor value and the clock frequency. However, the conversion rate depends on clock frequency and the required resolution. Normally, a few conversions can be done per second. High accuracy, high resolution, and slow speed make dual-slope A/D converters best suited for measuring steady or slowly changing quantities.
**Successive-approximation:** [7]

These are the most common A/D converters used in computers and data-acquisition applications. They are fairly fast, completing a conversion in one to several microseconds, with a typical resolution of eight bits, which provides 256 discrete values. Some successive-approximation converters now provide twelve or even sixteen bits of resolution, which yield 4096 and 65536 discrete values respectively. The conversion time of a successive-approximation converter increases as the number of bits increases. Figure 2.6 shows a block diagram of a typical unit.

![Block diagram of a Successive-approximation converter](image)

**Figure 2.6:** A block diagram of a Successive-approximation converter

The operation consists of comparing the analog input signal against the output of a digital-to-analog (D/A) converter. The conversion process is analogous to a weighing process using a set of known weights for an unknown weight. At the beginning of the conversion, the inputs of the D/A converter are set to zero. The D/A's MSB input is then turned on and the
output is compared with the input signal $V_{in}$. If the D/A's output is less than the input, that bit is left on, otherwise it is turned off. The same process is carried out on each bit in a descending order, until the least-significant bit has been compared. When the process is completed, the status line changes state to indicate that the conversion is finished and the contents of the output forms a binary digital code corresponding to the input signal.

Figure 2.7 shows a typical sequence for a four-bit successive-approximation converter. In this example, $V_{IN}$ is about 11 volts and the LSB represents one volt. First, bit eight is turned on and left on, because 8 is less than 11. Next, bit seven is turned on and then turned back off, because 12 is greater than 11. The process continues until the combination of bits closely approaches the input voltage.

![Figure 2.7: Example of successive-approximation conversion](image)

Accuracy, linearity, and speed of successive-approximation converters are affected by the properties of the D/A converter and the comparator. The settling time of the D/A converter and the response time of
the comparator are generally slower than the switching time of the logic circuits.

**Tracking Converters:** [7]

Tracking converters are slow, but they can follow small changes in input rapidly. Like the successive approximation converter, the tracking converter compares the input to a signal fed back from a D/A converter driven by a counter. Figure 2.8 shows a block diagram of a tracking A/D converter.

![Tracking A/D Converter Diagram](image)

**Figure 2.8:** Tracking A/D Converter

At the start of the conversion, the counter starts its count and continuous until the D/A output equal the input value. If the output of the D/A converter is less than the signal input, the counter counts up. If the D/A output is greater than the input signal, the counter counts down. If the
signal input is constant, the counter output checks back and forth between the two adjacent bit values, hence providing nearly instantaneous tracking of small input change. However, tracking of large changes is slow. The limited speed for a given resolution is a major disadvantage for these converters, since the conversion time for a full scale change is equal to the number of counts multiplied by clock cycle time. For example, a conversion could take as much as 400 μs for a 12 bit resolution and 10 MHz clock frequency.

Tracking converters are not often used for data acquisition, but are useful in track-and-hold applications, in which an input signal is followed until the clock is disabled. They also make excellent peak reading devices, by disabling the counter's down input, the converter will follow input increases and hold the highest reading until the counter is reset.

**Parallel Converters:** [7]

The parallel or "flash" converters provide almost instantaneous A/D conversion. They are fast, with sampling rates of 10 or 20 MHz. Some operate as fast as 100 MHz. Flash converters are mainly used for high speed processing of video data in applications including radars, digital oscilloscopes, and digital TV.

The basic circuit of a parallel converter is shown in figure 2.9. This circuit uses a precision voltage divider to make a series of equal reference voltage increments. An array of comparators compares each of those voltages to the input voltage, and each turns on when the input is greater than its particular reference. The comparators' outputs are applied to a logic circuit to form the digital output code. No clock is required for this
Figure 2.9: Parallel A/D Converter

type of converter; however, in most applications it is necessary to clock the output into a latch in order to hold the reading steady while it is read by a computer.

The disadvantage of the flash converter is the circuit complexity. A parallel A/D converter circuit's complexity doubles with each added bit. A one bit converter requires one comparator, two bits require three comparators, three bits require seven comparators, and so on. Similarly, the logic circuit's complexity increases with the increase of the number of bits. These two factors can affect and limit the size of a parallel A/D converter. For example, an eight bit unit requires 255 comparators. Circuit complexity can be reduced by using a "half-flash" or two-step flash technique, as shown in figure 2.10. Half-flash converters can double the number of resolution bits by using separate circuits for the most and the least significant parts of the analog input. Conversion time is also doubled.
when using this approach. The analog input is applied to the first comparator array to determine the most significant part of the input signal. The output of the first logic circuit is converted to analog form and subtracted from the input to obtain the least significant part. That signal is applied to a second array of comparators to produce the LSB outputs.

A strict precision requirement makes the flash and half-flash converters somewhat expensive.

**Microprocessor Interface:**

A/D converters usually update their outputs periodically and produce incorrect or no outputs between conversions. Computers, on the other hand,
operate sequentially according to a programmed set of instructions and may not be able to receive data when the converter is ready to send it. Therefore, communications must proceed in a handshake process.

Fig. 2.13 shows the basic interface circuit. First, the microprocessor must select the A/D converter through its address select decoder, then tells it to start the conversion by sending a convert signal. Conversion requires some time to complete, so the microprocessor must wait for it to finish. When the DATA-READY signal informs the microprocessor that conversion is complete, it can readdress the A/D to read its output via the data bus.

![Figure 2.11: Basic microprocessor interface for an A/D converter](image)

Data bus width can be a problem. If a twelve-bit converter is used with an eight-bit data bus, external buffers and latches are needed, as well as two read operations, in order to capture all bits.

The type of A/D converter used in this work is the parallel converter since it offers parallel conversion of analog signals with speed limited only
The type of A/D converter used in this work is the parallel converter since it offers parallel conversion of analog signals with speed limited only by the switching time of the comparators and gates. As the input changes, the output code changes. Thus, this is the fastest approach to conversion.

**Sample and Hold**:[2]

A sample and hold (S/H) is a device having an analog input, an analog output and a control input. It has two operating modes: Sample, and Hold. During Sample, S/H acquires the input signal as fast as possible until commanded to Hold, at which time it retains the last value of input signal that it acquired. In data acquisitions systems, S/H's are used either to freeze rapidly changing signals during conversion or to hold multiplexer outputs in a multi-channel analog multiplexed system.

The use of a S/H in data acquisition systems may introduce delays, errors and inaccuracies in data. These delays and errors are caused mainly by the characteristics of the S/Hs. Some of these characteristics are:

- Aperture time. This is the time delay between the hold command and the actual opening of the hold switch.
- Acquisition time. This is the time duration for which an input must be applied for sampling to the desired accuracy.
- Nonlinearity. This is the difference between the output voltage and the input voltage.
- Dynamic sampling error. This is the error in the hold output due to a changing analog input at the time the hold command is given.
- Droop rate. This is a drift of the output at a constant rate caused by the flow of current through the storage capacitor.

Depending on the type of S/H and the system, these characteristics may introduce an unacceptably long delay in a data acquisition system, as well as the inaccuracies mentioned above. Therefore these devices were not used in this work.
Chapter III

THE SYSTEM HARDWARE

The major function of the system is to capture analog data, digitize it and send it to a host computer for processing. Another function is to receive commands from the host computer in order to control devices. This chapter describes the system designed to meet these objectives.

System's block diagram:

The system consists of five major parts: acquisition unit, CPU, memory, control unit, and communication unit, as shown in fig. 3.1. A temporary memory buffer is used during the acquisition period. After acquisition, the data can be sent out serially to a host computer through an RS-232 interface. Communication with the outside world is accomplished by a Universal Synchronous and Asynchronous Receiver and Transmitter. The control unit is an 8-bit I/O port connected to a set of 8 solid state switches. All of these units are controlled by an 8-bit microprocessor.

Figure 3.1: System Block Diagram
The CPU: [4]

The Intel's 8085A-2 microprocessor was chosen for this system for the following reasons:

- It is inexpensive and easily available.
- Its Full TTL compatibility eliminates the need for extra TTL buffers and drivers; it interfaces directly to several peripheral components.
- It incorporates a clock generator on its chip, which requires only a crystal for its operation.
- It has 6 general purpose registers besides the accumulator which makes the programming easier and the processing faster.

In addition to these features, the 8085 has a complete family of peripheral devices such as memory, I/O, USART's, etc.

The 8085 uses a multiplexed data bus. The address is divided between the higher 8-bit address bus and the lower 8-bit address/data bus. In order to make the full 16-bit address available and to interface peripherals, the 8085 address/data bus must be demultiplexed. This is done by connecting a latch (74LS373) to the data bus and strobing it with the ALE signal. The latching of A0-A7 is shown in fig. 3.2.

The 74LS373 is an octal latch. With LE high and OE low, the LS373 acts as a transparent latch. When LE drops low, the LS373 latches its input onto the output lines. Fig. 3.3 shows the timing diagram for the address latching process. ALE goes high during the first clock cycle of a machine cycle. At the same time, the low order address is sent out on the
address/data bus. During the rest of the machine cycle, ALE goes low, therefore the address is latched, and the bus is used for data.

![Diagram of 8085 Low order Address Decoding](image)

**Figure 3.2:** 8085 Low order Address Decoding

![Diagram of 8085 Basic Address/Data timing diagram](image)

**Figure 3.3:** 8085 Basic Address/Data timing diagram
Address decoding: [5]

The 8085 has a 16 bit address bus, therefore it can address 64 K of memory space or I/O devices. A common practice used in addressing devices is to partially decode the address bus to generate chip selects. In doing that, every device is assigned a number of addresses.

In this system design, there are five devices to be addressed. These devices are: the ROM with the system monitor, a control port chip, analog input channel selector, RAM, and a USART. Each of these devices was assigned 2 K bytes of address space. Any one of these 2 K addresses can be specified by a pattern on the 11 address lines A0-A10. The rest of the address lines (A11-A15) are used to locate the 2 K blocks within the 64 K address space. A11-A15 are decoded in order to generate the chip selects. A 74LS138 3-to-8 decoder is used for the primary division of memory space into eight 2 K blocks. With this decoding method, each select line from the LS138 enables one chip. Figure 3.4 shows the system address decoding circuit. In this circuit, three of the select lines (Y3, Y6, Y7) are not used. Y1 select line is used to enable another 74LS138 decoder used to allocate the eight analog channels where they are treated as I/O ports.

A14 and A15 are used to enable the LS138 decoder for the primary decoding. They must be active low in order to enable the chip. Any address above 3FFFFH (i.e. A14=1 or A15=1) would disable the decoder. A11, A12 and A13 are used to select one of the eight output lines of the LS138. The address ranges selected by the eight output lines of the LS138 are shown in table 3.1.
Figure 3.4: Address decoding circuit

Not all devices are treated as memory space. The acquisition unit, control unit, and USART are looked at as I/O ports by the 8085. These ports can be accessed only by the use of IN and OUT instructions. The formats for these instructions are:

\begin{align*}
\text{IN} & \quad \text{nn} \\
\text{OUT} & \quad \text{nn}
\end{align*}

where \( \text{nn} \) is the I/O port number, a value between 00H and FFH.

<table>
<thead>
<tr>
<th>A13</th>
<th>A12</th>
<th>A11</th>
<th>Address range</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0000-07FFH</td>
<td>ROM</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0800-0FFFH</td>
<td>A/D</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1000-17FFH</td>
<td>USART</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1800-1FFFH</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2000-27FFH</td>
<td>CONTROL PORTS</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2800-2FFFH</td>
<td>RAM</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3000-37FFH</td>
<td>Not used</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3800-3FFFH</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Table 3.1: Address ranges for the system’s devices.
When the 8085 is accessing an I/O port, the port number is copied onto A8-A15 as well as A0-A7 during an IN or OUT instruction. For example, if the port number to be addressed is 2AH, the value on the address bus after executing an IN or OUT instruction is 2A2AH. Therefore, the 8085 can only access 256 ports. Since A11, A12, and A13 are used for the primary selection of devices, address lines A8, A9, and A10 are used for selecting port numbers. Table 3.2 shows the system's port ranges.

<table>
<thead>
<tr>
<th>Port Number</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>08-0FH</td>
<td>A/D</td>
</tr>
<tr>
<td>10-17H</td>
<td>USART</td>
</tr>
<tr>
<td>20-27H</td>
<td>Control</td>
</tr>
</tbody>
</table>

Table 3.2: The system port ranges.

The system memory allocation map, as well as the map of the I/O space is shown in figure 3.5.

**The acquisition unit:** [2, 7, 13]

This unit is made up of eight analog input channels. Each can be addressed and read individually. Each channel is an 8-bit A/D converter with a parallel data output. Since the system is a general purpose unit, a compromise had to be made between the cost and the conversion speed of the converters. In designing with A/Ds a general rule of thumb is: the price of the converter rises steeply as the converter performance in both speed and accuracy increases. A slow and inexpensive converter may limit the use of the unit to a certain application. On the other hand, fast converters extend this limit greatly but increases the cost. Hence a moderately high
speed converter of modest cost had to be chosen. For this design, the National Semiconductor ADC0820 was the appropriate choice. It is an 8-bit CMOS high speed half flash A/D converter with a sample-hold function, which eliminates the need for an external sample-and-hold. Another advantage of this converter is its ability to interface to microprocessors as a memory location or I/O port without the need for external interfacing logic. The ADC0820 uses two 4-bit flash converters to make an 8-bit conversion. As described in chapter 2, the analog input is applied to the
first ADC to get the four MSB result. It is then reconverted to an analog signal via a DAC and subtracted from the input. The difference in voltage is converted by the second 4-bit flash ADC to produce the four LSB's, giving an 8-bit data word out. The ADC0820 hookup to the CPU bus and its timing diagram are shown in figure 3.6. It operates as follows:

The converter is selected (CS' is low) when the CPU addresses it with a read command. The A/D converter requests a WAIT, by pulling the processor's READY line low after the falling edge of CS'. The processor will idle as long as the READY line remains low, giving the A/D converter time to respond to its request. RDY will go to the high Z state when the result of the conversion is strobed into the output latch. A 4.7KΩ resistor is provided for the READY line to force it high when RDY becomes inactive. The data is then read from the output latch, and the read instruction is complete.

The acquisition unit is an eight channel unit. It was more appropriate to use an A/D converter for each channel rather than using one A/D in a multiplexed system. The reasons for that are: at the 6-8 bit resolution level, the cost of an analog multiplexer is likely to be a considerable proportion of the cost of a converter at this resolution. The reverse is true at higher resolutions. Also, analog time division multiplexing demands a very high speed converter to achieve the desired sample rate. A slower converter on each channel with digital multiplexing is a better choice. Another reason is that wide dynamic ranges between channels are difficult to handle with analog multiplexing. Small signals less than 1 V would require differential low-level analog multiplexing (which is expensive). The alternative is fixed gain converters on each channel. Furthermore, since analog lines are prone to losses and noise interference, analog multiplexing
is best suited in making measurements at short distances, unlike digital multiplexing.

![Diagram of ADC0820 hookup circuit and its timing diagram]

**Figure 3.6**: ADC0820 hookup circuit and its timing diagram

Generally, analog multiplexers introduce errors into the selected signal path. These include leakage through switches and coupling of control signals into the analog channel. On the other hand, digital multiplexing onto a common bus requires more wiring, but its high switching speed makes possible operation at sampling rates far in excess of those of analog multiplexers. Therefore, digital multiplexing is used in this system to achieve a faster sampling rate. Figure 3.7 shows the design of the acquisition unit.
Figure 3.7: Design of the Acquisition Unit
Since the analog input channels are treated as I/O ports, they can only be selected with an IN or OUT instruction. The ports selected by the eight output lines of the secondary LS138 decoder are shown in table 3.3.

<table>
<thead>
<tr>
<th>Port No.</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>08H</td>
<td>Channel 1</td>
</tr>
<tr>
<td>09H</td>
<td>Channel 2</td>
</tr>
<tr>
<td>0AH</td>
<td>Channel 3</td>
</tr>
<tr>
<td>0BH</td>
<td>Channel 4</td>
</tr>
<tr>
<td>0CH</td>
<td>Channel 5</td>
</tr>
<tr>
<td>0DH</td>
<td>Channel 6</td>
</tr>
<tr>
<td>0EH</td>
<td>Channel 7</td>
</tr>
<tr>
<td>0FH</td>
<td>Channel 8</td>
</tr>
</tbody>
</table>

Table 3.3: Port assignment of the Analog Input Channels

**Control Unit:** [4]

As mentioned earlier, the function of the control unit is to guide devices and instruments. This kind of guidance is in the form of ON and OFF states. Eight devices may be guided at a time. The heart of the control unit in this system is the Intel 8155 static RAM with I/O ports. This chip has 256 bytes of RAM, three general purpose I/O ports, and a command register. The RAM in this chip is used as the monitor stack (stack registers). The IO/M pin selects either the memory portion or the command register and I/O ports. These ports can be programmed through the command register to be either INPUT or OUTPUT ports. The command register consists of eight latches, four of which define the mode of the ports (Input or Output). The other four define the interrupts and timer commands which are not used here. Figure 3.8 shows a block diagram of the 8155 hookup to the CPU.
Programming of the 8155 ports: [4]

The I/O section of the 8155 consists of two 8-bit ports (PA and PB) and one 6-bit port (PC). These ports are programmed as follows:

- **PA**: OUTPUT
- **PB**: INPUT
- **PC**: OUTPUT

The programming of these ports is accomplished by writing the appropriate word into the command register. The command register bit assignment is shown in Appendix B.

**Port A:**

This port is used for controlling the outside world (i.e., system and devices). Each bit of this port is connected through a buffer (74LS244) to a solid state switch in the form of an optoisolator. These optoisolators (MC3010) are optically coupled triacs. With such an isolated triac output stage, control of low power AC loads is possible. The most common use for a triac output is for triggering higher power triacs, allowing the
microprocessor to control thousands of watts of power. Figure 3.9 shows the hookup for one bit of port A.

![Figure 3.9: Port A hookup circuit to an optoisolator](image)

An LED is connected in series with each optoisolator of port A to indicate whether the corresponding switch is ON or OFF. The operation of these control switches is simple. A logic 1 triggers the triac while a logic 0 prevent it from firing.

**PORT B:**

This 8-bit port is programmed to be an input port. It is used to read the communication parameters set by the user to configure the system USART. The communication parameters or mode instruction is switch selectable. Initially all bits of port B are set to logical 1. In order to set a certain bit, it must be pulled down to logic 0 by opening the switch through a 4.7kΩ resistor. An eight bit Dip switch is provided for that purpose as shown in figure 3.10.
Each bit of this port has a special function or assignment. Table 3.4 shows these assignments.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB0</td>
<td>1 Stop bit</td>
</tr>
<tr>
<td>PB1</td>
<td>2 Stop bit</td>
</tr>
<tr>
<td>PB2</td>
<td>Odd Parity/disable</td>
</tr>
<tr>
<td>PB3</td>
<td>Even Parity/disable</td>
</tr>
<tr>
<td>PB4</td>
<td>5 Bit Word Length</td>
</tr>
<tr>
<td>PB5</td>
<td>6 Bit Word Length</td>
</tr>
<tr>
<td>PB6</td>
<td>7 Bit Word Length</td>
</tr>
<tr>
<td>PB7</td>
<td>8 Bit Word Length</td>
</tr>
</tbody>
</table>

Table 3.4: Port B bit assignments

Either one or two stop bits should be selected. If both are set by the user, the system will use 1 stop bit. On the other hand, if neither is selected, the system will issue an error signal through port C. Parity can be chosen as either Odd or Even. It is ignored if neither is selected, and is considered odd if both are selected. Word length must also be chosen. If more than one is selected, the system considers the smaller value. If no word length is specified, an error signal is issued.
**Port C:**

Only three bits of this 6-bit port are used. They serve as status or error indicators for the system. As shown in fig 3.11 each of the three bits is connected to an LED. Initially all bits are cleared. PC0 goes high when an unknown command is received from the host computer. This error is cleared once the system receives a valid command. PC1 and PC2 go high when the number of stop bits or the character length respectively are not specified. PC1 and PC2 can not be cleared unless the user selects the number of stop bits and/or character length and resets the system.

![Figure 3.11: Port C configuration](image)

**Serial Communication:** [4]

As indicated earlier, the user must be able to communicate with the system through a host computer. This is accomplished by using RS232C lines for serial data transfer. A very common serial communication chip is the Intel 8251A USART. It has all the needed functions to implement serial synchronous and asynchronous communication between devices. For this design, the format of transmission and reception is asynchronous since only a few characters need to be transmitted at a time. Appendix C shows the 8251A communication format. Data that is needed to be transmitted from
the system data bus to the outside world, for example, enters the USART
data bus buffer, then is transferred to the transmitter buffer. It is then serialized and transmitted on the TxD line after the appropriate framing information is added to it. Data that is received from the outside world, is collected serially in the receiver buffer through the RxD line, and converted to parallel data according to the appropriate format. Figure 3.12 shows the USART hookup to the system bus.

![USART hookup to the system bus](image)

Figure 3.12: USART interface to the system Bus

The 8251A has two control registers for its operation. They are the mode and the command registers. It also has a status register that can be read by the CPU to determine the status of the device. Instructions written to the mode register are always followed by a command instruction written to the command register. The 8251A distinguishes between accessing these control registers and the data register in the data bus buffer by the use of its C/D control line. Since there is only one line to access these registers, the control words must be written in sequence starting with the mode instruction and followed by command instructions. The mode instruction
must be written immediately after RESET. This instruction defines the baud rate of the device, the number of stop bits, parity type, character length, and the type of operation: synchronous or asynchronous. All subsequent control writes are command instructions. In a command instruction, the transmitter or receiver is enabled/disabled, the Error Flag is reset, a BREAK character is sent, or the whole device is reset and sent back to the mode instruction format. The status register is read when the control register is accessed in a Read mode. This 8-bit register tells when the transmitter is ready to send out a character, when the receiver is ready to send an assembled character to the CPU, or if BREAK or ERROR is detected. Appendix C shows a complete description of the two control registers and the status register.

The 8251A is programmed, and its status is read, using the CS', C/D', RD', and WR' input lines. The truth table for the USART's operation is shown in Table 3.5.

<table>
<thead>
<tr>
<th>CS'</th>
<th>C/D'</th>
<th>RD'</th>
<th>WR'</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Serial Data in =&gt; Data Bus</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Data Bus =&gt; Serial Data out</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>UART Status =&gt; Data Bus</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Data Bus =&gt; Mode/Command Reg.</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>No operation</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Table 3.5: Control signals for the 8251A USART

The transmitting and receiving speed for the 8251A is controlled by the TxC clock input for the transmitter, and RxC clock input for the receiver.
**Baud Rate Generator:** [1]  

Since it is required that the system communicate with a variety of computers, more than one baud rate is necessary. Figure 3.13 shows the circuit diagram for the baud rate generator. It consists of a CD4060 ripple counter and 4.9152 crystal to generate eight switch selectable baud rates. Both of the TxC and RxC pins of the 8251A are connected to the same clock output from the 4060.

![Baud Rate Generator Circuit](image)

**Figure 3.13:** Baud Rate Generator circuit

Baud rate in the 8251A asynchronous mode is a fraction of the actual input frequencies to TxC and RxC. A portion of the mode instruction selects this factor as 1, 1/16, or 1/64 of TxC or RxC. In order to get a 600 baud rate, for example, TxC and/or RxC must be connected to the "divide-by-512" pin of the 4060 (pin no. 13), and the mode instruction must be programmed for a 1/16 frequency factor. With the 4.9152 MHz crystal, the calculation is:

\[
f = \frac{4.9152 \text{ MHz}}{(512 \times 16)} = 600 \text{ Hz}
\]
Table 3.6 shows the frequencies produced by the 4060 and the corresponding baud rates.

<table>
<thead>
<tr>
<th>4060 Pin #</th>
<th>Divide-by</th>
<th>Frequency (Hz)</th>
<th>Baud rate (f/16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>16</td>
<td>307200</td>
<td>19200</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>153600</td>
<td>9600</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>76800</td>
<td>4800</td>
</tr>
<tr>
<td>6</td>
<td>128</td>
<td>38400</td>
<td>2400</td>
</tr>
<tr>
<td>14</td>
<td>256</td>
<td>19200</td>
<td>1200</td>
</tr>
<tr>
<td>13</td>
<td>512</td>
<td>9600</td>
<td>600</td>
</tr>
<tr>
<td>15</td>
<td>1024</td>
<td>4800</td>
<td>300</td>
</tr>
<tr>
<td>1</td>
<td>4096</td>
<td>1200</td>
<td>75</td>
</tr>
</tbody>
</table>

Table 3.6: System Baud rates

An 8-position DIP switch is used to select one of the eight rates.

**The RS-232C Interface:** [8, 9]

There are many standards used to define data communications interfaces and transmission. In terms of the present computer and communications world, the Electronic Industries Association (EIA) RS-232C is the most common serial interface between electronic components. The present system uses this standard for communication, and has a DB-25 connector since it has become almost universally associated with RS-232C. This serial interface has many signals which are not used for computer-to-computer or computer-to-peripheral communication. Only 3 out of the 25 pin connectors are of interest here. The Tx (pin 2) which is used to transmit from the device, Rx (pin 3) which receives data from external devices, and GND (pin 7) which is the signal ground. Figure 3.14 shows the system connection to a computer using the RS-232C bus.
In the RS-232C standard, signal levels are not TTL levels. For data, a positive voltage between 5 and 15 Volts on pin 2 or 3 with respect to pin 7 represents a logic 0 level, and a negative voltage between -5 and -15 Volts on either pin represents a logic 1. The voltage polarities are reversed for logic 0 and 1 on the control lines. Therefore, a voltage level shifting between the EIA and TTL levels must be made.

**RS-232C Drivers/Receivers: [13]**

Nearly all RS-232C applications need both line drivers and receivers that meet EIA specification. A common practice used in such applications is to employ a separate driver (transmitter) and a receiver that operate from a 12 Volt supply. The design of this data acquisition system however, uses a single RS-232C driver/receiver that operates from a +5 Volt supply. This newly designed chip (MAX232) has two drivers, two receivers, and onboard charge pump voltage converters which convert the +5V input power to the +10V needed to generate the RS-232C output levels. Only one driver and one receiver are used in this design. Figure 3.15 shows the MAX232 Driver/Receiver interface circuit.
Memory Buffer:

For certain operations, the system needs an intermediate storage buffer for data. Channel 1 and channel 2 of the acquisition unit are currently the only two channels programmed to use the memory buffer for data storage. These two channels can be used to continuously take data and store it in the memory. Once the memory buffer is full, the CPU can transfer the data from the buffer to the outside world serially. This method of storing data first before transmission makes the sampling rate much faster than taking one reading at a time and sending it out before doing the next. 2 Kbytes of RAM is used for this purpose. It allows 2000 readings to be taken before transmission. The memory buffer can be used in three different ways:
- To store data from channel 1.
- To store data from channel 2.
- To store data from channel 1 and 2 alternately.
This method of data acquisition makes the system appear as a low resolution two channel digital oscilloscope. A block diagram of the memory buffer circuit is shown in figure 3.16.

![Block diagram of the memory buffer circuit](image)

**Figure 3.16:** System Memory Buffer

**The System ROM:**

The system monitor program resides in a Read Only Memory chip. The 2716 2Kbyte EPROM used in the system contains the main program and all the subroutines needed by the CPU to make the system work properly. Figure 3.17 shows a block diagram of the ROM interface circuit.
In the next chapter, the system software is discussed.
Chapter IV

SOFTWARE

This chapter covers the software part of the design. The operating system is discussed in general, along with the layout of the subroutines that control the system's operation.

The discussion starts with the initialization procedures of the system, then it describes the UART and I/O port configurations, the command handling routines, and the communication routines.

Monitor Flowchart:

Figure 4.1 shows a general flowchart for the system monitor. It begins with initializing the system I/O ports and UART. Next a command is read from the UART and tested in order to determine the action the CPU must take before it waits for more commands. An error signal is issued for an unknown command.

System Initialization:

Upon power-up, the 8085 CPU enters what is called the 'Reset Cycle', during which the program counter register (PC) is reset to 0000. Following RESET, the 8085 will start executing instructions at location 0000 with the interrupt disabled. When power is on, the 8085 also can recognize a hardware RESET when the RESET-IN pin is pulled low. Here also, the PC is loaded with the reset address and execution restarts at location 0000.

The system initialization routine starts at address 0000. In this
Figure 4.1: General flowchart of the system monitor
Figure 4.1: General flowchart of the system monitor (continued)
routine, the I/O ports are configured properly, the control switches are reset (OFF), and the stack-pointer register (SP) is initialized to point at the top of the system stack. Also, three bytes in memory called CONtrol SWitch status (CONSW), DEFine OUTput (DEFOUT), and A/D SToRage (ADSTR) are cleared. The functions of these bytes are discussed later. Following the stack initialization, the program proceeds to configure the USART. The system initialization flowchart is shown in figure 4.2 and its program is as follows:

```
MVI A,ODH ;Initialize IO ports as
OUT IOCOM ; PA=out, PB=in, PC=out
MVI A,00H
OUT PORTA ;Reset all control switches
STA CONSW ;Save the switches status
LXI SP,20FFH ;Initialize the system stack
```

![Figure 4.2: System Initialization flowchart](image-url)
**USART Configuration:**

The configuration of the system USART begins by reading the communications format from port B. The asynchronous mode of communication is used in this system. Therefore the mode instruction includes character length and baud rate factor. As mentioned earlier, these formats are user switch selectable. The data byte read by the CPU from port B programs the number of stop bits, even/odd/or ignored parity, and character length. The CPU determines which bit is cleared by rotating the byte eight times in the accumulator through the carry, and testing the carry for 1 or 0. If either the stop bits or the character length bits are not selected, the CPU issues an error signal through port C and halts the execution of the monitor program until these bits are specified and the system is reset manually. Once the communication format bits are determined, the baud rate factor is then added to these bits and written to the USART command register.

The 8251A USART can not begin transmission until the TxEN (Transmitter ENable) bit is set in the command register and the device has received Clear To Send (CTS) input. The RxEN (Receiver Enable) bit must also be set in the command register so that the USART can receive command or data characters. These bits are written into the command register immediately after writing the communication parameters or mode instruction. Figure 4.3 shows a flowchart of the USART configuration subroutine.
Figure 4.3: Flowchart of USART configuration
Figure 4.3: Flowchart of USART configuration (continued)
System Commands:

When the monitor configures the USART it goes into the command routine, where a set of commands can be recognized. The first thing the monitor does in this routine is to read the USART status register to see if the RxRDY is set when data is being received. The monitor constantly checks this bit until it goes high as shown in the following program segment:

```
POLL IN UARCOM ;Read UART status register
ANI 02H ;AND RxRDY bit
JZ POLL ;Try again if not set
```

When RxRDY is set, the monitor proceeds to read the received command from the USART data register and execute the appropriate routine accordingly.

The system commands are classified into three categories: Data Acquisition Commands, Control Unit Commands, and Monitor Commands. A complete list of the system commands and their functions is given in table 4.1.

<table>
<thead>
<tr>
<th>Command</th>
<th>ASCII code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0011 0001</td>
<td>Read the appropriate analog input channel and send the equivalent digital data in either decimal or hexadecimal ASCII code depending on the user's choice.</td>
</tr>
<tr>
<td>2</td>
<td>0011 0010</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0011 0011</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0011 0100</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0011 0101</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0011 0110</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0011 0111</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0011 1000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0011 1001</td>
<td>Read all channels sequentially</td>
</tr>
</tbody>
</table>

Table 4.1a: Acquisition Unit commands
<table>
<thead>
<tr>
<th>Command</th>
<th>ASCII code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0100 0001</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0100 0010</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0100 0011</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0100 0100</td>
<td>turn appropriate</td>
</tr>
<tr>
<td>E</td>
<td>0100 0101</td>
<td>control switch ON</td>
</tr>
<tr>
<td>F</td>
<td>0100 0110</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0100 0111</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>0100 1000</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0110 0001</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0110 0010</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>0110 0011</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0110 0100</td>
<td>turn appropriate</td>
</tr>
<tr>
<td>e</td>
<td>0110 0101</td>
<td>control switch OFF</td>
</tr>
<tr>
<td>f</td>
<td>0110 0110</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>0110 0111</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>0110 1000</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1b: Control Unit commands

<table>
<thead>
<tr>
<th>Command</th>
<th>ASCII code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>0010 0011</td>
<td>decimal ASCII output</td>
</tr>
<tr>
<td>$</td>
<td>0010 0100</td>
<td>hexadecimal ASCII output</td>
</tr>
<tr>
<td>*</td>
<td>0010 1010</td>
<td>software RESET</td>
</tr>
<tr>
<td>+</td>
<td>0010 1011</td>
<td>read chans. 1 &amp; 2 continuously</td>
</tr>
<tr>
<td>-</td>
<td>0010 1101</td>
<td>read channel 1 continuously</td>
</tr>
<tr>
<td>/</td>
<td>0010 1111</td>
<td>set control unit ON</td>
</tr>
<tr>
<td>space bar</td>
<td>0010 0000</td>
<td>reset control unit (OFF)</td>
</tr>
</tbody>
</table>

Table 4.1c: Monitor commands

The ASCII code words for the above commands show some kind of consistency. For example, the upper four bits for all the acquisition unit commands are 0011, and for the monitor commands they are 0010. The system monitor takes advantage of this consistency when it is polling the received commands. The monitor first classifies the type of command by
testing the upper 4-bits of the received command. This is accomplished by masking off the lower nibble of the command first. It is then compared with the values of the four possible upper nibbles (2H, 3H, 4H, 6H). If the upper nibble of the command is not equal to any of the four values, an error signal is sent to port C2 to indicate that an unknown command was received. On the other hand, if the command is equal to one of the four values, the program will jump to the appropriate class of commands as shown in the program segment below. Figure 4.4 shows a flowchart of the command classification method.

![Flowchart](image-url)

**Figure 4.4: Commands Classification Flowchart**
IN UARIN ;read command data from USART
PUSH PSW ;save the command on stack
ANI FOH ;mask the lower nibble of command
CPI 20H ;compare with 20 (monitor command)
JZ MONITOR ;goto monitor routine if equal

CPI 30H ;compare with 30 (A/D command)
JZ ADC ;goto A/D routine

CPI 40H ;compare with 40 (control ON routine)
JZ ONSW ;goto switch on routine

CPI 60H ;compare with 60 (control OFF routine)
JZ OFFSW ;goto switch off routine

MVI A,04H ;send an error signal to port
OUT PORTC ;C2 (LED) if not a valid command
JMP POLL ;continue polling

Once the command class is determined, the system POP the saved command from the stack and proceeds to find out what the command is in that class, then it takes action. Again this is determined by comparing the received ASCII code word with a set of valid ASCII code commands. This classification technique and command polling is far more efficient than going through a list or table of all commands to determine what kind of action the CPU must take. An unknown command or character would cause the system to send an error signal through port C2.

**Monitor Commands:**

There are seven commands in this category that handle the monitor operation according to the user's choice. Each of these commands is discussed separately.
Software RESET: [4]

The system recognizes a software RESET as well as a hardware RESET. The software reset is accomplished by sending the a (*) to the system which causes the CPU to execute the RST 0 instruction. This command resets the CPU program counter, the control switches and the USART.

Control Unit SET/RESET:

When power is applied to the system, all of the control unit outputs are reset (off). This default can be changed before starting any process by sending a (/) which causes the unit outputs to be set (on). A blank or (space bar) would reset or turn off the control unit.

Characters and Data Output:

Data output sent to a host computer as ASCII code can be in either a hexadecimal or decimal format. The default format is hexadecimal. A (\(\ast\)) would change the default into decimal, and a (\($\)) would switch back to hexadecimal. When a \(\ast\) or \($\) is sent to the system, its value is stored in a reserved memory byte called DEFOUT (define output). The CPU examines this byte every time before data is sent out. If DEFOUT contains \(\ast\), the CPU directs the data to a subroutine called DECOUT (decimal output) in order to convert the data to decimal ASCII format. A \($\) value in DEFOUT sends the data to a subroutine called HEXOUT (hexadecimal output) to convert it to hexadecimal ASCII characters before being transmitted to the host computer.
Continuous Data Output:

In addition to reading a single data byte from the analog channels, the system can use channel 1 and 2 to read and store up to 2k-bytes of data continuously with a single command. This is analogous to a dual channel oscilloscope. A "-" command character would fill the entire memory of the system with data from channel 1 before transmitting it. On the other hand, a "+" command character causes the system to fill its memory with data from channel 1 and 2 alternately before transmission. This feature is useful for taking data from quickly changing analog signals or to compare two analog signals.

The continuous data routine begins by loading register pair B and C with the starting address of memory space where data is to be stored before transmission. Also, a counter is set for 2000 counts or locations in register pair D and E. Data is then read from channel 1 and stored in memory location pointed by B and C. Next, the memory location pointer (reg. B & C) is incremented and the counter is decremented and checked if its zero. This process is repeated until all 2000 memory location are filled (i.e. when the counter is zero). The next step is to send the data out to a host computer. This is done by also loading register pair B and C with the starting address of the data stored in memory and setting a counter for 2000 locations. Data is then read from memory location pointed by B and C and sent out. The pointer is then incremented and the counter is decremented and the process is repeated until all data are sent out. The following routine segment shows the process of continuous data acquisition and storing:

```
LXI B,2800H ;beginning address of memory
LXI D,07DH ;counter for 2000 location
```
For dual channel continuous data acquisition, data is loaded into memory from channel 1 and 2 alternately after the memory pointer and the counter are incremented and decremented respectively.

**Acquisition Unit Commands:**

These commands request readings from the analog input channels and send the digital equivalent ASCII code to the host computer. Only one analog conversion is sent out per request. For example, if data is requested from channel 1, a "1" command character must be sent from the host computer. The system would read channel 1 upon receiving the command and save the data byte in a temporary memory location called ADSTR. It would then
examine the DEFOUT byte to determine if decimal or hexadecimal output is required before sending the data back to the host computer. The ADC routine for channel 1 looks like this:

```
ADCO  POP PSW     ;restore the saved command
  CPI 31H       ;compare the command with ASCII "1"
  JNZ ADC1     ;try other channels if not equal
  IN 08H       ;read data from channel 1
  STA ADSTR    ;store data temporarily
  JMP CRTST    ;goto data output routine

ADC1  CPI 32H     ;compare the command with "2"
```

The ADC routines for the other channels are similar to the above, with the exception that the first instruction compares the command with the ASCII code for the particular channel number. Also, the read instruction (i.e. IN) would have the channel assigned port number for the operand.

The routine for examining DEFOUT is as follows:

```
CRTST LDA DEFOUT  ;load DEFOUT value into accumulator
  CPI 23H       ;compare with ASCII "#"
  JNZ HX       ;goto decimal conversion subroutine
  CALL DECRO    ;goto decimal conversion subroutine
  JMP POLL      ;continue polling

HX    CALL HEXRO  ;goto hex conversion subroutine
      JMP POLL   ;continue polling
```

**Hexadecimal Byte Output:** [10]

Data and commands are transmitted and received in ASCII representation codes on RS232C transmission lines. These codes are either in decimal or hexadecimal. In order to send the data in hexadecimal code, the upper nibble must be converted into an ASCII character and sent out
before the lower nibble. The structure of the hexadecimal data output subroutine can be seen in the following program:[10]

```assembly
HEXRO LDA ADSTR ;restore the saved data
RAR ;rotate right four times
RAR ; to move the upper nibble
RAR ; instead of the lower nibble
RAR
ANI 0FH ; mask the upper four bits off
ADI 90H ;decimal add 90 BCD
DAA ; form the 4 bit BCD digits
ACI 40H ;decimal add 40 BCD+carry
DAA
CALL OUT1 ; send the upper nibble out
LDA ADSTR ;restore the original data
ANI 0FH ; mask the upper four bits off
ADI 90H ;decimal add 90 BCD
DAA ; form the 4 bit BCD digits
ACI 40H ;decimal add 40 BCD+carry
DAA
CALL OUT1 ; send the upper nibble out
RET ; go back to main program
```

**Decimal Byte Output:** [12]

In this subroutine, data bytes are converted to their equivalent decimal ASCII code. Since there are 8 bits in a data word, the maximum possible decimal number is 255. The following relation shows the representation of a 3 digit decimal number:

\[ X \times 100 + Y \times 10 + R = N \]

From the above relation, a formula can be derived to determine the values of \( X, Y, \) and \( R \). First the data byte is divided by \((100)_d\) to determine \( X \) or the most significant digit. The next step is to divide by \((10)_d\) to determine the second digit \( Y \). Finally the remainder \( R \) is the least significant digit. These
digits are sent out to the host computer sequentially starting from the most significant digit and ending with the least significant digit.

The 8085 instruction set has no code for a division operation, therefore a subroutine called divide was written to accomplish the division technique. The structure of the entire decimal data output is shown in the following program:[12]

```
DECRO LDA ADSTR ;restore the saved A/D data
    MOV L,A ;load register pair H&L with
    MVI H,00H ;data to make 16 bit word
    MVI C,64H ;decimal 100 (divisor)
    CALL DIVID ;goto divide subroutine
    MVI C,0AH ;decimal 10 (divisor)
    CALL DIVID ;goto divide subroutine
    MOV A,L ;place remainder in Reg. A
    ADI 30H ;convert to ASCII
    CALL OUT1 ;goto data out subroutine
    RET ;return to main program

;DIVID
    MVI B,08H ;count=8
    DIV ;shift dividend, quotient
    DAD H ;is most significant part of dividend
    MOV A,H ;greater or equal to divisor?
    SUB C ;yes, subtract divisor
    JC CNT ;and add 1 to quotient
    MOV H,A
    INR L
    CNT ;decrement count
    DCR B
    JNZ DIV ;continue if not 0
    ADI 30H ;convert to ASCII
    MOV L,H
    MOV H,00
    CALL OUT1 ;goto data out subroutine
    RET ;return to the previous subroutine
```
**Output Subroutine**:  

All data that are available to be sent out whether in decimal or hexadecimal have to go through a data out subroutine. This short routine sends one data word at a time to the USART data register. It then reads the USART status register to see if the transmitter is ready for more data. This is done by checking TxRDY bit in the status register. If TxRDY bit is high more data can be sent out, otherwise the system must wait until TxRDY bit goes high before sending more data out. The data out subroutine is as follows:

```
OUT I       OUT UAROUT ;send data to USART data register
TRY         IN UARCOM ;read USART status register
ANI 01H     ;TxRDY bit ready yet?
JZ TRY      ;try again if not
RET         ;return to the previous routine
```

**Control Unit Commands**:  

Since the control unit consist of 8 switches, two sets of commands were necessary to turn these switches ON or OFF. An upper case character (A-H) sent to the system turns the appropriate switch ON, and a lower case character (a-h) turns it OFF. Each switch has one routine to turn it ON and another to turn OFF.

When a switch is to be turned ON, the monitor loads the switch's status byte in CONSW into the accumulator first. The particular switch bit is used to modify this byte, which is then sent out to port A before being saved in CONSW. The modification is done by ORing the contents of CONSW with a 1 in the bit that corresponds to the particular switch. This avoids
resetting any other switches. For example, if switch B is to be turned on the command "B" must be sent to the system. This command is tested by the control unit routine to see which switch must be turned on. Once this is determined, "switch B ON routine" is executed as shown below:

```
LDA CONSW ;load switches status in reg A
ORI 0000 0010 ;add switch B
OUT PORTA ;turn on switch B
STA CONSW ;save the switches status
JMP POLL ;continue polling
```

The opposite is true when turning a switch OFF. The switch bit in CONSW that corresponds to the particular switch is ANDed with a zero and saved before being sent to portA. The other switches are not affected by this procedure. Again each switch has its own "OFF routine" as shown in the following program segment for switch B:

```
LDA CONSW ;load switches status in reg A
ORI 1111 1101 ;subtract switch B
OUT PORTA ;turn off switch B
STA CONSW ;save the switches status
JMP POLL ;continue polling
```

A complete listing of the system monitor program is shown in Appendix A.

This conclude the discussion on the software part of the design. Following in the next chapter, is the presentation of the results and conclusions drawn from testing the system.
Chapter V

CONCLUSION

In the data acquisition and control system, the hardware is organized into ten blocks. The system was assembled and built using wire wrapping techniques. The whole assembly took one circuit board (figure 5.1).

![Diagram of hardware design](image)

**Figure 5.1**: Organization of the hardware design

**Testing the system:**

Testing of the system was done by initiating software requests from a host computer. For this purpose, a Tandy 200 portable computer was used as the host computer. It has a serial port (RS232C) and multi-communication parameters that can be set by the user to suit any need. The computer and the system were connected via their RS232C ports with 10 feet of unshielded cable and their communication parameters were matched. Three programs written in BASIC on the Tandy 200 computer were then
executed requesting data and controlling a low power AC load. All three programs worked perfectly in obtaining data and controlling the load.

The first program requests temperature readings from a thermistor through channel 1 and plots them on a temperature-time graph on the screen of the host computer. The entire program takes approximately 13 seconds to request and plot 200 data points. The program would take less time to execute if it were written in machine language on a faster computer.

The second program reads the temperature from channel 1 constantly and controls a low power AC motor through switch A. Initially, the AC motor is off. It is turned on when the temperature being measured rises above 70°C and is turned off if it is below that temperature.

The third program uses a single command to read data from all channels and display the result on the screen. When it was tested, no inputs were connected to the analog channels; therefore the system sent random data.

The execution of these programs was repeated with all the available baud rates of the system, and they always ran flawlessly. Appendix D shows the listings for these programs.

**System's Hardware limitations:**

The system was designed with the intention of having a fairly fast data acquisition unit. Its 8085A-2 CPU can run with a 10MHz clock. On the other hand, the 8251A USART cannot run at a clock rate higher than 6.144MHz; therefore a wait state was introduced in the early stages of the design. It required more hardware. This wait state delay caused a failure
to respond to commands in some rare instances or even caused the sending out of incorrect data. Because of this, and because the wait state only increases the CPU processing speed by a small percentage, a 6.144MHz clock with no wait state was used instead of the 10MHz clock. This speed limitation imposed by the system UART has no effect on either the A/D's conversion time or the transmission speed.

Because of the many components involved in the design, power transients may affect the operation of the system. Therefore 0.1 μF disk capacitors were connected on different parts of the power bus between +5V and GND. There is an average of one capacitor for each IC chip.

Another limitation to the system is the signal voltage range applied to the analog channels. The analog input signal must be between 0 and +5V DC. Incorrect data would result for signals below or above that range.

**Software characteristics:**

All software commands were tested and worked perfectly. The control unit software routines are efficient and fast in responding to its commands. Data acquisition in the continuous sampling mode is also fast.

Perfect software can never be achieved. The monitor program has many routines and subroutines, causing delays for certain operations. Although high speed A/D converters are used in the system, acquisition time can be long for several reasons. First, the system monitor takes time to decode the acquisition command. Once that is done, data is read from the requested channel. It is then converted into decimal or hexadecimal ASCII code words. Finally, these words are transmitted serially to a host.
computer. Serial transmission is the most time consuming factor in this work. The following formula determines the maximum transmission speed the system can use for one data word:

\[
\frac{\text{Number of bits in each data word}}{\text{Maximum baud rate}} = \text{Time for one data word}
\]

Data words transmitted and received by the system are usually 9 bits including a start bit and a stop bit. Using the above formula gives:

\[
\frac{9 \text{ bits}}{19200 \text{ bit/sec}} = 469 \mu\text{sec}
\]

The number above represents the minimum amount of time it takes the system to transmit or receive one data word. While it takes the A/D converter 2.5 \mu\text{sec} to do the conversion, the system uses 936 \mu\text{sec} to transmit the data in hexadecimal ASCII, and 1404 \mu\text{sec} in decimal ASCII. In addition, more time is needed for processing before transmission. The only case in which the data acquisition can be done very quickly is when the system reads and stores a block of data in RAM before the ASCII conversion and transmission. Table 5.1 shows the time it take the system commands to execute and give results. These numbers were calculated by adding the machine cycles for all the instructions for the particular command and multiplying this by the CPU cycle time.
<table>
<thead>
<tr>
<th>Command type</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Unit OFF</td>
<td>85 µsec</td>
</tr>
<tr>
<td>Control unit ON</td>
<td>79 µsec</td>
</tr>
<tr>
<td>Single data acquisition</td>
<td></td>
</tr>
<tr>
<td>in decimal ASCII</td>
<td>2070 µsec</td>
</tr>
<tr>
<td>in hexadecimal ASCII</td>
<td>1577 µsec</td>
</tr>
<tr>
<td>Continuous acquisition</td>
<td></td>
</tr>
<tr>
<td>channel 1</td>
<td>18 µsec</td>
</tr>
<tr>
<td>channel 1 and 2</td>
<td>24 µsec</td>
</tr>
</tbody>
</table>

Table 5.1: Execution time of the system commands

Note that the execution time for the continuous data acquisition represents the time period for each data read and stored in RAM before transmission. In addition to all the above numbers, 469 µsec is required for the system to receive a command from the host computer.

Data from channel 1 and 2 in the continuous mode can not be transmitted at a speed higher than 4800 baud. There is also a delay introduced by the number of subroutines the CPU has to go through while the USART transmitter is waiting for more data. The OUT1 subroutine (appendix D) shows that the CPU waits for the transmitter to send a data word before going back to continue. This process can be made more efficient if the CPU is allowed to do the ASCII conversion while the transmitter is busy. This could be done by using the USART's (READY pin) to interrupt the CPU when the transmitter is ready.

The entire monitor program occupies just over half of the reserved ROM space. It is made up of several subroutines so that its operation is fast. It could be made smaller and take less ROM space, but slower operation would result.
Range of applications:

The data acquisition and control system discussed in this thesis is one of the easiest I/O boards to program and run. With its multiple baud rate and serial port compatibility it can be interfaced to any computer with an RS232C communication port and may even be used with a "dumb" terminal.

Applications of the system are numerous and are only limited by the speed of the system and the analog input voltage range. Some applications include: data logging, process control, energy management, meteorology, robotics, security systems, product testing, laboratory and medical instrumentation. In general, the system can handle many continuous signals produced by various types of sensors and transducers. During testing, the system was used around the clock for almost three days to monitor conditions (specifically temperature) and transmit test data to the host computer. It has proven to be very reliable in spite of the extreme conditions it was subjected to during the tests.

Expanding the system:

Expanding the system is possible with some hardware and software changes. More input channels may be added. By using the free select lines from the primary address decoder, a secondary decoder can be enabled to choose more A/D. Additional RAM memory can also be used. The analog input voltage range can be extended with a pre-amp or voltage divider. Finally, remote operation of the system is possible through the use of a modem.
Summary:

This system can monitor 8 analog data channels and respond to a change in them by controlling 8 low power AC output circuits. It can also continuously take data from one or two channels for up to 2000 readings. It has been used to take temperature readings and to turn a small fan on and off.
Appendix A

OPERATING SYSTEM LISTING
ABSTRACT

This program is a small monitor for the data acquisition and control unit interfaced to any computer via an RS-232C port. The program responds to a set of commands from a host computer to read and send back data from 8 different channels and control a set of 8 solid state switches.

PROGRAM ORGANIZATION

The program is organized as follows:

1) System I/O Ports Configuration
2) System UART Configuration
3) Data Acquisition Routine
4) Control Unit Routine

MONITOR EQUATES

CONSW EQU 2002H ;CONTROL UNIT STATUS BYTE
DEFOUT EQU 2004H ;HEX OR DEC OUTPUT BYTE
ADSTR EQU 2006H ;A/D OUTPUT TEMPORARY STORAGE BYTE
DAUSP EQU 20FFH ;START OF MONITOR STACK
UAROUT EQU 10H ;USART DATA BUS OUT REGISTER
UARIN EQU 11H ;USART DATA BUS IN REGISTER
UARCOM EQU 11H ;USART MODE/STATUS REGISTER
IOCOM EQU 20H ;8155 I/O PORTS COMMAND REGISTER
PORTA EQU 21H ;PORT A ADDRESS
PORTB EQU 22H ;PORT B ADDRESS
PORTC EQU 23H ;PORT C ADDRESS

BEGIN PROGRAM

;SYSTEM I/O PORTS CONFIGURATION

ORG 0
MVI A,0DH ;INITIALIZE I/O PORTS
OUT IOCOM ;Pa=OUT, Pb=IN, Pc=OUT
MVI A,0DH
OUT IOCOM
MVI A,0DH
OUT IOCOM
STA CONSW ;SAVE THE SWITCHES STATUS
MVI A,23H
STA DEFOUT ;DEFINE OUTPUT CHARACTERS (HEX)
AVOCET SYSTEMS 8085/280 ASSEMBLER - VERSION 1.05M SERIAL #00355

SOURCE FILE NAME: DACON.ASM

0010 31FF20  LXI SP,DAUSP ; INITIALIZE DAU STACK

0013 0822  UART IN PORTB ; READ COMMUNICATION FORMATS
0015 1F   H0 ; ROTATE RIGHT THRU CARRY
0016 0A1F00 JC BIT2 ; GOTO BIT2 IF CARRY=1
0019 1640 MVI D,40H ; 1 STOP BIT (MODE BYTE)
001B 1F   H0 ; ROTATE (IGNORE THE NEXT BIT)
001C C32500 JMP BIT3 ; TEST BIT3
001F 1F   H0 ; ROTATE RIGHT
0020 DA6800 JC ERR2 ; IF CARRY, SEND AN ERROR SIGNAL
0023 16C0 MVI D,0CH ; 2 STOP BITS (MODE BYTE)
0025 1F   H0 ; ROTATE
0026 DA3300 JC BIT4 ; GOTO BIT4 IF CARRY
0029 F5 PUSH PSW ; SAVE PARAMETER BYTE
002A 7A MOV A,D ; PLACE USART MODE BYTE IN A
002B F610 ORI 10H ; ADD ODD PARITY BIT
002D 57 MOV D,A ; SAVE UART MODE BYTE
002E F1 POP PSW ; RESTORE PARAMETER BYTE
002F 1F   H0 ; SHIFT (IGNORE THE NEXT BIT)
0030 C33D00 JMP BIT5 ; TEST BIT 5
0033 1F   H0 ; ROTATE TO TEST BIT 4
0034 DA3D00 JC BIT5 ; IF CARRY (DISABLE PARITY)
0037 F5 PUSH PSW ; SAVE PARAMETER BYTE
0038 7A MOV A,D ; PLACE USART MODE BYTE IN ACC
0039 F630 ORI 30H ; ADD EVEN PARITY BIT
003B 57 MOV D,A ; SAVE UART MODE BYTE
003C F1 POP PSW ; RESTORE PARAMETER BYTE
003D 1F   H0 ; SHIFT (IGNORE NEXT BIT)
003E DA4700 JC BIT6 ; IF CARRY TRY BIT 6
0041 7A MOV A,D ; PLACE USART MODE BYTE
0042 F602 ORI 02H ; ADD (5 BIT CHAR LENGTH) TO MODE
0044 C36200 JMP UAREG ; GOTO USART REG ROUTINE
0047 1F   H0 ; ROTATE TO TEST BIT 6
0048 DA5100 JC BIT7 ; IF CARRY TRY BIT 7
004B 7A MOV A,D ; PLACE USART MODE BYTE IN A
004C F605 ORI 0BH ; ADD (6 Bit CHAR LENGTH) TO MODE
004E C36200 JMP UAREG ; GOTO USART REG ROUTINE
0051 1F   H0 ; ROTATE TO TRY BIT 7
0052 DA5B00 JC BIT8 ; TRY BIT 8 IF CARRY
0055 7A MOV A,D ; PLACE USART MODE BYTE IN A
0056 F60A ORI 0AH ; ADD (7 BIT CHAR LENGTH) TO MODE
0058 C36200 JMP UAREG ; GOTO USART REG ROUTINE
005B 1F   H0 ; ROTATE TO TRY BIT 8
IN UARIN ; READ COMMAND DATA FROM USART
PUSH PSW ; SAVE THE COMMAND
ANI 0FH ; MASK THE LOWER NIBBLE

CPI 20H ; IS IT A MONITOR COMMAND?
JZ MONITOR ; IF SO GOTO MONITOR ROUTINE

CPI 30H ; IS IT A/D COMMAND
JZ ADC ; GOTO A/D ROUTINE

CPI 40H ; IS IT AN ON CONTROL SWITCH
JZ ONSW ; GOTO ON SWITCH ROUTINE

CPI 60H ; IS IT AN OFF CONTROL SWITCH
JZ OFFSW ; GOTO OFF SWITCH ROUTINE

MVI A, 04H ; SEND AN ERROR SIGNAL TO PORT
OUT PORTC ; C3 (LED) IF NOT A VALID COMMAND
JMP POLL ; CONTINUE POLLING

OUT PORTA ; C3 (LED) IF NOT A VALID COMMAND

MVI A, 00H ; IF IT IS TURN ALL CONTROL SWITCHES OFF
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

MVI A, OFFH ; IF IT IS TURN ALL CONTROL SWITCHES ON
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

OUT PORTA ; C3 (LED) IF NOT A VALID COMMAND

MVI A, 00H ; IF IT IS TURN ALL CONTROL SWITCHES OFF
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

MVI A, OFFH ; IF IT IS TURN ALL CONTROL SWITCHES ON
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

OUT PORTA ; C3 (LED) IF NOT A VALID COMMAND

MVI A, 00H ; IF IT IS TURN ALL CONTROL SWITCHES OFF
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

MVI A, OFFH ; IF IT IS TURN ALL CONTROL SWITCHES ON
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

OUT PORTA ; C3 (LED) IF NOT A VALID COMMAND

MVI A, 00H ; IF IT IS TURN ALL CONTROL SWITCHES OFF
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

MVI A, OFFH ; IF IT IS TURN ALL CONTROL SWITCHES ON
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

OUT PORTA ; C3 (LED) IF NOT A VALID COMMAND

MVI A, 00H ; IF IT IS TURN ALL CONTROL SWITCHES OFF
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

MVI A, OFFH ; IF IT IS TURN ALL CONTROL SWITCHES ON
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

OUT PORTA ; C3 (LED) IF NOT A VALID COMMAND

MVI A, 00H ; IF IT IS TURN ALL CONTROL SWITCHES OFF
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING

MVI A, OFFH ; IF IT IS TURN ALL CONTROL SWITCHES ON
STA CONSW ; SAVE CONTROL SWITCHES STATUS
JMP POLL ; CONTINUE POLLING
SOURCE FILE NAME: DACON.ASM

005C D7C00 JC ERR3 ;SEND AN ERROR SIGNAL
005F 7A MOV A, D ;PLACE USART MODE BYTE IN A
0062 F40E ORI OEH ;ADD (8 BIT CHAR LENGTH) TO MODE

******************************************************************************

;USART CONFIGURATION
;THIS ROUTINE WRITES THE COMMUNICATION FORMATS INTO
;THE USART MODE REGISTER THEN PROGRAM THE COMMAND REGISTER

0062 D311 UAREG OUT UARCOM ;WRITE PARAMETERS INTO MODE REG
0064 3E07 MV A, 07H ;SET TXEN, DTR, RXEN
0066 D311 OUT UARCOM ;WRITE TO COMMAND REG
0068 C38000 JMP POLL ;GO TO POLL ROUTINE

******************************************************************************

;USART ERROR SIGNALS
;THIS ROUTINE SENDS AN ERROR SIGNAL TO PORTC1 IF NO
;STOP BIT IS SELECTED

0068 3E01 ERR2 MV A, 01H ;SEND NO STOP BIT ERROR SIGNAL
006A D323 OUT PORTC ;TO PC1
006F CDC303 CALL DELAY ;GOTO DELAY SUBROUTINE
0072 3E00 MV A, 00H ;CLEAR A REG (MAKE ERROR LED
0074 D323 OUT PORTC ;TO FLASH)
0076 CDC303 CALL DELAY ;GOTO DELAY SUBROUTINE
0079 C36000 JMP ERR2 ;REPEAT

******************************************************************************

;THIS ROUTINE SENDS AN ERROR SIGNAL TO PORTC2 IF THE
;CHARACTER LENGTH IS NOT SELECTED

007C 3E02 ERR3 MV A, 02H ;SEND NO CHARACTER LENGTH
007E D323 OUT PORTC ;BIT SPECIFIED ERROR SIGNAL
0080 CDC303 CALL DELAY ;TIME DELAY
0083 3E00 MV A, 00H ;CLEAR A REG (MAKE ERROR LED
0085 D323 OUT PORTC ;TO FLASH)
0087 CDC303 CALL DELAY ;TIME DELAY
008A C37000 JMP ERR3 ;REPEAT

******************************************************************************

;RECEIVED COMMANDS POLL
;THIS ROUTINE CLASSIFIES THE RECEIVED COMMANDS AS
;DATA AQUISITION COMMANDS, CONTROL COMMANDS, OR
;MONITOR COMMANDS SUCH AS SOFTWARE RESET, DECIMAL
;OR HEXADECIMAL OUT...ETC AND EXECUTE THE
;APPROPRIATE ROUTINE

008D DB11 POLL IN UARCOM ;READ USART STATUS REG
008F E502 ANI 02H ;RECEIVER READY YET?
0091 C80000 JZ POLL ;TRY AGAIN IF NOT

0094 3E00 MV A, 00H ;CLEAR THE CHAR ERROR LED
0096 D323 OUT PORTC ;AT PC1
SINGLE CHANNEL OSCILLOSCOPE ROUTINE

THIS ROUTINE ACTS LIKE AN OSCILLOSCOPE BY FILLING A 2K-BYTE OF MEMORY WITH DATA FROM CHANNEL 1 FIRST; THEN SEND THEM OUT TO THE HOST COMPUTER.

BEGINNING ADDRESS OF MEMORY : COUNTER FOR 2000 LOCATION

LOAD CHANNEL HEX OR DEC DEFINITION

HEXADECIMAL OUTPUT?

GOTO DECRO IF NOT

RESTORE CONTENT INTO REG A

SAVE VALUE TEMPORARLY

SEND DATA TO HEXRO FOR CONVERSION

INCREMENT MEM LOCATION POINTER

DECREMENT COUNTER

CHECK THE COUNTER

IF IT'S ZERO YET

SEND THE NEXT BYTE

CONTINUE POLLING

LOAD NEXT BYTE

CONTINUE POLLING

SEND THE NEXT BYTE
DUAL CHANNEL OSCILLOSCOPE ROUTINE

This routine acts as a dual channel oscilloscope. It begins by filling a 2K-byte of memory with data from channel 1 and 2 alternately, then sending them out to the host computer.

0139 C38D00  JMP POLL ;CONTINUE POLLING

013C FE28  SCOP2  CPI 28H ;IS IT "++" DUAL CHANNEL SCOPE
013E C28901  JNZ ERR1 ;SEND AN ERROR SIGNAL IF NOT
0141 010228  LXI B,2800H ;BEGIN MEMORY ADDRESS
0144 11E803  LXI D,03E8H ;SET UP A COUNTER
0147 0B08  INPUT2  IN 08H ;SAMPLE CHANNEL 1
0149 02  STAX B ;SAVE IN MEM LOCATION POINTED BY B
014A 03  INX B ;INCREMENT MEMORY POINTER
014B 0909  IN 09H ;SAMPLE CHANNEL 2
014D 02  STAX B ;SAVE IN SUBSEQUENT MEM LOCATION
014E 03  INX B ;INCREMENT MEMORY LOCATION
014F 18  Dcx D ;DECREMENT COUNTER
0150 7A  MOV A,D ;CHECK THE COUNTER
0151 B3  ORA E ;IF IT'S ZERO YET
0152 C24701  JNZ INPUT2 ;SAMPLE AGAIN IF COUNTER NOT ZERO
0155 010228  LXI B,2800H ;BEGINNING OF MEM ADDRESS
0158 110307  LXI D,0700H ;SET UP THE COUNTER
015B 3A0420  LDA DEFOUT ;LOAD HEX OR DEC DEFINITION
015E FE24  CPI 2AH ;HEXADECIMAL OUT?
0160 C27401  JNZ OUTPUT4 ;GO TO DECRO IF NOT

0163 0A  OUTPUT3  LDA X B ;RESTORE CONTENT INTO REG A
0164 320620  STA ADSR ;SAVE VALUE TEMPORARILY
0167 CDE103  CALL HEXRO ;SEND DATA TO HEXRO
016A 03  INX B ;INCREMENT MEM LOCATION POINTER
016B 18  Dcx D ;DECREMENT COUNTER
016C 7A  MOV A,D ;CHECK COUNTER
016D B3  ORA E ;IF IT'S ZERO YET
016E C25301  JNZ OUTPUT3 ;SEND THE NEXT BYTE
0171 C38D00  JMP POLL ;CONTINUE POLLING

0174 0A  OUTPUT4  LDA X B ;RESTORE CONTENT INTO REG A
0175 320620  STA ADSR ;SAVE VALUE TEMPORARILY
0178 C5  PUSH B ;SAVE REG B & C
0179 05  PUSH D ;SAVE REG D & E
017A C00204  CALL DECRO ;SEND DATA TO DECRO
017D 01  POP D ;RESTORE REG D & E
017E C1  POP B ;RESTORE REG B & C
017F 03  INX B ;INCREMENT MEM LOCATION POINTER
0180 1B  Dcx D ;DECREMENT COUNTER
0181 7A  MOV A,D ;CHECK COUNTER
0182 B3  ORA E ;IF IT'S ZERO YET
0183 C27401  JNZ OUTPUT4 ;SEND THE NEXT BYTE
AVOCET SYSTEMS 8085/Z80 ASSEMBLER - VERSION 1.05M SERIAL #00355

SOURCE FILE NAME: DACON.ASM

0186 C38000 JMP POLL ;CONTINUE POLLING
0189 3E04 ERR1 MVI A, 04H ;SEND AN ERROR SIGNAL
0188 0233 OUT PORTC ;TO PORT C3 OTHERWISE
018D C38000 JMP POLL ;CONTINUE POLLING

:*******************************************************************************
: ADC ROUTINE
: THIS SEGMENT SAMPLES THE SYSTEM'S REQUESTED A/D CHANNEL AND SEND OUT THE DATA EITHER IN DECIMAL
: OR HEXADECIMAL ASCII, DEPENDING ON THE USER'S CHOICE

0190 F1 ADC POP PSW ;RESTORE THE SAVED COMMAND
0191 FE31 CPI 31H ;CHANNEL 1?
0193 C29E01 JNZ CHAN1 ;CONTINUE CHECKING IF NOT
019E DB08 IN 08H ;SAMPLE CHANNEL 1
0198 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
019B C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT
019E FE32 CHAN2 CPI 32H ;CHANNEL 2?
01A0 C2AB01 JNZ CHAN2 ;CONTINUE CHECKING IF NOT
01A3 DB09 IN 09H ;SAMPLE CHANNEL 2
01A5 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01A8 C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT
01AB FE33 CHAN3 CPI 33H ;CHANNEL 3?
01A0 C2B001 JNZ CHAN3 ;CONTINUE CHECKING IF NOT
01B0 DB0A IN 0AH ;SAMPLE CHANNEL 3
01B2 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01B5 C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT
01BB FE34 CHAN4 CPI 34H ;CHANNEL 4?
01BA C2C501 JNZ CHAN4 ;CONTINUE CHECKING IF NOT
01BD DB0B IN 0BH ;SAMPLE CHANNEL 4
01BF 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01C2 C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT
01C5 FE35 CHAN5 CPI 35H ;CHANNEL 5?
01C7 C2D001 JNZ CHAN5 ;CONTINUE CHECKING IF NOT
01CA DB0C IN 0CH ;SAMPLE CHANNEL 5
01CC 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01CF C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT
01D2 FE36 CHAN6 CPI 36H ;CHANNEL 6?
01D4 C2DF01 JNZ CHAN6 ;CONTINUE CHECKING IF NOT
01D7 DB0D IN 0DH ;SAMPLE CHANNEL 6
01D9 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01DC C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT
01DF FE37 CHAN7 CPI 37H ;CHANNEL 7?
01E1 C2EC01 JNZ CHAN7 ;CONTINUE CHECKING IF NOT
SOURCE FILE NAME: DACON.ASM

01E4 DB0E IN 0EH ;SAMPLE CHANNEL 7
01E6 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01E9 C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT

01EC FE38 CHAN8 CPI 38H ;CHANNEL 8?
01EE C2F901 JNZ CHAN9 ;CONTINUE CHECKING IF NOT
01F1 DB0F IN OFH ;SAMPLE CHANNEL 8
01F3 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
01F6 C3CD03 JMP CRTST ;CHECK FOR HEX OR DEC OUTPUT

01F9 FE39 CHAN9 CPI 39H ;ALL CHANNELS?
01FB CA0502 JZ ALLCH ;SAMPLE ALL CHANNELS
01FE 3ED4 MVI A,0AH ;IF NOT A VALID COMMAND
0200 D323 OUT PORTC ;THEN SEND AN ERROR SIGNAL
0202 C3B000 JMP POLL ;CONTINUE POLLING

;***************************************************************************
;THIS PART SAMPLES ALL CHANNELS SEQUENTIALLY AND SEND
;THE DATA OUT IN DECIMAL OR HEXADECIMAL DEPENDING ON
;THE USER CHOICE.

0205 3A0420 ALLCH LDA DEOUT ;DECIMAL?
0206 FE23 CPI 23H ;DECIMAL?
020A CA5002 JZ ALLDC ;SAMPLE ALL CHANNEL DECIMAL

020D DB08 ALLHX IN 08H ;SAMPLE CHANNEL 1
020F 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0212 CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE

0215 DB09 IN 09H ;SAMPLE CHANNEL 2
0217 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
021A CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE

021D DB0A IN 0AH ;SAMPLE CHANNEL 3
021F 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0222 CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE

0225 DB0B IN 0BH ;SAMPLE CHANNEL 4
0227 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
022A CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE

022D DB0C IN 0CH ;SAMPLE CHANNEL 5
022F 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0232 CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE

0235 DB0D IN 0DH ;SAMPLE CHANNEL 6
0237 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
023A CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE

023D DB0E IN 0EH ;SAMPLE CHANNEL 7
AVOCET SYSTEMS 8085/Z80 ASSEMBLER - VERSION 1.03 SERIAL #00355

SOURCE FILE NAME: DACON.ASM

023F 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0242 CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE
0245 DB0F IN OFH ;SAMPLE CHANNEL 8
0247 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
024A CDE103 CALL HEXRO ;GOTO HEX SUBROUTINE
024D C38000 JMP POLL ;CONTINUE POLLING

0250 DB0B ALLDC IN 08H ;SAMPLE CHANNEL 1
0252 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0255 CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0258 DB0B IN 09H ;SAMPLE CHANNEL 2
025A 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
025D CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0260 DB0A IN 0AH ;SAMPLE CHANNEL 3
0262 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0265 CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0268 DB0B IN 0BH ;SAMPLE CHANNEL 4
026A 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
026D CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0270 DB0C IN 0CH ;SAMPLE CHANNEL 5
0272 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0275 CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0278 DB0D IN 0DH ;SAMPLE CHANNEL 6
027A 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
027D CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0280 DB0E IN 0EH ;SAMPLE CHANNEL 7
0282 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
0285 CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0288 DB0F IN OFH ;SAMPLE CHANNEL 8
028A 320620 STA ADSTR ;SAVE VALUE TEMPORARILY
028D CD0204 CALL DECRO ;GOTO DECIMAL SUBROUTINE
0290 C38000 JMP POLL ;CONTINUE POLLING

******************************************************************************
CONTROL SWITCHES ROUTINE (ON)
;THIS PART Responds TO THE USER'S COMMAND OF SWITCHING
;THE REQUIRED CONTROL UNIT SWITCH ON.
;******************************************************************************
0293 F1 ONSW POP PSM ;RESTORE COMMAND
0294 FE41 CPI 41H ;SWITCH A?
0296 C2A502 JNZ SWB ;CONTINUE CHECKING
SOURCE FILE NAME: DACON.ASM

LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 01H ;ADD SWITCH A
STA CONSW ;SAVE NEW SWITCHES STATUS
JMP POLL ;CONTINUE POLLING

CPI 42H ;SWITCH B?
JNZ SWC ;CONTINUE CHECKING
LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 02H ;ADD SWITCH B
STA CONSW ;SAVE NEW SWITCHES STATUS
JMP POLL ;CONTINUE POLLING

CPI 43H ;SWITCH C?
JNZ SWD ;CONTINUE CHECKING
LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 04H ;ADD SWITCH C
OUT PORTA ;TURN SWITCH C ON
STA CONSW ;SAVE NEW SWITCHES STATUS
JMP POLL ;CONTINUE POLLING

CPI 44H ;SWITCH D?
JNZ SWE ;CONTINUE CHECKING
LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 08H ;ADD SWITCH D
OUT PORTA ;TURN SWITCH D ON
STA CONSW ;SAVE NEW SWITCHES STATUS
JMP POLL ;CONTINUE POLLING

CPI 45H ;SWITCH E?
JNZ SWF ;CONTINUE CHECKING
LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 10H ;ADD SWITCH E
OUT PORTA ;TURN SWITCH E ON
STA CONSW ;SAVE NEW SWITCHES STATUS
JMP POLL ;CONTINUE POLLING

CPI 46H ;SWITCH F?
JNZ SWG ;CONTINUE CHECKING
LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 20H ;ADD SWITCH F
OUT PORTA ;TURN SWITCH F ON
STA CONSW ;SAVE NEW SWITCHES STATUS
JMP POLL ;CONTINUE POLLING

CPI 47H ;SWITCH G?
JNZ SWH ;CONTINUE CHECKING
LDA CONSW ;LOAD SWITCHES STATUS IN REG A
ORI 40H ;ADD SWITCH G
OUT PORTA ;TURN SWITCH G ON
SOURCE FILE NAME: DACON.ASM

;AVOCET SYSTEMS 8085/Z80 ASSEMBLER - VERSION 1.05M SERIAL #00355

030C 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
030F C38000 JMP POLL ;CONTINUE POLLING

0312 FE48 5WH CPI 48H ;SWITCH H?
0314 C2403 JNZ SWIH ;CONTINUE
0317 3A0220 LDA CONSW ;LOAD SWITCHES STATUS IN REG A
031A F680 ORI 80H ;ADD SWITCH H
031C D321 OUT PORTA ;TURN SWITCH H ON
031E 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
0321 C38000 JMP POLL ;CONTINUE POLLING

0324 3E04 SWI MVI A,04H ;SEND AN ERROR SIGNAL
0326 D323 OUT PORTC ;IF NOT A VALID COMMAND
0328 C38000 JMP POLL ;CONTINUE POLLING

;**********************************************************************
; CONTROL SWITCHES ROUTINE (OFF)
; THIS SEGMENT SWITCH THE REQUIRED CONTROL UNIT OFF
;**********************************************************************

032B F1 OFFSW POP PSW ;RESTORE THE COMMAND
032C FE61 CPI 61H ;SWITCH A?
032E C23E03 JNZ SWFA ;CONTINUE CHECKING
0331 3A0220 LDA CONSW ;LOAD SWITCHES STATUS IN REG A
0334 E6FE ANI 0FH ;SUBTRACT SWITCH A
0336 D321 OUT PORTA ;TURN SWITCH A OFF
0338 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
033B C38000 JMP POLL ;CONTINUE POLLING

033E FE62 SWBF CPI 62H ;SWITCH B?
0340 C25003 JNZ SWFB ;CONTINUE CHECKING
0343 3A0220 LDA CONSW ;LOAD SWITCHES STATUS IN REG A
0346 E6FD ANI 0FH ;SUBTRACT SWITCH B
0348 D321 OUT PORTA ;TURN SWITCH B OFF
034A 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
034D C38000 JMP POLL ;CONTINUE POLLING

0350 FE63 SWCF CPI 63H ;SWITCH C?
0352 C26203 JNZ SWFC ;CONTINUE CHECKING
0355 3A0220 LDA CONSW ;LOAD OLD SWITCHES STATUS
0358 E6FB ANI 0FH ;SUBTRACT SWITCH C
035A D321 OUT PORTA ;TURN SWITCH C OFF
035C 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
035F C38000 JMP POLL ;CONTINUE POLLING

0362 FE64 SWDF CPI 64H ;SWITCH D?
0364 C27403 JNZ SWFD ;CONTINUE CHECKING
0367 3A0220 LDA CONSW ;LOAD OLD SWITCHES STATUS
036A E67F ANI 0FH ;SUBTRACT SWITCH D
036C D321 OUT PORTA ;TURN SWITCH D OFF
036E 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
0371 C38000 JMP POLL ;CONTINUE POLLING
SOURCE FILE NAME: DACON.ASM

0374 FE65 SWF CPI 65H ;SWITCH E?
0376 C280D1 JNZ SWFF ;CONTINUE CHECKING
0379 3A0220 LDA CONSW ;LOAD OLD SWITCHES STATUS
037C E6EF ANI 0EFH ;SUBTRACT SWITCH E
037E 0321 OUT PORTA ;TURN SWITCH E OFF
0380 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
0383 C38000 JMP POLL ;CONTINUE POLLING

0386 FE66 SWFF CPI 66H ;SWITCH F?
0388 C280D1 JNZ SWGF ;CONTINUE CHECKING
038B 3A0220 LDA CONSW ;LOAD OLD SWITCHES STATUS
038E E5DF ANI 0DFH ;SUBTRACT SWITCH F
0390 0321 OUT PORTA ;TURN SWITCH F OFF
0392 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
0395 C38000 JMP POLL ;CONTINUE POLLING

0398 FE67 SWGF CPI 67H ;SWITCH G?
039A C280D1 JNZ SWFH ;CONTINUE CHECKING
039D 3A0220 LDA CONSW ;LOAD OLD SWITCHES STATUS
03A0 E6BF ANI 0BFH ;SUBTRACT SWITCH G
03A2 0321 OUT PORTA ;TURN SWITCH G OFF
03A4 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
03A7 C38000 JMP POLL ;CONTINUE POLLING

03AA FE68 SWFH CPI 68H ;SWITCH H?
03AC C280D3 JNZ SWFI ;CONTINUE
03AF 3A0220 LDA CONSW ;LOAD OLD SWITCHES STATUS
03B2 E7F ANI 0FH ;SUBTRACT SWITCH H
03B4 0321 OUT PORTA ;TURN SWITCH H OFF
03B6 320220 STA CONSW ;SAVE NEW SWITCHES STATUS
03B9 C38000 JMP POLL ;CONTINUE POLLING

03BC 3ED4 SWFI MVI A,04H ;SEND AN ERROR SIGNAL
03BE 0323 OUT PORTC ;IF NOT A VALID CHAR
03C0 C38000 JMP POLL ;CONTINUE POLLING

;*****************************************************************
; DELAY SUBROUTINE
; THIS IS A 1 SEC DELAY USED TO MAKE ERRORS LED FLASHS
;*****************************************************************

03C3 1FFFFFF DELAY LXI D,OFFFH ;SET UP DELAY VALUE
03C5 1D LOOP DCX D ;DECREMENT COUNTER
03C7 7A MOV A,D ;LOAD D REG INTO REG A
03CB 83 ORA E ;OR REG A WITH REG E
03CD C2C603 JNZ LOOP ;IF COUNTER NOT 0, CONTINUE
03CC C9 RET ;RETURN

;*****************************************************************
; OUTPUT TEST ROUTINE
;*****************************************************************
;This routine determine whether the output should be in decimal or hexadecimal numbers

03CD 3A0420 CRTST LDA DEFOUT ;Load hex or dec definition
03D0 FE23 CPI 23H ;Decimal?
03D2 C2DB03 JNZ HX
03D5 CD0204 CALL DECRO ;Goto decimal output subroutine
03D8 C38000 JMP POLL ;Continue polling
03DE C38000 HX CALL HEXRO ;Goto hex output subroutine
03D7 1F JMP POLL ;Continue polling

;*****************************
;Hexadecimal subroutine (hex to ASCII)
;This subroutine converts data to its equivalent hexadecimal ASCII code

03E1 3A0620 HXRO LDA ADSTR ;Restore the saved data
03E4 1F RAR ;Rotate right 4 times
03E5 1F RAR ;In order to move the upper nibble instead of the lower nibble
03E6 1F RAR ;Mask the upper 4 bits
03E7 1F RAR ;Decimal add 90 BCD
03E8 E60F ANI 0FH ;Decimal add 40 BCD+CARRY
03EA C690 ADI 90H ;Decimal add 90 BCD
03EC 27 DAA ;Form two 4-bit BCD digits
03ED CE40 ACI 40H ;Decimal add 40 BCD+CARRY
03EE 27 DAA
03F0 CD3204 CALL OUT1 ;Send the upper nibble out
03F3 3A0620 LDA ADSTR ;Restore the saved data
03F6 E60F ANI 0FH ;Mask the upper 4 bits
03F8 C690 ADI 90H ;Decimal add 90 BCD
03FA 27 DAA
03FB CE40 ACI 40H ;Decimal add 40 BCD+CARRY
03FD 27 DAA
03FE CD3204 CALL OUT1 ;Send the lower nibble out
0401 C9 RET ;Return

;*****************************
;Decimal subroutine
;This subroutine converts the binary data to their equivalent decimal ASCII code

0402 3A0620 DECRO LDA ADSTR ;Restore the saved data
0405 6F MOV L,A ;Load reg pair H&L with
0406 260D MVI H,00H ;Data to make 16 bit word
0408 1664 MVI 0,54H ;Decimal 100 (divisor)
040A CD1904 CALL DIVID ;Goto divide subroutine
040D 160A MVI 0,0AH ;Decimal 10 (divisor)
040F CD1904 CALL DIVID ;Goto divide subroutine
0412 7D MOV A,L ;Place remainder in reg A
0413 C630 ADI 30H ;Convert to ASCII
0415 CD3204 CALL OUT1 ;Goto output subroutine
Source File Name: DACON.ASM

0418 C9 RET ;RETURN

0419 4A DIVD MOV C,D
041A 0608 MVI B,08H ;COUNT=8
041C 29 DIV DAD H ;SHIFT DIVIDEND, QUOTIENT
041D 7C MOV A,H ;IS MOST SIG PART OF DIVIDEND
041E 91 SUB C ;GREATER OR = DIVISOR
041F DA2404 JC CNT ;NO, GOTO NEXT STEP
0422 67 MOV H,A ;YES, SUBTRACT DIVISOR
0423 2C INR L ;AND ADD 1 TO QUOTIENT
0424 05 CNT DCR B ;DECREMENT COUNT
0425 C21C04 JNZ DIV ;CONTINUE IF NOT 0
0428 7D MOV A,L
0429 C630 ADI 30H ;CONVERT TO ASCII
042B 6C MOV L,H
042C 283D MVI H,00H
042E CD2204 CALL OUT1 ;GOTO OUTPUT SUBROUTINE
0431 C9 RET ;RETURN

;******************************************************************************************
; OUT1 SUBROUTINE
; THIS SUBROUTINE SENDS THE DECIMAL OR HEXADECIMAL DATA OUT TO THE HOST COMPUTER THROUGH THE USART
;******************************************************************************************

0432 D310 OUT1 OUT UAROUT ;SEND DATA TO USART DATA REG.
0434 DB11 TRY IN UARCOM ;TEST USART STATUS REG FOR TxRDY
0436 E501 AND 01H ;READY YET?
0438 CA3404 JZ TRY ;TRY AGAIN IF NOT
0439 C9 RET ;RETURN IF READY

;******************************************************************************************

0000 END
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>0190</td>
<td>CHAN6</td>
<td>01D2</td>
<td>HEXRO</td>
<td>03E1</td>
<td>OUTPU2</td>
<td>0127</td>
<td>SWF</td>
<td>02EE</td>
</tr>
<tr>
<td>ADSTR</td>
<td>2006</td>
<td>CHAN7</td>
<td>01DF</td>
<td>HX</td>
<td>0308</td>
<td>OUTPU3</td>
<td>0163</td>
<td>SWFF</td>
<td>0386</td>
</tr>
<tr>
<td>ALLCH</td>
<td>0205</td>
<td>CHAN8</td>
<td>01EC</td>
<td>INPUT1</td>
<td>00FE</td>
<td>OUTPU4</td>
<td>0174</td>
<td>SWG</td>
<td>0300</td>
</tr>
<tr>
<td>ALLDC</td>
<td>0250</td>
<td>CHAN9</td>
<td>01F9</td>
<td>INPUT2</td>
<td>0147</td>
<td>POLL</td>
<td>0080</td>
<td>SWGF</td>
<td>0398</td>
</tr>
<tr>
<td>ALLHX</td>
<td>020D</td>
<td>CNT</td>
<td>0424</td>
<td>ICOM</td>
<td>0020</td>
<td>PORTA</td>
<td>0021</td>
<td>SWH</td>
<td>0312</td>
</tr>
<tr>
<td>BIT2</td>
<td>001F</td>
<td>CONSW</td>
<td>2002</td>
<td>LOOP</td>
<td>03C5</td>
<td>PORTB</td>
<td>0022</td>
<td>SWHF</td>
<td>03AA</td>
</tr>
<tr>
<td>BIT3</td>
<td>0025</td>
<td>CRST</td>
<td>03CD</td>
<td>MON1</td>
<td>008F</td>
<td>PORTC</td>
<td>0023</td>
<td>SWI</td>
<td>0324</td>
</tr>
<tr>
<td>BIT4</td>
<td>0033</td>
<td>DAUSP</td>
<td>20FF</td>
<td>MON2</td>
<td>00CE</td>
<td>SCOP2</td>
<td>013C</td>
<td>SWIF</td>
<td>03BC</td>
</tr>
<tr>
<td>BIT5</td>
<td>003D</td>
<td>DECRD</td>
<td>0402</td>
<td>MON3</td>
<td>00DD</td>
<td>SWB</td>
<td>02A6</td>
<td>TRY</td>
<td>0434</td>
</tr>
<tr>
<td>BIT6</td>
<td>0047</td>
<td>DEFOUT</td>
<td>2004</td>
<td>MON4</td>
<td>00E8</td>
<td>SWBF</td>
<td>033E</td>
<td>UARCOM</td>
<td>0011</td>
</tr>
<tr>
<td>BIT7</td>
<td>0051</td>
<td>DELAY</td>
<td>03C3</td>
<td>MON5</td>
<td>00F3</td>
<td>SWC</td>
<td>02BB</td>
<td>UAREG</td>
<td>0062</td>
</tr>
<tr>
<td>BIT8</td>
<td>0058</td>
<td>DIV</td>
<td>041C</td>
<td>MONTOR</td>
<td>0088</td>
<td>SWCF</td>
<td>0350</td>
<td>UARIN</td>
<td>0010</td>
</tr>
<tr>
<td>CHAN2</td>
<td>019E</td>
<td>DIVID</td>
<td>0419</td>
<td>OFFSW</td>
<td>032B</td>
<td>SWD</td>
<td>02CA</td>
<td>UART</td>
<td>0010</td>
</tr>
<tr>
<td>CHAN3</td>
<td>01A8</td>
<td>ERR1</td>
<td>0189</td>
<td>ONSW</td>
<td>0293</td>
<td>SWDF</td>
<td>0362</td>
<td>UART</td>
<td>0013</td>
</tr>
<tr>
<td>CHAN4</td>
<td>0188</td>
<td>ERR2</td>
<td>006B</td>
<td>OUT1</td>
<td>0432</td>
<td>SME</td>
<td>02DC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAN5</td>
<td>01C5</td>
<td>ERR3</td>
<td>007C</td>
<td>OUTPU1</td>
<td>0116</td>
<td>SWEF</td>
<td>0374</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

***** NO ERRORS DETECTED *****
Appendix B

TEMPERATURE MONITOR PROGRAM LISTING

This program written in BASIC on a TANDY 200 computer, reads data from channel 1 which is connected to a temperature sensor, and plot the results on a temperature-time graph on the screen.

10 CLEAR
20 MAXFILES=2
30 CLS
**** Establish communication by opening the COM: port for both
**** input and output.
40 OPEN "COM:9711DNN" FOR INPUT AS 1
50 OPEN "COM:9711DNN" FOR OUTPUT AS 2
**** Draw X and Y axes and label them.
60 LINE (33,3)-(33,108)
65 LINE -(235,108)
70 FOR J=0 TO 10
74 X=108-(J*10)
78 PSET (31,X):PSET (32,X)
80 NEXT J
90 FOR K=0 TO 20
94 M=33+(K*10)
98 PSET (M,109):PSET (M,100)
100 NEXT K
110 PRINT@520,"0"
115 PRINT@280,"50"
120 PRINT@40,"100"
130 FOR K=0 TO 9
134 T=565+(3*K)
138 PRINT@T,K
140 NEXT K
**** Send command to request data in decimal ASCII
150 PRINT#2,"#";
**** Print the current time at the beginning of the conversion
160 PRINT@31,TIME$
170 FOR I=1 TO 200
**** Send command for voltage measurement from channel 1
180 PRINT #2,"1";
**** Get three character string, P$
190 P$=INPUT$(3,1)
**** Convert to temperature in °C
200 Y=INT(50.19-0.4565*(VAL(P$)-76))*2
**** Turn control unit 1 on if Temp> 100 °C
210 IF Y>120 THEN PRINT#2,"A"; ELSE PRINT#2,"a";
220 IF Y>133 OR Y<0 THEN 240
**** Plot the Temp data on the screen
230 PSET (I+33,132-Y)
240 NEXT I
**** Print current time at end of readings
250 PRINT@7,1,TIME$
260 END
Appendix D

DATA SHEETS
ADC0820 8-Bit High Speed μP Compatible A/D Converter with Track/Hold Function

General Description
By using a half-flash conversion technique, the 8-bit ADC0820 CMOS A/D offers a 1.5 μs conversion time and dissipates only 75 mW of power. The half-flash technique consists of 32 comparators, a most significant 4-bit ADC and a least significant 4-bit ADC. The input to the ADC0820 is tracked and held by the input sampling circuitry eliminating the need for an external sample-and-hold for signals moving at less than 100 mV/μs.

For ease of interface to microprocessors, the ADC0820 has been designed to appear as a memory location or I/O port without the need for external interfacing logic.

Features
- Built-in track-and-hold function
- No missing codes
- No external clocking
- Single supply—5 VCC
- Easy interface to all microprocessors, or operates stand-alone
- Latched TRI-STATE® output
- Logic inputs and outputs meet both MOS and TTL voltage level specifications
- Operates radionically or with any reference value equal to or less than VCC
- 0V to 5V analog input voltage range with single 5V supply
- No zero or full-scale adjust required
- Overflow output available for cascading
- 0.3° standard width 20-pin DIP

Key Specifications
- Resolution 8 Bits
- Conversion Time 2.5 μs Max (RD Mode)
- Input signals with slew rate of 100 mV/μs converted without external sample-and-hold to 8 bits
- Low Power 75 mW Max
- Total Unadjusted Error ± ½ LSB and ± 1 LSB

Connection Diagram

Functional Diagram

![Connection Diagram](image1)

![Functional Diagram](image2)
### Description of Pin Functions

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( V_{\text{IN}} )</td>
<td>Analog input range: ( \text{GND} \leq V_{\text{IN}} \leq V_{\text{CC}} )</td>
</tr>
<tr>
<td>2 DB0</td>
<td>TRI-STATE data output—bit 0 (LSB)</td>
</tr>
<tr>
<td>3 DB1</td>
<td>TRI-STATE data output—bit 1</td>
</tr>
<tr>
<td>4 DB2</td>
<td>TRI-STATE data output—bit 2</td>
</tr>
<tr>
<td>5 DB3</td>
<td>TRI-STATE data output—bit 3</td>
</tr>
<tr>
<td>6 WR/RDY</td>
<td>WR-RD Mode</td>
</tr>
<tr>
<td>7 Mode</td>
<td>Mode selection input—it is internally tied to GND through a 50 ( \mu )A current source.</td>
</tr>
<tr>
<td>8 RD</td>
<td>RD Mode: When mode is low. WR-RD Mode: When mode is high</td>
</tr>
<tr>
<td>9 INT</td>
<td>WR-RD Mode</td>
</tr>
</tbody>
</table>

**Function**

- **INT** going low indicates that the conversion is completed and the data result is in the output latch. INT will go low, \(-800\) ns after the rising edge of WR, or INT will go low after the falling edge of RD, if RD goes low prior to the 800 ns time out. INT is reset by the rising edge of RD or CS.

- **RD Mode**
  - **INT** going low indicates that the conversion is completed and the data result is in the output latch. INT is reset by the rising edge of RD or CS.

- **WR Mode**
  - **INT** going low indicates that the conversion is completed and the data result is in the output latch. INT is reset by the rising edge of RD or CS.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 INT</td>
<td>WR-RD Mode</td>
</tr>
<tr>
<td>10 GND</td>
<td>Ground</td>
</tr>
<tr>
<td>11 V_\text{REF}(-)</td>
<td>The bottom of resistor ladder, voltage range: ( \text{GND} \leq V_{\text{REF}(\neg)} \leq V_{\text{REF}(\neg)} ) (Note 5)</td>
</tr>
<tr>
<td>12 V_\text{REF}(+)</td>
<td>The top of resistor ladder, voltage range: ( V_{\text{REF}(\neg)} \leq V_{\text{REF}(\neg)} \leq V_{\text{CC}} ) (Note 5)</td>
</tr>
<tr>
<td>13 CS</td>
<td>CS must be low in order for the RD or WR to be recognized by the converter.</td>
</tr>
<tr>
<td>14 DB4</td>
<td>TRI-STATE data output—bit 4</td>
</tr>
<tr>
<td>15 DB5</td>
<td>TRI-STATE data output—bit 5</td>
</tr>
<tr>
<td>16 DB6</td>
<td>TRI-STATE data output—bit 6</td>
</tr>
<tr>
<td>17 DB7</td>
<td>TRI-STATE data output—bit 7 (MSB)</td>
</tr>
<tr>
<td>18 OFL</td>
<td>Overflow output—if the analog input is higher than the ( V_\text{REF}(+) ), OFL will be low at the end of conversion. It can be used to cascade 2 or more devices to have more resolution (9, 10-bit).</td>
</tr>
<tr>
<td>19 NC</td>
<td>No connection</td>
</tr>
<tr>
<td>20 V_{\text{CC}}</td>
<td>Power supply voltage</td>
</tr>
</tbody>
</table>

### 1.0 Functional Description

#### 1.1 GENERAL OPERATION

The ADC0820 uses two 4-bit flash A/D converters to make an 8-bit measurement. Each flash ADC is made up of 15 comparators which compare the unknown input to a reference ladder to get a 4-bit result. To take a full 8-bit reading, one flash conversion is done to provide the most significant data bits (via the MS flash ADC). Driven by the 4 MSBs, an internal DAC recreates an analog approximation of the input voltage. This analog signal is then subtracted from the input, and the difference voltage is converted by a second 4-bit flash ADC (the LS ADC), providing the 4 least significant bits of the output data word.

The internal DAC is actually a subsection of the MS flash converter. This is accomplished by using the same resistor ladder for the A/D as well as for generating the DAC signal. The DAC output is actually the tap on the resistor ladder which most closely approximates the analog input. In addition, the "sampled-data" comparators used in the ADC0820 provide the ability to compare the magnitudes of several analog signals simultaneously, without using input summing amplifiers. This is especially useful in the LS flash ADC, where the signal to be converted is an analog difference.
The MAX230 family of line drivers/receivers is intended for all RS-232 and V.28/V.24 communications interfaces, and in particular for those applications where ±12V is not available. The MAX230 and MAX236 are particularly useful in battery powered systems since their low power shutdown mode reduces power dissipation to less than 5μW. The MAX233 and MAX235 use no external components and are recommended for applications where printed circuit board space is critical.

All of the MAX230 family except the MAX231 and MAX239 need only a single +5V supply for operation. The MAX230 family RS-232 driver/receivers have on-board charge pump voltage converters which convert the +5V input power to the ±10V needed to generate the RS-232 output levels. The MAX231 and MAX239, designed to operate from +5V and +12V, contain a +12V to -12V charge pump voltage converter.

Since nearly all RS-232 applications need both line drivers and receivers, the MAX230 family includes both receivers and drivers in one package. Since the wide variety of RS-232 applications require differing numbers of transmitters and receivers, Maxim offers a wide selection of RS-232 driver/receiver combinations in order to minimize the package count (see table below).

Both the receivers and the line drivers (transmitters) meet all EIA RS-232C and CCITT V.28 specifications.

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Power Supply Voltage</th>
<th>No. of RS-232 Drivers</th>
<th>No. of RS-232 Receivers</th>
<th>External Components</th>
<th>Low Power Shutdown and TTL 3-State</th>
<th>No. of Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX230</td>
<td>+5V</td>
<td>5</td>
<td>0</td>
<td>4 capacitors</td>
<td>Yes</td>
<td>20</td>
</tr>
<tr>
<td>MAX231</td>
<td>+5V and +7.5V to 13.2V</td>
<td>2</td>
<td>2</td>
<td>2 capacitors</td>
<td>No</td>
<td>14</td>
</tr>
<tr>
<td>MAX232</td>
<td>+5V</td>
<td>2</td>
<td>2</td>
<td>4 capacitors</td>
<td>No</td>
<td>16</td>
</tr>
<tr>
<td>MAX233</td>
<td>+5V</td>
<td>2</td>
<td>2</td>
<td>None</td>
<td>No</td>
<td>20</td>
</tr>
<tr>
<td>MAX234</td>
<td>+5V</td>
<td>4</td>
<td>0</td>
<td>4 capacitors</td>
<td>No</td>
<td>16</td>
</tr>
<tr>
<td>MAX235</td>
<td>±5V</td>
<td>5</td>
<td>5</td>
<td>None</td>
<td>Yes</td>
<td>24</td>
</tr>
<tr>
<td>MAX236</td>
<td>+5V</td>
<td>4</td>
<td>3</td>
<td>4 capacitors</td>
<td>Yes</td>
<td>24</td>
</tr>
<tr>
<td>MAX237</td>
<td>+5V</td>
<td>5</td>
<td>3</td>
<td>4 capacitors</td>
<td>No</td>
<td>24</td>
</tr>
<tr>
<td>MAX238</td>
<td>+5V</td>
<td>4</td>
<td>4</td>
<td>4 capacitors</td>
<td>No</td>
<td>24</td>
</tr>
<tr>
<td>MAX239</td>
<td>+5V and +7.5V to 13.2V</td>
<td>3</td>
<td>5</td>
<td>2 capacitors</td>
<td>Yes</td>
<td>24</td>
</tr>
</tbody>
</table>

* Patent Pending

---

**General Description**

**Features**
- Operates from Single 5V Power Supply (+5V and +12V — MAX231 and MAX239)
- Meets All RS-232C and V.28 Specifications
- Multiple Drivers and Receivers
- Onboard DC-DC Converters
- ±5V Output Swing with ±5V Supply
- Low Power Shutdown — <1μA
- 3-State TTL/CMOS Receiver Outputs
- ±10V Receiver Input Levels

**Applications**
- Computers
- Peripherals
- Modems
- Printers
- Instruments

---

**Selection Table**

---

Maxim Integrated Products
# +5V Powered RS-232 Drivers/Receivers

## Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Conditions</th>
<th>Min</th>
<th>Ty</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC Power Supply Current</td>
<td>No load, ( T_a = +25^\circ C )</td>
<td>5</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>V( ^+ ) Power Supply Current</td>
<td>No load, ( V^+ = 12V ), MAX231 and MAX233 only</td>
<td>5</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Shutdown Supply Current</td>
<td>Figure 1, ( T_a = +25^\circ C )</td>
<td>1</td>
<td>10</td>
<td>( \mu A )</td>
<td></td>
</tr>
<tr>
<td>Input Logic Threshold Low</td>
<td>( T_{IN} )</td>
<td>0.8</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input Logic Threshold High</td>
<td>( T_{OH} )</td>
<td>2.0</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic Pullup Current</td>
<td>( T_{PS} = 0V )</td>
<td>15</td>
<td>200</td>
<td>( \mu A )</td>
<td></td>
</tr>
<tr>
<td>RS-232 Input Voltage Operating Range</td>
<td>( V_{CC} = 5V ), ( T_a = +25^\circ C )</td>
<td>-30</td>
<td>( -30 )</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>RS-232 Input Threshold Low</td>
<td>( V_{CC} = 5V ), ( T_a = +25^\circ C )</td>
<td>0.8</td>
<td>1.2</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>RS-232 Input Threshold High</td>
<td>( V_{CC} = 5V ), ( T_a = +25^\circ C )</td>
<td>1.7</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>RS-232 Input Hysteresis</td>
<td>( V_{CC} = 5V )</td>
<td>0.2</td>
<td>0.5</td>
<td>1.0</td>
<td>V</td>
</tr>
<tr>
<td>RS-232 Input Resistance</td>
<td>( T_a = +25^\circ C )</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>k( \Omega )</td>
</tr>
<tr>
<td>TTC/CMOS Output Voltage Low</td>
<td>( I_{OUT} = 3.2mA )</td>
<td>0.4</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTC/CMOS Output Voltage High</td>
<td>( I_{OUT} = -1.0mA )</td>
<td>3.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTC/CMOS Output Leakage Current</td>
<td>( E_N = V_{CC}, G = 0V )</td>
<td>0.01</td>
<td>( \mu A )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Enable Time</td>
<td>MAX235, MAX236, MAX238, Figure 2</td>
<td>400</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Disable Time</td>
<td>MAX235, MAX236, MAX238, Figure 2</td>
<td>250</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Propagation Delay</td>
<td>RS-232 to TTL</td>
<td>0.5</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instantaneous Slew Rate</td>
<td>( C_L = 10pF ), ( R_L = 3-7k\Omega ), ( T_a = +25^\circ C ) (Note 1)</td>
<td>30</td>
<td>( V/\mu s )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transition Region Slew Rate</td>
<td>( R_L = 3k\Omega ), ( C_L = 10\mu F ), Measured from +3V to -3V or -3V to +3V</td>
<td>3</td>
<td>( V/\mu s )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Resistance</td>
<td>( V_{CC} = V^+, V^- = 0V ), ( V_{OUT} = \pm 2V )</td>
<td>300</td>
<td>( \Omega )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS-232 Output Short Circuit Current</td>
<td>( V_{CC} = V^+, V^- = 0V ), ( V_{OUT} = \pm 2V )</td>
<td>( \leq 10 )</td>
<td>mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Sample tested.

---

**ELECTRICAL CHARACTERISTICS**

\( V_{CC} = 5V \pm 10\% \), \( V^+ = 7.5V \) to 13.2V (MAX231 and MAX233 Only), \( T_a \) * Operating Temperature Range, Figures 2-3-12, unless otherwise noted.* **MAXIM**
+5V Powered
RS-232 Drivers/Receivers

Figure 5: MAX232 Typical Operating Circuit

Figure 6: MAX233 Typical Operating Circuit
Review of EIA Standard
RS-232-C and CCITT

--- Recommendations V.28 and V.24

The most common serial interface between electronic equipment is the "RS232" interface. This serial interface has been found to be particularly useful for the interface between units made by different manufacturers since the voltage levels are defined by the EIA Standard RS-232-C and CCITT Recommendation V.28. The RS-232 specification also contains signal circuit definitions and connector pin assignments, while CCITT circuit definitions are contained in a separate document, Recommendation V.24. Originally intended to interface modems to computers and terminals, these standards have many signals which are not used for computer-to-computer or computer-to-peripheral communication. V.24 contains 43 circuit definitions, while RS-232 contains 22 circuit definitions. These circuit names and pin assignments are shown in Table 1, which is an extract of Figures 3.1 and 4.1 of the RS-232C specification, and Table 1 of CCITT recommendation V.24. Most computer-to-computer or computer-to-peripheral communication circuits only use the subset shown in Table 2 of this data sheet.

Serial interfaces can be used with a variety of transmission formats. The most popular by far is the asynchronous format, generally at one of the standard baud rates of 300, 600, 1200, etc. The maximum recommended baud rate for RS-232 and V.28 is 20,000 baud, and the fastest commonly used baud rate is 19,200 baud. Asynchronous serial links use a variety of combinations of the number of data bits, what type (if any) of parity bit, and the number of stop bits. A typical combination is 7 data bits, even parity, and 1 stop bit.

RS232/V28 physical links are also suitable for synchronous transmission protocols. These higher level protocols often use the standard RS-232C/V28 voltage levels. Note that one type of physical link (such as RS-232/V28 voltage levels) can be used for a variety of higher level protocols. Table 3 summarizes the voltage levels and other requirements of V.28 and RS-232.

--- Comparison of RS-232C/V28 with other Standards

The other two most common serial interface specifications are the EIA RS423 and RS422/RS485 (CCITT recommendations V10 and V11). While the RS-232 or V28/V24 interface is the most common interface for communication between equipment made by different manufacturers, the RS423/V10 interface and RS422/V11 interfaces can operate at higher baud rates. In addition, the RS485 interface can be used for low cost local area networks.

The RS423 and V10 interfaces are unbalanced or dual-voltage, and use the single-ended signaling format. RS485 uses a partially balanced interface, and requires the use of a differential receiver. Single-ended links are limited in length because it is the summation of the noise present on both sides of the transmission line. One solution is to use a very fast clock rate relative to the transmitted signal. Dual-ended links are limited in length due to the effect of noise on one side of the transmission line.

--- Application Hints

Operation at High Baud Rates

V.28 states that "the time required for the signal to pass through the transition region during a change in state shall not exceed 1 millisecond or 3 per cent of the nominal element period on the interchange circuit, whichever is less." RS-232C allows the transition time to be 4 per cent of the duration of a signal element. At 19,200 baud, the "nominal element period" is approximately 50μs, of which 3 per cent is 1.5μs. Since the transition region is from -3V to +3V, this means the V28 slew rate would ideally be faster than 6V/1.5μs = 4V/μs at 19.2 kbaud and 2V/μs at 9600 baud. The RS-232 requirement is equivalent to 3V/μs at 19.2 kbaud and 1.5V/μs at 9600 baud. The slew rate of the MAX230 series devices is about 3V/μs with the maximum recommended load of 2500pF. In practice, the effect of less than optimum slew rate is a distortion of the recovered data, where the 1's and 0's no longer have equal width. This distortion generally has negligible effect and the MAX230 family can be reliably used for 19.2 kbaud serial links when the cable capacitance is kept below 2500pF. With very low capacitive loading, the MAX230 and MAX234-239 may even be used at 38.4 kbaud, since the typical slew rate is 5V/μs when loaded with 500pF in parallel.

--- +5V Powered
RS-232 Drivers/Receivers

"single-ended" interfaces which use a differential receiver. This standard is intended for data signaling rates up to 100 kbit/s (100 kilobaud). It achieves this higher baud rate through more precise requirements on the waveshape of the transmitters and through the use of differential receivers to compensate for ground potential variations between the transmitting and receiving equipment. With certain limitations, this interface is compatible with RS-232 and V.28. The limitations are:

1) less than 20,000 baud rate.
2) maximum cable lengths determined by RS-232 performance.
3) RS423/V10 DTE and DCE signal return paths must be connected to the the RS232/V28 signal ground.
4) the RS-232 transmitter output voltages must be limited to ±12V, or additional protection must be provided for the RS423/V10 receivers, and
5) not all RS232/V28 receivers will show proper power-off detection of V10 transmitter outputs.

Maxim's MAX230 and MAX232-MAX238 meet restrictions 4 and 5 over the entire range of recommended operating conditions. The MAX231 and MAX239 meet restrictions 4 and 5 provided that the V" voltage is 12.5V or less.

The RS422, RS485, and V10 interfaces are balanced double-current interchanges suitable for baud rates up to 10 Mbit/s. These interfaces are not compatible with RS-232 or V28 voltage levels.
### +5V Powered
#### RS-232 Drivers/ Receivers

Table 1. Circuit Definitions/ Pin Assignments

<table>
<thead>
<tr>
<th>PIN</th>
<th>EIA RS-232 CIRCUIT</th>
<th>CCITT V.24 CIRCUIT</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AA</td>
<td>101</td>
<td>Protective Ground</td>
<td>Ground</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>BA</td>
<td>103</td>
<td>Transmitted Data</td>
<td>Data</td>
<td>DTE</td>
</tr>
<tr>
<td>3</td>
<td>BB</td>
<td>104</td>
<td>Received Data</td>
<td>Data</td>
<td>DCE</td>
</tr>
<tr>
<td>4</td>
<td>CA</td>
<td>103</td>
<td>Request To Send</td>
<td>Control</td>
<td>DTE</td>
</tr>
<tr>
<td>5</td>
<td>CB</td>
<td>108</td>
<td>Clear To Send</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>6</td>
<td>CC</td>
<td>107</td>
<td>Data Set Ready</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>7</td>
<td>AB</td>
<td>102</td>
<td>Signal Ground/Return</td>
<td>Ground</td>
<td>Ground</td>
</tr>
<tr>
<td>8</td>
<td>CF</td>
<td>109</td>
<td>Received Line Signal Detector</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>Reserved for Data Set</td>
<td>Testing</td>
<td>Testing</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>Reserved for Data Set</td>
<td>Testing</td>
<td>Testing</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>Unassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>SCF</td>
<td>122</td>
<td>Sec. Received Line Signal Detector</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>13</td>
<td>SCB</td>
<td>121</td>
<td>Secondary CTS</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>14</td>
<td>SBA</td>
<td>118</td>
<td>Sec. Transmit Data</td>
<td>Data</td>
<td>DCE</td>
</tr>
<tr>
<td>15</td>
<td>DB</td>
<td>114</td>
<td>Transmission Signal Element Timing (DCE)</td>
<td>Timing</td>
<td>DCE</td>
</tr>
<tr>
<td>16</td>
<td>SBB</td>
<td>119</td>
<td>Sec. Received Data</td>
<td>Data</td>
<td>DCE</td>
</tr>
<tr>
<td>17</td>
<td>DD</td>
<td>115</td>
<td>Receiver Signal Element Timing (DCE)</td>
<td>Timing</td>
<td>DCE</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>Unassigned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>SCA</td>
<td>120</td>
<td>Secondary RTS</td>
<td>Control</td>
<td>DTE</td>
</tr>
<tr>
<td>20</td>
<td>CD</td>
<td>108.2</td>
<td>Data Terminal Ready</td>
<td>Control</td>
<td>DTE</td>
</tr>
<tr>
<td>21</td>
<td>CG</td>
<td>112</td>
<td>Signal Quality Det.</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>22</td>
<td>CE</td>
<td>125</td>
<td>Ring Indicator</td>
<td>Control</td>
<td>DCE</td>
</tr>
<tr>
<td>23</td>
<td>CH/CI</td>
<td>111/112</td>
<td>Data Signal Rate Selector (DCE/DCE)</td>
<td>Control</td>
<td>DTE/DCE</td>
</tr>
<tr>
<td>24</td>
<td>DA</td>
<td>113</td>
<td>Transmit Signal Element Timing (DCE)</td>
<td>Timing</td>
<td>DTE</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td>Unassigned</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Circuits Commonly Used for RS-232C and V.24 Asynchronous Interfaces

<table>
<thead>
<tr>
<th>PIN</th>
<th>CIRCUIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Protective Ground Connect to Earth Ground</td>
</tr>
<tr>
<td>2</td>
<td>Transmit Data (TD) Data from DTE</td>
</tr>
<tr>
<td>3</td>
<td>Receive Data (RO) Data from DCE</td>
</tr>
<tr>
<td>4</td>
<td>Request To Send (RTS) Handshake from DTE</td>
</tr>
<tr>
<td>5</td>
<td>Clear To Send (CTS) Handshake from DCE</td>
</tr>
<tr>
<td>6</td>
<td>Data Set ready (DSR) Handshake from DCE</td>
</tr>
<tr>
<td>7</td>
<td>Reference Point for Signals</td>
</tr>
<tr>
<td>8</td>
<td>Received Line Signal Detector Handshake from DCE</td>
</tr>
<tr>
<td>11</td>
<td>Printer Busy Signal Handshake from Printer</td>
</tr>
<tr>
<td>20</td>
<td>Data Terminal Ready Handshake from DTE</td>
</tr>
<tr>
<td>22</td>
<td>Ring Indicator Handshake from DCE</td>
</tr>
</tbody>
</table>

Table 3. Summary of RS-232C and V.28 Electrical Specifications

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SPECIFICATION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver Output Voltage</td>
<td>+5V to +15V With 3-7kΩ load</td>
<td></td>
</tr>
<tr>
<td>conditioner</td>
<td>-5V to -15V With 3-7kΩ load</td>
<td></td>
</tr>
<tr>
<td>Max. output</td>
<td>±25V Max. No Load</td>
<td></td>
</tr>
<tr>
<td>Receiver Input Thresholds</td>
<td>(data and clock signals)</td>
<td></td>
</tr>
<tr>
<td>On level</td>
<td>-3V to -25V</td>
<td></td>
</tr>
<tr>
<td>Off level</td>
<td>-3V to -25V</td>
<td></td>
</tr>
<tr>
<td>Receiver Thresholds</td>
<td>RTS, DSR, DTR</td>
<td></td>
</tr>
<tr>
<td>On level</td>
<td>+3V to +25V Detects Power</td>
<td></td>
</tr>
<tr>
<td>Off level</td>
<td>Open Circuit or -3V to -25V Off Condition at Driver</td>
<td></td>
</tr>
<tr>
<td>Receiver Input Resistance</td>
<td>3kΩ to 7kΩ</td>
<td></td>
</tr>
<tr>
<td>Driver Output Resistance</td>
<td>300Ω Min.</td>
<td></td>
</tr>
<tr>
<td>condition</td>
<td>Vout &lt; ±2V</td>
<td></td>
</tr>
<tr>
<td>Driver Slew Rate</td>
<td>30V/µs Max.</td>
<td></td>
</tr>
<tr>
<td>3kΩ &lt; Rl ≤ 7kΩ; 0pF &lt; Cll ≤ 2500pF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Signalling Rate</td>
<td>Up to 20kbps/sec.</td>
<td></td>
</tr>
<tr>
<td>Cable Length</td>
<td>50/16 m. Recommended</td>
<td></td>
</tr>
<tr>
<td>Maximum Length</td>
<td>Longer cables permissible, if Cll = 5 2500pF</td>
<td></td>
</tr>
</tbody>
</table>

MAXIM
with 5kΩ. Under no circumstance will the slew rate exceed the RS-232/V.28 maximum spec of 30V/μs and, unlike the MC1488 driver, no external compensation capacitors are needed under any load condition.

**Driving Long Cables**

The RS-232 standard states that “The use of short cables (each less than approximately 50 feet or 15 meters) is recommended; however, longer cables are permissible, provided that the load capacitance . . . does not exceed 2500pF.” Baud rate and cable length can be traded off: use lower baud rates for long cables, use short cables if high baud rates are desired. For both long cables and high baud rates, use RS422/V.11. The maximum cable length for a given baud rate is determined by several factors, including the capacitance per meter of cable, the slew rate of the driver under high capacitive loading, the receiver threshold and hysteresis, and the acceptable bit error rate. The MAX230 family receivers have 0.5V of hysteresis, and the transmitters are designed such that the slew rate reduction caused by capacitive loading is minimized (see Typical Characteristics).

**+5V Powered RS-232 Drivers/Receivers**

Occasionally a non-inverting transmitter or receiver is needed instead of the inverting transmitters and receivers of the MAX230 family. Simply use one of the receivers as a TTL/CMOS inverter to get the desired operation (Figure 15). If the logic output driving the receiver input has less than 1mA of output source capability, then add the 2.2kΩ pullup resistor.

The MAX230 series receiver TTL outputs can directly drive the input of another receiver to form a non-inverting RS-232 receiver.

**Protection for Shorts to ±15V Supplies**

All MAX230 family devices transmitter outputs except the MAX231, MAX232 and MAX233 are protected against short circuits to ±15V, which is the maximum allowable loaded output voltage of an RS-232/V.28 transmitter. The MAX231, MAX232, and MAX233 can be protected against short circuits to ±15V power supplies by the addition of a series 220Ω resistor in each output. This protection is not needed to protect against short circuits to most RS-232 transmitters such as the 1488, since they have an internal short circuit current limit of 12mA.

The power dissipation of the MAX230 and MAX234-MAX239 is about 200mW with all transmitters shorted to ±15V.

**Isolated RS-232 Interfaces**

RS-232 and V.28 specifications require a common ground connection between the two units communicating via the RS-232/V.28 interface. In some cases, there may be large differences in ground potential between the two units, and in other cases it may be desired to avoid ground loop currents by isolating the two grounds. In other cases, a computer or control system must be protected against accidental connection of the RS-232/V.28 signal lines to 110/220VAC power lines. Figure 16 shows a circuit with this isolation. The power for the MAX233 is generated by a MAX635 DC-DC converter. When the MAX635 regulates point “A” to -5V, the isolated output at point “B” will be semi-regulated to +5V. The two optocouplers maintain isolation between the system ground and the RS-232 ground while transferring the data across the isolation barrier. While this circuit will not withstand 110VAC between the RS-232 ground and either the receiver or transmitter lines, the voltage difference between the two grounds is only limited by the optocoupler and DC-DC converter transformer breakdown ratings.
Mode

Instruction Format, Asynchronous Mode

Command Instruction Format

APPLICATIONS OF THE 8251A

Asynchronous Serial Interface to CRT Terminal, DC—9600 Baud
REFERENCES

8. "Understanding Data Communications," Texas Instruments Information Publishing Center, 1984