21540

MICROCOMPUTER-ANALOG CONVERTER

SOFTWARE & HARDWARE INTERFACING

BLACKSBURG CONTINUING EDUCATION SERIES™
edited by Titus, Titus, Rony & Larsen

The Blacksburg Continuing Education Series™ of books provide a laboratory — or experiment-oriented approach to electronic topics. Present and forthcoming titles in this series include the following:

- The 8080A Bugbook® : Microcomputer Interfacing and Programming
- DBUG: An 8080 Interpretive Debugger
- Design of Op-Amp Circuits, With Experiments
- 555 Timer Applications Sourcebook, With Experiments
- Design of Active Filters, With Experiments
- 8080/8085 Software Design
- Logic & Memory Experiments Using TTL Integrated Circuits (2 Volumes)
- Design of Phase-Locked Loop Circuits, With Experiments
- Interfacing and Scientific Data Communication Experiments
- NCR Data Processing Concepts Course
- NCR Data Communications Concepts
- NCR Basic Electronics Course, With Experiments
- Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming Interfacing (2 Volumes)

In most cases, thses books provide both text material and experiments, which permit one to demonstrate and explore the concepts that are covered in the book. These books remain among the very few that provide step-by-step instructions concerning how to learn basic electronic concepts, wire actual circuits, test microcomputer interfaces, and program computers based upon the popular 8080-type microprocessor chip. We have found that the books are very useful to the electronic novice who desires to join the "electronics revolution," with minimum time and effort.

It is our pleasure to introduce this new book which covers interfacing A/D and D/A to microcomputers as well as the software that is required to control them. Complete hardware schematic diagrams and completely assembled program examples have been included. We have not tried to describe all of the analog/digital conversion techniques and how they operate, but rather we have concentrated upon the use of analog/digital converters, in small computer systems. We have chosen commercially available modules and integrated circuits for use in the examples and in the experiments. The topics of data acquisition systems, sample-and-hold devices and multiplexer circuits are also covered. The book is aimed at the microcomputer user who is interested in interfacing his computer to "real-world" analog signals for data acquisition, control, display, plotting, etc.

Our books have been well accepted in the United States and abroad. Selected books are now being translated into German, Italian, Chinese, and Japanese. If you are interested in further details concerning these translations, contact the series editors. Both domestic and foreign short courses are available in conjunction with the Extension Division at Virginia Polytechnic Institute and State University. Plese write or call Dr. Linda Leffel, Continuing Education Center, VPI & SU, Blacksburg, VA 24061. Phone (703) 951-5241. Short courses on microcomputer interfacing and microcomputer software development are given by Tychon, Inc., Blacksburg, VA 24060. For more information on these courses, write or call Dr. Christopher Titus, at Tychon, phone (703) 951-9030.

We continue to be interested in identifying other authors who could contribute books to this series. If you have an interest in writing or publishing such a text, please contact one of the series editors.

Jonathan A. Titus, Christopher A. Titus, David G. Larsen
and Peter R. Rony
"The Blacksburg Group"

Microcomputer—Analog Converter Software And Hardware Interfacing

by

Jonathan A. Titus
Peter R. Rony

Christopher A. Titus
David G. Larsen

Originally Published as
Microcomputer—Analog Converter
Software and Hardware Interfacing
With Experiments for 8080A/Z80/8085 Systems
by E&L Instruments, Inc.

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

FIRST EDITION
FIRST PRINTING—1978

International Standard Book Number: 0-672-21540-1
Library of Congress Catalog Card Number: 78-57201

Printed in the United States of America.

Preface

Our purpose in writing this book has been to introduce you to the concepts and techniques of interfacing digital computers to analog electronic devices. Actually, the ideas presented in this book can probably be extended to interfaces and programs for many types of computers. We have concentrated upon interfacing analog-to-digital and digital-to-analog converters to an 8080/8085-based microcomputer. The designs and programs are equally applicable to all of the 8080-type microcomputers, the 8080A, 8085, Z80, etc. It has not been our purpose to detail the inner workings of the analog and digital converter modules, but rather, to treat them the way we now treat the integrated-circuit digital electronic devices such as those in the SN7400 transistor-transistor logic (TTL) family or the CD4000 complementary metal-oxide semiconductor (CMOS) family.

You will not learn how to build analog-to-digital or digital-to-analog converters in this book. We have chosen, instead, to show you how to use the modules and devices that are already commercially available. We present many examples of converter interfacing with complete hardware schematic diagrams and program listings. Perhaps one of these designs will answer one of your needs with little additional work on your part. If you require something special, we hope that you will find that the concepts presented are applicable to your work. If so, we have done our job.

We have assumed that you will be able to condition (amplify, filter, etc.) your analog signals so that they are compatible with the A/D converters that we present in the examples, and that you will be able to use the voltages that are output by the D/A converters that we have used. The topics of *signal conditioning, filters, noise, ground* *loops,* and *amplifiers* are beyond the scope of this book. We are looking forward to presenting these topics in future books.

If you are interested in the internal operation of A/D and D/A converters, there are two good references available which will help you understand how they work. These are:

Analog-Digital Converter Notes, D. H. Sheingold, ed.,
Analog Devices, Inc., Norwood, MA 02062, 1977.
Data Conversion Handbook, Donald B. Bruck, Hybrid
Systems Corporation, Burlington, MA 01803, 1974.

In all of the examples, we have tried to use commercially available converter modules or integrated circuits (chips). There are dozens of analog/digital module and device manufacturers, some of whom are mentioned in the following units or in the appendices. Our choices of specific manufacturers, or specific modules, were made at random, but with an eye toward presenting those modules which are easy to use and which provide a number of interfacing schemes.

The choice of specific modules does not imply our endorsement and it should not be taken to mean that these modules may necessarily be the best ones for your specific applications.

We have assumed that you have had some experience programming microcomputers at the assembly language level, and that you are familiar with the 8080's internal registers and I/O operations. Both the memory-mapped and the accumulator I/O techniques will be discussed, although we tend to favor the accumulator I/O technique because of its simplicity. You should be familiar with the use of latches at output ports, the use of three-state buffers at input ports, and the use of various decoder circuits for device address decoding. All of the examples and experiments assume the use of an 8080-based microcomputer with an uninverted, bidirectional data bus.

The program examples in the text and in the experiments are shown in the byte-per-line output that is produced by the Tychon Editor/Assembler (TEA) that was written by Dr. Christopher A. Titus. We find that it is much easier to read and interpret than the instruction-per-line outputs of other assemblers. Although we favor the octal numbering system, along with the Heath Company and others, the TEA program can also provide a hexadecimal format. Write to Chris if you would like more information about TEA.

We have found wide acceptance of our books in formal classes as well as by individual users, worldwide. Selected books are being translated into German, Japanese, French, Italian, Chinese and Malaysian. If you are interested in further details concerning these translations, or in translating the books into other languages, please contact us.

The Blacksburg Continuing Education Series™ continues to expand with additional titles being added in the past few months. A list of the current series is given inside the front cover of this book. We continue to be interested in identifying and working with authors who think that their book ideas would fit into the Blacksburg Continuing Education Series. If you have an idea that you are interested in working on, please contact us here in Blacksburg.

Many of the concepts that are presented in this book have been incorporated into the material taught at seminars that are presented by Tychon here in Blacksburg. Three courses are currently being taught: *Microprocessor Interfacing* (*628*), *Introduction to Assembly Language Programming for 8080/8085 Processors* (*685*), and *Intermediate Assembly Language Programming for 8080/8085 Processors* (*687*). If you are interested in these courses, write to The Course Director, Tychon, Inc., Box 242, Blacksburg, VA 24060. Courses are also provided through the Center for Continuing Education and the Extension Division at Virginia Polytechnic Institute and State University, Blacksburg, VA. Call Dr. Linda Leffel at (703) 951-6208 for further information.

We have received assistance from a number of companies that have provided information, technical assistance, and permission to reproduce information from their data sheets and books. We would like to acknowledge the assistance of ADAC Corporation, Analog Devices, Inc., Burr-Brown Research Corporation, Data Translation, Inc., Datel Systems, Inc., National Semiconductor Corporation, Signetics Corporation, and Siliconix.

We also greatly appreciate the efforts of Mr. Murray Gallant, president of E&L Instruments, Inc., and his continuing support in this writing.

JONATHAN A. TITUS, CHRISTOPHER A. TITUS, PETER R. RONY, AND DAVID G. LARSEN

Contents

UNIT 1

UNIT 2

UNIT 3

UNIT 4

UNIT 5

UNIT 6

UNIT 7

1

Interfacing Digital-to-Analog Converters

INTRODUCTION TO THIS UNIT

Most digital-to-analog converters (DACs or D/A converters) may be considered to be voltage or current output devices. The output is in direct proportion to the value of the digital input, generally in binary form. In this unit you will see how digital-to-analog converters are interfaced to microcomputers and some of the uses to which the analog outputs may be put. The D/A converters are treated as modules and we will consider ourselves to be ignorant of their internal operation. Understanding how the converters actually work may be important, but we do not believe that it will greatly increase your interfacing and software development skills.

OBJECTIVES

At the end of this unit you will be able to do the following:

- Calculate the step-voltage for any *n*-bit D/A converter having an output range of *x* volts.
- Describe the operation of a simple 8-bit D/A converter interface.
- Describe how D/A converters may be used to generate the following types of output:
  Positive and negative ramps (sawtooths)

Triangular waves
Square waves
Complex waveforms

- Design an interface for a double-buffered, 10-bit D/A converter.
- Write software to control two D/A converters for an X *vs.* Y display of a data file.

Most digital-to-analog converters (DACs or D/A converters) provide an analog output that is proportional to some type of digital input. Typical digital inputs are in binary code and each individual input may have only one of two values, a logic 1 or a logic 0. If we consider a 4-bit D/A converter with input bits A, B, C and D, which have binary weights of 1, 2, 4 and 8, respectively, we find that there are 16 possible states from 0000_2 to 1111_2, inclusive. Let us assume that each binary input will contribute a voltage to the total which is proportional to its binary weight. The easiest voltages to use in this example are A = 1 volt, B = 2 volts, C = 4 volts and D = 8 volts. If a binary 0 is present in a particular column, the corresponding voltage is not added to the total. If a binary 1 is present, then the voltage is added to the total.

In this 4-bit D/A converter the voltage range will be from zero volts to 15 volts, as shown in Table 1-1.

Table 1-1. Four-Bit D/A Converter With Voltage Range From 0 Volts to 15 Volts

Binary Code Input				Output Voltage
D = 8	C = 4	B = 2	A = 1	(Volts)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Most real D/A converters use a series of resistors in an R, 2R, 4R, 8R, etc., binary weighted combination or in an R-2R ladder-type network. The ladder network of R and 2R values is preferred because of the ease of manufacture and improved electrical characteristics. The voltages are switches electronically, but a discussion of how this is done is beyond our scope. Current-output D/A converters are also available, their general advantage being fast *settling time,* generally a few hundred nanoseconds. For a further discussion of D/A converters and factors such as *linearity, offset,* and *resolution* we recommend the references mentioned in the introduction. These and other technical terms are defined in Appendix 1.

Digital-to-analog converters usually have transistor-transistor logic (TTL) compatible inputs which make them very easy to interface to the standard 7400-series logic and most microprocessor or microcomputer intergrated circuits. Some of the newer D/A converters are available with complementary metal-oxide-semiconductor (CMOS) or emitter-coupled-logic (ECL) compatible inputs. Although straight-binary coded D/A converters will be used in all of our examples and in the experiments, converters do come with other types of coded inputs. The types generally available are:

- Two's Complement
- Complementary Binary
- Complementary Offset Binary
- Offset Binary
- Binary Coded Decimal (BCD)

These codes may be useful in some applications but we have found that the binary-coded converters are the ones most widely used in microcomputer interfacing. Voltage outputs of 0–10 volts, 0–5 volts, ±2.5 volts, ±5 volts and ±10 volts are common. In short, the variety of D/A converters is so great that you can probably find one that will answer your specific needs. For nonstandard ranges, an operational amplifier with the necessary gain may be used to "customize" the output voltage to your needs. Few people design and build their own D/A converters because of the wide variety available. Converters with digital inputs of 8, 10, 12 and 14 binary bits are common with prices ranging from about $15 to several hundred dollars.

D/A CONVERTER OUTPUTS

When a parallel digital input is presented to a D/A converter, the converter will respond by outputting a corresponding voltage that is proportional to the value of the digital input. How can we determine what the output voltage will be for a given *n*-bit D/A converter with a range of *x* volts?

We will start by using a D/A converter with eight inputs coded in straight binary and with an output voltage range of 0 to 1 volt. The

Table 1-2. Output Voltages for Various Binary Inputs on a 0- to 1-Volt Scale D/A Converter

Binary Input	Output Voltage (Volts)
00000000	0.00390625
00000001	0.00781250
00000010	0.01171875
00000011	0.01562500
00000100	0.01953125
....	....
....	....
11111011	0.98046875
11111100	0.98437500
11111101	0.98828125
11111110	0.99218750
11111111	0.99609375

eight bits, *n,* provide 256 values from 0 to 255. The output can be divided into 256 individual steps of 0.00390 volt, or 3.9 millivolts per step. We can now determine the voltage output by the D/A converter for any input value, X:

$$V_{out} = \text{Output Voltage} = (\text{1-volt full scale}) \cdot (X/256)$$

or the following formula may be used:

$$V_{out} = \text{Output Voltage} = (X \text{ steps}) \cdot (0.00390 \text{ volt/step})$$

Some of the output voltages by the 0- to 1-volt full scale D/A converter are shown in Table 1-2.

It is important to note that while the D/A converter used in this example has a specified range from 0 to 1 volt, the analog output does not actually reach one volt. The reason for this becomes apparent if we examine the "weights" or voltages assigned to each of the eight bits. Since we are using the binary numbering system, each bit is an integral power of two. Thus when we go from the most significant bit (MSB), or the left-most bit, to the least significant bit (LSB), or the right-most bit, each bit position will have a weight that is one-half that of its left neighbor and twice that of its right neighbor.

Let us now consider the actual voltage weights assigned to the bits:

Full-Scale Voltage = 1 Volt

Least Significant Bit	$0.00390625 = \frac{1}{256}$	volt
	$0.00781250 = \frac{1}{128}$	volt
	$0.01562500 = \frac{1}{64}$	volt
	$0.03125000 = \frac{1}{32}$	volt
	$0.06250000 = \frac{1}{16}$	volt
	$0.12500000 = \frac{1}{8}$	volt
	$0.25000000 = \frac{1}{4}$	volt
Most Significant Bit	$0.50000000 = \frac{1}{2}$	volt

GENERATING ANALOG OUTPUTS

The D/A converter modules will quickly convert the binary value applied at the inputs to a voltage, but these inputs must be maintained for as long as the output voltage is required. If we attempt to interface a D/A converter to a microcomputer system simply by connecting the 8-bit data bus to the eight D/A converter inputs, we would observe a constantly changing voltage output by the D/A converter. The various output voltages would be caused by the many different 8-bit values presented to the D/A converter's inputs as the microcomputer uses the bus to transfer data and instructions. It is not only necessary to present the data to the D/A converter's inputs, but it must also be held or captured for as long as it is needed. A simple latch circuit may be used to capture the data from the data source, usually the microcomputer's data bus. Digital-to-analog converters generally require little more than latches and device decoding circuitry, plus the converter itself, for a complete interface.

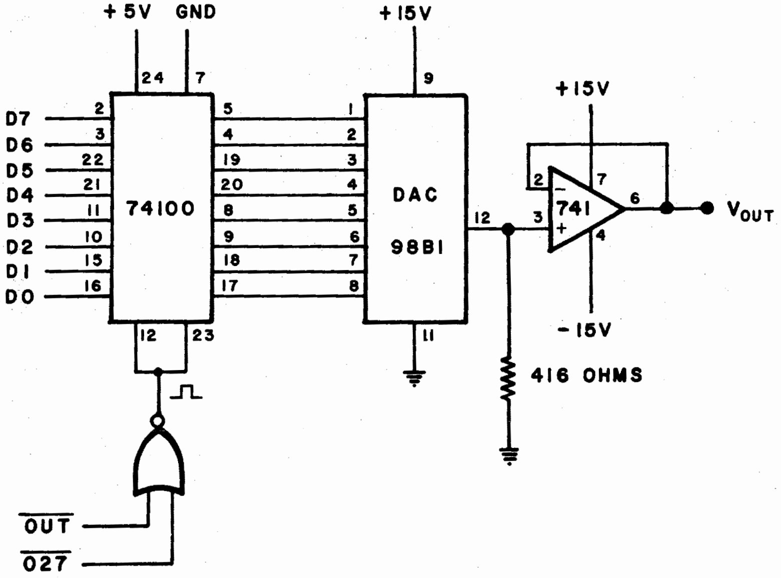


Fig. 1-1. Interfacing a Datel DAC-98BI D/A converter to an 8080-based microcomputer using a latch circuit.

A typical D/A converter interface is shown in Fig. 1-1. A Datel Systems, Incorporated DAC-98BI module is used as the 8-bit D/A converter. The DAC-89BI has a current output, so circuitry has been added to provide a 0- to 1-volt output range.

For more information about device decoding and device addressing such as that used in Fig. 1-1 to activate the latch, we refer you to *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 2,* Unit No. 17.

There are many uses for the analog outputs generated by computers using circuits similar to the one shown in Fig. 1-1. Some of these are listed below:

- Drive a servo motor
- Drive a strip-chart recorder
- Control a voltage-to-frequency converter
- Control a programmable power supply
- Drive an analog meter

In some applications, it is necessary to produce a linearly increasing voltage ramp to be used to control a test process, move a recorder pen, or move an oscilloscope beam. In the following example we will use the D/A converter to generate a linear ramp.

GENERATING A VOLTAGE RAMP

The voltage ramp output by a D/A converter is probably the simplest output that may be generated using fairly simple software. Even with a limited number of 8080 instructions in your software vocabulary, you can probably suggest a method of generating the linearly increasing binary values which are to be output to the D/A converter. In this example we will use the D/A converter which has been interfaced to the 8080-based computer, as shown in Fig. 1-1. An OUT 027 instruction transfers the contents of the 8080's A register to the D/A converter's latch circuit.

One of the easiest techniques that we can use to generate a ramp output uses one of the register increment instructions. A typical ramp is shown in Fig. 1-2.

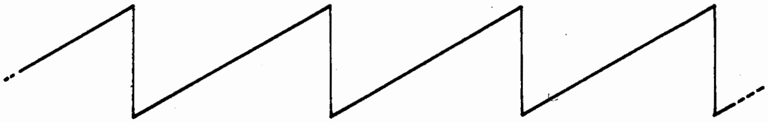


Fig. 1-2. Typical ramp waveform.

By incrementing the value in one of the 8080's registers and then latching it in the D/A converter's latch circuit, the 256 steps may be generated in about 3.2 milliseconds (this assumes an 8080 clock rate of 500 nanoseconds). The time period of each ramp and thus the slope may be slowed down by introducing a time delay subroutine into the ramp generation software. The software in Example 1-1 shows the steps necessary to generate the ramp, while the software in Example 1-2 shows how a time delay may be introduced by using a call to a time delay subroutine. The time delay subroutine may be a series of no-operation instructions (NOPs) or it may be a more complex, programmable time delay, which may answer many needs. In this second example, a general-purpose time delay subroutine has been written and assembled.

```
                       /EXAMPLE 1-1
                       /TYPICAL SOFTWARE FOR A POSITIVE RAMP OUTPUT

                       *003 000
003 000 074  START,    INRA       /INCREMENT THE CONTENTS OF A
003 001 323            OUT        /OUTPUT IT TO THE DAC
003 002 027            027        /DAC'S DEVICE CODE
003 003 303            JMP        /KEEP DOING IT AGAIN AND AGAIN
003 004 000            START
003 005 003            0
```

```
                       /EXAMPLE 1-2
                       /POSITIVE RAMP OUTPUT SOFTWARE WITH TIME DELAY

                       *003 000
003 000 061            LXISP      /LOAD A STACK ADDRESS SO THAT SUBROUTINES
003 001 377            377        /MAY BE USED
003 002 003            003
003 003 074  LOOP,     INRA       /INCREMENT A
003 004 323            OUT        /OUTPUT IT TO DAC
003 005 027            027
003 006 315            CALL       /CALL THE TIME DELAY SUBROUTINE
003 007 014            DELAY
003 010 003            0
003 011 303            JMP        /DO IT AGAIN
003 012 003            LOOP
003 013 003            0

                       /THIS IS THE DELAY SUBROUTINE

003 014 365  DELAY,    PUSHPSW    /SAVE REG A & FLAGS
003 015 325            PUSHD      /SAVE D&E
003 016 021            LXID       /LOAD THE TIMING BYTES
003 017 065            065        /065 TO REGISTER E
003 020 001            001        /001 TO REGISTER D
003 021 033  DEC,      DCXD       /DECREMENT REG PAIR D
003 022 172            MOVAD
003 023 263            ORAE       /IF D OR E IS NON-ZERO, WE
003 024 302            JNZ        /DECREMENT THE PAIR AGAIN
003 025 021            DEC
003 026 003            0
003 027 321            POPD       / GET REG VALUES BACK
003 030 361            POPPSW     /RESTORE A & FLAGS, TOO
003 031 311            RET        /RETURN
```

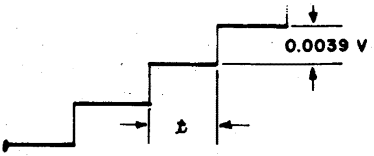


Fig. 1-3. Expanded linear output of D/A voltage converter.

It is important to remember that the D/A converter is still generating discreet voltage step outputs of 0.0039 volt, so that when the linear output is expanded it appears as shown in Fig. 1-3.

The time period, *t,* depends upon how fast we want to have the computer output the analog information. The time delay subroutine shown in Example 1-2 could be used to increase the time period, *t,* to several seconds. Filtering or integrating the voltage may be useful if the step function is not acceptable and a more linear output is necessary. In most applications, the output is left unfiltered, although amplification may be used to increase the range to 0 to 10 volts or perhaps to a range of 0 to 2.35 volts for a specific application.

In some applications, a negative ramp may be needed that decreases from a high potential to a low potential. This may be accomplished by adding an inverting, unity-gain amplifier and some additional components to the D/A converter interface, thus providing a hardware solution. In our case, it is more reasonable, however, to change the software to decrement the register, thus causing the output to the D/A converter to decrease in value. This is the software solution. The software for the negative ramp is shown in Example 1-3.

```
                        /EXAMPLE 1-3
                        /TYPICAL SOFTWARE FOR A NEGATIVE RAMP OUTPUT

                        *003 000
003 000 075   START,    DCRA        /DECREMENT THE CONTENTS OF A
003 001 323             OUT         /OUTPUT IT TO THE DAC
003 002 027             027
003 003 303             JMP         /KEEP GOING
003 004 000             START
003 005 003             0
```

Of course, the additional call to the time delay subroutine could have been added, but we have not shown it in this example.

You should note that when one of the 8080's 8-bit, internal registers contains 377, or 11111111_2, and is incremented, it becomes 000, or 00000000_2. Thus, the largest value becomes the smallest value in one step to start the ramp at the "bottom" again. Likewise, when 000, or 00000000_2, is decremented it becomes 377, or 11111111, the smallest value becoming the largest value, again in a single step. No additional software is necessary to reinitialize the register to these starting values.

COMPLEX RAMP OUTPUTS

There may be applications in which a complete sweep, *i.e.*, from minimum to maximum value, of the ramp's output is not needed. For example, suppose that we have interfaced a D/A converter to our computer and that the converter's output ranges from 0 to 10 volts full scale. Our application requires that the output sweep between a specific low voltage limit and a specific high voltage limit. In this example, we will set the lower limit at 2.0 volts and the higher limit at 8.0 volts. Using the full 0- to 10-volt range, we could use operational amplifiers to attenuate the output range and to offset it, too. We will use software instead.

This is a typical "hardware/software tradeoff." If an operational amplifier scheme is used, more hardware is added to the system and it becomes more difficult to change the high and low limits than if software is used. The hardware solution still has merit, though, since it will compress the 256 voltage points into smaller increments when output through the added operational amplifiers. This means that the 2.0- to 8.0-volt range will be divided into 256 points or steps. The software solution will result in the usual 0.0039 volt steps for a 0- to 1-volt D/A converter or 0.039 volt steps for a 0- to 10-volt D/A converter.

The first task in solving this problem is to determine the binary values of the limiting voltages. This may be done empirically by breadboarding a D/A converter and slowly increasing the binary inputs and noting their values when the D/A converter's output reaches 2.0 and 8.0 volts. The limits may also be calculated as follows:

$$\frac{\text{10 Volts Full Scale Output}}{\text{256 Steps}} = 0.0390 \text{ volt/step}$$

and

$$\frac{\text{2.0 Volts}}{\text{0.0390 volt/step}} = 51.3 \text{ steps}$$

It is, of course, impossible to have fractional steps. The only steps that are allowed are those with integral values between 0 and 255. We must decide if we want the lower limit to be 51 steps (1.989 volts) or 52 steps (2.028 volts). In this case the 1.989-volt value is acceptable as the lower limit. In a similar way we determine that the upper voltage limit is represented by 205 steps.

The software shown in Example 1-4 will start at the lower limit and increase the voltage output by the D/A converter to the upper limit of 8.0 volts in discreet 39-millivolt steps.

Table 1-3. Upper and Lower Voltage Limits and Their Binary, Octal, and Hexadecimal Equivalents

Voltage	Steps	Binary	Octal	Hex
2.0 (1.989)	51	00110011	063	33
8.0 (7.995)	205	11001101	315	CD

```
                        /EXAMPLE 1-4
                        /RAMP OUTPUT WITH UPPER & LOWER LIMITS
                        *003 000
003 000 076   START,    MVIA       /LOAD REG A WITH STARTING VALUE
003 001 063             063        /063 = 051 DECIMAL = 2.0 VOLTS
003 002 323   LOOP,     OUT        /OUTPUT IT TO THE DAC
003 003 027             027
003 004 074             INRA       /INCREMENT THE VALUE
003 005 000             NOP        /THREE NOP'S LEFT FOR ADDITION OF
003 006 000             NOP        /A CALL TO A TIME DELAY
003 007 000             NOP
003 010 376             CPI        /COMPARE THE VALUE TO THE UPPER LIMIT
003 011 315             315        /315 = 205 DECIMAL = 8.0 VOLTS
003 012 302             JNZ        /IF NOT EQUAL, DO THE LOOP AGAIN
003 013 002             LOOP
003 014 003             0
003 015 303             JMP        /IF EQUAL, REINITIALIZE AND DO IT
003 016 000             START      /AGAIN
003 017 003             0
```

The software shown in Example 1-4 will cause the D/A converter to output a maximum voltage which is within 39 millivolts of the upper limit. This occurs since the comparison between the limit value and the actual register value is performed after the register is incremented, but before the value is output to the D/A converter. If the D/A converter's output is to actually reach the upper limit, the comparison value must be changed from 315 to 316.

This simple program may be used to cycle a ramp between any two voltages that are within the range of the D/A converter's output voltage. The ramp may also be a negative-going one by using the decrement instruction, comparing the register value to the lower limit and when it is reached, resetting the register to the upper limit.

The slope of the output is determined by the time required for the computer to execute all of the instructions. Three no-operation instructions (NOP=000) are provided in Example 1-4 so that a call to a time delay subroutine may be inserted. For optimum speed, the NOPs should be removed and the program condensed.

The time delay subroutine could be a more complex program than a simple time delay loop. A flag, interrupt, or other stimulus could govern whether or not the computer proceeds to the next step. Software could also be added to wait for an external event to take place before starting another ramp cycle.

TRIANGULAR WAVE OUTPUT

The previous software examples for ramp generation could serve as the basis for additional software which permits the D/A converter to generate a triangular wave output. In this example, a complete 0- to 10-volt full scale triangular wave output will be assumed. The first attempt at a program that will output this type of a waveform is shown in Example 1-5.

```
                                /EXAMPLE 1-5
                                /TRIANGULAR WAVE OUTPUT PROGRAM #1
                                *003 000
003 000 257            XRAA     /CLEAR REG A
003 001 074   UP,      INRA     /INCREMENT REG A
003 002 323            OUT      /OUTPUT IT TO THE DAC
003 003 027            027
003 004 302            JNZ      /IF STILL NON-ZERO, KEEP GOING
003 005 001            UP
003 006 003            0
003 007 075   DOWN,    DCRA     /DECREMENT REG A
003 010 323            OUT      /OUTPUT IT TO THE DAC
003 011 027            027
003 012 302            JNZ      /IS IT = 0?
003 013 007            DOWN     /NO, GO DOWN ONE MORE
003 014 003            0
003 015 303            JMP      /YES, START UP AGAIN
003 016 001            UP
003 017 003            0
```

If we attempt to run this program, a "glitch" is observed at the apex of the triangle. The glitch is caused by the program. After incrementing the count register to its maximum value of 377, the software must again increment it to 000 and output it to the D/A converter before the JNZ instruction is used to test the value.

When the value 000 is reached, the JNZ instruction is "ignored" and the DOWN section starts its execution. The illustration in Fig. 1-4 shows what the D/A converter's output would look like when executing the software from Example 1-5. These examples assume an ideal output from the D/A converter.

The glitch may be removed by simply reversing the position of the INRA and the OUT 027 instructions. The modified software, shown in Example 1-6, will produce a reasonable triangular wave output. This new program, Triangular Wave Output Program #2, now checks the incremented value prior to the output instruction, pre-

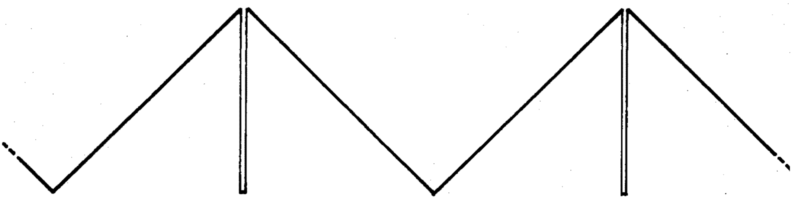


Fig. 1-4. D/A converter output when software from Example 1-5 is executed.

venting the output of the sequence of values 377, 000, 377 generated in the first program, Example 1-5.

```
                            /EXAMPLE 1-6
                            /TRIANGULAR WAVE OUTPUT PROGRAM #2
                            *003 000
003 000 257                    XRAA        /CLEAR REG A
003 001 323     UP,            OUT         /OUTPUT VALUE IN REG A TO DAC
003 002 027                    027
003 003 074                    INRA        /INCREMENT REG A
003 004 302                    JNZ         /IF NOT ZERO, GO UP AGAIN
003 005 001                    UP
003 006 003                    0
003 007 075     DOWN,          DCRA        /DECREMENT REG A
003 010 323                    OUT         /OUTPUT IT TO THE DAC
003 011 027                    027
003 012 302                    JNZ         /IF NOT ZERO, GO DOWN AGAIN
003 013 007                    DOWN
003 014 003                    0
003 015 303                    JMP         /IF ZERO, GO UP
003 016 001                    UP
003 017 003                    0
```

There is, however, another problem with this second program. The high value of 377 and the low value of 000 are both output to the D/A converter twice in succession, giving the triangular output a plateau at the high and low points. These plateaus, illustrated in Fig. 1-5, are twice as long as the other voltage steps generated by the software.

This problem, too, is caused by the software. After the value 377 is output to the D/A converter by the OUT 027 instruction in the UP software loop, the accumulator, or A register, is incremented to 000, checked by the JNZ instruction and then decremented back to 377. This value is then output in the DOWN section of the program. Thus, the value 377 has been output to the D/A converter twice. Clearly some modification must be made in the software to correct this.

The software shown in Example 1-7 has some additional increment and decrement instructions to eliminate the "double output" of the values 377 and 000. This program provides "glitchless" output

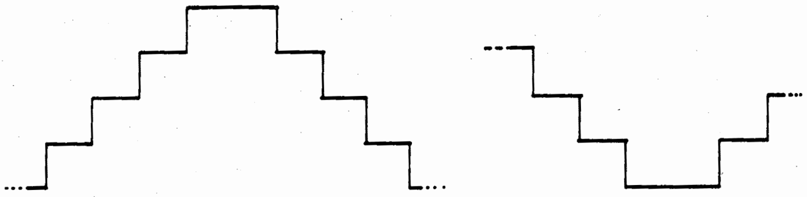


Fig. 1-5. Triangular output with high and low plateaus.

of a triangular wave. It should be noted that there are probably a number of other software solutions to this problem.

```
                        /EXAMPLE 1-7
                        /TRIANGULAR WAVE OUTPUT PROGRAM #3
                        *003 000
003 000 323   UP,       OUT       /OUTPUT REG A TO DAC
003 001 027             027
003 002 074             INRA      /INCREMENT REG A
003 003 302             JNZ       /IF NOT ZERO, GO UP AGAIN
003 004 000             UP
003 005 003             0
003 006 075             DCRA      /IF ZERO, DECREMENT IT TO 377
003 007 075   DOWN,     DCRA      /DECREMENT IT AGAIN
003 010 323             OUT       /OUTPUT IT TO DAC
003 011 027             027
003 012 302             JNZ       /IF NOT ZERO, GO DOWN AGAIN
003 013 007             DOWN
003 014 003             0
003 015 074             INRA      /IF ZERO, INCREMENT IT TO 001
003 016 303             JMP       /GO BACK UP AGAIN
003 017 000             UP
003 020 003             0
```

The triangular wave output by the software shown in Example 1-7 has equal positive and negative ramp periods since the two software loops, UP and DOWN, contain the same number of instructions with equal execution times. For nonsymmetrical outputs where slopes are to be different, NOP instructions or calls to time delay subroutines could be easily added to the program. For example, if 11 NOP instructions are added to the DOWN loop between the OUT 027 and the JNZ instruction, the ratio of positive ramp period to negative ramp period is approximately 3 to 7.

SQUARE WAVES AND OTHER OUTPUTS

Square waves are generated easily by using a D/A converter and software to alternate between a low and a high voltage. The periods and frequencies may be determined in the software with either NOP instructions or calls to time delay subroutines. A typical example of a square wave output program is shown in Example 1-8.

```
                       /EXAMPLE 1-8
                       /TYPICAL SQUARE WAVE OUTPUT PROGRAM
                       *003 000
003 000 076  START,    MVIA        /LOAD REG A WITH LOWER LIMIT
003 001 023            023
003 002 323            OUT         /OUTPUT IT TO THE DAC
003 003 027            027
003 004 315            CALL        /DO A TIME DELAY ROUTINE
003 005 021            DELAY
003 006 003            0
003 007 076            MVIA        /LOAD REG A WITH UPPER LIMIT
003 010 307            307
003 011 323            OUT         /OUTPUT IT TO THE DAC
003 012 027            027
003 013 315            CALL        /DO ANOTHER DELAY LIKE THE ONE
003 014 021            DELAY       /SHOWN IN EXAMPLE 1-2
003 015 003            0
003 016 303            JMP         /DO IT AGAIN
003 017 000            START
003 020 003            0

003 021 000  DELAY,    0           /DELAY SUBROUTINE IS LOCATED HERE
```

By changing the software to use two different time delay subroutines, rectangular wave outputs may be generated. Pulses may also be generated by using a very short time delay program for the pulse and a much longer time delay program for the interval between pulses. Depending upon the frequency of the 8080 microprocessor chip's crystal clock, very short pulses may be generated. Example 1-9 shows the software which may be used to generate a 20-microsecond-long 10-volt pulse with a repetition rate of about 10 milliseconds.

```
                       /EXAMPLE 1-9
                       /TYPICAL PULSE OUTPUT PROGRAM USING A DAC
                       *003 000
003 000 076  LOOP,     MVIA        /LOAD REG A WITH HIGHEST VALUE
003 001 377            377
003 002 323            OUT         /OUTPUT IT TO THE DAC
003 003 027            027
003 004 076            MVIA        /LOAD REG A WITH LOWEST VALUE
003 005 000            000
003 006 323            OUT         /OUTPUT IT TO THE DAC
003 007 027            027
003 010 315            CALL        /CALL A DELAY ROUTINE
003 011 016            DELAY
003 012 003            0
003 013 303            JMP         /GO BACK AND DO IT AGAIN
```

```
003 014 000          LOOP
003 015 003          0
003 016 000  DELAY,  0         /10 MILLISECOND DELAY MAY BE PLACED
003 017 000          0         /HERE, SEE EXAMPLE 1-2.
```

Complex waveforms may be generated by treating the square wave, the triangular wave and the ramp output software as subroutines. Linking these subroutines together with a series of call instructions and perhaps calls to time delay subroutines will produce complex outputs. These outputs are generally difficult to generate with standard test equipment and they may be useful for testing or cycling power supplies, electromechanical devices, etc. While the output may be slow, since it is limited by the execution speed of the 8080's instructions, it would be difficult to generate these complex outputs in other ways.

DATA DISPLAYS AND OUTPUTS

Many applications require that the computer be used to output a file of data for later evaluation. The data can take two forms: a list or numeric printout of the data or a graphic representation presented on a chart, drawing, etc. In many cases it will be much easier to distinguish trends and changes in the values that have been stored in the computer if the graphical output is chosen. The graph may be output to an oscilloscope, an X–Y plotter, or a strip-chart recorder. The example in Fig. 1-6 illustrates the difference between the numeric output and the graphic output.

Our next task will be to develop a program to output some data in graphic form. A file of 8-bit binary words or bytes will be set up in the computer's memory between addresses 007 000 and 007 143, inclusive. This will provide us with 100_{10} points to display in some way. A strip-chart recorder is available which is compatible with the voltage output of the 8-bit D/A converter previously interfaced to our 8080 microcomputer. The data points are to be output once and since the computer will be able to output all 100 data points much faster than the strip-chart's servo motor can respond, a delay of 100 milliseconds will be inserted into the software between the output of each point. The output of all 100 points will take 10 seconds.

To use a file of data stored in the 8080's memory, address pointers must be set up in the H and L registers (register pair H) to provide the 16-bit address necessary to point to locations 007 000 through 007 143. We will also need to have access to a subroutine that provides the 100 millisecond delay. Although important in some applications, we will ignore the time necessary to actually get the data from the memory and output it to the D/A converter. This will cause the

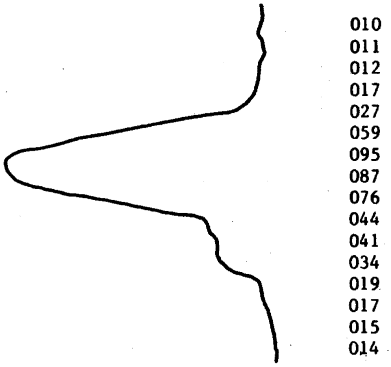


Fig. 1-6. Difference between numeric output and graphic output.

real time between points to be somewhat longer than 100 milliseconds, but by a negligible amount. A counter register is also needed to keep track of the number of points that have been output so that the program will stop when all 100 of the points have been plotted. Since subroutines are to be used, a stack area must be set up in read/write memory. It is a good idea to locate the stack in an area of read/write memory distant from that used for program and data storage. A flowchart of the 100-point output software is provided in Fig. 1-7.

The software for the 100-point plot program is provided in Example 1-10 in its fully assembled form:

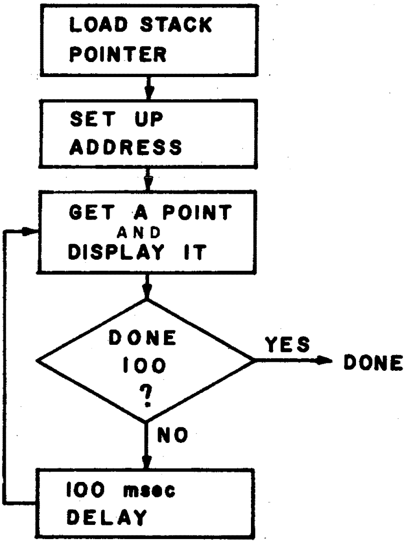


Fig. 1-7. Flowchart for a 100-point data display program or plotting program that uses an 8-bit D/A converter.

```
                        /EXAMPLE 1-10
                        /ONE HUNDRED POINT DISPLAY SOFTWARE
                        *003 000
003 000 061             LXISP     /LOAD A STACK POINTER
003 001 377             377
003 002 003             003
003 003 041   DISPLA,   LXIH      /LOAD A POINT COUNT
003 004 000             DATA1     /OF THE DATA STORAGE AREA
003 005 007             0
003 006 006             MVIB      /LOAD A POINT COUNT
003 007 144             144       /144 = 100 DECIMAL
003 010 176   LOOP,     MOVAM     /GET AN 8-BIT DATA POINT
003 011 323             OUT       /OUTPUT IT
003 012 027             027
003 013 043             INXH      /INCR THE ADDR
003 014 005             DCRB      /DECREMENT POINT COUNT
003 015 312             JZ        /IF DONE, GET OUT OF THE LOOP
003 016 026             DONE
003 017 003             0
003 020 315             CALL      /IF NOT DONE, DO A 100-MILLI-
003 021 027             HUNMIL    /SECOND DELAY
003 022 003             0
003 023 303             JMP       /DO THE NEXT POINT
003 024 010             LOOP
003 025 003             0
003 026 166   DONE,     HLT       /WHEN FINISHED DISPLAYING, HALT
003 027 000   HUNMIL,   0         /100 MSEC DELAY SOFTWARE MAY BE ADDED
003 030 000             0         /HERE. SEE EXAMPLE 1-2 FOR A TYPICAL
003 031 000             0         /TIME DELAY SUBROUTINE
                        *007 000
007 000 000   DATA1,    0         /DATA STARTS HERE FOR 100 ADDRESSES
```

This program will output the 100-point data file in approximately 10 seconds, or 0.1 seconds per point. The 8080's B register or any other unused register may be used to count the 100 points, but is this really necessary? We certainly need a counter, but the L register is already being used as a counter since it is incremented in each loop through the software. It will increase from 000 to 143 during the course of the program.

The L register may be used to perform two tasks: (1) to provide the low address of the data points and (2) to provide an increasing point count. By comparing the value contained in the L register to 144_8 we can determine when the file output has been completed. Remember that the 8080's comparison instructions do not alter either of the two data bytes being compared.

In this example, the contents of the L register will be transferred to the A register and then compared to a data byte in a compare-immediate (CPI) instruction. There is no need to transfer the value in the A register back to the L register since the data is *copied* during register–register transfers.

In the software in Example 1-11, the L register is used as the counter and as the low address value, thus permitting us to save the B register for future use.

```
                            /EXAMPLE 1-11
                            /IMPROVED 100-POINT DISPLAY SOFTWARE
                            /SHOWING ONLY THE MAIN PROGRAM
                            *003 000
003 000 061                 LXISP
003 001 377                 377
003 002 003                 003
003 003 041     DISPLA,     LXIH
003 004 000                 DATA1
003 005 007                 0
003 006 176     LOOP,       MOVAM
003 007 323                 OUT
003 010 027                 027
003 011 043                 INXH        /INCREMENT MEMORY ADDRESS
003 012 175                 MOVAL       /MOVE CONTENTS OF REG L TO REG A
003 013 376                 CPI         /COMPARE IT TO
003 014 144                 144         /144 = 100 DECIMAL
003 015 312                 JZ          /IF ALL POINTS DISPLAYED
003 016 026                 DONE        /WE'RE DONE
003 017 003                 0
003 020 315                 CALL
003 021 027                 HUNMIL
003 022 003                 0
003 023 303                 JMP
003 024 006                 LOOP
003 025 003                 0
```

HIGH SPEED CRT DISPLAYS WITH D/A CONVERTERS

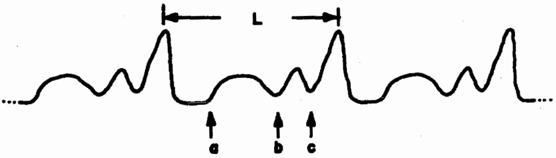
In the previous example, the data stored in the 100-point, 8-bit data file was output only once, at low speed. There are many applications where we may wish to output the data very quickly and often more than a single time. This is the case when a cathode ray tube or oscilloscope is to be used to display the data.

By removing the three-byte call to the 100-millisecond time delay subroutine in the strip-chart output program, Example 1-11, the 100-point data file could be output once, very quickly, perhaps to a storage oscilloscope. If, however, a normal oscilloscope is the only type available, the computer will have to be set up to output the data points again and again so that the display will be maintained.

Instead of jumping to the address labeled DONE in the strip-chart software, Examples 1-10 and 1-11, the computer must be reprogrammed to repeatedly display the complete 100-point data file. This may be done by having the computer jump back to that point in the program at which the address of the data file is loaded into register pair H. This is shown in Example 1-12 in which the time delay subroutine call has been removed.

```
                             /EXAMPLE 1-12
                             /ONE HUNDRED POINT DISPLAY SOFTWARE FOR HIGH
                             /SPEED DISPLAY DEVICES
                             *003 000
003 000 061                  LXISP      /LOAD A STACK POINTER
003 001 377                  377
003 002 003                  003
003 003 041   DISPLA,        LXIH       /LOAD THE DATA ADDRESS
003 004 000                  DATA1
003 005 007                  0
003 006 176   LOOP,          MOVAM      /GET AN 8-BIT DATA POINT
003 007 323                  OUT        /OUTPUT IT TO THE DAC
003 010 027                  027
003 011 043                  INXH       /INCR MEMORY ADDRESS
003 012 175                  MOVAL      /MOVE L TO A
003 013 376                  CPI        /COMPARE IT TO 100 DECIMAL
003 014 144                  144
003 015 302                  JNZ        /IF NOT EQUAL, DO IT AGAIN
003 016 006                  LOOP
003 017 003                  0
003 020 303                  JMP        /IF DONE, REINITIALIZE AND
003 021 003                  DISPLA     /START IT AGAIN
003 022 003                  0
```

This program will display the 100 points in the data file again and again to provide a display which might look like this:

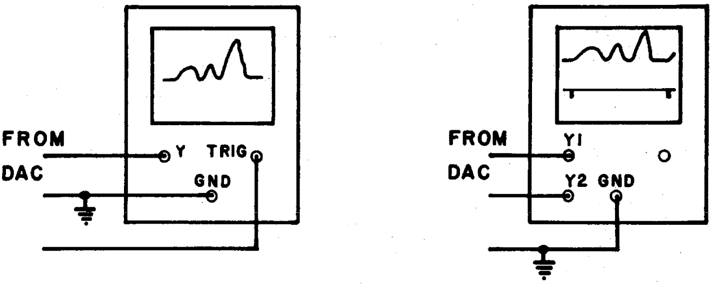


When the data file is being displayed it is difficult to determine where the 100-point data file starts and where it ends. We can probably assume that the file's length is *L,* since this is the distance between repetitive features being displayed. The length might be more properly called the period, *T,* or ΔT, since it is a time period which is being measured on the oscilloscope. The top of the peak is probably not the start of the data file; the file might start at points *a, b* or *c.*

Most data files will be discontinuous enough at their beginning and end so that the displayed file will not run together, but this possibility exists and it must be considered. Some method of defining the start of the data file is required. The first point in the file could be preset to a known value, say, 377, but this wastes one of the points and it may be "lost" in among the other data values in the file. An easy way to overcome this problem is to have the computer generate an output which may be used to trigger the oscilloscope. It could also be used on a second trace, if one is available on the oscilloscope, to identify the file's starting point. The output command could be inserted at the start of the DISPLAY software as shown in Example 1-13.

```
                              /EXAMPLE 1-13
                              /GENERATING A TRIGGER PULSE IN THE DISPLAY
                              /SOFTWARE
                              *003 000
003 000 061                   LXISP     /LOAD A STACK POINTER
003 001 377                   377
003 002 003                   003
003 003 041     DISPLA,       LXIH      /LOAD THE DATA ADDRESS
003 004 000                   DATA1
003 005 007                   0
003 006 323                   OUT       /GENERATE A TRIGGER PULSE AT THE
003 007 030                   030       /START OF THE FILE
003 010 176     LOOP,         MOVAM     /GET AN 8-BIT DATA POINT
```

The OUT 030 pulse is used to trigger the oscilloscope or to provide a "mark" at the start of the file (Fig. 1-8).



(A) Triggering the oscilloscope.

(B) Providing a "mark."

Fig. 1-8. OUT 030 pulse is used to trigger the oscilloscope or to provide a "mark" at the start of a file.

These techniques for the display of the data in the file rely upon the oscilloscope's time base or the strip-chart's motor to generate the time axis for the data output. This is accomplished by either moving the oscilloscope's electron beam or the strip-chart's paper at a fixed, known rate. A second D/A converter could also be used to provide the time base or X-axis information.

The second D/A converter could be used to generate the ramp potential to move the oscilloscope's electron beam by using the software shown in Example 1-1. Using two D/A converters to provide computer controlled output of both data and time-base information provides what is called an X–Y or X *vs.* Y display since the computer can control each axis independent of the other (Fig. 1-9).

The software must now include program steps to generate the ramp as well as to output the data. For 100 data points, the X or time-axis D/A converter will only have a range of 100/256th of its full scale range since only the first 100 voltage steps will be output. This limitation is overcome by using the oscilloscope's gain controls to expand the X-axis to fill the screen. The software which controls the X–Y display of data is provided in Example 1-14.

```
                          /EXAMPLE 1-14
                          /SOFTWARE FOR RAMP AND DATA OUTPUT USING A
                          /DUAL D/A CONVERTER INTERFACE
                          *003 000
003 000 006   DISPLA,     MVIB       /SET UP A REGISTER FOR THE RAMP VALUES
003 001 000               000
003 002 041               LXIH       /SET UP THE DATA ADDRESS
003 003 000               DATA1
003 004 007               0
003 005 004   LOOP,       INRB       /INCREMENT RAMP DATA
003 006 170               MOVAB      /MOVE RAMP DATA TO REG A
003 007 323               OUT        /OUTPUT IT TO X-AXIS DAC
003 010 036               036
003 011 176               MOVAM      /GET THE FIRST DATA POINT
003 012 323               OUT        /OUTPUT IT TO Y-AXIS DAC
003 013 027               027
003 014 043               INXH       /INCREMENT MEMORY ADDRESS
003 015 175               MOVAL      /MOVE L TO A
003 016 376               CPI        /COMPARE IT TO 100 DECIMAL
003 017 144               144
003 020 302               JNZ        /IF NOT ZERO, GET ANOTHER POINT
003 021 005               LOOP       /AND DISPLAY IT
003 022 003               0
003 023 303               JMP        /ALL 100 POINTS OUTPUT, SO RE-
003 024 000               DISPLA     /REINITIALIZE AND DO IT AGAIN
003 025 003               0
                          *007 000
007 000 000   DATA1,      0          /DATA STARTS HERE AND GOES FOR 100
007 001 000               0          /POINTS
```

Each time that the computer goes through the software loop, the value in the B register is incremented and output as the time-base or X-axis data. This software may work fairly well, but upon close examination there may be streaks of light on the oscilloscope's screen. These streaks of light are caused by the short time that that each point

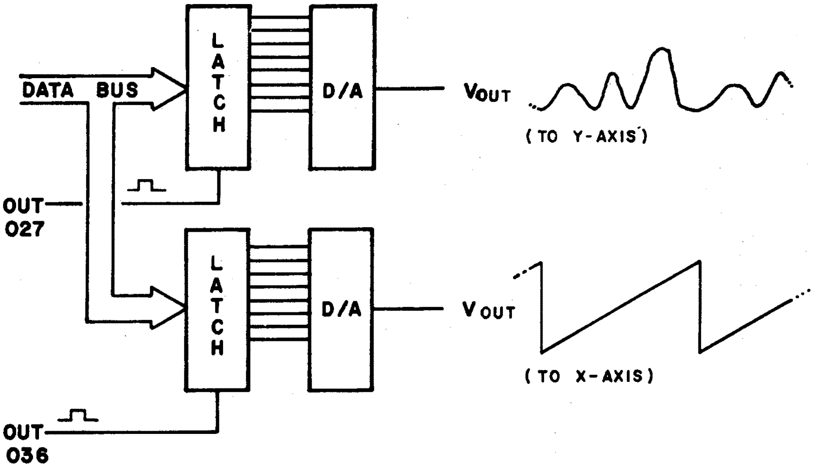


Fig. 1-9. Using two 8-bit D/A converters to generate X- and Y-axis voltages for plotting or graphic display.

is displayed in relation to the time that the oscilloscope spends in moving the electron beam to the next point's position. We may need to introduce a short time delay into the software, perhaps between the OUT and the INXH instructions, so that the beam will stop and "intensify" each point. This time delay might be a few NOP instructions or an actual call to a time delay subroutine.

This program is interesting since the value in the B register and the value in the L register will always be equal. In a previous example we saw that the value in the L register could be used as both a counter and as a memory address. In this case the value in the L register may be used as the memory address and as the value for the X-axis D/A converter. This eliminates the use of register B.

Even if the data field is offset or biased to have a starting address of 007 100, the value in the L register may still be used simply by subtracting (or adding) the offset prior to the use of the value as the data to be output to the X-axis D/A converter. This mode of operation is shown in Example 1-15.

```
                          /EXAMPLE 1-15
                          /SOFTWARE FOR AN X-Y DAC DISPLAY OF DATA WHERE
                          /THE MEMORY ADDRESS ALSO GENERATES A RAMP
                          *003 000
003 000 061               LXISP     /SET UP A STACK
003 001 377               377
003 002 003               003
003 003 041    DISPLA,    LXIH      /SET UP NEW ADDRESS POINTERS
003 004 100               NEWPNT
```

```
003 005 007            0
003 006 175  LOOP,     MOVAL      /GET THE LOW ADDRESS
003 007 376            CPI        /COMPARE IT TO 144 POINTS (100 DECIMAL)
003 010 244            244        /PLUS OFFSET OF 100 ADDRESSES (OCTAL)
003 011 312            JZ         /IF EQUAL, GO BACK AND START AGAIN
003 012 003            DISPLA     /IF NOT EQUAL, KEEP ON GOING
003 013 003            0
003 014 326            SUI        /SUBTRACT THE OFFSET OF
003 015 100            100        /100 ADDRESSES (OCTAL OFFSET)
003 016 323            OUT        /OUTPUT THIS AS THE RAMP DATA TO
003 017 036            036        /THE X-AXIS DAC
003 020 176            MOVAM      /GET A DATA POINT
003 021 323            OUT        /OUTPUT IT TO THE Y-AXIS DAC
003 022 027            027
003 023 315            CALL       /WAIT A BIT SO THE BEAM CAN
003 024 032            TIMER      /INTENSIFY THE SPOT
003 025 003            0
003 026 043            INXH       /INCREMENT THE MEMORY ADDRESS
003 027 303            JMP        /GO BACK AND TRY AGAIN
003 030 006            LOOP
003 031 003            0

003 032 006  TIMER,    MVIB       /LOAD B WITH A NUMBER
003 033 200            200
003 034 005            DCRB       /DECREMENT IT
003 035 302            JNZ        /NOT ZERO, JUMP BACK 1
003 036 034            TIMER+2
003 037 003            0
003 040 311            RET        /COUNT=0, RETURN
                       *007 000
007 100 000  NEWPNT,   0          /START OF THE NEW, OFFSET DATA FILE
```

You may have asked yourself why we have chosen to add another D/A converter to the microcomputer when the external time bases provided by the oscilloscope and the strip-chart recorder worked so well. The utility of an X–Y display is that it allows the computer to do complex graphics which are not possible with Y–T displays.

There are two types of X-Y data displays: (1) where the data is to be displayed using a uniform time base with constant time intervals between each point and (2) where two data files are used to define a point's position for drawings, maps, circuit layouts, etc.

When used for the second type of display, co-ordinate points are assigned to the various points on the oscilloscope screen and it becomes possible to output squares, circles, alphanumeric characters and other shapes and forms not possible without two D/A converters. Using two, 8-bit D/A converters there are 65,536 points which may be "addressed" by the computer through the D/A converters. The utility and simplicity of a Y-T display is still maintained since one of the D/A converters may still be used to output a ramp, as in previous examples.

We will now discuss the use of two D/A converters to output X-Y data from two independent 100-point data files. The length of the files may be increased or decreased as needed. If the software is to be used with an X-Y plotter, pen-up and pen-down controls would have to be added to the interface and a time delay would be needed so that the servo motor could keep up with the computer. These have been implemented in the software. A flowchart has been used to illustrate the operation of the program and a completely assembled software listing is also provided. The program will continuously display the data in the file. If the program is to be used with a plotter the necessary halt instruction should be added (Fig. 1-10). The halt instruction is used for plotter output with the final jump being used if the data points are to be output again.

In most cases register H is used as the pointer to indicate which memory location will be involved in data transfer to or from one of the 8080's internal registers. There are also instructions which allow the contents of register pair B or register pair D to be used as memory address pointers. In the X-Y data output software shown in Example 1-16, register pair H is used to point to the Y-axis data file and register pair D is used to point to the X-axis data file. The Y-axis data file will reside between addresses 007 000 and 007 177 and the X axis data will reside between 007 200 and 007 377. This allows for data files of 128 points, although only 100 will be used in our example.

When using an oscilloscope with the X-Y data display software in Example 1-16, the call to the WAIT1 subroutine may be eliminated by substituting three NOP instructions in place of the three-byte call instruction. The pen-up and pen-down commands may be useful if you wish to turn the oscilloscope's electron beam on and off. This is called *Z-axis modulation.*

```
                            /EXAMPLE 1-16
                            /SOFTWARE FOR A DISPLAY OF X–Y DATA USING TWO,
                            /INDEPENDENT DATA FILES, X DATA & Y DATA
                            *003 000
003 000 061                 LXISP      /SET UP A STACK
003 001 377                 377
003 002 003                 003
003 003 041     START,      LXIH       /SET UP POINTER FOR Y DATA
003 004 000                 YDATA
003 005 007                 0
003 006 021                 LXID       /SET UP POINTER FOR X DATA
003 007 200                 XDATA
003 010 007                 0
003 011 176     MORE,       NOVAM      /GET Y DATA
003 012 323                 OUT        /OUTPUT IT
003 013 027                 027
003 014 032                 LDAXD      /GET X DATA
003 015 323                 OUT        /OUTPUT IT
```

```
003 016 036          036
003 017 315          CALL      /WAIT FOR PLOTTER TO MOVE
003 020 100          WAIT1
003 021 003          0
003 022 323          OUT       /OUTPUT PULSE PUTS PEN DOWN
003 023 030          030       /OR TURNS SCOPE BEAM ON
003 024 315          CALL      /WAIT FOR POINT TO BE PLOTTED OR
003 025 130          WAIT2     /INTENSIFIED
003 026 003          0
003 027 323          OUT       /OUTPUT PULSE LIFTS PEN OR TURNS
003 030 031          031       /SCOPE BEAM OFF
003 031 043          INXH      /INCREMENT MEMORY POINTERS
003 032 023          INXD
003 033 175          MOVAL     /GET LOW ADDRESS FOR Y DATA
003 034 376          CPI       /ARE ALL POINTS OUTPUT?
003 035 144          144
003 036 302          JNZ       /NO, DO MORE
003 037 011          MORE
003 040 003          0
003 041 303          JMP       /YES, DISPLAY THE FILE AGAIN
003 042 003          START     /SUBSTITUTE A HALT FOR THE JUMP
003 043 003          0         /TO OUTPUT THE DATA ONLY ONCE
                     *003 000
003 100 000  WAIT1,  0         /APPROPRIATE TIME DELAY GOES HERE
                     *003 130
003 130 000  WAIT2,  0         /AND HERE, TOO
                     *007 000
007 000 000  YDATA,  0         /THE Y-AXIS DATA STARTS HERE
                     *007 200
007 200 000  XDATA,  0         /THE X-AXIS DATA STARTS HERE


                     /EXAMPLE 1-17
                     /SOFTWARE FOR AN X–Y DATA DISPLAY USING A SINGLE
                     /FILE CONTAINING X AND Y DATA
                     *003 000
003 000 061          LXISP
003 001 377          377
003 002 003          003
003 003 041  START,  LXIH      /LOAD THE MEMORY DATA POINTER
003 004 000          XYDATA
003 005 007          0
003 006 176  LOOP,   MOVAM     /GET THE Y DATA
003 007 323          OUT       /OUTPUT IT
003 010 027          027
003 011 043          INXH      /INCREMENT THE ADDRESS POINTER
003 012 176          MOVAM     /GET THE X DATA FROM THE NEXT LOC'N
003 013 323          OUT       /OUTPUT IT
003 014 036          036
003 015 315          CALL      /THIS IS THE SAME SOFTWARE AS THAT IN
003 016 100          WAIT1     /EXAMPLE 1-16.
003 017 003          0
```

```
003 020 323           OUT
003 021 030           030
003 022 315           CALL
003 023 130           WAIT2
003 024 003           0
003 025 323           OUT
003 026 031           031
003 027 043           INXH      /INCREMENT ADDRESS POINTER AGAIN
003 030 175           MOVAL     /CHECK REG L
003 031 376           CPI       /COMPARE IT TO DECIMAL 200
003 032 310           310       /310 = 200 DECIMAL
003 033 302           JNZ
003 034 006           LOOP
003 035 003           0
003 036 303           JMP
003 037 003           START
003 040 003           0
                      *007 000
007 000 000 XYDATA,   0         /DATA STARTS HERE WITH Y DATA
007 001 000           0         /X-AXIS DATA
007 002 000           0         /Y-AXIS DATA
007 003 000           0         /ETC. . . . . . . .
```

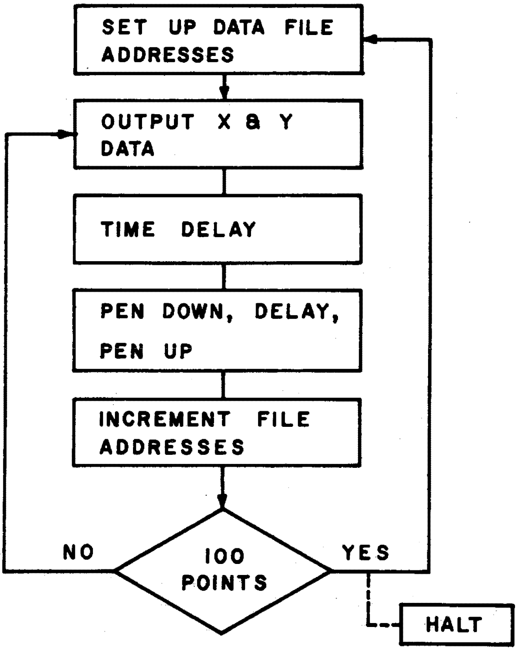


Fig. 1-10. The flowchart for a general X-Y display software routine that may be used to drive an X-Y plotter with pen-up and pen-down control.

The computer could have made use of a single data file in which the X-axis and the Y-axis information is placed in alternating memory locations, eliminating the need for the register pair D pointer address. An example of this type of file manipulation is shown in Example 1-17. Remember that the single data file will be twice as long as either the X-axis or the Y-axis files that were used in Example 1-16, but that the same number of memory locations are required to store the entire file. One disadvantage of this type of "data filing" is that it will require additional software to access the points in the file for further manipulation.

We have only provided examples of point plotting. Line plotting is much more complex since additional points must be plotted to "create" the straight line. Mathematical routines are generally used to add these points to a plot, filling in the "space" between two real data points. Line plots are beyond our present discussion.

INTERFACING A 10-BIT D/A CONVERTER

Many applications that require analog outputs from a computer system will need more resolution than the one part in 256 that is provided by an 8-bit D/A converter, similar to the one used in our examples. There are 10-, 12- and 14-bit D/A converters readily available and these, too, may be interfaced to microcomputers.

How, you may ask, is it possible to interface a 10, 12-, or 14-bit device to an 8-bit computer? Actually, it is easier than it may appear at first glance. A 10-bit D/A converter with a resolution of one part in 1024 will be used in this example. This is more than enough resolution for most applications.

To interface a 10-bit D/A converter to an 8-bit computer, the interface is built so that the computer first transfers eight bits to the converter and then, in another operation and with separate circuitry, the computer transfers the final two bits to the converter. This first attempt at interfacing is shown in Fig. 1-11.

The 10 bits of data that are to be displayed will be stored in two consecutive memory locations as shown:

ADDRESS				*DATA BITS*					
A	D7	D6	D5	D4	D3	D2	D1	D0	10-Bit Word
A+1	X	X	X	X	X	X	D9	D8	
A+2	D7	D6	D5	D4	D3	D2	D1	D0	10-Bit Word
A+3	X	X	X	X	X	X	D9	D8	

X = Unused bit positions

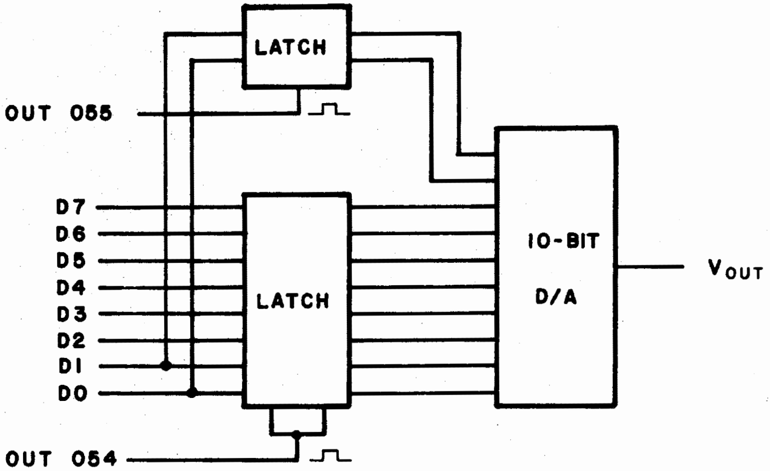


Fig. 1-11. Block diagram of a 10-bit D/A converter interfaced to an 8-bit computer.

The eight least significant bits (LSBs) are stored in memory loca tion A. The two most significant bits of the 10-bit data word ar stored in memory location A+1 in bit positions D1 and D0. The othe six bits in this most significant bit (MSB) storage location are no used.

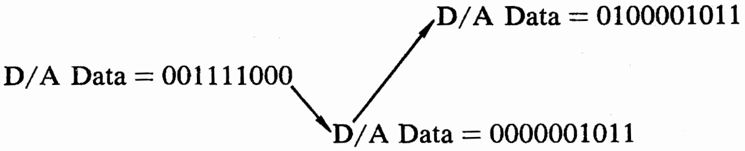
We have assumed that only one 10-bit D/A converter has bee interfaced to the computer, although more could have been added Part of a typical 10-bit D/A converter output program is shown a follows. It has been written to control the 10-bit D/A converter inter face shown in Fig. 1-11.

```
                  *000 000
000 000 041       LXIH      /LOAD MEMORY POINTER
000 001 000       000
000 002 007       007
000 003 176       MOVAM     /GET 8 LSB'S
000 004 323       OUT       /OUTPUT THEM TO THE DAC
000 005 054       054
000 006 043       INXH      /INCREMENT THE ADDRESS
000 007 176       MOVAM     /GET THE 2 MSB'S
000 010 323       OUT       /OUTPUT THEM TO THE DAC
000 011 055       055
```

Suppose that the converter is now outputting the voltage corresponding to the 10-bit binary value 0011110000_2. The next 10-bit binary value that is to be output to the converter is 0100001011_2. What will the output of the D/A converter look like when the computer outputs this new value, using the program section listed above?

We would expect that the computer would cause the 10-bit D/A converter to output a new voltage that is somewhat higher than the current voltage now being output. The second binary value is larger than the first, thus the output voltage should be higher. While this is the end result of outputting the new data word, 0100001011_2, there is a "glitch" present in the output.

The actual output is shown as follows, plotted with respect to time. The time scale is greatly exaggerated:



Why do we observe this unexpected glitch and the data word, 0000001011_2, associated with it? The glitch is observed because all of the bits in the new 10-bit word are not applied to the D/A converter's inputs at the same time. The following portion of a program that may be used to update the D/A converter illustrates what happens:

```
LXIH           /OLD DATA                        = 0011110000
000
007
MOVAM          /GET NEW DATA
OUT            /OUTPUT 8 LSB'S OF NEW DATA      = 0000001011
054
INXH
MOVAM
OUT            /OUTPUT 2 MSB'S OF NEW DATA      = 0100001011
055
```

You will notice that the intermediate word, 0000001011_2, is "made up of" the eight least significant bits of the new 10-bit data word and the two most significant bits of the old data word. It is this "intermediate" 10-bit data word which causes the glitch.

While we wish to output the data word, 0100001011_2, to the D/A converter, the present interface can only transfer eight bits of data at a time. Thus, the MSBs of the old data word will always be present while the eight LSBs are being changed to those of the new data word. With the present hardware and software there is no way to avoid this problem.

An additional 10-bit latch may be added to the interface to prevent the D/A converter from generating the glitch due to the "overlap" between the old and the new data words. The technique of using two

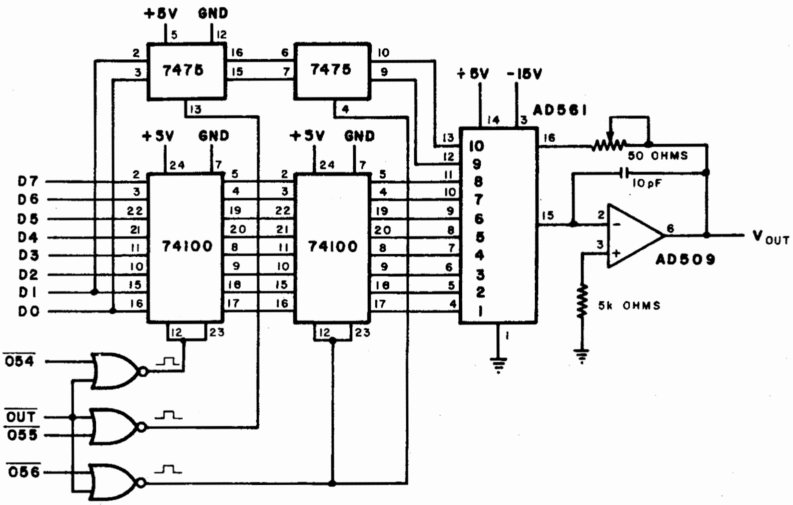


Fig. 1-12. A double-buffered 10-bit digital-to-analog converter using the Analog Devices AD561.

latches in series is called *double buffering*. The complete interface for a double-buffered D/A converter is shown in Fig. 1-12. The additional latch may be clocked or enabled with a separate OUT command, but it may also be set up so that the eight LSBs are transferred to the D/A converter's inputs when the computer outputs the two MSBs. In this example we have chosen to use a separate command to control the 10-bit latch.

The following sequence is used when data is being output to the 10-bit D/A converter:

1. The eight LSBs of the new data word are transferred to the 74100 latch that is connected to the data bus. The OUT 054 command performs this function.
2. The two MSBs of the new data word are transferred to the 7475 latch that is connected to the data bus. The OUT 055 command performs this function.
3. The complete, new 10-bit data word latched in the 74100 and 7475 latch is transferred to the 10-bit latch with an OUT 056 command. The 10-bit latch is constructed with another 7475 and another 74100 device.

This type of a double-buffered interface uses the same type of software shown in the previous example, except that an additional output command has been added to provide the pulse necessary to enable the 10-bit latch.

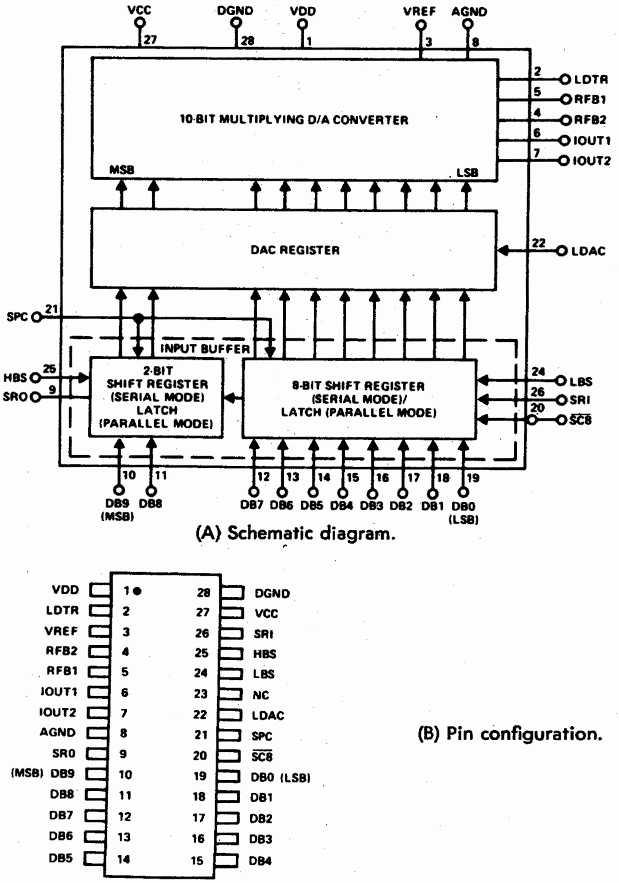
```
/EXAMPLE 1-18
/TYPICAL 10-BIT DAC OUTPUT SOFTWARE
```

```
                   *003 000
003 000 041 START, LXIH     /SET UP DATA ADDRESS
003 001 000        DATA
003 002 007        0
003 003 176 DAC,   MOVAM    /GET THE 8 LSB'S
003 004 323        OUT      /OUTPUT THEM
003 005 054        054
003 006 043        INXH     /INCREMENT THE ADDRESS
003 007 176        MOVAM    /GET THE 2 MSB'S
003 010 323        OUT      /OUTPUT THEM
003 011 055        055
003 012 323        OUT      /TRANSFER THE 10-BIT WORD TO THE DAC
003 013 056        056      /ALL AT ONCE
003 014 043        INXH     /INCREMENT ADDRESS AGAIN
003 015 175        MOVAL    /GET THE L VALUE
003 016 376        CPI      /IS IT = 310?
003 017 310        310      /310 = 200 DECIMAL
003 020 302        JNZ      /HAVE ALL 100, 10-BIT WORDS BEEN OUTPUT
003 021 003        DAC      /NO, DO MORE
003 022 003        0
003 023 303        JMP      /YES, REINITIALIZE AND DO IT AGAIN
003 024 000        START
003 025 003        0
                   *007 000
007 000 000 DATA,  0        /8 LSB'S STORED HERE (WORD 1)
007 001 000        0        /2 MSB'S STORED HERE (WORD 1)
007 002 000        0        /8 LSB'S STORED HERE (WORD 2), ETC. . .
```

In one of the newer D/A converter devices, the Analog Devices AD7522, the double-buffering is provided within the integrated circuit. The AD7522 is shown in Fig. 1-13. Three device-select pulses are used with this device, an 8-bit transfer or low-byte strobe pulse (LBS, pin 24), a 2-bit or high-byte strobe pulse (HBS, pin 25) and a load D/A converter pulse (LDAC, pin 22).

Since the AD7522 has built-in double buffering, we have chosen to use it in many of the experiments presented in Unit 7. To make your interfacing tasks easier, it has been combined with the necessary gating logic and amplifier circuit in the form of an Outboard which can be plugged into a solderless breadboard socket. The complete schematic for the LR-35 10-bit D/A converter Outboard is shown in Fig. 1-14. A photograph of the LR-35 is shown in Fig. 1-15.

We expect that manufacturers will continue to provide new and useful features within their D/A modules to make interfacing easier in the future. Another recent development is the Signetics NE5018 integrated circuit. This is a single, 22-pin device which incorporates the latch circuits, analog reference, and output amplifier along with an 8-bit D/A converter. Interfacing requires two or three external components and a device select pulse. This device will be discussed in more detail in the experiments in Unit 7.



(A) Schematic diagram.

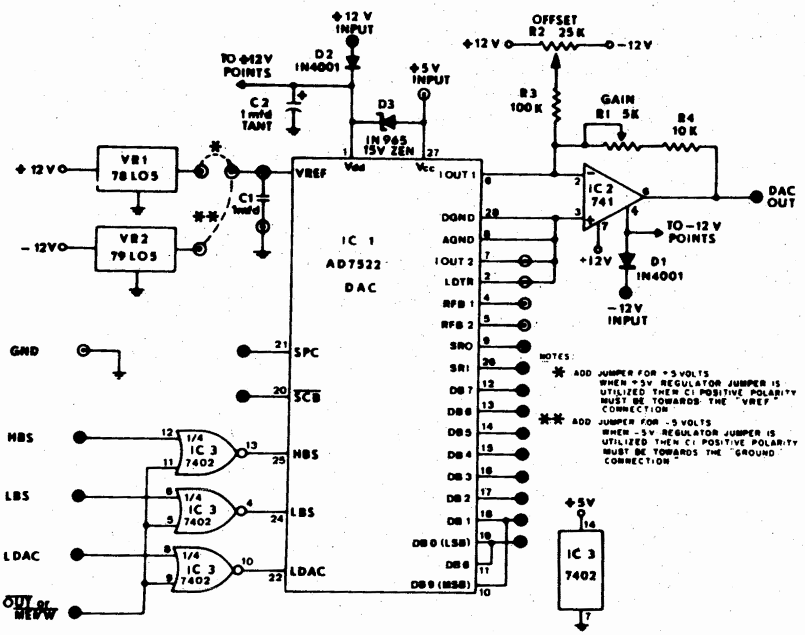
(B) Pin configuration.

Courtesy Analog Devices, Inc.

Fig. 1-13. The Analog Devices AD7522 10-bit D/A converter integrated-circuit schematic and pin configuration.

MEMORY MAPPED I/O TECHNIQUES AND D/A CONVERTERS

The memory mapped input/output (I/O) technique may be readily applied to D/A converters such as those shown in the examples in this unit. To convert the interface circuits so that the memory-mapped I/O technique is used rather than the accumulator I/O technique, the $\overline{\text{OUT}}$ signals must be changed to the memory write signal, $\overline{\text{MW}}$ or $\overline{\text{MEMW}}$, and the device addresses must be changed to 16-bit memory addresses rather than 8-bit I/O addresses. The actual techniques of memory-mapped I/O are covered in detail in Units 21 and 22 of *In-*



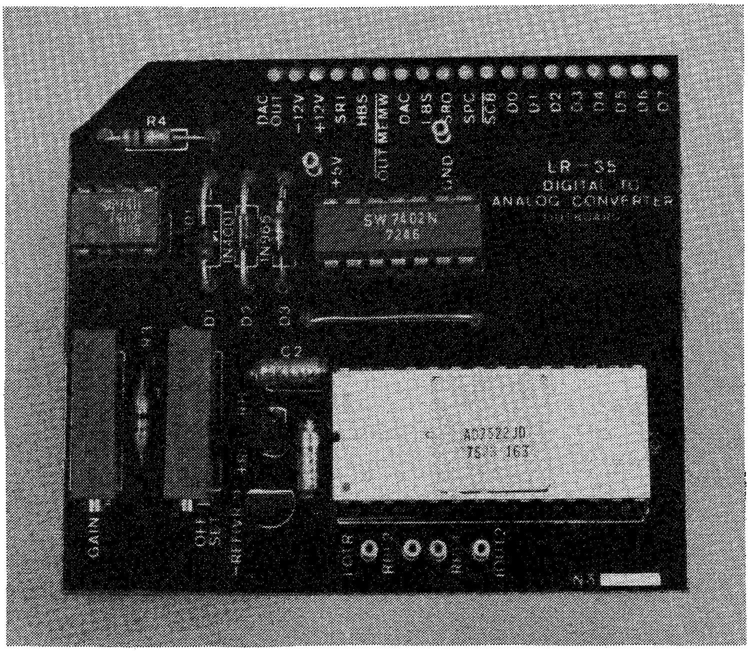
Courtesy E&L Instruments, Inc.

Fig. 1-14. Schematic diagram of the LR-35 10-bit D/A converter Outboard®.

troductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 2, published by Howard W. Sams & Co., Inc. The software examples must also be changed so that memory reference instructions are used to output data to the D/A converters. The main advantage of memory-mapped I/O is that there are a large number of 8080 instructions which may be used to transfer data between I/O devices and the 8080 chip.

The new programmable peripheral interface (PPI) integrated circuits, such as the Intel 8255, provide an easy means of implementing memory-mapped I/O for D/A converters. These devices incorporate 8-bit latches and they may be programmed to be either input or output ports.

Using the PPI-type integrated circuits with D/A converters that have more than eight bits of digital input may pose problems, since the PPI chips are not double buffered. The 10-bit D/A converter which required two sets of latches, Fig. 1-12, could not be readily interfaced using a PPI-type chip. If a PPI-type chip was used, the output of the D/A converter would show glitches since the 10-bit data word would not be transferred to the 10 D/A converter inputs simultaneously.



Courtesy E&L Instruments, Inc.

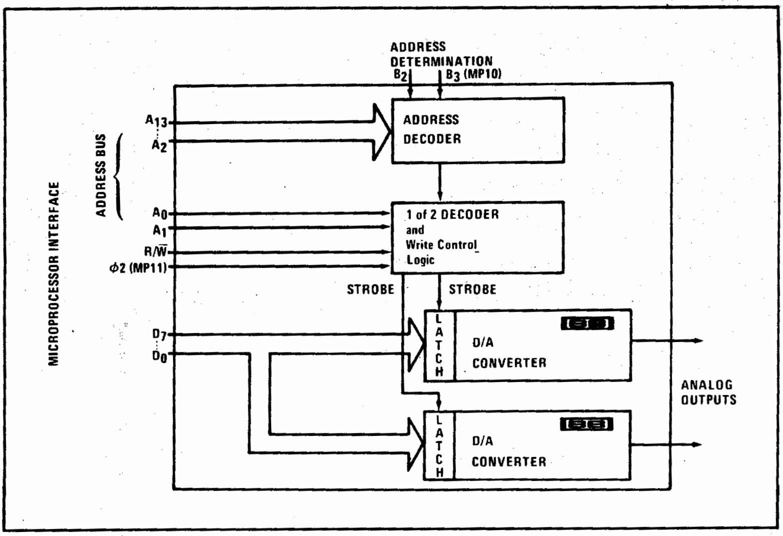
Fig. 1-15. Photograph of the LR-35 Outboard®.

It is our opinion that you will be better off by using standard TTL integrated circuits for your interfacing where the task is well defined or where more than eight bits will be transferred at one time. The D/A converters with built-in double-buffered latches are your best choice if you wish to avoid interfacing problems with D/A converters having more than eight bits of input. You will find that the cost of the PPI-type interface chips will soon be equal to the price of converters with built-in latches and other interface elements.

Both the memory-mapped and accumulator I/O techniques will be used in the experiments presented in Unit 7. You will find that both of the techniques are useful and that the choice of interfacing techniques depends upon your experience with hardware and software and also upon the problem to be solved.

OTHER DIGITAL-TO-ANALOG CONVERTERS

Another digital-to-analog converter which deserves attention is the Burr-Brown MP-10 module. This converter has been designed and built specifically for use with 8080-type microcomputers such as the 8085, 8048, Zilog Z80, 8080A, and others. The MP-10 module is of particular interest since it incoporates two 8-bit D/A converters within the module by using a programmable peripheral interface (PPI) integrated circuit similar to the Intel 8255 for the latch portion of the interface. The 8255 PPI chip has three ports which may be configured as either input or output ports depending upon the commands sent to it under software control. Thus, the chip is truly programmable.



Courtesy Burr-Brown Research Corp.

Fig. 1-16. Block diagram of the Burr-Brown MP-10 dual, 8-bit D/A converter module.

By incorporating this type of an integrated circuit into their MP-10 D/A converter package, Burr-Brown has been able to offer two 8-bit D/A converters in a one-inch by one-and-three-quarter-inch package having 32 contact pins. A block diagram of the MP-10 module is shown in Fig. 1-16.

The digital inputs to the MP-10 are all compatible with the TTL signal levels present on the microcomputer's address, data, and control signal buses. The necessary device-address-decoding logic is provided within the module. The MP-10 may be configured to respond to any one of four groups of addresses when the memory-mapped I/O technique is used. Each of the four groups contains three device addresses which correspond to the three functions present within each module.

The three functions are: (1) D/A converter #1, (2) D/A converter #2, and (3) the internal PPI control register. The addresses available are as follows:

High Address	Low Address	Function
X X 1 1 1 1 1 1	1 1 1 1 B B 0 0	D/A #1
X X 1 1 1 1 1 1	1 1 1 1 B B 0 1	D/A #2
X X 1 1 1 1 1 1	1 1 1 1 B B 1 1	Control Register

The address bits noted as "B" are defined by the user with hardwired connections so that up to four groups of addresses may be used with one computer system. The "B" address bits are selected as either 00, 01, 10, or 11. The "X" address bits are not used within the MP-10 module.

Whenever a programmable peripheral interface integrated circuit is used in an interface it is necessary to initialize it. The Burr-Brown MP-10 module is no exception and it is initialized by having the 8080 computer transfer the control word 10000000_2 to the MP-10's control register. *This must be done prior to the use of either D/A converter.*

When the memory-mapped I/O technique is used, either of the two following software examples may be used for the MP-10 initialization:

```
MVIA                    LXIH
200                     CONTROL
STA                     0
CONTROL                 MVIM
0                       200
```

In the left-hand example, the 8080's A register is loaded with the control word which is then stored in the memory address specified by CONTROL, the address of the D/A converter's control register. In the right-hand example, register pair H is loaded with the address of the MP-10's control register and then the control word is transferred to the control register with the MVIM instruction. In these two software examples the notation "CONTROL" followed by "0" is used to represent the two 8-bit bytes of the 16-bit address assigned to the control register within the MP-10 module, say, $00111111\ 11110011_2$.

Since the two D/A converters within the MP-10 module have been assigned successive addresses, two 8-bit bytes of data may be easily transferred, one byte to each converter, using the SHLD instruction. This is a three-byte instruction that is used to transfer the contents of the H and the L register to two successive memory locations, X and X+1. The address, X, is specified in the last two bytes of the SHLD instruction. In this way, two D/A converters may be updated with new data by using a single instruction.

The MP-10 module has no provision for decoding the two most significant address bits, A15 and A14. Thus, if the MP-10 module is to be interfaced using the memory-mapped I/O technique, it will be addressed in the last 256-byte block in each of the four 16K blocks of memory available within the 8080's memory space. This is seen with the aid of the following address chart:

Address Decoding for the MP-10 D/A Converter Module

High Address	Low Address
0 0 1 1 1 1 1 1	1 1 1 1 0 0 0 0
0 1 1 1 1 1 1 1	1 1 1 1 0 0 0 0
1 0 1 1 1 1 1 1	1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1	1 1 1 1 0 0 0 0

The four addresses in the address chart will *all* address a single D/A converter. We have assumed that the "B" address select inputs are both hardwired to logic zero. If you are using 16K or more memory in a single block this will cause problems since the same address may be assigned to a real memory location and to a D/A converter.

To correct this problem we would suggest using the gating scheme shown in Fig. 1-17. This will have the overall effect of "moving" the addresses of the MP-10 D/A converters and control register to the last 256-byte block of memory addressable by the 8080, that is to say, within addresses 377 000 and 377 377 (FFOO and FFFF, hexadecimal). This modification requires a single SN7400 quad-NAND gate package.

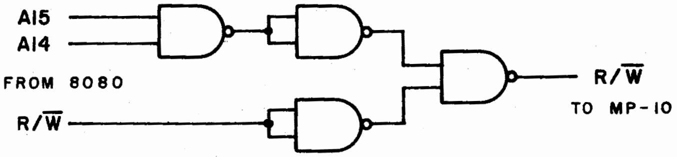


Fig. 1-17. Using address gating to move the MP-10 converter memory to higher memory addresses.

This circuit will enable the MP-10's read/write input only when address bits A15 and A14 are both logic one. Since all of the other address bits, with the exception of A3 through A0, must be logic 1s too, the addresses of the devices will be changed to:

1 1 1 1 1 1 1 1   1 1 1 1 B B C C

Here, the "B" address bits must still be defined and hardwired by the user and the "C" address bits are used to define the internal device that will be addressed.

The MP-10 module may also be used with the accumulator I/O technique by substituting the 8080's $\overline{\text{OUT}}$ signal for the memory-write signal applied to the MP-10. In this case, address inputs A13 through A8 must be hardwired to logic one. The addresses of the D/A converters and the control register are the same as those used in the memory-mapped examples, except that the high address is ignored. The low address is incorporated in the program as the second byte in the output instructions. *Remember that if accumulator I/O is used, the control register within the MP-10 module must still be initialized with the value* 10000000_2.

When the accumulator I/O technique is used with the MP-10 module, the following steps may be used to initialize the control register within the module:

```
MVIA      /LOAD THE A REGISTER WITH DATA
200       /DATA = 200
OUT       /OUTPUT IT TO THE CONTROL REGISTER
CONTROL   /EIGHT-BIT ADDRESS OF REGISTER
```

Burr-Brown has made the microcomputer-to-converter interfacing task somewhat easier by incorporating the latches and device-address control logic within their MP-10 module. We expect that this trend will continue.

It is important to remember, though, that while the interfacing may become easier, someone must still write the software necessary to make the interfaces operate correctly.

2

Interfacing Analog-to-Digital Converters

INTRODUCTION TO THIS UNIT

Analog-to-digital converters (ADCs or A/D converters) play an important role in many microcomputer systems. They allow a digital system to convert a voltage signal into a digital quantity that represents the unknown voltage. This is particularly useful when temperature, pressure, weight, position, distance, or some other unknown is to be measured and the measuring sensor has an output which is either an analog voltage or current. Hundreds of A/D converter modules are available in many configurations to solve general or specific problems. This unit includes descriptions and examples of several types of converters and how they are interfaced to an 8-bit microcomputer. Software is also provided in the examples.

OBJECTIVES

- Describe the operation of a ramp A/D converter.
- Describe the operation of a successive approximation A/D converter.
- Develop software to input data from both ramp and successive approximation A/D converters.
- Describe the interface required for an 8-bit and a 10-bit A/D converter module.
- Describe the differences between delay loop, interrupt, and polled, real-time clock timers for data acquisition programs.

- Discuss the use of A/D and D/A converters for data-acquisition and data-display applications.

ANALOG-TO-DIGITAL CONVERTERS

The most widely used types of analog-to-digital converters operate by matching or comparing an unkonwn voltage to a voltage produced by a digital-to-analog converter. The generation of the known voltage and the comparison process is straightforward. This unit will introduce software-based analog-to-digital conversion techniques and methods for interfacing modular A/D converters to microcomputers.

Analog-to-digital converters are available with a variety of input and output configurations and with various features, such as three-state outputs and amplifiers already built into them. Most A/D converters are voltage measuring devices with input ranges of 0–5 volts, 0–10 volts, ±5 volts or ±10 volts. Some converters are available with all of these ranges or with a variable gain feature so that almost any voltage range can be measured. The outputs of A/D converters are usually compatible with transistor-transistor logic (TTL) levels, being coded in straight-binary or binary-coded decimal (BCD) form. Converters with two's-complement binary and offset binary are also available if these codes are needed.

SOFTWARE A/D CONVERSION TECHNIQUES

When working with microcomputers, or any small computer for that matter, it becomes apparent that there are many hardware/software tradeoffs which can be made. For example, the following operations may be performed with standard integrated circuits or with a series of software instructions:

- Binary or decade counters
- Inverters
- Shift registers
- Code converters
- Flip-flops
- Multiplexers

In fact, there are many other tasks that may be done by software or by using hardware. The analog-to-digital conversion process is one of the tasks which may be done equally well in either hardware or software. We will explore both the ramp-type A/D converter and the successive-approximation-type converter to see how the conversion process works and how it may be accomplished with software.

LINEAR RAMP CONVERTERS

The ramp-type of A/D converter operates by matching an unknown voltage with the voltage represented by a steadily increasing

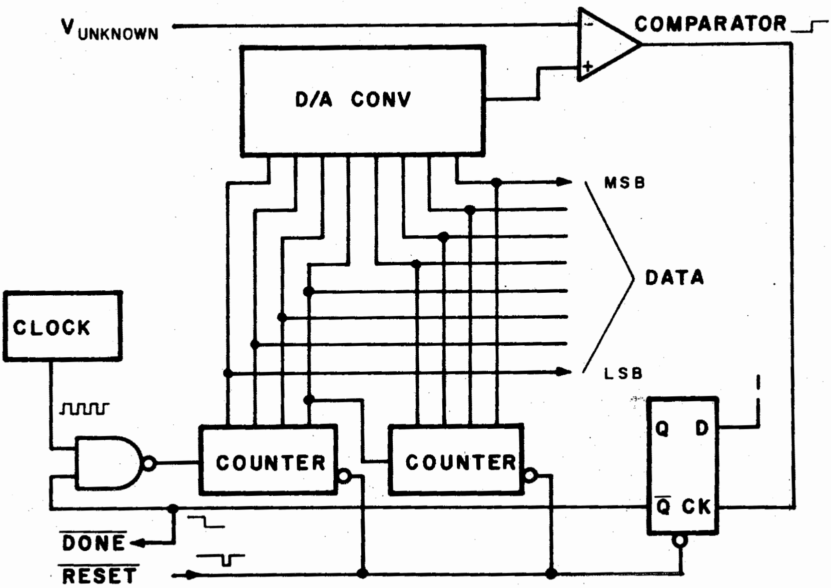


Fig. 2-1. Hardware representation of a ramp-type A/D converter.

voltage ramp. The unknown voltage is constantly compared to the ramp voltage until they are found to be equal. When they are equal, the process is stopped. If counters are used with a D/A converter to generate the ramp, the binary code present at the inputs to the D/A converter will represent the binary code for the unknown voltage. A block diagram representation of a ramp A/D converter is shown in Fig. 2-1.

A short logic zero pulse is applied to the $\overline{\text{RESET}}$ input to clear both of the binary counters and the D-type flip-flop. When the flip-flop is cleared, its Q output is a logic one, enabling the NAND gate to pass clock pulses through to the binary counters. The parallel count applied to the D/A converter generates a ramp output. The ramp output and the unknown voltage are compared by the comparator, which will output a logic one when the two voltage inputs are equal. The logic one output by the comparator will clock or pulse the flip-flop so that the Q output will become a logic zero, disabling the NAND gate and thus preventing any further clock pulses from reaching the counters.

When the count is maintained, as indicated by the end-of-conversion or $\overline{\text{DONE}}$ flag being in the logic zero state, the eight parallel outputs represent the binary code of the unknown voltage. The D input to the flip-flop is held at a logic one. A timing diagram for this type of ramp conversion process is provided in Fig. 2-2.

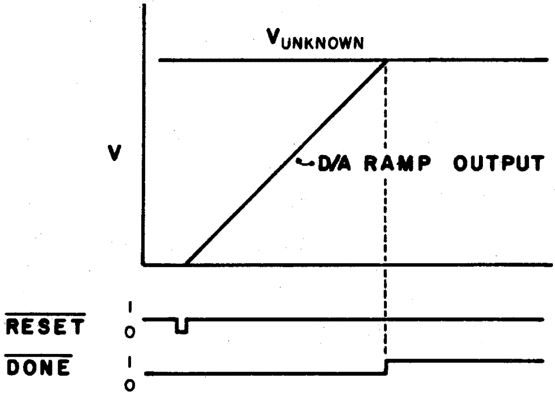


Fig. 2-2. Timing diagram for a ramp A/D converter such as that shown in Fig. 2-1.

We have assumed that the unknown voltage is within the range of the D/A converter's ramp output. In converters of this type, synchronous counters are preferred to ripple or asynchronous counters. The synchronous counter's outputs will all change to their respective states simultaneously. This avoids counting spikes, which are produced by ripple counters such as the 4-bit binary counter, the SN7493. If a BCD output is required, a BCD D/A converter and BCD counters may be used. The SN74161 binary counter and the SN74160 BCD counter may be used in converters of this type.

Since we have discussed interfacing an 8-bit D/A converter to a microcomputer in a previous unit, we will now attempt to use it to generate the ramp under software control. Software might also be used to test the state of the comparator's output. In this case an input port could be used to input the logical state of the comparator's output. A typical interface is shown in Fig. 2-3 in block diagram form.

It is now possible to substitute software for hardware. The computer will be used to generate a ramp output by the D/A converter

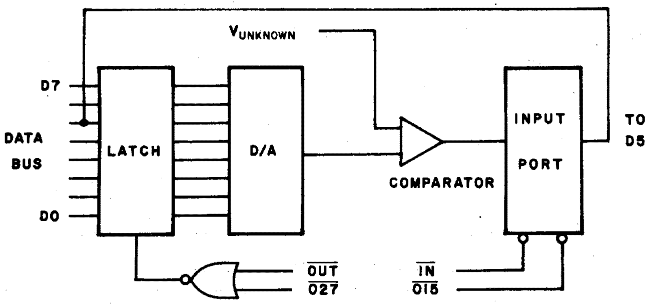


Fig. 2-3. A typical software-based ramp A/D converter using a D/A converter, an input port, and a comparator.

using the software steps that were described in the previous unit. The comparator's output will be treated as an external flag which will be sensed or tested with logical and decision-making instructions.

```
                          /EXAMPLE 2-1
                          /TYPICAL SOFTWARE FOR A RAMP A/D CONVERTER
                          /USING A D/A & COMPARATOR
                          *003 000
003 000 006   START,      MVIB        /SET B = 0
003 001 000               000
003 002 170   DALOOP,     MOVAB       /MOVE B TO A
003 003 323               OUT         /OUTPUT IT TO THE D/A
003 004 027               027
003 005 333               IN          /INPUT THE COMPARATOR STATUS
003 006 015               015
003 007 346               ANI         /MASK OUT UNWANTED BITS
003 010 040               040
003 011 302               JNZ         /STILL = 0?
003 012 020               DONE        /NO, WE HAVE A MATCH
003 013 003               0
003 014 004               INRB        /YES, ADD 1 AND DO IT AGAIN
003 015 303               JMP
003 016 002               DALOOP
003 017 003               0
003 020 170   DONE,       MOVAB       /FINISHED, GET THE VALUE
003 021 323               OUT         /OUTPUT IT TO A DISPLAY
003 022 002               002
003 023 166               HLT         /THEN HALT
```

The software shown in Example 2-1 will work with the hardware shown in Fig. 2-3, but the software is slow, taking up to 7 milliseconds to perform a conversion since up to 255 passes through the DALOOP section of the program may be required, depending upon the unknown voltage. Higher voltages will take longer to convert than lower voltages since the ramp must be incremented further to reach the higher voltages necessary for an equal condition to exist.

Conversion times would be even longer for higher resolution 10-, 12-, or 14-bit converters implemented in this way. It should also be noted that the computer is dedicated to this task for the many milliseconds that are required by the software to perform the conversion. Even for a slowly varying signal, the conversion may not be finished quickly enough to "catch" the portion of the signal that we wish to measure.

For example, we can use a peak detector to signal that the maximum of an instrument's output has been reached so that it may be digitized. The peak detector could be very sophisticated, or it could be a simple timer indicating to the A/D converter or to the computer that it must digitize the instrument's output. This relationship is shown in Fig. 2-4.

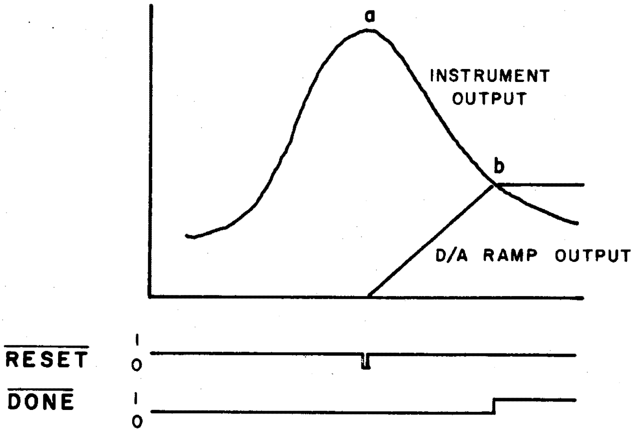


Fig. 2-4. The timing relationship between the output of an instrument and the attempt by a ramp A/D converter to digitize the peak value.

The $\overline{\text{RESET}}$ signal is generated by the instrument to indicate to either the hardware ramp A/D converter (Fig. 2-1) or to the computer that the maximum has occurred. The voltage that we wish to digitize is that present at point *a*. The converter or the computer senses this and starts to generate the ramp voltages. Since the ramp output changes slowly we find that during this time the instrument's output has also changed.

The coincidence of the ramp output and the instrument's output actually takes place at *b*, some time after the peak value has passed by. If we blindly treat the digitized value as the peak value, the measurement will be in error by a considerable amount. We have not really measured the peak at all, but something that occurred after the peak appeared.

The software-generated ramp A/D conversion technique is slow and it is not a method which is usually used to perform A/D conversions in microcomputers. The ramp A/D conversion technique is still valid, however, and there are prepackaged converter modules which have relatively fast conversion times and they are available at low cost. This makes them ideal for general experimental and laboratory use.

Three examples are:

Analog Devices, Inc., Norwood, MA 02062	
ADC 8S	8-bit, 1-millisecond conversion time
Datel Systems, Inc., Canton, MA 02021	
ADC Econoverter	6-bit, 50-microsecond conversion time
ADC 98A	8-bit, 20-microsecond conversion time

The hardware and software interfacing techniques required to control A/D converter modules such as these will be discussed in the *Interfacing Analog-to-Digital Converters* section of this unit.

Successive-Approximation Converters

In many cases, we wish to be able to perform analog-to-digital conversions accurately and in such a way that it will not take hundreds of microseconds or even milliseconds to do a simple 8- or 10-bit conversion. The successive-approximation technique is one that provides both accuracy and speed.

Let us examine how this technique works. We will assume that we wish to guess the value of a number in the range of zero to 255, the range possible with eight binary bits. If the ramp technique is used it may take many steps to reach the unknown number since each step is only one step above the previous one. In contrast, the successive-approximation method performs tests that are based upon the weights of each binary bit position, going from the most significant bit (MSB) to the least significant bit (LSB). This method "homes in" on the unknown number in a very few tests. We will assume that a comparator is still available to indicate the greater-than or less-than conditions.

The individual tests are performed with the binary weights such that the binary weight is added to, or excluded from, the total value depending upon the comparison of the accumulating total with the unknown number. Table 2-1 is an example of an 8-bit conversion of the number 113 into a binary equivalent.

Table 2-1. Converting Decimal Number 113 Into a Binary Equivalent

Test Value	Response	Sum
128	Too high, do not add it to the sum	0
64	Too low, add it to the sum and go on	64
32	Sum of 64+32 is still too low, add it, go on	96
16	Sum of 16+32+64 is still too low, add it	112
8	Sum too high, do not add it	112
4	Sum too high, do not add it	112
2	Sum too high, do not add it	112
1	Sum of 64+32+16+1 is just right	113

Since the weights of each binary bit have been used in the approximation, it becomes easy to convert the information in the previous table into a true binary representation of the decimal number 113. This is done by placing a one in each bit position that contributed to the total and a zero in all those positions for which the weight was not added to the sum:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	1	1	1	0	0	0	1

The successive-approximation technique took only eight tests to match the unknown number, while the ramp method would have taken 113 tests. The successive-approximation technique only requires one test per bit. Thus, a 10-bit A/D converter, using the successive-approximation technique, would require only 10 steps to match any integer value between zero and 1024.

In a successive-approximation converter, a digital-to-analog converter is used to provide the test voltages in discreet weighted steps. The D/A converter may be controlled with either hardware or software. We will attempt a software solution and we will not discuss the logic circuits used to implement a hardware design.

The computer interface used in this example will be the same as that used with the software-based ramp converter (Fig. 2-3). No hardware modifications are required since the basic steps of outputting the digital value and inputting the state of the comparator's output will be the same.

The software method, or *algorithm,* is easy to visualize. The computer will start by having all of the D/A converter's input bits set to logic zero. The most significant bit (MSB) will be set to a logic 1 and then the comparator's output will be input and tested. If the D/A converter has output a voltage that exceeds the unknown voltage, the comparator will indicate this condition with a logic 1 output. This will be sensed by the computer and the bit position being tested will be reset to zero. If the D/A converter's output does not exceed the unknown voltage the bit will remain at a logic 1. The next least significant bit will be tested in a similar fashion until all of the bits have been tested.

The flowchart in Fig. 2-5 shows how the program could operate. While this may look like a long, complicated program, it may be simplified by using logical instructions, rotate instructions, and software loops. The actual program will use several of the 8080's registers:

- Register A I/O data transfers and bit manipulations
- Register B Test data storage
- Register C Accumulating result storage
- Register D Loop counter

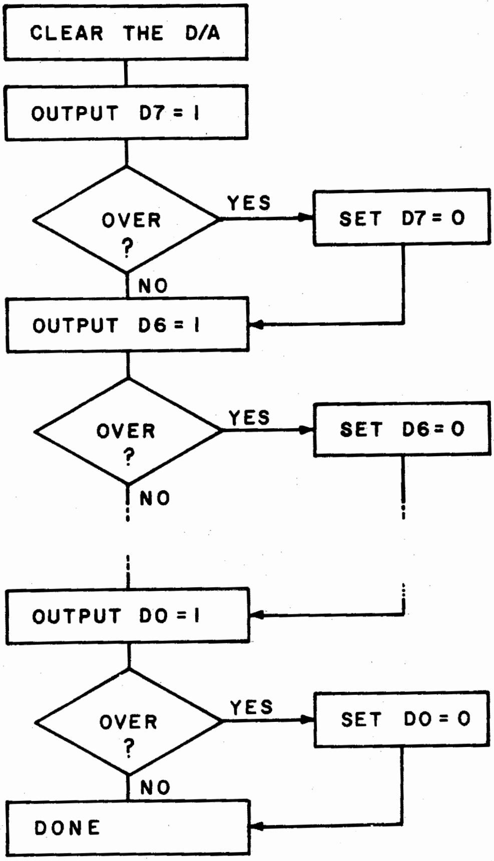


Fig. 2-5. Flowchart for a software-based successive-approximation A/D converter with eight bits.

You may wish to follow the program step-by-step using paper and pencil to note the contents of each register as you go along, trying to match an unknown number. The DBUG program now contained in the Blacksburg Continuing Education Series, *"DBUG; An 8080 Interpretive Debugger"* may also be used with a teletypewriter or terminal to follow the flow of the program. The completely assembled program is shown in Example 2-2.

```
                         /EXAMPLE 2-2
                         /PROGRAM FOR 8-BIT SUCCESSIVE APPROXIMATION
                         /A-TO-D CONVERTER
                         *003 000
003 000 227   START,     SUBA        /CLEAR A
003 001 001              LXIB        /LOAD REG PAIR B
```

```
003 002 000           000       /C = 0
003 003 200           200       /B = 10000000
003 004 026           MVID      /SET UP COUNTER = 8
003 005 010           010
003 006 260  AGAIN,   ORAB      /OR A WITH B
003 007 117           MOVCA     /STORE DATA IN C
003 010 323           OUT       /OUTPUT IT TO DAC
003 011 027           027
003 012 333           IN        /INPUT COMPARATOR STATUS
003 013 001           001
003 014 346           ANI       /MASK OUT UNWANTED BITS
003 015 040           040
003 016 312           JZ        /IS IT GREATER THAN UNKNOWN VOLTAGE?
003 017 025           OK        /NO, GO TO OK
003 020 003           0
003 021 170           MOVAB     /YES, GET TEST PATTERN
003 022 057           CMA       /COMPLEMENT IT
003 023 241           ANAC      /SET BIT TESTED TO A 0
003 024 117           MOVCA     /STORE DATA IN C
003 025 170  OK,      MOVAB     /GET TEST PATTERN
003 026 037           RAR       /ROTATE TEST PATTERN RIGHT
003 027 107           MOVBA     /PUT IT BACK
003 030 171           MOVAC     /GET THE DATA TO REG A
003 031 025           DCRD      /DECREMENT LOOP COUNT, D = 0?
003 032 302           JNZ       /NO, TEST THE NEXT BIT
003 033 006           AGAIN
003 034 003           0
003 035 166  DONE,    HLT       /HALT WITH FINAL VALUE IN REG A
```

This program performs an 8-bit successive-approximation conversion. Programs for more than eight bits become more complex since multiple bytes are required for the test-bit patterns and the accumulating result. A 10-bit successive-approximation A/D converter experiment is included in Unit 7. Of course, a 10-bit D/A converter is required, too.

Using the software shown in Example 2-2, an 8-bit conversion will take about 240 microseconds (we have assumed a 500-nanosecond 8080 clock). The conversion time will be the same, regardless of the value of the unknown voltage being digitized, since each bit position must be tested. The sampling and digitizing may be represented as shown in Fig. 2-6.

If you look at Fig. 2-6 carefully, you will notice that some of the voltages output by the D/A converter were above the unknown voltage and others were below the unknown voltage. The matching process takes place until the D/A converter's output finally "homes in" on the unknown voltage.

The successive-approximation A/D converter may be applied to the instrument signal digitizing problem first shown in Fig. 2-4, where a ramp-type A/D converter was used to digitize a peak volt-

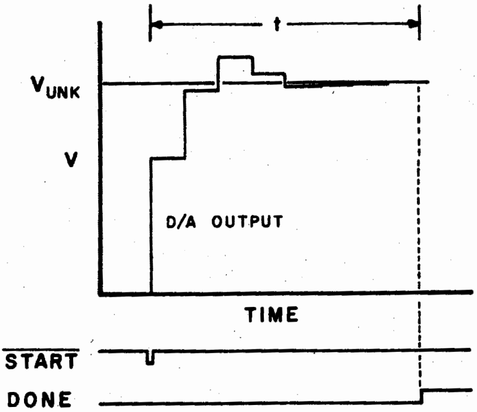
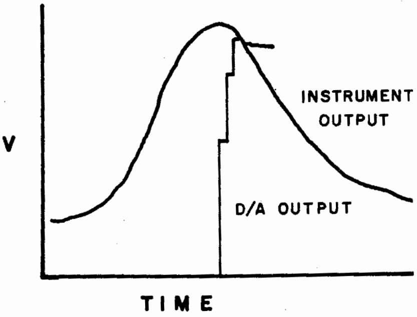


Fig. 2-6. Timing diagram for a successive-approximation A/D converter.

age. When a faster, successive-approximation A/D converter is applied to the problem, the outputs look like those shown in Fig. 2-7.

Again the software-based, successive-approximation converter is too slow to "catch" the actual peak value and the actual value present at the end of the conversion is lower than the real peak value. There is an important difference between the attempts of the ramp A/D converter and the successive-approximation converter to perform the conversion. When the ramp converter was used, the conversion was completed when the ramp voltage and the instrument's output coincided. When the successive-approximation A/D converter is used there is coincidence, but the converter is still testing bits when the coincidence occurred. *The conversion process will continue until all of the bits have been tested.* This means that the final four or five bit-testing sequences will attempt to match the decreasing instrument output. Thus, at the end of the successive-approximation conversion

Fig. 2-7. Timing relationship between the output of an instrument and the attempt of a software-based, successive-approximation A/D converter to digitize the peak voltage.



there is a large difference between the actual instrument output and the value provided by the conversion software. This is a real problem and it must always be considered when there is the possibility of the signal changing faster than the converter can keep up with it. There are ways to solve this problem, as you will see later.

There are some applications where it may be effective to use a software-based, successive-approximation A/D converter. However, successive-approximation A/D converter modules with high conversion speeds are available at moderate cost. Here are some representative samples from three manufacturers:

Analog Devices, Inc., Norwood, MA 02062

ADC10Z	10-bit	10-μsecond conversion time
AD7570	10-bit	20-μsecond conversion time

Burr-Brown Research Corp., Tucson, AZ 85734

ADC80AG-10	10-bit	18-μsecond conversion time
ADC80AG-12	12-bit	25-μsecond conversion time

Datel Systems, Inc., Canton, MA 02021

ADCHY12BC	12-bit	8-μsecond conversion time

With the exception of the Analog Devices AD7570 converter, al of the others have many input ranges, ±2.5 volts, ±5 volts, ±1 volts, 0–5 volts and 0–10 volts being typical. The AD7570 devic is interesting since it is a monolithic complementary metal-oxide semiconductor circuit with three-state outputs. It is easily interfaced t microcomputer systems. Many A/D converter modules are availabl with various output codes. The binary and binary-coded decima (BCD) codes will be the only ones considered in this book.

It has been our purpose in this section to show you how the ram and successive-approximation analog-to-digital conversion method work and to show you how software may be used to perform th conversions. Most people will be interested in interfacing some o the converter modules rather than in using the software-based converters. The next section of this unit describes the interfacing circuitry and software used with modular A/D converter interfacing

INTERFACING A/D CONVERTER MODULES

While the software-controlled A/D converters may be useful i some situations, most users will probably choose the prepackage A/D converter modules. In this section we will investigate how a 8-bit and a 10-bit A/D converter module may be interfaced to microcomputer. We will also discuss what we can do with the converter and such things as data-display files and data-acquisition timing.

Interfacing an 8-Bit Converter

Most analog-to-digital converters do not perform conversions one right after another as fast as they can, so there is more to A/D converter interfacing than just requesting a value and inputting it. Converters will generally perform a single conversion when they are signaled to do so by the computer. This signal is a TTL-compatible START or CONVERT pulse, which is easily generated under software control. The pulse may be either a logic zero or a logic one, depending upon the individual converter module chosen.

Since the A/D converter takes some time to perform the conversion, from a few microseconds to many milliseconds, we must not expect that the unknown signal has been digitized as soon as the START pulse has been received by the A/D converter. In most cases the A/D converter will have a *status* or flag output, which indicates that it is BUSY or that it is DONE. When the converter signals that it is done, the digital outputs are a true representation of the unknown voltage.

A typical interface for an 8-bit A/D converter requires an 8-bit, three-state input port for the data, a three-state input port for the DONE/$\overline{\text{BUSY}}$ flag and an output pulse to start the converter. A typical interface is shown in Fig. 2-8. The software which may be used to control the module is also shown in Example 2-3. The program is written as a subroutine that starts a conversion within the A/D

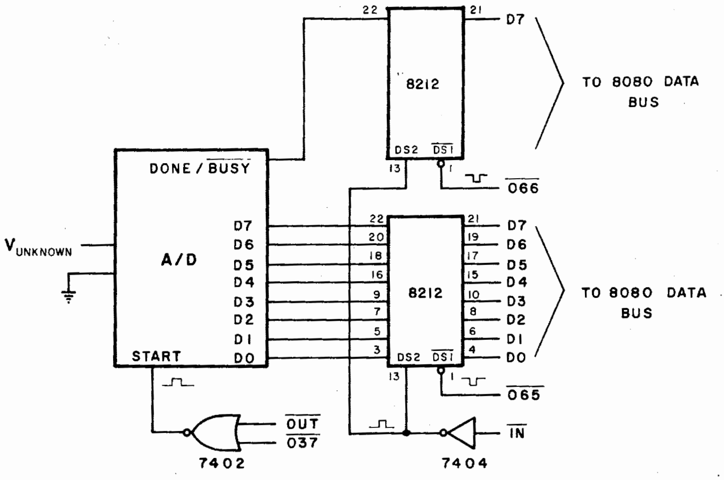


Fig. 2-8. A typical 8-bit A/D converter module interfaced to a microcomputer.

converter module by pulsing the START input and that senses the condition of the DONE/$\overline{\text{BUSY}}$ flag. When the conversion is completed, the 8-bit value is input into the 8080's A register.

The subroutine does not store the data in the computer's memory, but leaves it in the A register. Steps to store the data in a memory location could be added if they are needed. Actually, interfaces and software for 8-bit A/D converters are fairly simple.

The 8-bit A/D converters have a definite place in some systems, but some users require more *resolution,* or the ability to divide their signals into smaller portions, as provided by 10- or 12-bit A/D converters. These converters are more difficult to interface since multiple bytes are required for data transfer and data storage. Since these multibyte devices are the more general case, we have chosen to explore them in more detail than the 8-bit converters.

```
                          /EXAMPLE 2-3
                          /TYPICAL 8 BIT ADC CONTROLLED SUBROUTINE
                          /EIGHT BIT BINARY VALUE IS RETURNED IN
                          /THE A REGISTER
                          *000 000
000 000 323   CONV,       OUT        /START THE CONVERTER
000 001 037               037
000 002 333   TEST,       IN         /INPUT THE STATUS BIT
000 003 066               066
000 004 346               ANI        /MASK OUT ALL UNWANTED BITS
000 005 200               200
000 006 312               JZ         /IS IT DONE?
000 007 002               TEST       /NO, FLAG = 0, TEST AGAIN
000 010 000               0
000 011 333               IN         /YES, FLAG = 1, INPUT DATA
000 012 065               065
000 013 311               RET        /DONE, RETURN WITH VALUE IN A
```

Interfacing a 10-Bit A/D Converter

The 10-bit A/D converters are used in situations where their resolution of one part in 1024 is required. Interfacing a 10-bit A/D converter to a microcomputer is not difficult since the same interface elements used with the 8-bit A/D converter are still used with the 10-bit device. Three-state input ports are used to input data and flag information and an output pulse is still used to start the conversion process.

Input and output of data words that are longer than eight bits requires that the data be transferred in two or more 8-bit bytes. This was the case for the 10-bit D/A converter that was described in Unit 1; two 8-bit bytes were used to transfer data to the 10-bit converter, even though only two bits in one of the data words were used.

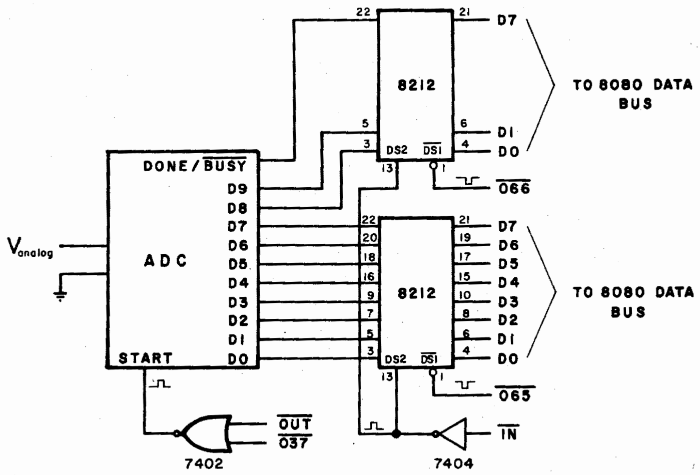


Fig. 2-9. A typical 10-bit A/D converter module interfaced to an 8-bit microcomputer. through the data bus.

When we attempt to interface a 10-bit A/D converter to the computer, we may find that an additional input port is required for the two additional bits of data. We can use the same type of interface as that used with the 8-bit A/D converter since there are spare, unused bits available on one of the input ports. The complete interface is shown in Fig. 2-9.

Since two bytes of data are to be input, the software used to control the A/D converter must be changed to reflect this. A complete program listing is provided in Example 2-4.

The A/D converter's status flag output may be tested in the same way that it was tested in the 8-bit A/D converter program, with an AND instruction being used to mask out the unwanted bits. However, if two of the unused bits of the three-state input port used for the flag condition input are now being used to input data, too, the AND operation will "destroy" the two data bits and this information will have to be input once again. An addition instruction has been chosen to replace the AND instruction. By adding a one to the most significant bit, an overflow or *carry* will take place if the MSB is a one. If the MSB is a zero, no carry will take place. This is a particularly useful operation for us since zeros will be added to the other bit positions. It should be noted that 0 + X = X, so the other seven bits will be unaffected by this addition operation.

Additional instructions have been incorporated in the software shown in Example 2-4 to store the A/D converter's data within the B and C registers. Register B will contain the A/D converter's two most significant bits in positions D1 and D0. Register C will contain the other eight bits, D7 through D0.

The program also includes PUSH and POP instructions so that the contents of the A register and the flags are preserved during the A/D converter control process. In cases such as this, it is useful to have the main program call the ADC subroutine and then store the data, rather than having the ADC subroutine perform all of these functions. This will keep the ADC subroutine as general as possible so that other sections of the program may have access to it, too.

```
                         /EXAMPLE 2-4
                         /ADC INPUT ROUTINE FOR A 10-BIT
                                   /CONVERTER. TWO MSB'S RETURNED IN REG B
                                   /AND THE 8 LSB'S RETURNED IN REGISTER C
                         *003 000
003 000 365  ADC,        PUSHPSW   /SAVE REG A & FLAGS
003 001 323              OUT       /START A CONVERSION
003 002 037              037
003 003 333  TEST,       IN        /INPUT STATUS & 2 MSB'S
003 004 066              066
003 005 306              ADI       /ADD 1 TO THE FLAG BIT TO
003 006 200              200       /CAUSE A CARRY IF IT IS SET
003 007 322              JNC       /NO OVERFLOW, CHECK IT AGAIN
003 010 003              TEST
003 011 003              0
003 012 107              MOVBA     /FLAG = 1, SO SAVE MSB'S
003 013 333              IN        /INPUT LSB'S
003 014 065              065
003 015 117              MOVCA     /STORE THEM IN REG C
003 016 361              POPPSW    /RESTORE REG A & FLAGS
003 017 311              RET       /RETURN
```

Two additional programs are also provided, Example 2-5 and Example 2-6. In each, instructions are provided to store in read/write memory the data obtained from the 10-bit A/D converter. The program in Example 2-5 shows one way in which this may be done. If registers B and C contained needed values before this subroutine was called, the data stored in these two registers will be "lost" since the registers are also being used in the ADC subroutine for temporary data storage.

```
                         /EXAMPLE 2-5
                         /THIS PROGRAM CALLS THE "ADC" SUBROUTINE
                         /AND THEN STORES THE 10 BITS OF DATA IN
                         /A FILE IN R/W MEMORY
                         *004 000
004 000 061              LXISP     /LOAD A STACK POINTER VALUE
004 001 000              000
004 002 060              060
```

```
004 003 041       LXIH     /LOAD THE MEMORY POINTER
004 004 000       000      /LOW ADDR= 000
004 005 061       061      /HIGH ADDR= 061
004 006 315       CALL     /CALL THE CONVERSION ROUTINE
004 007 000       ADC
004 010 003       0
004 011 161       MOVMC    /STORE THE 8 LSB'S IN MEMORY
004 012 043       INXH     /INCREMENT THE ADDRESS
004 013 043       MOVMB    /STORE THE S MSB'S
004 014 160       INXH     /INCREMENT AGAIN
004 015 000       0        /ADDITIONAL SOFTWARE WILL CONTINUE
004 012 043       INXH     /FROM THIS POINT
```

The program shown in Example 2-6 shows how PUSH and POP instructions may be used in the *main program* to save the contents of the B and C registers and then restore the values after the ADC subroutine and the data storage section of the main program have completed their execution. The PUSHB and POPB instructions could not be used in the ADC subroutine if we wished to have access to the 10-bit value from the A/D converter. If the PUSHB and POPB instructions are used in the subroutine, the following sequence would take place whenever the ADC subroutine is used; the old values in the B and C registers would be stored on the stack, the B and C registers would be used for storage of the A/D converter's data, the pop operation would then restore the old values back into registers B and C and the return operation would return control to the main program. Thus, the data from the A/D converters would not be available. You must always be cautious in your use of the stack operations. Remember: *The stack pointer must be set to an available area of read/write memory before any stack operations are used.*

```
                  /EXAMPLE 2-6
                  /THIS SOFTWARE WILL PERFORM A CONVERSION
                  /AND STORAGE OF THE DATA, BUT REGISTERS
                  /B & C ARE PRESERVED
                  *004 000
004 000 061       LXISP    /LOAD THE STACK POINTER
004 001 000       000
004 002 060       060
004 003 041       LXIH     /LOAD THE MEMORY POINTER
004 004 000       000
004 005 061       061
004 006 305       PUSHB    /SAVE REG B & C ON THE STACK
004 007 315       CALL     /CALL THE CONVERSION ROUTINE
004 010 000       ADC
004 011 003       0
004 012 161       MOVMC    /THIS STORES THE DATA
004 013 043       INXH
```

```
004 014 160        MOVMB
004 015 043        INXH
004 016 301        POPB      /THIS RESTORES REG B & C
004 017 000        0         /ADDITIONAL SOFTWARE WILL BE
                             /ADDED HERE
```

USING A/D CONVERTERS FOR DATA ACQUISITION

Using analog-to-digital converters with microcomputers allows data to be acquired at a maximum rate of about 20,000 points per second. This approximation was made using an 8-bit converter with a simple input program and program steps which would be used to store the data and perform various software "overhead" tasks such as counting the number of data points acquired. Data cannot be acquired for very long if this data acquisition rate is to be maintained since the computer will quickly run out of available read/write memory in which to store the data. Slower data rates of 10 to 25 points per second are more reasonable for many measurements.

In this section, we will examine the use of a small computer to acquire temperature data from a greenhouse over a 24-hour period. Temperature data will be taken every 10 *minutes* using a 10-bit A/D converter which has been interfaced according to the schematic presented previously in Fig. 2-9.

We will assume that the sensor is normalized to have an output of 0–5 volts, which is the range of the A/D converter's analog input. A thermocouple or bridge circuit could be used, but newer, solid-state devices such as the National Semiconductor LM3911 temperature-controller chip might be easier to use; the LM3911 provides a voltage output that is directly proportional to degrees centigrade (Celsius).

Since the data from the temperature sensor is to be acquired over a 24-hour period with a 10-minute interval between points, 144 measurements will be made by the 10-bit A/D converter. Each 10-bit binary value will require two 8-bit memory locations for storage. The computer is expected to store the data for the 24-hour period and then halt so that the data may be examined.

A 10-minute time-delay program has been written and it will be used to "delay" the computer between the acquisition of the temperature values. The actual time required to execute the data acquisition steps and the data storage steps, plus the time needed by the converter to perform the conversion is small (100 to 200 microseconds) when compared to the 10-minute time interval between temperature readings. The 10-minute time delay program is a general one and it does not take these relatively short periods into account. A completely assembled data-acquisition program is provided in Example 2-7.

```
                          /EXAMPLE 2-7
                          /DATA ACQUISITION PROGRAM FOR A 10-BIT
                          /ADC SAMPLING 144 POINTS AT ONE POINT
                          /EVERY 10 MINUTES
                          *003 000
003 000 061  START,       LXISP      /LOAD STACK POINTER FOR SUBROUTINES
003 001 377               377
003 002 003               003
003 003 026               MVID       /LOAD D AS A 144 POINT COUNTER
003 004 220               220        /220 = 144 DECIMAL
003 005 041               LXIH       /LOAD H & L AS MEMORY POINTERS
003 006 000               000
003 007 002               002
003 010 323  CONVERT,     OUT        /START A CONVERSION
003 011 037               037
003 012 333  TEST,        IN         /INPUT FLAG BIT
003 013 066               066
003 014 306               ADI        /ADD 1 TO FLAG BIT TO CAUSE
003 015 200               200        /A CARRY IF IT IS SET
003 016 322               JNC        /NO OVERFLOW, CHECK IT AGAIN
003 017 012               TEST
003 020 003               0
003 021 107               MOVBA      /SAVE THE MSB'S
003 022 333               IN         /INPUT 8 LSB'S
003 023 065               065
003 024 167               MOVMA      /STORE THEM IN MEMORY
003 025 043               INXH       /INCREMENT MEMORY POINTER
003 026 160               MOVMB      /STORE 2 MSB'S, TOO
003 027 043               INXH
003 030 025               DCRD       /DECREMENT THE POINT COUNT
003 031 312               JZ         /IS IT = 0?
003 032 073               DONE       /YES, DONE
003 033 003               0
003 034 315               CALL       /NO, CALL TIMER DELAY
003 035 042               DELAY
003 036 003               0
003 037 303               JMP        /GO BACK AND DO IT AGAIN
003 040 010               CONVRT
003 041 003               0
                          /THIS IS A 10-MINUTE DELAY SUBROUTINE
003 042 365  DELAY,       PUSHPSW    /SAVE REGISTERS AND FLAGS
003 043 305               PUSHB
003 044 325               PUSHD
003 045 021               LXID       /DO 1200 LOOPS OF 0.5 SEC EACH
003 046 260               260        /DELAY LOOP
003 047 004               004
003 050 001  TIME1,       LXIB       /DELAY TIME LOOP 0.5 SEC
003 051 303               303        /CONSTANTS FOR 500 NSEC CLOCK
003 052 242               242
003 053 013  TIME2,       DCXB       /DECREMENT REG PAIR B
003 054 170               MOVAB      /GET B
003 055 261               ORAC       /OR IT WITH C
003 056 302               JNZ        /IF NOT 0, DO IT AGAIN
```

```
003 057 053           TIME2
003 060 003           0
003 061 033           DCXD      /DECREMENT SECOND COUNTER
003 062 172           MOVAD     /GET D
003 063 263           ORAE      /OR IT WITH E
003 064 302           JNZ       /IF NOT DONE, GO BACK TO
003 065 050           TIME1
003 066 003           0
003 067 321           POPD      /RESTORE REGISTERS & FLAGS
003 070 301           POPB
003 071 361           POPPSW
003 072 311           RET       /RETURN AFTER 600 SECONDS
                      /THE PROGRAM HALTS HERE WHEN IT IS "DONE."
003 073 166   DONE,   HLT       /END OF PROGRAM
```

DATA-ACQUISITION TIMING

The program provided in Example 2-7 will work well, acquiring a 10-bit temperature value every 10 minutes, but the DELAY subroutine will "tie-up" the computer and prevent it from doing other tasks In a program such as this, the computer is *dedicated* to the temperature measurement software, but there are hardware and software methods which will help to eliminate this problem.

You will quickly find that software timing loops monopolize the computer's time. Certainly the computer cannot be performing other software tasks while it is also executing a time-delay program. For short time delays it may be possible to use the computer solely for the delay of the program. Even long delays such as the 10-minute DELAY subroutine are acceptable if the computer has no other tasks to perform. If interrupts are used in a computer system and a programmer has relied upon software time-delay loops, the actual time delays may be longer than expected. This happens when the time-delay loops are interrupted by an external device. Since the interrupting device's software will require a finite amount of computer execution time, the overall time delay becomes longer than anticipated. In some systems this is an important consideration.

In the greenhouse temperature measurement example, there are currently no interrupting devices connected to the computer. To avoid future problems, it might be best to consider an alternative to the time-delay software such as that used in the DELAY subroutine.

An external *real-time clock* will be used instead of the software delay loop for timing the 10-minute intervals. This type of a clock is independent of software execution time and it will accurately time periods which will not be affected by interrupts. Generally, a crysta oscillator and an appropriate series of digital divider or counter circuits are used to provide a series of accurate frequencies or time periods. A typical clock is shown in Fig. 2-10.

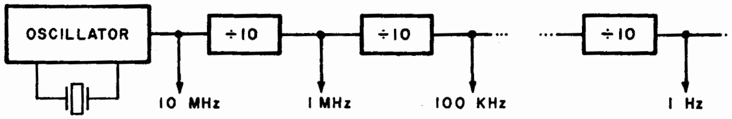


Fig. 2-10. Block diagram of a stable crystal oscillator and frequency dividers used in a real-time clock circuit.

Additional dividers or counters (they perform the same function) could be added to the crystal clock to provide other time periods or frequencies. The greenhouse temperature data-acquisition system would require a 10-minute, or 600-second, period, so additional counters would be needed. Many fast data-acquisition systems would require only the higher frequencies.

The time periods provided by a real-time clock are usually sensed in one of two ways; with a polled flag that is sensed under program control or with an interrupt that is independent of program control. Both methods are important and they will be discussed in detail.

Polled Flag Timers

In this example of a polled flag timer, a simple flip-flop will be used to detect the positive edge of one of the crystal clock's divider outputs. The clock period is assumed to be 10 minutes, so a positive edge will occur only once every 10 minutes. The necessary circuitry to implement a simple polled flag timer is shown in Fig. 2-11.

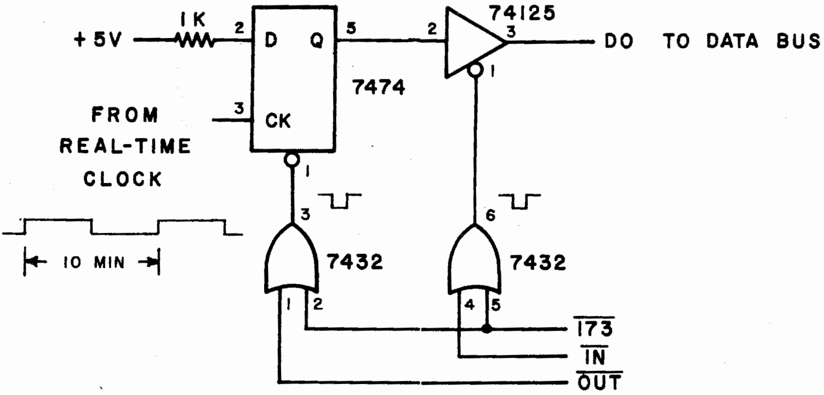


Fig. 2-11. A typical interface for a polled flag timer. The state of the flag is input to the computer on the data bus, bit DO.

A short software example that illustrates the use of an external flag-timer circuit is shown in Example 2-8. When this portion of the program is executed, the status of the flag is input and tested. If the flip-flop's output is a logic zero (still timing), the computer executes instructions that cause it to jump around the A/D converter control steps to CONT where it continues program execution. If the flip-flop's output is a logic one (timer finished), the computer executes the calls to the ADC and the STORE subroutines. The ADC subroutine inputs the A/D converters 10 bits of data and the STORE subroutine stores the data in read/write memory. This software would be incorporated in the computer's main program, MAIN TASK. The ADC subroutine has been encountered before in Example 2-4. The STORE subroutine is not illustrated since we are not sure how we want to store the data. This will be explored in further examples.

```
                        /EXAMPLE 2-8
                        /FLAG SENSING SOFTWARE USED WITH THE ADC
                        /SUBROUTINE AND A STORE SUBROUTINE
                        *003 126
003 126 333             IN        /INPUT THE TEST FLAG
003 127 173             173
003 130 346             ANI       /MASK OUT ALL OTHER BITS
003 131 001             001       /MASK= 00000001
003 132 312             JZ        /NO FLAG, CONTINUE
003 133 145             CONT
003 134 003              0
003 135 323             OUT       /CLEAR THE FLAG FLIP-FLOP
003 136 173             173
003 137 315             CALL      /FLAG FOUND, DO A CONVERSION
003 140 000             ADC
003 141 100             0
003 142 315             CALL      /STORE THE DATA IN MEMORY USING
003 143 000             STORE     /THIS CALL TO A STORAGE SUBROUTINE
003 144 200             0         /WHICH YOU ADD TO THE PROGRAM
003 145 000             0         /CONTINUE WITH THIS INSTRUCTION
                                  /IF NO FLAG IS FOUND OR AFTER AN
                                  /ADC INPUT
```

In the second example of a polled flag timer's software, Example 2-9, a single subroutine, ADCSTR, is used to replace the ADC and STORE subroutines discussed in Example 2-8. The new ADCSTR subroutine is completely *transparent* since all of the registers that will be used are first stored in the stack with PUSH instructions and later restored with POP instructions. The ADCSTR subroutine will perform a single, 10-bit conversion using the A/D converter previously interfaced to the computer. The 10-bit data value is then stored in locations 005 000 and 005 001. This software does not set up a data file, although that could be done. *Only the current 10-bit data value is retained.* Perhaps the MAIN TASK program uses it in some way, adding it to the data in the data file.

```
                          /EXAMPLE 2-9
                          /TYPICAL FLAG SENSING SOFTWARE USED WITH
                          /THE ADC AND STORE SUBROUTINES
                          *003 126
003 126 333               IN        /INPUT THE TEST FLAG
003 127 173               173
003 130 346               ANI       /MASK OUT ALL OTHER BITS
003 131 001               001       /MASK= 00000001
003 132 312               JZ        /NO FLAG, CONTINUE
003 133 142               CONT
003 134 003               0
003 135 323               OUT       /CLEAR THE FLAG FLIP-FLOP
003 136 173               173
003 137 315               CALL      /CALL A SUBROUTINE TO INPUT AND
003 140 000               ADCSTR    /STORE A NEW VALUE IN A MEMORY
003 141 010               0         /LOCATION CALLED NEWVAL
003 142 000   CONT,       0         /CONTINUE WITH THIS INSTRUCTION
                                    /IF NO FLAG IS FOUND OR AFTER AN
                                    /ADC INPUT
                          *010 000
010 000 365   ADCSTR,     PUSHPSW   /SAVE REGISTER A & FLAGS
010 001 305               PUSHB     /SAVE REGISTERS B & C
010 002 345               PUSHH     /SAVE REGISTERS H & L
010 003 323               OUT       /START THE CONVERTER
010 004 037               037
010 005 333   CHK,        IN        /INPUT STATUS BIT
010 006 066               066
010 007 306               ADI       /CHECK THE FLAG
010 010 200               200       /BY ADDING 1 TO IT
010 011 322               JNC       /IF NOT DONE, CHECK AGAIN
010 012 005               CHK
010 013 010               0
010 014 117               MOVCA     /2 MSB'S ARE SAVED IN C
010 015 333               IN        /INPUT 8 LSB'S
010 016 065               065
010 017 041               LXIH      /SET UP MEMORY POINTERS
010 020 000               NEWVAL
010 021 005               0
010 022 167               MOVMA     /STORE THE 8 LSB'S
010 023 043               INXH      /INCREMENT THE MEMORY POINTER
010 024 161               MOVMC     /STORE THE 2 MSB'S
010 025 341               POPH      /RESTORE THE REGISTERS
010 026 301               POPB
010 027 361               POPPSW
010 030 311               RET       /RETURN TO THE MAIN PROGRAM
                          *005 000
005 000 000   NEWVAL,     0         /8 LSB'S STORED HERE
005 001 000               0         /2 MSB'S STORED HERE
```

When the flag's logic-one state is detected and when the computer starts to execute the ADCSTR subroutine, it will take some time for the computer to complete the tasks in the ADCSTR subroutine and resume execution of the MAIN TASK program at symbolic address CONT. The time taken to execute ADCSTR will delay the execution of the MAIN TASK program. In this case, the time is not significant, but it is precisely this uncertainty in computer program execution times that led to the development of the real-time clock.

In both of these software examples (Examples 2-8 and 2-9), the flag-checking and decision-making steps had to be added to the MAIN TASK program so that the timer could be checked, and its condition acted upon by the computer. If the MAIN TASK program is executed very quickly, so much the better, but lengthy tasks may cause some uncertainty in the time delay since it is impossible to know when the computer will get back to the added flag-checking portion of the program. The real-time clock itself is independent of the software being executed, but software is still needed to check the flag. If MAIN TASK can go through this portion of the program every 50 or 100 milliseconds, then the maximum time between temperature readings would be 10 minutes and 1/10th second. If, however, MAIN TASK takes a few minutes to get back to the flag-checking steps, the maximum time between temperature readings may be 12, 13, or more minutes. This is unreasonable.

Interrupt Timers

Interrupts appear to be an attractive alternative to flag timers since the interrupts will be serviced almost immediately by the computer. If interrupts are to be used with an 8080-based system, it will be our responsibility to enable the interrupt and to construct an interrupt flag and an interrupt instruction port so that a single-byte restart instruction, RST7, will be gated into the 8080 when an interrupt occurs. We will assume that there is only this single interrupting device.

You should recall that a restart instruction will cause the 8080 microcomputer to call a subroutine at a special vector address. In this case, it will be a call to the subroutine at location 000 070. This means that a restart instruction of 377 (RST7) must be gated into the 8080 when the interrupt is acknowledged. Remember: *A stack must be established in read/write memory before you use restart instructions.*

The same flag circuitry used with the polled flag timer will be used in this example. The circuit is shown in Fig. 2-12. You will note that the only difference between this circuit and the polled flag circuit (Fig. 2-11) is the addition of an interrupt signal, TO INTERRUPT. This is connected to the 8080 microprocessor chip's interrupt-input pin, pin 14. An interrupt-instruction port is shown in Fig. 2-13. An SN74S240 8-bit, three-state buffer has been used to supply the 377 instruction (RST7) when the buffer is enabled with the interrupt acknowledge signal, $\overline{\text{INTA}}$.

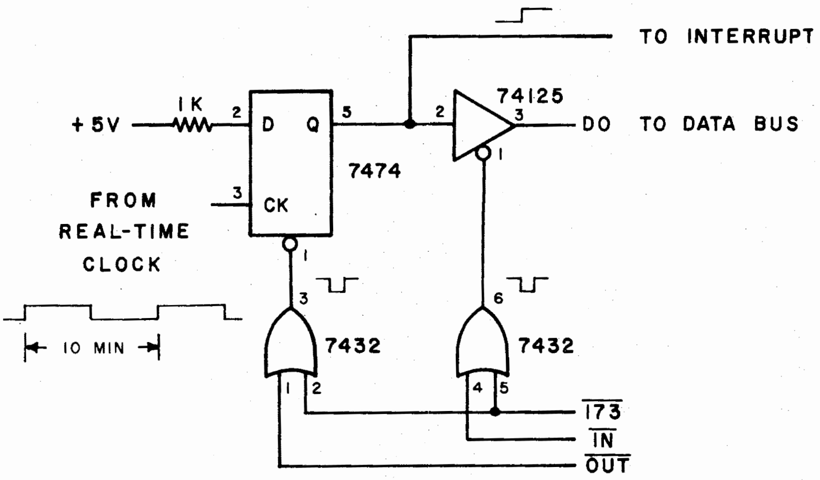


Fig. 2-12. Interrupt flag circuit. The flag may be tested using either an interrupt or a polled software operation.

A positive edge applied to the clock input of the D-type flip-flop will now cause an interrupt that will be serviced or acknowledged by the 8080 immediately after it finishes executing its current instruction. This response time is in the order of a few microseconds. When the 8080 acknowledges the interrupt, the RST7 instruction is gated onto

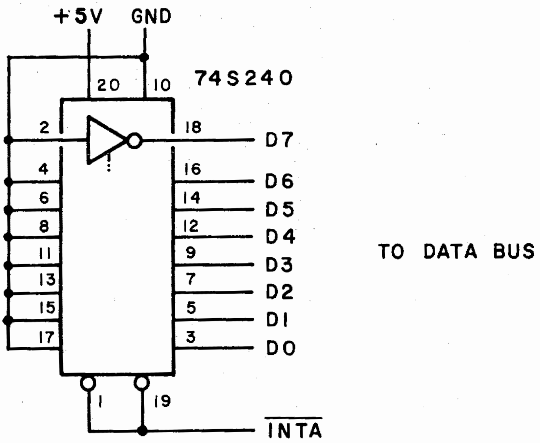


Fig. 2-13. Interrupt-instruction port set up for the RST7 or 377 instruction. An SN74S240 integrated circuit is used as a three-state input.

the data bus and into the 8080's *instruction register* where it is decoded to instruct the 8080 to perform the call to the subroutine starting at address 000 070. The analog-to-digital converter's service program starts at that address.

The interrupt-service subroutine shown in Example 2-10 has some interesting features. Notice that the A/D converter is started before the two PUSH instructions are executed. Since the OUT instruction does not alter any of the 8080's registers, this is certainly valid. Starting the conversion first saves time since the converter will be performing the conversion while the two PUSH instructions are being executed. When an 8080 with a 500-nanosecond clock period is used, this saves 11 microseconds, so the A/D conversion is almost complete if we are using a fast, successive-approximation A/D converter such as the Analog Devices AD 7570. The LHLD instruction loads register pair H with the address to point to the data storage locations. This adds another eight microseconds to the time between the start of the conversion and the end-of-conversion flag-checking steps.

If enough instructions are placed between the command used to start the conversion process and the commands used to check the state of the DONE/$\overline{\text{BUSY}}$ flag, the flag checking may not be necessary. The converter will have completed the conversion process by the time that the computer is ready to input the data. In this case you may be able to do away with the flag-checking steps and input the data directly. This technique should be used only when you know exactly how long the conversion process will take and when that time is shorter than the time that the computer will require for its software "overhead" tasks.

The software in Example 2-10 will continue to input and store temperature values every 10 minutes until we disable it in some way. Additional software might be added to count the 144 data points over 24 hours, or this could be included in the MAIN TASK program. MAIN TASK could access and use the address value stored at POINT to determine how many points had been input and stored. The point-counting software might also have been added to the A/D converter service subroutine, ADCSVC. The program may be disabled simply by disabling the interrupt enable flag with a disable interrupt instruction, DI, 363. This brings up an important point. Only an enable-interrupt instruction and a load-stack-pointer instruction are needed in MAIN TASK to implement this interrupt-based solution to the 10-minute timing problem. There is no software to check a flag or to delay the execution of MAIN TASK.

The second interrupt software example (Example 2-11) is complete, since it includes instructions which are used to build a 144-point data file and the program will branch to DONE when all of the points have been acquired.

```
                          /EXAMPLE 2-10
                          /10-BIT ADC INTERRUPT SERVICE SUBROUTINE
                          /DATA FILE STARTS AT THE ADDRESS STORED AT POINT
                          *000 070
000 070 323  ADCSVC,      OUT        /START THE CONVERTER
000 071 037               037
000 072 365               PUSHPSW    /SAVE THE REGISTER & FLAGS
000 073 345               PUSHH      /SAVE REGISTERS H & L
000 074 052               LHLD       /GET MEMORY POINTERS INTO H & L
000 075 000               POINT      /SO THE DATA MAY BE STORED
000 076 120               0
000 077 333  TEST,        IN         /INPUT THE FLAG BIT
000 100 066               066
000 101 306               ADI        /ADD A 1 TO THE FLAG BIT
000 102 200               200
000 103 322               JNC        /IS IT A 1?
000 104 077               TEST       /NO, NOT DONE YET
000 105 000               0
000 106 107               MOVBA      /YES, ADC DONE, STORE 2 MSB'S IN B
000 107 333               IN         /INPUT 8 LSB'S
000 110 065               065
000 111 167               MOVMA      /STORE THEM IN MEMORY
000 112 043               INXH       /INCREMENT THE MEMORY POINTER
000 113 160               MOVMB      /STORE THE 2 MSB'S IN MEMORY
000 114 043               INXH       /INCREMENT MEMORY POINTER AGAIN
000 115 042               SHLD       /SAVE THE STORAGE AREA ADDRESS
000 116 000               POINT
000 117 120               0
000 120 341               POPH       /RESTORE REGISTERS H & L
000 121 361               POPPSW     /RESTORE REGISTER A & FLAGS
000 122 323               OUT        /CLEAR THE INTERRUPT FLAG
000 123 173               173
000 124 373               EI         /RE-ENABLE THE INTERRUPT
000 125 311               RET        /RETURN TO MAIN PROGRAM
                          *120 000
120 000 000  POINT,       000        /THIS IS WHERE THE ADDRESS OF THE ADC
120 001 020               020        /STORAGE AREA IS KEPT. IN THIS PROGRAM
                                     /THE STORAGE AREA STARTS AT
                                     /ADDRESS 020 000. YOU COULD PLACE YOUR
                                     /OWN POINTER ADDRESS HERE, BUT THESE
                                     /TWO LOCATIONS MUST BE IN R/W MEMORY

                          /EXAMPLE 2-11
                          /THIS IS THE ADC INTERRUPT SERVICE SOFTWARE
                          /WITH THE ADDITIONAL STEPS FOR A POINT
                          /COUNTER AND EXIT TO A "DONE" ROUTINE
                          /WHEN ALL POINTS ARE ACQUIRED
                          *000 070
000 070 323  ADCSVC,      OUT        /START THE CONVERTER
000 071 037               037
000 072 365               PUSHPSW    /SAVE REGISTER A & FLAGS
000 073 325               PUSHD      /SAVE REGISTERS D & E
000 074 305               PUSHB      /SAVE REGISTERS B & C
000 075 345               PUSHH      /SAVE REGISTERS H & L
```

```
000 076 052           LHLD      /GET THE POINT COUNT
000 077 002           COUNT
000 100 120           0
000 101 353           XCHG      /EXCHANGE REG PAIR D & H
000 102 052           LHLD      /GET MEMORY POINTERS INTO H & L
000 103 000           POINT     /SO THE DATA MAY BE STORED
000 104 120           0
000 105 333  TEST,    IN        /INPUT THE FLAG BIT
000 106 066           066
000 107 306           ADI       /ADD A 1 TO THE FLAG BIT
000 110 200           200
000 111 322           JNC       /IS IT A 1?
000 112 105           TEST      /NO, NOT DONE YET
000 113 000           0
000 114 107           MOVBA     /YES, ADC DONE, STORE 2 MSB'S IN B
000 115 333           IN        /INPUT 8 LSB'S
000 116 065           065
000 117 167           MOVMA     /STORE THEM IN MEMORY
000 120 043           INXH      /INCREMENT THE MEMORY POINTER
000 121 160           MOVMB     /STORE THE 2 MSB'S IN MEMORY
000 122 043           INXH      /INCREMENT MEMORY POINTER AGAIN
000 123 025           DCRD      /DECREMENT POINT COUNT
000 124 312           JZ        /ALL POINTS TAKEN?
000 125 146           EXIT      /YES, GO TO EXIT
000 126 000           0
000 127 042           SHLD      /SAVE THE STORAGE AREA ADDRESS
000 130 000           POINT
000 131 120           0
000 132 353           XCHG      /EXCHANGE REG PAIRS D & H
000 133 042           SHLD      /SAVE THE POINT COUNT
000 134 002           COUNT
000 135 120           0
000 136 341           POPH      /RESTORE REGISTERS H & L
000 137 301           POPB      /RESTORE REGISTERS B & C
000 140 321           POPD      /RESTORE REGISTERS D & E
000 141 361           POPPSW    /RESTORE REGISTER A & FLAGS
000 142 323           OUT       /CLEAR THE INTERRUPT FLAG
000 143 173           173
000 144 373           EI        /RE-ENABLE THE INTERRUPT
000 145 311           RET       /RETURN TO MAIN PROGRAM
000 146 341  EXIT,    POPH      /RESTORE THE REGISTERS
000 147 301           POPB
000 150 321           POPD
000 151 361           POPPSW
000 152 063           INXSP     /INCREMENT STACK POINTER PAST
000 153 063           INXSP     /THE RETURN ADDRESS
000 154 000  DONE,    0         /PLACE ADDITIONAL SOFTWARE HERE
                                /AND ON UP

                      *120 000
120 000 000  POINT,   000       /THIS IS WHERE THE ADDRESS OF THE ADC
120 001 120           020       /STORAGE AREA IS KEPT. IN THIS PROGRAM
                                /THE STORAGE AREA STARTS AT
                                /ADDRESS 020 000. YOU COULD PLACE YOUR
```

```
                                     /OWN POINTER ADDRESS HERE, BUT THESE
                                     /TWO LOCATIONS MUST BE IN R/W MEMORY
120 002 000  COUNT,   000            /THE POINT COUNTER IS STORED HERE IN THE
120 003 220           220            /SECOND BYTE. THE FIRST BYTE IS NOT USED.
                                     /220 = 144 DECIMAL
```

This program is slightly more complicated than the program shown in Example 2-10 since we are going to branch out of a subroutine and we have no intention of using the return address stored on the stack during the computer's execution of the restart or one-byte call instruction. Two stack-pointer-increment instructions are used to move the stack pointer beyond the two addresses used to store the complete, two-byte return address. If you wish to have the computer use the return address, simply remove the two INXSP instructions and substitute two no-operation instructions in their places.

The area of memory at symbolic address DONE has been left open so that an output or display software routine could be added to output the 144 temperature readings. You should note that by branching to DONE, you will probably cease operation of the MAIN TASK program.

Real-time clocks are very useful peripherals to have on a small computer system where accurate timing is important. Clocks may be as simple as the oscillator and counter chain that we have discussed previously, or more complex devices such as the Intel 8253 programmable interval-timer integrated circuit or the Texas Instruments TMS-5501 multifunction I/O controller integrated circuit; each of these devices having programmable registers that may be set to a particular count. After the programmed number of counts has taken place, the device will either generate an interrupt or set a flag which may be sensed with software commands. Counting rates are determined by an external crystal clock.

USING THE DATA

Once the data from the temperature sensor has been acquired and stored in the computer's memory, we must decide what we want to do with it. It could be output to a teletypewriter for a supervisor's records, but this may not be meaningful since the data is stored as 10-bit binary voltage values and not as temperatures. Additional routines could be added to the software to convert the 10-bit binary data to temperatures, but a simpler output means is available.

We will use a 10-bit D/A converter to output the data to a small oscilloscope for display after the 24-hour data acquisition period. This will give the farmer or nurseryman a profile of the temperature over the past 24 hours. The data might also be output to a strip-chart recorder to form a permanent record.

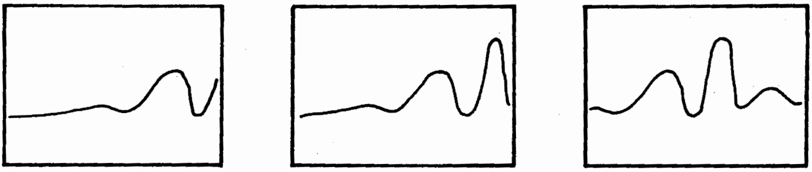
The software to be used for the display of the 10-bit values is similar to that provided in Example 1-18. In this next software example, we will use the 10-minute software delay loop since it simplifies the overall software. Interrupt or flag timers could be used, but we have chosen not to use them for simplicity in the example. The software duplicates the program steps shown in Example 2-7, except that the display steps have been added in place of the halt instruction at DONE. The complete program is provided in Example 2-12.

```
                            /EXAMPLE 2-12
                            /DATA ACQUISITION & DISPLAY PROGRAM FOR
                            /144 POINTS
                            *003 000
003 000 061   START,        LXISP      /LOAD STACKER POINTER FOR SUBROUTINES
003 001 377                 377
003 002 003                 003
003 003 026                 MVID       /LOAD D AS A 144 POINT COUNTER
003 004 220                 220        /220 = 144 DECIMAL
003 005 041                 LXIH       /LOAD H & L AS MEMORY POINTERS
003 006 000                 000
003 007 002                 002
003 010 323   CONVRT,       OUT        /START A CONVERSION
003 011 037                 037
003 012 333   TEST,         IN         /INPUT FLAG BIT
003 013 066                 066
003 014 306                 ADI        /ADD 1 TO FLAG BIT TO CAUSE
003 015 200                 200        /A CARRY IF IT IS SET
003 016 322                 JNC        /NO OVERFLOW, CHECK IT AGAIN
003 017 012                 TEST
003 020 003                 0
003 021 107                 MOVBA      /SAVE THE MSB'S
003 022 333                 IN         /INPUT 8 LSB'S
003 023 065                 065
003 024 167                 MOVMA      STORE THEM IN MEMORY
003 025 043                 INXH       /INCREMENT MEMORY POINTER
003 026 160                 MOVMB      /STORE 2 MSB'S, TOO
003 027 043                 INXH
003 030 005                 DCRB       /DECREMENT THE POINT COUNT
003 031 312                 JZ         /IS IT = 0?
003 032 100                 DONE       /YES, DONE
003 033 003                 0
003 034 315                 CALL       /NO, CALL TIMER DELAY
003 035 042                 DELAY
003 036 003                 0
003 037 303                 JMP        /GO BACK AND DO IT AGAIN
003 040 010                 CONVRT
003 041 003                 0
                                       /THIS IS WHERE THE 10 MINUTE DELAY
                                       /SUBROUTINE WOULD GO
003 042 000   DELAY,        0
                                       /THE DISPLAY SOFTWARE IS NOW SUBSTITUTED
                                       /FOR THE HALT INSTRUCTION
```

```
                              *003 100
003 100 041   DONE,   LXIH     /RESET THE ADDRESS POINTER
003 101 000           000
003 102 002           002
003 103 026           MVID     /RESET THE POINT COUNTER
003 104 220           220
003 105 176   LOOP,   MOVAM    /GET THE 8 LSB'S
003 106 323           OUT      /OUTPUT THEM TO THE DAC
003 107 054           054
003 110 043           INXH     /INCREMENT MEMORY POINTER
003 111 176           MOVAM    /GET THE 2 MSB'S
003 112 323           OUT      /OUTPUT THEM, TOO
003 113 055           055
003 114 323           OUT      /STROBE THE DOUBLE BUFFERED
003 115 056           056      /DAC MODULE
003 116 043           INXH     /INCREMENT POINTER AGAIN
003 117 025           DCRD     /DECREMENT POINT COUNT
003 120 312           JZ       /IS IT = 0?
003 121 100           DONE     /YES, REINITIALIZE AND DISPLAY
003 122 003           0        /THE FILE AGAIN
003 123 303           JMP      /NO, KEEP GOING
003 124 105           LOOP
003 125 003           0
```

This software will work nicely to display the data for the past 24-hour period, but it is limited in its usefulness, since new data will not be displayed until the current 24-hour period has elapsed and all 144 of the new temperature readings have been acquired. Can the program be modified so that instead of displaying the data only at the end of the 24-hour acquisition period it continually updates the data and displays the data for the *past 24 hours?* Fig. 2-14 shows what a typical updated display might look like.

While these updated displays are somewhat exaggerated, they do illustrate what the hardware and software will be required to do. New temperature points or readings are added to the display on the right side as they are acquired and old points are removed from the left side of the display. This means that only data from the *latest* 24-hour



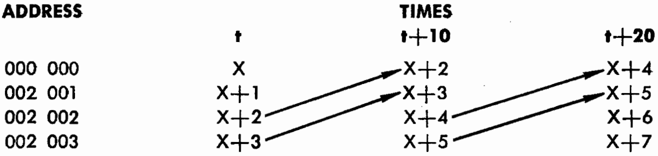
(A) Time t_0.

(B) Time t_1.

(C) Time t_2.

Fig. 2-14. An updated display showing the data at three times, t_0, t_1, and t_2. The data seems to "move" from right to left.

period are displayed. The movement of the data in the display storage area of read/write memory is shown graphically as follows:



ADDRESS	TIMES		
	t	t+10	t+20
000 000	X	X+2	X+4
002 001	X+1	X+3	X+5
002 002	X+2	X+4	X+6
002 003	X+3	X+5	X+7

Note that data values at X and X+1 are both lost at t+10, because each temperature value is 10 bits, requiring two, 8-bit bytes for storage.

New points are added to the end of the temperature data file and, since the oscilloscope's beam moves from left to right, the latest temperature value to be acquired is displayed on the right of the display. The software necessary to move the data down by two positions is shown in Example 2-13. Only 143 of the data points need to be moved since the 144th point will be the latest reading added to the end of the data file. However, it is just as easy to move 144 data points.

```
                         /EXAMPLE 2-13
                         /THIS SUBROUTINE MOVES THE DATA VALUES
                         /DOWN TWO ADDRESSES IN R/W MEMORY
                         *003 000
003 000 052  UPDATE,     LHLD      /GET THE FILE ADDRESS
003 001 100              POINT
003 002 003              0
003 003 072              LDA       /GET THE POINT COUNT
003 004 102              COUNT
003 005 003              0
003 006 137              MOVEA     /STORE IT IN REG E
003 007 043              INXH      /INCREMENT ADDRESS TWICE
003 010 043              INXH
003 011 006  NXTPNT,     MVIB      /SET B= # BYTES PER POINT
003 012 002              002
003 013 116  BYTE2,      MOVCM     /GET THE DATA
003 014 053              DCXH      /DECREMENT THE ADDRESS TWICE
003 015 053              DCXH
003 016 161              MOVMC     /PUT THE DATA THERE
003 017 043              INXH      /MOVE ADDR UP BY THREE
003 020 043              INXH
003 021 043              INXH
003 022 005              DCRB      /ALL BYTES DONE?
003 023 302              JNZ       /NO, DO ANOTHER TRANSFER
003 024 013              BYTE2
003 025 003              0
003 026 035              DCRE      /YES, DECREMENT POINT COUNT
003 027 302              JNZ       /MORE POINTS?
```

```
003 030 011           NXTPNT    /YES, DO ANOTHER ONE
003 031 003           0
003 032 311           RET       /NO, RETURN
                      *003 100
003 100 000  POINT,   0         /ADDRESS POINTER IS STORED HERE
003 101 000           0         /IN THIS EXAMPLE
003 102 000  COUNT,   0         /POINT COUNT IS STORED HERE
```

If we are to use the computer to continuously display the temperature data and update it as new temperature values are acquired, it should be obvious that the computer cannot also be used to perform the 10-minute time-delay subroutine operations which we reintroduced in Example 2-12.

We will now treat the display steps as the MAIN TASK program and we will use an interrupt timer to signal the end of each 10-minute period. The A/D converter software and the file-updating subroutine (Example 2-13) will be part of the interrupt service subroutine.

Two flowcharts, Figs. 2-15 and 2-16, show the execution of the display and interrupt programs. The interrupt may occur at any time during the execution of the display software, so the flowcharts do not show a direct connection to the interrupt service subroutine, INTSVC.

The main operating program now displays the latest 144 temperature values stored in the data file. The A/D converter acquires new values and the file is updated only when the interrupt timer signals that a 10-minute period has elapsed. The data in the file will be set initially to zero, using the software instructions at CLEAR. This will produce a straight line when the data is displayed by the oscilloscope.

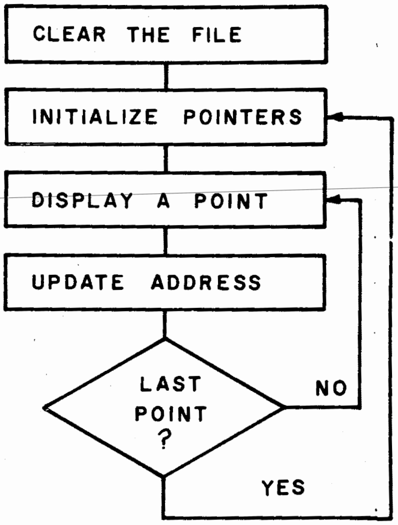


Fig. 2-15. The MAIN TASK program will continuously display all of the 144 temperature values.

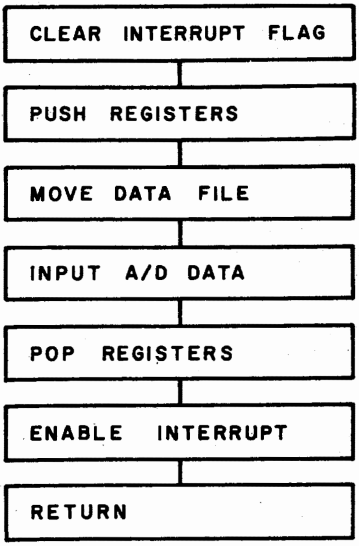


Fig. 2-16. Interrupt-based A/D converter and data-file manipulation program.

In this manner, false temperature readings due to the "random data" found in the computer's read/write memory upon power-up will not occur.

```
                      /EXAMPLE 2-14
                      /THIS IS A COMPLETE INTERRUPT BASED DATA
                      /ACQUISITION PROGRAM WITH THE DISPLAY SOFT-
                      /WARE USED AS THE "MAIN TASK"
                      *003 000
003 000 061  START,   LXISP    /LOAD STACK POINTER
003 001 370           370
003 002 004           004
003 003 373           EI       /ENABLE INTERRUPT
003 004 052           LHLD     /GET THE ADDRESS OF THE DATA FILE
003 005 054           POINT
003 006 003           0
003 007 072           LDA      /GET THE POINT COUNT
003 010 060           COUNT
003 011 003           0
003 012 137           MOVEA    /MOVE IT TO REG E
003 013 257  CLEAR,   XRAA     /CLEAR A
003 014 167           MOVMA    /STORE IT
003 015 043           INXH     /INCREMENT THE POINTER
003 016 167           MOVMA    /DO IT AGAIN
003 017 043           INXH
003 020 035           DCRE     /DECREMENT THE COUNT
003 021 302           JNZ      /IF COUNT IS NOT 0
003 022 013           CLEAR    /CLEAR THE NEXT LOC'N
003 023 003           0
```

```
003 024 052  DISPL,    LHLD      /GET THE FILE ADDRESS
003 025 054            POINT
003 026 003            0
003 027 072            LDA       /GET THE POINT COUNT
003 030 060            COUNT
003 031 003            0
003 032 137            MOVEA     /MOVE IT TO REG E
003 033 176  MORE,     MOVAM     /GET THE 8 LSB'S OF A POINT
003 034 323            OUT       /OUTPUT TO DAC
003 035 054            054
003 036 043            INXH      /INCREMENT THE ADDR POINTER
003 037 176            MOVAM     /GET THE 2 MSB'S OF A POINT
003 040 323            OUT       /OUTPUT TO THE DAC
003 041 055            055
003 042 323            OUT       /STROBE THE DAC
003 043 056            056
003 044 043            INXH      /INCREMENT THE ADDR AGAIN
003 045 035            DCRE      /DECREMENT THE COUNT
003 046 302            JNZ       /MORE POINTS TO BE OUTPUT?
003 047 033            MORE      /YES
003 050 003            0
003 051 303            JMP       /NO, DISPLAY IT ALL AGAIN
003 052 024            DISPL
003 053 003            0

003 054 000  POINT,    000       /LOW ADDRESS OF THE DATA FILE
003 055 020            020       /HIGH ADDRESS OF THE DATA FILE
003 056 036  NEWPNT,   036       /FIRST ADDRESS OF THE LAST
003 057 021            021       /TWO WORD DATA POINT
003 060 220  COUNT,    220       /NUMBER OF DATA POINTS, UP TO
003 061 000            000       /377 = 255 DECIMAL
                                 /220 = 144 DECIMAL

                       /THIS IS THE ADC SUBROUTINE. IT STARTS A
                       /CONVERSION, WAITS FOR THE DONE FLAG AND
                       /STORES THE DATA IN THE LAST LOCATION IN
                       /THE FILE

003 062 323  CONVRT,   OUT       /START A CONVERSION
003 063 037            037
003 064 333  CHK,      IN        /INPUT THE FLAG BIT
003 065 066            066
003 066 306            ADI       /ADD A 1 TO IT
003 067 200            200
003 070 322            JNC       /IF NO CARRY, CHECK AGAIN
003 071 064            CHK
003 072 003            0
003 073 117            MOVCA     /STORE THE 2 MSB'S
003 074 333            IN
003 075 065            065       /INPUT THE 8 LSB'S
003 076 052            LHLD      /LOAD THE ADDRESS OF THE
003 077 056            NEWPNT    /NEW OR LATEST POINT
003 100 003            0
003 101 167            MOVMA     /STORE 8 LSB'S
003 102 043            INXH      /INCREMENT THE ADDRESS
```

```
003 103 161            MOVMC     /STORE 2 MSB'S
003 104 311            RET
                       /THIS SUBROUTINE MOVES THE DATA VALUES
                       /DOWN TWO ADDRESSES IN R/W MEMORY
003 105 052  UPDATE,   LHLD      /GET THE FILE ADDRESS
003 106 054            POINT
003 107 003            0
003 110 072            LDA       /GET THE POINT COUNT
003 111 060            COUNT
003 112 003            0
003 113 137            MOVEA     /STORE IT IN REG E
003 114 043            INXH      /INCREMENT ADDRESS TWICE
003 115 043            INXH
003 116 006  NXTPNT,   MVIB      /SET B= # BYTES PER POINT
003 117 002            002
003 120 116  BYTE2,    MOVCM     /GET THE DATA
003 121 053            DCXH      /DECREMENT THE ADDRESS TWICE
003 122 053            DCXH
003 123 161            MOVMC     /PUT THE DATA THERE
003 124 043            INXH      /MOVE ADDR UP BY THREE
003 125 043            INXH
003 126 043            INXH
003 127 005            DCRB      /ALL BYTES DONE?
003 130 302            JNZ       /NO, DO ANOTHER TRANSFER
003 131 120            BYTE2
003 132 003            0
003 133 035            DCRE      /YES, DECREMENT POINT COUNT
003 134 302            JNZ       /MORE POINTS?
003 135 116            NXTPNT    /YES, DO ANOTHER ONE
003 136 003            0
003 137 311            RET       /NO, RETURN
                       /INTERRUPT SERVICE SUBROUTINE FOR THE REAL
                       /TIME CLOCK.
                       *000 070
000 070 323  INTSVC,   OUT       /CLEAR INTERRUPT FLAG
000 071 173            173
000 072 365            PUSHPSW   /SAVE FLAG AND REGISTERS
000 073 305            PUSHB
000 074 325            PUSHD
000 075 345            PUSHH
000 076 315            CALL      /MOVE THE DATA DOWN A POINT
000 077 105            UPDATE
000 100 003            0
000 101 315            CALL      /DO A CONVERSION
000 102 062            CONVRT
000 103 003            0
000 104 341            POPH      /RESTORE THE REGISTERS
000 105 321            POPD
000 106 301            POPB
000 107 361            POPPSW
000 110 373            EI        /RE-ENABLE INTERRUPT
000 111 311            RET       /RETURN TO THE MAIN TASK
```

The time spent servicing the interrupting device, the A/D converter in this case, will be short since a fast, successive-approximation converter is used. For other converters that may take longer, say a digital panel meter which may take 0.1 second, the time spent in the interrupt service portion of the program becomes long. To avoid spending long periods of time in the interrupt service software, and thus away from the MAIN TASK program, a multilevel or priority interrupt scheme may be used.

When using a converter or other interrupting device that has a long response time, two interrupts may be used. In the case of the slow A/D converter, one interrupt would be caused by the clock and the other would be caused by the A/D converter when it finished a conversion. The interrupt caused by the clock at 10-minute intervals would *vector* or point the computer to a subroutine which would only start the A/D converter, clear the clock's interrupt flag, re-enable the interrupt and then return to the MAIN TASK which displays the data in the file. When the slow A/D converter completed a conversion, it would interrupt the 8080, but with a different restart instruction, say RST6. The interrupt service subroutine pointed to by the RST6 instruction starts at address 000 060. This subroutine is used to update the data file, input the data from the A/D converter, and then store the data in the file. In this way the computer can effectively use the time during which the conversion is being performed for other tasks. The only assumption that we have made is that the converter will have finished its current conversion before the next conversion is requested by the interrupting time. Thus, we could not use a converter with a conversion time of 0.1 second with a data acquisition system that will require more than 10 points per second. The greenhouse data acquisition system will have 10-minute periods between conversions so even a very slow A/D converter could be used. The use of interrupts has been covered in greater detail in Unit No. 23 of *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 2,* published by Howard W. Sams & Co., Inc.

A circuit for two interrupts is shown in Fig. 2-17. The clock will generate an interrupt and the RST7 instruction while the A/D converter's DONE/$\overline{\text{BUSY}}$ flag will generate an interrupt and the RST6 instruction. Remember that the RST7 instruction is a 377 and the RST6 instruction is a 367. This scheme is *not* a priority interrupt. We have assumed that the clock and the A/D converter will not interrupt at the same time.

The software necessary to implement the above multi-interrupt system is shown in Example 2-15. The display software in the MAIN TASK program and the UPDATE subroutines are not affected by the change to a two interrupt system. The clock service routine, CLKSVC, replaces the interrupt service subroutine, INTSVC, shown in Example 2-14. The A/D converter's interrupt service subroutine, ADSVC; is new. The old A/D converter program, CONVRT, is not used. You should note that the new A/D converter subroutine, ADSVC, does not contain any flag checking steps. Why? The A/D converter's DONE flag will now generate the interrupt restart instruction, RST6, so that the computer vectors directly to the ADSVC subroutine when the conversion is complete.

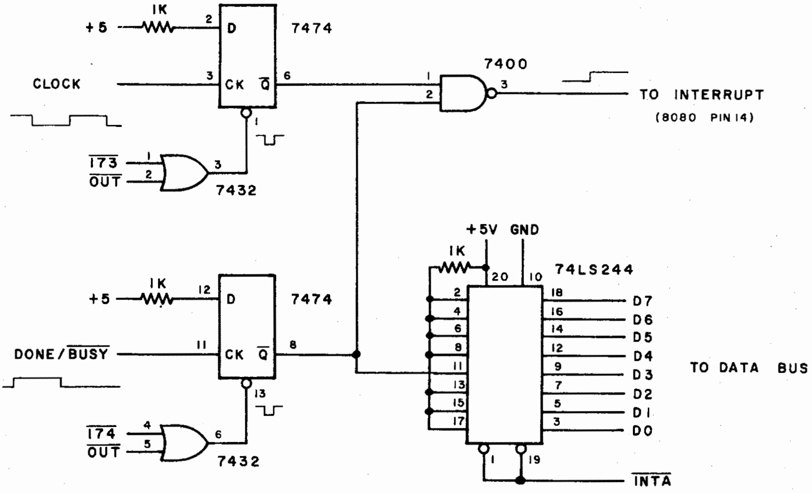


Fig. 2-17. Simple interrupt scheme for two interrupting devices, one generating a RST6 instruction (367) and the other generating a RST7 instruction (377). The interrupts cannot be simultaneous for proper operation of this circuit.

```
                       /EXAMPLE 2-15
                       /INTERRUPT SUBROUTINES FOR CLOCK AND
                       /CONVERTER
                       *000 060
000 060 303  ADSVC,    JMP      /NOT ENOUGH ROOM HERE, SO GO TO
000 061 100            AHEAD    /A HIGHER ADDRESS SO WE DON'T RUN INTO
000 062 000            0        /ADDRESS 000 070, THE OTHER INTERRUPT
                                /VECTOR ADDRESS
                       *000 070
000 070 323  CLKSVC,   OUT      /CLEAR THE CLOCK'S INTERRUPT FLAG
000 071 173            173
000 072 323            OUT      /START A CONVERSION
000 073 037            037
000 074 373            EI       /RE-ENABLE THE INTERRUPT
000 075 311            RET      /RETURN TO MAIN PROGRAM
                       *000 100
000 100 323  AHEAD,    OUT      /CLEAR THE CONVERTER'S INTERRUPT
000 101 174            174      /FLAG
```

```
000 102 305        PUSHB     /SAVE REGISTERS
000 103 325        PUSHD
000 104 345        NOP
000 105 000        NOP
000 106 000        PUSHH     /SAVE H & L
000 107 365        PUSHPSW   /SAVE A & FLAGS
000 110 315        CALL      /MOVE THE DATA DOWN ONE POINT
000 111 105        UPDATE    /OR TWO LOCATIONS TO MAKE ROOM
000 112 003        0         /FOR THE NEW POINT
000 113 052        LHLD      /GET MEMORY POINTER FOR NEW
000 114 056        NEWPNT    /DATA POINTS
000 115 003        0
000 116 333        IN        /INPUT 8 LSB'S
000 117 065        065
000 120 167        MOVMA     /STORE THEM
000 121 043        INXH
000 122 333        IN        /INPUT 2 MSB'S
000 123 066        066
000 124 167        MOVMA     /STORE THEM, TOO.
000 125 361        POPPSW    /RESTORE REGISTERS
000 126 341        POPH
000 127 321        POPD
000 130 301        POPB
000 131 373        EI        /RE-ENABLE INTERRUPT
000 132 311        RET
```

If you look carefully at the programs provided in Example 2-14 and Example 2-15, you will see that good use has been made of the 8080's SHLD, LHLD, and STA instructions. These instructions have allowed us to store address and counter information in read/write memory in a set of locations which are accessed by the programs. The address of the latest temperature point, for example, is stored in location NEWPNT and in the next location as well. *Remember that an address is 16 bits, requiring two 8-bit bytes of storage.* When using an address storage technique such as this, it becomes easy to change the starting address of the data file stored at POINT, and the number of points in the data file, the count being stored at COUNT. We must, however, calculate the addresses of the last two locations in the data file which are to be used to store the newest data point. Keep in mind that the address of the last point will not vary once it is determined, since the latest data will always be stored in the same place in the file: the last two locations.

In many applications data storage needs change, so the starting addresses of the data file, the number of points in the data file, and the addresses of the locations to be used for storage of the last data point will also change. Instead of calculating and recalculating these addresses each time that the program is used in a different application, we can rely upon the computer to do this for us.

In the next software example, Example 2-16, the software at START (from Example 2-14) has been changed so that the program will now calculate the addresses of the last data point (two locations) once the starting address of the data file and the number of data points to be acquired are stored in POINT (two address bytes) and COUNT (one count byte). These values could be input through front panel controls, a teletypewriter, or they could be preset by the programmer. Since the remainder of the software from Example 2-14 will remain the same, only the changes between addresses START and CLEAR have been shown in Example 2-16. Remember that if the starting address of the file and the number of data points to be acquired will be changed, the memory area used to store the values at POINT, NEWPNT, and COUNT must be read/write memory.

Some of the software steps shown in Example 2-16 could have been combined into other subroutines, but we have avoided doing this to keep you from having to "jump" back and forth between various segments of the program as you follow through the steps in the example. We hope that you will disect these programs and attempt other software solutions which may be more efficient or applicable to your specific needs.

```
                          /EXAMPLE 2-16
                          /SOFTWARE TO INITIALIZE ADDRESS AND COUNT
                          /(MAY BE USED WITH EXAMPLE 2-14)
                          /THIS ASSUMES THAT TWO BYTES ARE USED
                          /FOR EACH DATA VALUE
                          *003 000
003 000 061    START,     LXISP     /LOAD STACK POINTER
003 001 370               370
003 002 004               004
003 003 373               EI        /ENABLE INTERRUPT
003 004 052               LHLD      /NOW, CALCULATE LAST POINT'S ADDR
003 005 060               COUNT     /GET THE COUNT
003 006 003               0
003 007 046               MVIH      /SET REG H TO ZERO
003 010 000               000
003 011 051               DADH      /ADD IT TO ITSELF (DOUBLE IT)
003 012 353               XCHG      /PUT DOUBLED VALUE IN D & E
003 013 052               LHLD      /GET STARTING ADDRESS OF THE FILE
003 014 054               POINT
003 015 003               0
003 016 031               DADD      /ADD THE ADDRESS COUNT
003 017 053               DCXH      /SUBTRACT 2
003 020 053               DCXH
003 021 042               SHLD      /SAVE ADDRESS OF LAST POINT IN
003 022 056               NEWPNT    /NEWPNT
003 023 003               0
003 024 052               LHLD      /GET ADDR OF DATA FILE
003 025 054               POINT
```

```
003 026 003            0
003 027 072            LDA        /GET THE POINT COUNT
003 030 060            COUNT
003 031 003            0
003 032 137            MOVEA
003 033 257  CLEAR,    XRAA       /AND SO ON FROM HERE ON OUT
                       *003 054
003 054 000  POINT,    0          /IF YOU EXPECT TO MAKE CHANGES, AND YOU
003 055 000            0          /WISH TO USE THIS SOFTWARE, THESE
003 056 000  NEWPNT,   0          /LOCATIONS MUST BE IN READ/WRITE
003 057 000            0          /MEMORY
003 060 000  COUNT,    0
```

The software examples provided for the greenhouse temperature data-acquisition application have all assumed a maximum count of 256 data points. This has simplified the program since only single-precision or one-byte math operations have been required to count the points. If you wish to work with larger files of data, the programs will have to be modified.

CLOSING THE LOOP—CONTROL APPLICATIONS

The greenhouse temperature measurement example illustrates how a small computer may be used in an *open-loop* application, or one that does not require any control. The data and control signals flow in one direction; to the computer. We would now like to explore the possibility of closing the loop and allowing the computer to perform some type of control function to keep the temperature within certain

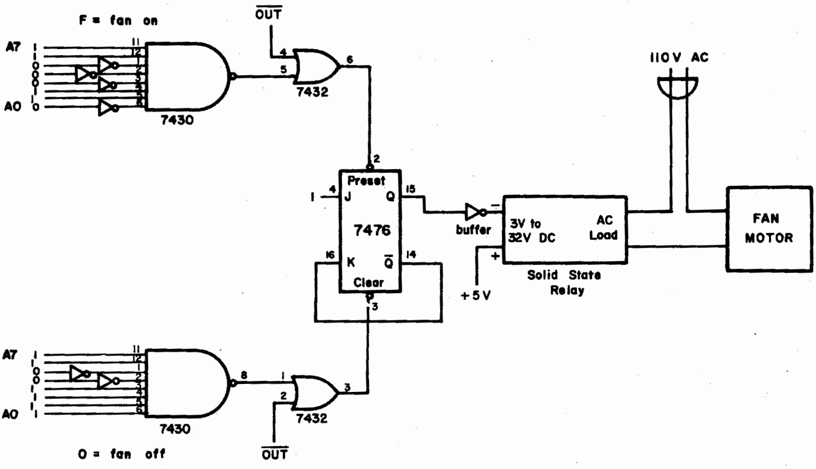


Fig. 2-18. The use of a solid-state relay to control a fan under program control. The OUT 306 command turns the fan on while the OUT 317 command turns off.

limits. A fan for cooling and a heater are available and each may be controlled by the computer. A typical control interface is shown in Fig. 2-18 where a solid-state relay is used to control the fan. A similar interface would be used to control the heater.

The control program could be written as a subroutine, which might be added to the A/D converter's interrupt service subroutine, INTSVC, right after the CONVRT subroutine is called. Thus, the control operation would be performed based upon the newest temperature value. We will add a simple control subroutine with arbitrary values of 001 035 for the low set point and 002 310 for the high set point. Remember that these are both octal equivalents of the 10-bit values 0100011101 and 1011001000, respectively. Depending upon the temperature sensor used, these values could correspond to the temperature values of 15°C (59°F) and 25°C (77°F).

```
                        /EXAMPLE 2-17
                        /SUBROUTINE FOR HEATER/FAN CONTROL
                        *003 200
003 200 323   STPNT,    OUT       /FAN OFF
003 201 317             317
003 202 323             OUT       /HEATER OFF
003 203 320             320
003 204 052             LHLD      /GET THE HIGH 10-BIT VALUE
003 205 251             HIVAL
003 206 003             0
003 207 174             MOVAH     /THESE STEPS FORM ITS NEGATIVE
003 210 057             CMA       /VALUE
003 311 127             MOVDA
003 212 175             MOVAL
003 213 057             CMA
003 214 137             MOVEA
003 215 023             INXD
003 216 052             LHLD      /GET THE LOW 10-BIT VALUE
003 217 253             LOVAL
003 220 003             0
003 221 174             MOVAH     /THESE STEPS FORM ITS NEGATIVE
003 222 057             CMA       /VALUE, LESS 1
003 223 107             MOVBA
003 224 175             MOVAL
003 225 057             CMA
003 226 117             MOVCA
003 227 052             LHLD      /GET THE NEW DATA VALUE
003 230 255             NEWPNT
003 231 003             0
003 232 345             PUSHH     /STORE IT
033 233 031             DADD      /ADD D&E TO H&L
003 234 322             JNC       /IF NO CARRY, CHECK FOR A TOO LOW
003 235 243             TOLOW     /VALUE
003 236 003             0
003 237 323             OUT       /CARRY, MUST BE TOO HIGH, SO
003 240 306             306       /FAN ON
```

```
003 241 341          POPH      /GET H OFF THE STACK, DON'T USE IT
003 242 311          RET
003 243 341 TOLOW,   POPH      /GET H BACK TO TEST VALUE
003 244 011          DADB
003 245 330          RC        /IF THERE IS A CARRY, OK
003 246 323          OUT       /NO CARRY, TOO LOW
003 247 307          307       /HEATER ON
003 250 311          RET

003 251 310 HIVAL,   310       /LOW 8 BITS OF HIGH SET POINT
003 252 002          002       /HIGH 2 BITS OF  "    "    "
003 253 035 LOVAL,   035       /LOW 8 BITS OF LOW SET POINT
003 254 001          001       /HIGH 2 BITS OF  "    "    "
003 255 000 NEWPNT,  0
003 256 000          0
```

The control subroutine, STPNT, will perform the comparisons and either turn the fan or heater on or off as required. Each device could be turned on for a preset period or they could be turned on and left on until the next temperature reading is taken. The actual method used would depend upon the size of the greenhouse. A small greenhouse might overheat or overcool if either device is left on for the entire 10-minute sample period. The program in Example 2-17 has assumed a large greenhouse, so the fan or the heater will remain on for the entire 10-minute sample interval.

While this may seem like a simple example, since the temperature could just as easily be controlled by a thermostat, the display and control program can be very useful. The temperature value for each point could be integrated to provide a value for degree days, an indication of the heat produced by the sun on a given day. With the current emphasis on solar heating and cooling, this type of computer system could be used for solar collector positioning and pump sequencing control, as well as for data acquisition. A report from the Copper Development Association mentions the use of a Tektronix Model 31 programmable calculator for just this purpose.*

* *CDA Decade 80 Solar House, Application Data Sheet,* Cooper Development Association, Inc., 405 Lexington Avenue, New York 10017, 1977.

3

Dual-Slope Analog-to-Digital Converters and Digital Panel Meters

INTRODUCTION TO THIS UNIT

The low cost of digital panel meters (DPMs) is making them an attractive means of interfacing analog signals to small computers. They provide a direct readout of the units measured and many models have binary-coded decimal (BCD) outputs, which facilitate interfacing. The *dual-slope integrating analog-to-digital conversion technique* is generally used in digital panel meters. Dual-slope A/D converters are also available in a variety of prepackaged modules. The dual-slope conversion technique has some advantages over the ramp and successive-approximation conversion techniques that make it attractive in some applications.

DUAL-SLOPE ANALOG-TO-DIGITAL CONVERTERS

The dual-slope conversion method operates upon the principle of indirectly measuring an unknown voltage by converting it to a time period. A block diagram of a typical dual-slope converter is shown in Fig. 3-1.

When a conversion is to be performed, the dual-slope converter connects the unknown voltage to an integrator circuit through an

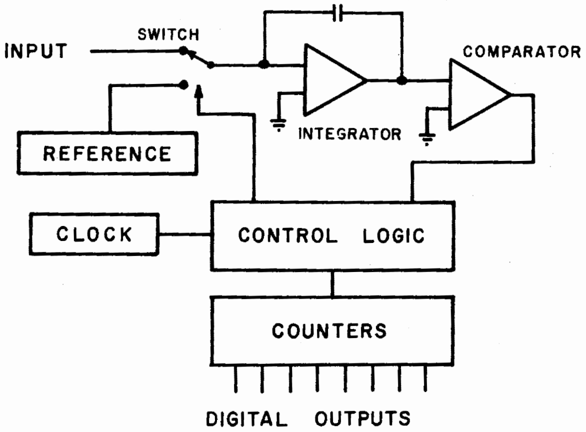


Fig. 3-1. Block diagram for a typical dual-slope, analog-to-digital converter with 8-bit output.

electronic switch. The control logic allows the integrator to integrate, or sum, the voltage for a fixed period, T1. At the end of this period, the integrator's input is switched by the control logic to a stable reference voltage with a polarity that is opposite to that of the unknown signal being measured. Since the input to the integrator is now negative, this has the effect of subtracting from the sum accumulated during the integration period, T1.

Since the reference potential is known and stable, the *slope* of the integrator's output will be constant with respect to time. The dual-slope converter measures the time period, T2, that is necessary for the integrator's output to reach zero volts. These timing relationships are shown in Fig. 3-2. The timing for two different voltages, A and B, is shown.

When a higher input voltage, A, is applied to the integrator the sum accumulated at the end of the integration period, T1, is also

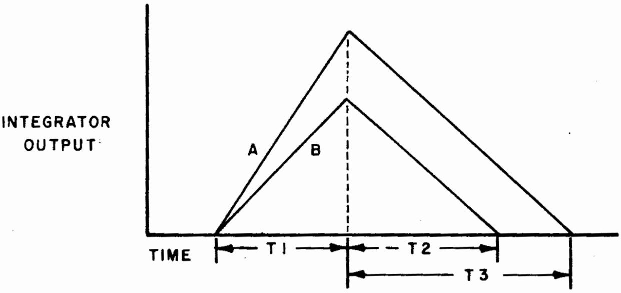


Fig. 3-2. Timing diagram for a dual-slope analog-to-digital converter.

larger. Since the reference potential remains the same, and thus the "discharge" slope remains the same, the time necessary for the integrator's output to reach zero increases to T3. Thus, higher unknown voltages give proportionally longer discharge periods, an indirect measurement of the unknown voltage applied to the converter.

During the discharge period, the clock signal is gated through the control logic to the counters. The period, T2, is measured as the number of pulses counted at a rate of, for example, 100 pulses per volt. The accumulated count then becomes an indirect measure of the voltage applied to the dual-slope converter. For a longer discharge period, T3, the pulse gate is open for a longer period and more pulses are counted, indicating a higher unknown voltage. The clock rate and the control logic are set so that the voltage applied as an input to the converter and the clock pulses gated through the control logic are proportional.

The counters may be decimal ones for use in a digital panel meter (DPM), or they may be binary devices for computer and control applications where a BCD readout is not required.

The dual-slope conversion technique is analogous to measuring the flow rate of water coming out of a pipe by filling a bucket for 60 seconds. The volume of water collected is an indication of the flow rate, but instead of measuring this directly, the bucket is emptied at a fixed rate of, say, one liter per second. The flow rate is then indirectly found by using the following formula:

$$\text{Unknown Flow Rate} = \frac{\text{Emptying time}}{\text{60 seconds}} \times 1 \text{ liter/second}$$

For voltages this converts to:

$$E_{\text{in}} = \frac{\text{T2}}{\text{T1}} \times (V_{\text{ref}})$$

The T1 period is the integration time and the T2 period is the discharge period.

Most A/D converter users will purchase dual-slope converters rather than attempt to construct them themselves. Some representative samples are:

Analog Devices, Inc., Norwood, MA 02062		
ADC 1100	3½ digits Range of ±0.1999 volt	42-millisecond conversion time
ADC 141I	14 bits Range of ±10 volts	40-millisecond conversion time
Analogic Corporation, Wakefield, MA 01880		
AN 2313	10 bits	6.7-millisecond conversion time
AN 2317	14 bits	67-millisecond conversion time
	Both devices have a range of ±2 volts.	

Datel Systems, Inc., Canton, MA 02021
ADC E10B 10 bits 1.25-millisecond conversion time
Ranges of ±1, ±5 and ±10 volts

Dual-slope A/D converters can take a long time to perform a single conversion since the unknown and reference integration periods may be long. Some of the less expensive 10- and 12-bit models have conversion times of from 5 to 10 milliseconds, while higher resolution units can perform conversions every 100 to 120 milliseconds. In spite of their relatively slow speed, dual-slope A/D converters can be very accurate.

By integrating an unknown voltage over a specific period of time, random and periodic noise may be integrated, or summed, to zero. In other words, the noise is averaged. For example, consider the measurement of the signal, shown in Fig. 3-3, at point *X*. The voltage is supposed to be a fairly steady dc level, but superimposed upon it is some 60-Hz *ripple* or noise.

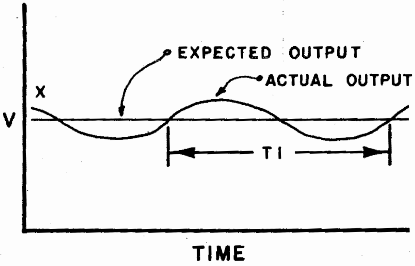


Fig. 3-3. Diagram of an expected signal output and the actual signal output, showing 60-Hz noise superimposed upon the signal.

If a fast, successive-approximation converter is triggered to start a conversion at point *X*, the digitized value will be too high. A dual-slope analog-to-digital converter will be quite insensitive to the 60-Hz noise if the integration period, T1, is an integer multiple of the 60-Hz period of 16.67 milliseconds.

The 60-Hz noise does not affect the dual-slope converter's final digitized value since the average of the input signal is what is actually converted, as shown in Fig. 3-4. The integrated value, or the area under the curve shown in Fig. 3-4, is equal to the area under the straight, dashed line since the 60-Hz noise both adds to and subtracts from the accumulating total. In the same way, random noise is averaged, decreasing the possibility of sampling and converting a signal during a noise spike. This is called *normal-mode noise rejection,* which is defined as the elimination of noise superimposed upon a single-ended or one-wire signal.

Many dual-slope A/D converters are set up for 60-Hz normal-mode noise rejection of about 40 dB. Models are also available for

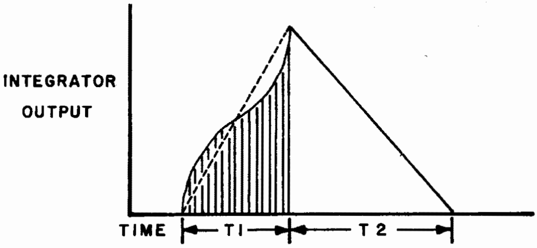


Fig. 3-4. Integration of a signal with 60-Hz noise superimposed upon it. Integration is performed for one 16.66-millisecond period.

50-Hz normal-mode noise rejection, 50 Hz being the line frequency used elsewhere in the world.

Dual-slope A/D converters are easily interfaced to microcomputers using the same techniques that were developed for interfacing successive-approximation converter modules. The control and data signals are so similar that there is no need to cover this subject further. The only real difference between dual-slope and successive-approximation converters is in the conversion technique used. Our main interest in dual-slope converters concerns their use in digital panel meters.

DIGITAL PANEL METERS

Digital panel meters (DPMs) are analog-to-digital converters that are slightly different from those discussed previously, since a display is an integral part of the converter. Most digital panel meters are binary-coded-decimal devices. The use of a DPM is frequently preferable to the use of another type of A/D converter, particularly in those applications where a visual, *decimal* indication of voltage, pressure, temperature, or some other variable is needed; a numeric readout is part of all digital panel meters.

Many DPMs are used to replace magnetic movement or analog meters of one type or another. As such, they do not necessarily lend themselves to computer interfacing, since the BCD code is used and parallel digit outputs may not be available. Some DPMs have features which make it possible to interface them to computer systems. The basic requirements for DPM interfacing are:

- Parallel digit outputs in transistor-transistor logic (TTL) compatible form.
- A status signal to indicate the end of an A/D conversion, also in TTL-compatible form.

Most digital panel meters, unlike other A/D converters, are free-running devices. This means that they are constantly performing conversions to update the displays. These conversions are controlled by an internal clock, but many DPMs have an external connection so that interfaces and other devices may command the DPM to perform a conversion. Since the parallel digit outputs from the DPM will be in binary-coded-decimal (BCD) form, a quick review of the BCD coding system is provided in the next section. If you are already familiar with the BCD numbering system, you may wish to skip the next section. This is only a *review* of the BCD numbering system and not a detailed examination.

BINARY-CODED-DECIMAL NUMBERING

In the binary-coded-decimal numbering system, the decimal grouping of the digits is retained, but each digit is converted into its equivalent binary number without regard to its actual decimal position within the number. In this way, a seven is always indicated by 0111_2, whether it is the seven in 107 or the seven in the number 7325. The range of digits is always limited to zero through nine.

The binary-coded-decimal numbering is easily performed with the aid of Table 3-1. After you perform a few conversions, you should be able to do additional conversions without the aid of the table.

Table 3-1. Binary Coded Decimal Numbering

Decimal Digit	Binary Code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Two examples of decimal-to-BCD conversions are provided:

Number to be converted = 7325

0111 = 7   0011 = 3   0010 = 2   0101 = 5

Number to be converted = 8039

1000 = 8   0000 = 0   0011 = 3   1001 = 9

It is important for you to realize that since each decimal digit value zero through nine may be represented by four binary bits, an 8-bit microcomputer could store two BCD digits in each 8-bit memory location. This is very useful since it will have the storage space needed if we were to assign each digit its own memory location. When two BCD digits are stored together in this way we call the data *packed BCD data.* Software is most often used to *pack* and *unpack* the data.

DIGITAL PANEL METER CONSIDERATIONS

Most of the general-purpose digital panel meters are 3½- or 4½-digit devices. The one-half digit indicates that the most significant digit may be either a zero or a one. Thus, for a 3½-digit DPM, the range would be from 0000 to 1999. Some DPMs have a *unipolar* range of from zero to their full scale, say, +1.999 volts, while other DPMs have a *bipolar* range which goes from a negative full scale value to a positive full scale value, for example, ±1.999 volts.

Other digital panel meters may be programmed to have various ranges with a resistor or external digital connections being used to determine the range. Some of the newer digital panel meters are *autoranging.* These DPMs automatically select the correct input range, and can convert analog signals in the ranges of ±19.99 volts, ±1.999 volts, and ±199.9 millivolts. The autoranging DPMs are slightly more expensive than the fixed range units, but they are particularly useful in applications where signals may vary over several voltage decades.

Digital panel meters are generally thought of as having long conversion times when compared to the fast successive-approximation and ramp A/D converters. Most DPMs will take several milliseconds to perform a conversion and repetition rates range from about 2 to about 20 conversions per second. For some systems this may be considered slow, but in many other systems where high-speed A/D conversions are not necessary, DPMs are widely used. In our greenhouse temperature measurement example, a 3½-digit DPM could have been used just as successfully as the 10-bit successive-approximation A/D converter. Samples were taken every 10 minutes, and it is doubtful that the temperature would undergo quick changes during the multimillisecond conversion time required by a DPM.

We can summarize this section by listing the important attributes of digital panel meters:

- Parallel BCD outputs, TTL-compatible (optional on some units)
- Available in various ranges, some of which are programmable
- End-of-conversion (EOC) flag output
- External conversion start input (optional on some units)
- Slow; 2 to 20 conversions per second
- Digital display of data in volts or other engineering units
- Low cost A/D converter with built-in analog signal conditioning

DIGITAL PANEL METER INTERFACING

Digital panel meters are interfaced to computers in a manner which is very similar to that used in the previous A/D converter interfacing examples. The outputs from the DPM are available in parallel, TTL-compatible form and the DPM provides an end-of-conversion flag output to indicate that the parallel digital outputs are valid and that they may be sampled by the computer. Depending upon the DPM being interfaced, either the internal DPM logic or the computer will provide the pulse to initiate a conversion.

In some DPMs, however, there are other signals that must be connected as well. Some of these signals may be hard wired to either a logic one or a logic zero in order to select a range, program a decimal point on the display, etc. These inputs do not require any further action and it is doubtful that they would ever be changed once they are hard wired in a system.

There may be other important signals output by the DPM to indicate range, overflow, polarity, etc. These signals are not generally found in small modular converters, so we have not discussed them in any of the previous A/D converter interfacing examples.

We will provide two examples of digital panel meter interfacing; (1) a simple interface using a standard DPM with a single range, and (2) a more complex example in which an autoranging DPM is used. These interfaces are not limited to DPMs alone, since other BCD data sources may be interfaced in a similar manner. Frequency meters, clocks, event counters, and other devices, including analytical instruments such as spectrometers and chromatographs, fall into this category.

Digital Panel Meter Interfacing Example No. 1

In this example, a digital panel meter with a single bipolar range of ±1.999 volts will be interfaced to an 8080-based microcomputer. The DPM to be used in this example is an Analog Devices, Inc., AD2009 3½-digit DPM.

The software to acquire data from the AD2009 will be written as a general-purpose subroutine that may be used by other programs. The hardware will be constructed such that each BCD digit will have its own three-state input port. The AD2009 may be operated in either an externally triggered or an internally triggered mode. We have chosen to use the internal trigger, which will perform six conversions per second (or one conversion every 167 milliseconds). The external trigger may be useful when it is necessary to have the DPM perform a conversion upon command of the computer.

With the aid of the internal trigger, the software will be able to input the BCD data from either the conversion that has just taken place or from the conversion currently taking place. Thus, if the computer happens to request data from the DPM just after a conversion has been completed, the data will be available and input into the computer. If, however, the computer requests data while the converter section of the DPM is processing the analog signal, the computer must wait until the end of the current conversion before the new data will be available. This will give us a maximum uncertainty of 167 milliseconds between data points. In most slow applications, this is acceptable.

Interfacing the AD2009—The AD2009 digital panel meter has a number of input and output lines that are compatible with the standard TTL levels present in the 8080-based computer that we are using. Some of the DPMs signals do not need to be controlled by the computer so they may be hard wired to either a logic zero or a logic one, depending upon the contrôl needed. The important interfacing signals for the AD2009 are as follows:

Digital Outputs		
BCD Data		12 lines
OVERRANGE	(Most significant digit)	1 line
	Logic 1 = 0*nnn* volts	
	Logic 0 = 1*nnn* volts	
	n = 0 through 9	
OVERLOAD	Logic 1 = In range	1 line
	Logic 0 = Out of range	
POLARITY	Logic 1 = Positive voltage input	1 line
	Logic 0 = Negative voltage input	
STATUS	Logic 1 = Converting (Busy)	1 line
	Logic 0 = Done	
Digital Inputs		
HOLD	Logic 1 = Normal conversion mode	1 line
	Logic 0 = Hold data, disable trigger	

The HOLD input is very useful in this application, since it allows the computer to disable the conversion process so that the data may be read under program control. When the DPMs HOLD input is asserted at logic zero, no further conversions can take place and the data outputs are held steady or latched so that they may be read.

The interface for the AD2009 has been constructed using four 4-bit, three-state input ports. The three BCD digits are input one at a time under software control. The four status signals, OVERRANGE, OVERFLOW, POLARITY, and STATUS are input through a separate 4-bit, three-state input port. Even though the 8080 is an 8-bit computer, it can still easily handle smaller 4-bit bytes. While it may appear inefficient to interface the DPM in this way, it is easy to do and it costs about the same as two 8-bit input ports. You will notice that a flip-flop is included in the interface to control the state of the DPMs HOLD input. An OUT 061 instruction forces the DPM into the HOLD state while an OUT 060 instruction removes the HOLD signal and allows the DPM to function normally. A complete schematic of the DPM interface is shown in Fig. 3-5. The pinout of only one of the 8095 three-state buffers is shown in the diagram. The remaining devices have exactly corresponding pin configurations.

Software for the AD2009—Software to control the AD2009 digital panel meter interface is probably the most important part of the overall scheme. The following requirements must be met by the general-purpose DPM interface control subroutine:

1. Input the data from the DPM.
2. Store the data in five successive memory locations:

Location	*Data*
n	Polarity bit (Sign)
n+1	Overrange Bit
n+2	100s
n+3	10s
n+4	1s

3. If an OVERLOAD condition exists, exit from the subroutine back to the main program with the CARRY bit set to a logic one, otherwise return with the data stored in read/write memory and the CARRY bit equal to a logic zero.

The polarity bit and the overrange bit will be stored in bit position D0 in their respective memory locations. The remaining seven bits in each location will not be used. Remember that the carry bit is an internal 8080 flag, generally associated with the accumulator or A register.

It will be the responsibility of the main program that calls the DPM subroutine to check the carry flag for an overload condition. Since the decimal point is understood to be between the hundreds and the thousands digits, there is no need to store a character to represent the decimal point.

The DPM subroutine software would have to be modified to detect and to act upon an overload condition if the subroutine is to be used under interrupt control. Since interrupts may occur at any time, we could not be sure that the currently operating software would be able to act upon an overload condition. Remember, too, that if the carry flag is changed during the subroutine and an interrupt service subroutine returns to the main program with the carry, or other registers, in the wrong state, the effect upon the program may be disastrous.

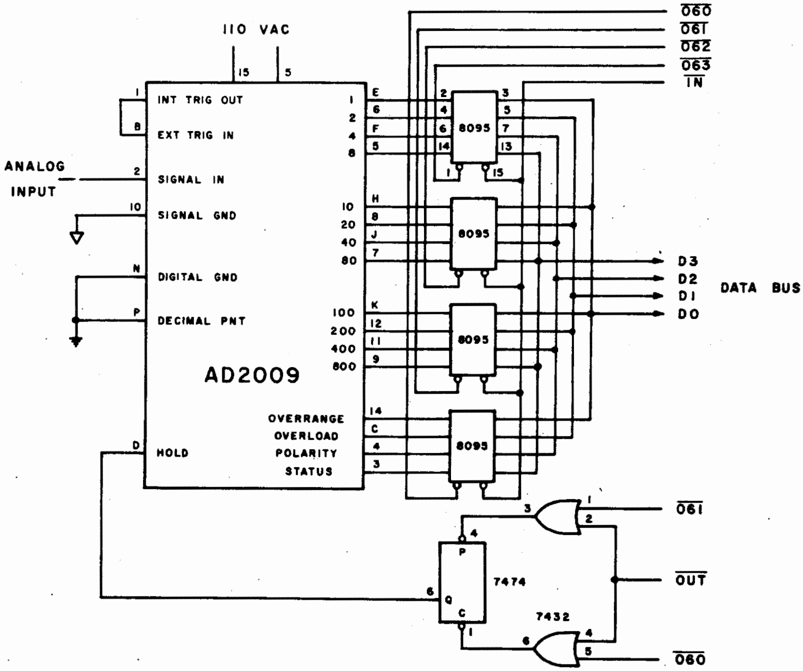


Fig. 3-5. The complete interface for the Analog Devices AD2009 digital panel meter.

Most DPM applications will be slow enough not to warrant the use of an interrupt.

The complete DPM subroutine is shown in Example 3-1. The subroutine monitors the STATUS line to determine that the DPM has completed its current conversion. Once the end-of-conversion condition is sensed, the HOLD line to the DPM is asserted and the computer inputs the BCD data. The computer program first checks the OVERLOAD input since an overload condition will mean simply setting the carry flag to a logic one and returning to the main program. Only if an overload condition is not found will the computer progress to the data input and storage steps.

```
                         /EXAMPLE 3-1
                         /SIMPLE DPM INTERFACE CONTROLLER SUBROUTINE
                         *030 000
030 000 345   DPM,       PUSHH      /SAVE REGISTERS ON THE STACK
030 001 305              PUSHB
030 002 041              LXIH       /POINT TO STORAGE AREA
030 003 000              STORE
030 004 040              0
030 005 016              MVIC       /SET UP A MASK FOR THE 4 LSB'S
```

```
030 006 017           017     /MASK=00001111
030 007 333   TEST,   IN      /INPUT AND TEST STATUS BIT
030 010 060           060
030 011 346           ANI     /CHECK STATUS
030 012 010           010
030 013 302           JNZ     /IF NOT DONE, TEST AGAIN
030 014 007           TEST
030 015 030           0
030 016 323           OUT     /DONE, PUT DPM IN HOLD MODE
030 017 060           060
030 020 333           IN      /INPUT UPDATED STATUS
030 021 060           060
030 022 107           MOVBA   /STORE IT
030 023 346           ANI     /TEST FOR OVERLOAD
030 024 002           002
030 025 302           JNZ     /IF NO OVERLOAD, GO ON TO OK
030 026 034           OK
030 027 030           0
030 030 301           POPB    /IF OVERLOAD, RESTORE REGISTERS
030 031 341           POPH
030 032 067           STC     /SET THE CARRY AND
030 033 311           RET     /RETURN
030 034 170   OK,     MOVAB   /GET THE STATUS WORD
030 035 037           RAR     /ROTATE POLARITY BIT TO D0
030 036 037           RAR
030 037 346           ANI     /MASK OUT ALL ELSE
030 040 001           001
030 041 167           MOVMA   /STORE IT IN MEMORY
030 042 043           INXH    /INCREMENT MEMORY POINTER
030 043 170           MOVAB   /GET STATUS AGAIN
030 044 346           ANI     /MASK OUT ALL BUT OVERRANGE BIT
030 045 001           001
030 046 167           MOVMA   /STORE IT
030 047 043           INXH
030 050 333           IN      /INPUT NEXT DIGIT
030 051 061           061
030 052 241           ANAC    /MASK OUT UNWANTED BITS
030 053 167           MOVMA   /STORE IT
030 054 043           INXH
030 055 333           IN      /DO IT FOR NEXT DIGIT
030 056 062           062
030 057 241           ANAC
030 060 167           MOVMA
030 061 043           INXH
030 062 333           IN      /DO IT FOR LAST DIGIT
030 063 063           063
030 064 241           ANAC
030 065 167           MOVMA
030 066 323           OUT     /CLEAR THE HOLD MODE
030 067 061           061
030 070 067           STC     /CLEAR THE CARRY WITH A SET
030 071 077           CMC     /AND COMPLEMENT
030 072 301           POPB    /RESTORE REGISTERS
030 073 341           POPH
```

```
030 074 311                RET
                           /DPM DATA STORAGE AREA. THIS MUST BE IN
                           /A READ/WRITE MEMORY AREA
                           *040 000
040 000 000  STORE,        0         /SIGN STORED HERE
040 001 000                0         /1000'S (OVERRANGE BIT)
040 002 000                0         /100'S
040 003 000                0         /10'S
040 004 000                0         /1'S
```

Digital Panel Meter Interfacing Example No. 2

In this example we have chosen to use an automatic ranging or autoranging digital panel meter with three measurement ranges of ±19.99 volts, ±1.999 volts, and ±0.1999 volt (±199.9 millivolts). The Datel Systems, Inc., Model DM2000AR will answer these needs.

The following digital outputs and inputs are provided for interfacing or other external use:

Digital Outputs

BCD Data (Three digits)		12 lines
"½" Digit Data		1 line
POLARITY	Logic 1 = Positive voltage input	1 line
	Logic 0 = Negative voltage input	
OVERFLOW	Logic 1 = In range	1 line
	Logic 0 = Out of range	
DONE	Logic 1 = Converting (Busy)	1 line
	Logic 0 = Done	
DECIMAL POINT		3 lines
Range = 19.99 volts, logic 0 at pin A6		
Range = 1.999 volts, logic 0 at pin A5		
Range = .1999 volt, logic 0 at pin A7		

Note: Only one signal is at logic zero at one time.

Digital Inputs

MODE RANGE	Logic 1 = External range	1 line
	Logic 0 = Autorange	
R1 and R2, External Range Control		2 lines
Range = 19.99 volts, R1 = 0, R0 = 1		
Range = 1.999 volts, R1 = 1, R0 = 0		
Range = .1999 volt, R1 = 0, R0 = 0		

Note: R1 = R0 = 1 is not allowed.

START	A positive transition resets the DPM	1 line
	A negative transition starts a conversion	
	Minimum pulse width = 100 nanoseconds	

INTERNAL START 1 line
Logic 1 = Run
Logic 0 = Stop

There are a few other connections to the DM2000AR, but they are not used in this interface configuration. The microcomputer interface must be able to start a conversion, sense that the conversion has been completed, and input the data. In this case, there are 13 numeric data lines for the 3½ digits, one line for overload, one line for polarity, and three lines for decimal point position. The DM2000AR's DONE flag must also be input.

We have decided to hard wire the RANGE MODE signals, R1 and R0, to ground, or logic zero, and the INTERNAL START signal to +5 volts, or logic 1, to configure the DPM in the autoranging mode. Some of these digital inputs could be controlled by the computer, if that is desired, by using a latched output port to provide the necessary logic signals. In the current application, this is not necessary. The DPMs conversion is started with an OUT 200 pulse provided by gating $\overline{\text{OUT}}$ with an unused device address, $\overline{200}$.

We have chosen to use accumulator I/O for this example, although memory-mapped I/O is equally valid in an application such as this. A typical interface for the DM2000AR is shown in Fig. 3-6. This interface uses three 8-bit input ports constructed with Texas Instruments SN74LS244 three-state buffers. Since this is a relatively new integrated circuit, a schematic diagram is shown in Fig. 3-7.

Device codes of 200, 201, and 202 have been assigned to the three-input ports. You should be able to see direct similarities between this interface, the interface for the 10-bit A/D converter presented in Unit 2, and the previous DPM interface. The software is also similar.

Software for the DM2000AR DPM—The software used to control the DM2000AR digital panel meter can be very simple or it can be complex, depending upon the task to be performed. Again, the three basic tasks are:

- Start a conversion.
- Sense the end-of-conversion flag (done).
- When done, input the data.

The major difference between the DM2000AR and the AD2009 digital panel meters is that the DM2000AR has additional outputs for decimal point position, an indication of the voltage range currently in use by the DPM. Since the DM2000AR was chosen for this application, we must assume that this information is useful.

In this example we will actually use the DPM, the microcomputer, and the interface to construct a data logger. Data loggers are devices or systems that store data in a form that will be useful at a later time

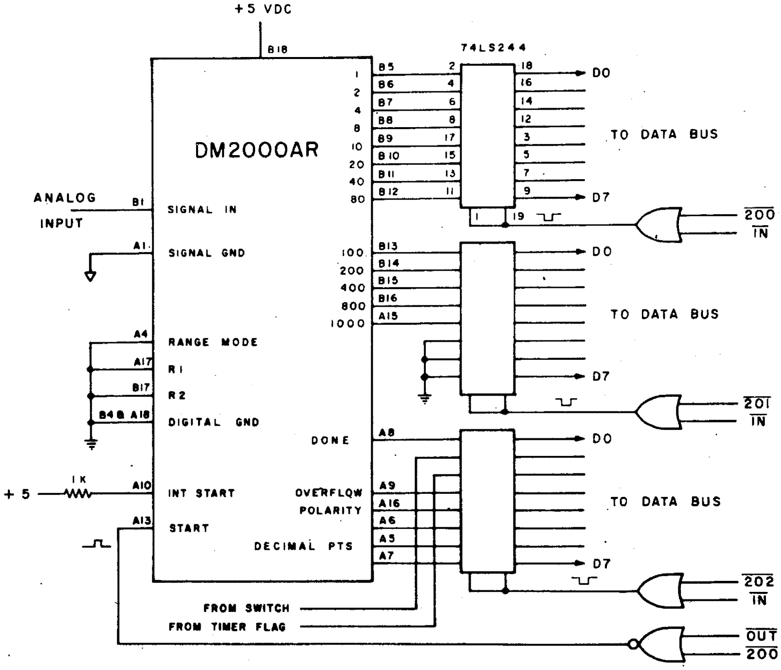
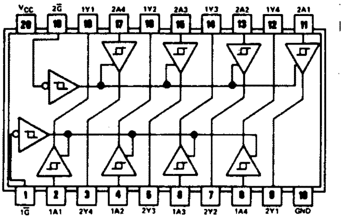


Fig. 3-6. The complete interface for the Datel Systems DM200AR digital panel meter.

when the data is actually evaluated. Data loggers usually do not make decisions based upon the data they are collecting, so they do not require a sophisticated computer for data processing. The microcomputer will answer this need nicely.

In this case, we will print the data being collected on a teletypewriter for later evaluation. Since most automatic send/receive (ASR) teletypewriters will also punch characters on a paper tape at the same time that the numbers and letters are being printed, it is possible to



SN54LS244 (J) **SN74LS244 (J, N)**

Courtesy Texas Instruments Inc.

Fig. 3-7. Schematic diagram of the SN74-LS244 three-state buffer integrated circuit.

generate a complete, permanent, computer-compatible record of the information logged.

Now that the DPM has been interfaced to our microcomputer, as shown in Fig. 3-6, it is necessary to develop software to perform the logging functions. The software requirements are:

- Start logging the data only after a switch is closed.
- Sample the data at a rate set by an external clock. The sample rate will be from two samples per second to one sample per minute.
- Print the data as ±*nnnn,* with the decimal point in the correct position for the voltage range being measured.
- Print five columns of data across the teletypewriter's page with three spaces between columns.
- Print "OVER" when an overflow condition exists.

We will assume that you will be able to provide the clock circuit and the switch closure to logic zero or ground. These concepts have been discussed in Unit 2. The switch closure will also be used to reset the clock circuit to synchronize the clock to the computer. In this way we will be sure that the first clock period starts at $t = 0$, and not at some other time. The clock rate is determined by connecting the clock flag to the proper oscillator–divider network output, as shown previously in Figs. 2-10 and 2-11. Two of the unused bit positions at Input Port 202 are used for sensing the switch closure and the clock flag. Bit position D1 is assigned to the switch while bit position D2 is assigned to the clock flag. Since a flip-flop flag is used in the timer circuit, a flag-clearing pulse must be provided in the interface. In this case we have chosen to make the OUT 200 pulse perform two functions; not only will it start a conversion in the DM2000AR DPM, it will also clear the clock flag flip-flop.

The clock and flag circuitry will not be discussed further. Our main emphasis will be on the software necessary to perform the data-logging functions. A simplified flowchart is shown in Fig. 3-8. The software will actually be somewhat more complex since the BCD data must be converted into the ASCII codes corresponding to each decimal digit, the sign must be output as a + or a −, and the decimal point must be placed correctly between the digits. The overflow must also be detected and acted upon.

The conversion of BCD data into the corresponding ASCII code (American Standard Code for Information Interchange) is straightforward. The ASCII code is the one used with most teletypewriters and terminal devices. The 8-bit binary ASCII codes and their octal equivalents for the decimal characters zero through nine are shown in Table 3-2. It should be easy to see that the last four bits in each ASCII code are the BCD equivalent of the decimal digit being repre-

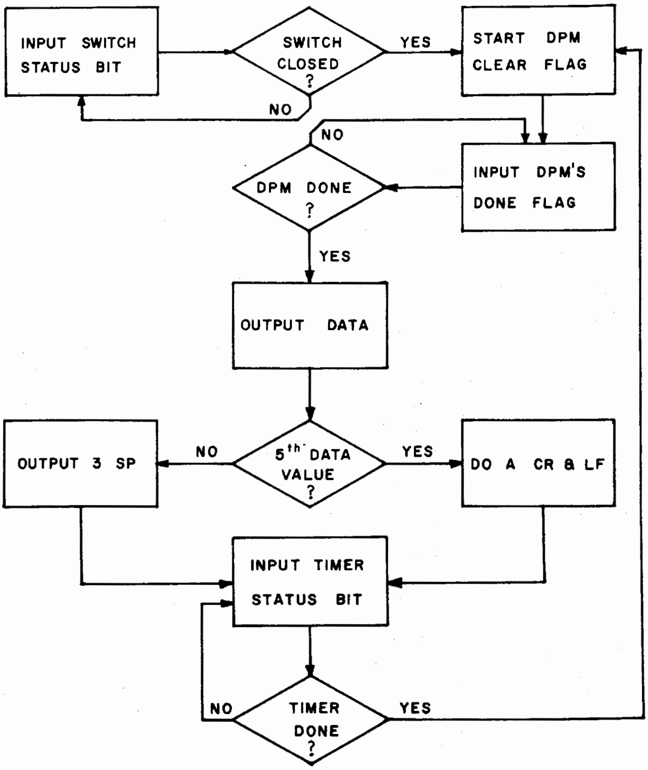


Fig. 3-8. The flowchart for the data-logger control program.

Table 3-2. ASCII Characters and Their Octal Equivalents for the Decimal Digits Zero Through Nine

		ASCII	
Decimal	BCD	Binary	Octal
0	0000	10110000	260
1	0001	10110001	261
2	0010	10110010	262
3	0011	10110011	263
4	0100	10110100	264
5	0101	10110101	265
6	0110	10110110	266
7	0111	10110111	267
8	1000	10111000	270
9	1001	10111001	271

sented. The conversion takes place simply by adding the value 10110000_2, or 260_8, to the BCD value.

The software for the teletypewriter output and data formatting is more complex than the data-acquisition portion of the program. It is shown in flowchart form in Fig. 3-9. A complete program listing for the data-acquisition and output program used in the data logger is provided in Example 3-2.

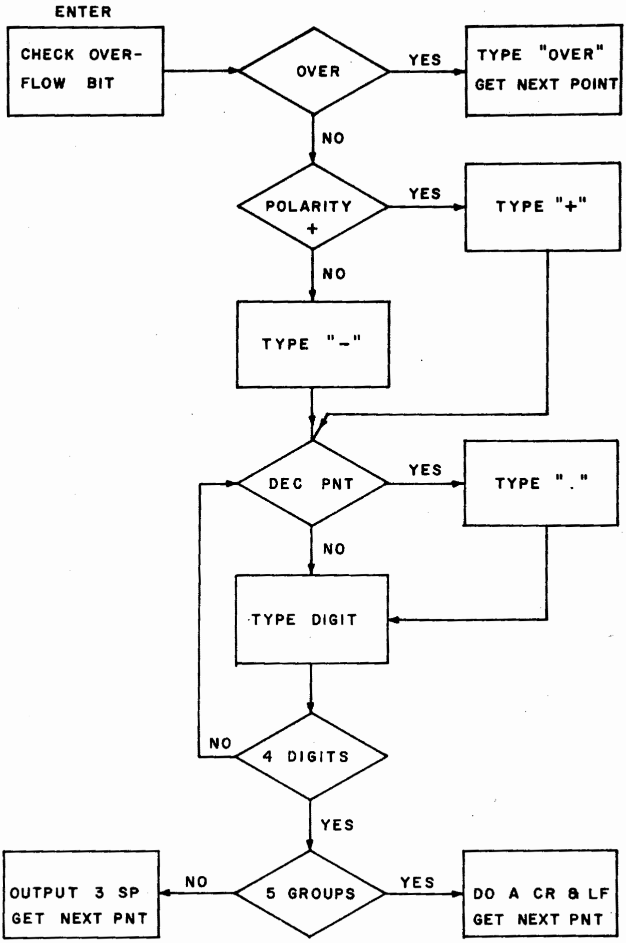


Fig. 3-9. The flowchart for the teletypewriter portion of the data-logger control program.

Some of the software that is used in the data logger example may be of interest since it can be used in other related problems. Three of the software sections are discussed in the following sections.

HANDLING BCD DATA

Binary-coded-decimal data requires only four bits per digit, so rather than "wasting" the other four bits in each 8-bit word, the data is packed as two 4-bit BCD data words per 8-bit word. This is done initially at the interface by inputting two 4-bit BCD digits at each 8-bit input port. In each case, the most significant digit (MSD) of the two is represented in bit positions D7 through D4. To convert these four bits into the ASCII character that represents the correct digit, the bits D7 through D4 are rotated to the right and into bit positions D3 through D0. This new data word is then logically ANDed with 00001111_2 to mask-out the unwanted bits D7 through D4. Addition of 10110000_2 to the four bits of the data, now in bit positions D3 through D0, converts the BCD digit into the corresponding ASCII code.

```
                         /EXAMPLE 3-2
                         /USING THE MICROCOMPUTER AS A DATA LOGGER
                         *030 000
303 000 061              LXISP
030 001 377              377
030 002 030              030
030 003 315              CALL      /START WITH A CLEAN LINE
030 004 163              CRLF
030 005 030              0
030 006 333   TEST,      IN        /TEST FOR THE SWITCH CLOSURE
030 007 202              202
030 010 346              ANI
030 011 002              002
030 012 302              JNZ
030 013 006              TEST
030 014 030              0
030 015 056   INIT,      MVIL      /INITIALIZE THE COLUMN COUNT
030 016 005              005
030 017 333   TIMER,     IN        /TEST FOR THE TIMER FLAG
030 020 202              202
030 021 346              ANI
030 022 004              004
030 023 312              JZ
030 024 017              TIMER
030 025 030              0
030 026 323              OUT       /CLEAR TIMER FLAG & START DPM
030 027 200              200
030 030 333   FLAG,      IN        /TEST FOR THE EOC FLAG
030 031 202              202
030 032 107              MOVBA     /STORE STATUS BITS
030 033 017              RRC
030 034 332              JC
030 035 030              FLAG
030 036 030              0
030 037 333              IN        /CONVERSION FINISHED, INPUT THE
```

```
030 040 200          200      /FIRST EIGHT BIT WORD
030 041 137          MOVEA    /STORE IT IN REG E
030 042 333          IN       /INPUT THE SECOND EIGHT BIT WORD
030 043 201          201
030 044 127          MOVDA    /STORE IT IN REG D
030 045 170          MOVAB    /GET THE STATUS BITS
030 046 346          ANI      /TEST FOR OVERFLOW
030 047 010          010
030 050 312          JZ       /IF OVER, JUMP
030 051 220          OVER
030 052 030          0
030 053 170          MOVAB    /GET THE STATUS BITS AGAIN
030 054 016          MVIC     /SET REG C FOR "+"
030 055 053          "+"
030 056 170          MOVAB    /GET THE STATUS BITS AGAIN
030 057 346          ANI      /TEST FOR THE SIGN
030 060 020          020
030 061 302          JNZ      /JUMP IF SIGN IS +
030 062 066          PLUS
030 063 030          0
030 064 014          INRC     /IT IS — SO ADD 2
030 065 014          INRC
030 066 171   PLUS,  MOVAC    /GET THE + OR —
030 067 315          CALL     /PRINT IT
030 070 245          TTYOUT
030 071 030          0
030 072 315          CALL     /CHECK FOR A DECIMAL POINT
030 073 176          DECM
030 074 030          0
030 075 172          MOVAD    /SET UP THE FIRST CHARACTER
030 076 017          RRC
030 077 017          RRC
030 100 017          RRC
030 101 017          RRC
030 102 315          CALL     /PRINT THE NUMBER
030 103 210          NUMOUT
030 104 030          0
030 105 315          CALL     /CHECK FOR ANOTHER DECIMAL POINT
030 106 176          DECM
030 107 030          0
030 110 172          MOVAD    /GET THE NEXT NUMBER
030 111 315          CALL     /PRINT THE NUMBER
030 112 210          NUMOUT
030 113 030          0
030 114 315          CALL     /CHECK FOR THE LAST DECIMAL POINT
030 115 176          DECM
030 116 030          0
030 117 173          MOVAE
030 120 017          RRC
030 121 017          RRC
030 122 017          RRC
030 123 017          RRC
030 124 315          CALL     /PRINT THE LAST NUMBER
030 125 210          NUMOUT
```

```
030 126 030              0
030 127 173              MOVAE      /GET THE LAST NUMBER
030 130 315              CALL
030 131 210              NUMOUT
030 132 030              0
030 133 055   COUNT,     DCRL       /HAVE FIVE COLUMNS BEEN PRINTED?
030 134 312              JZ         /IF SO, JUMP
030 135 155              NXTLIN
030 136 030              0
030 137 046              MVIH       /IF NOT, PRINT 3 SPACES
030 140 003              003
030 141 076   MRSP,      MVIA
030 142 040              040
030 143 315              CALL
030 144 245              TTYOUT
030 145 030              0
030 146 045              DCRH
030 147 302              JNZ
030 150 141              MRSP
030 151 030              0
030 152 303              JMP        /AFTER 3 SPACES DO ANOTHER POINT
030 153 017              TIMER
030 153 030              0
                         /THIS ROUTINE OUTPUTS A CARRIAGE RETURN AND
                         /A LINE FEED AND POINTS BACK TO THE
                         /REINITIALIZING SOFTWARE
030 155 315   NXTLIN,    CALL
030 156 163              CRLF
030 157 030              0
030 160 303              JMP
030 161 015              INIT
030 162 030              0
                         /OUTPUT A CARRIAGE RETURN AND LINE FEED
030 163 076   CRLF,      MVIA
030 164 015              015
030 165 315              CALL
030 166 245              TTYOUT
030 167 030              0
030 170 076              MVIA
030 171 012              012
030 172 315              CALL
030 173 245              TTYOUT
030 174 030              0
030 175 311              RET
                         /SUBROUTINE TO TEST DECIMAL STATUS BITS
                                    /AND OUTPUT A "." IF ONE IS FOUND
030 176 170   DECM,      MOVAB      /GET THE STATUS WORD
030 177 027              RAL        /ROTATE IT
030 200 107              MOVBA      /PUT IT BACK
030 201 330              RC         /IF A CARRY IS FOUND, RETURN
030 202 076              MVIA       /NO CARRY, TYPE A "."
030 203 056              056
```

```
030 204 315             CALL
030 205 245             TTYOUT
030 206 030             0
030 207 311             RET

                        /CONSTRUCT AN ASCII NUMBER FROM THE
                        /BCD VALUE PRESENT IN 4 LSB'S OF REG A

030 210 346  NUMOUT,    ANI         /MASK OUT UNWANTED BITS
030 211 017             017
030 212 306             ADI
030 213 260             260
030 214 315             CALL
030 215 245             TTYOUT
030 216 030             0
030 217 311             RET

                        /ROUTINE TO OUTPUT "OVER" IF AN OVERFLOW
                        /CONDITION IS FOUND

030 220 345  OVER,      PUSHH
030 221 041             LXIH
030 222 261             MSGI
030 223 030             0
030 224 176  MLOOP,     MOVAM
030 225 376             CPI
030 226 000             000
030 227 302             JNZ
030 230 236             OUTPUT
030 231 030             0
030 232 341             POPH
030 233 303             JMP         /TREAT THIS AS A REGULAR OUTPUT
030 234 133             COUNT       /SO COUNT IT
030 235 030             0
030 236 315  OUTPUT,    CALL
030 237 245             TTYOUT
030 240 030             0
030 241 043             INXH
030 242 303             JMP
030 243 224             MLOOP
030 244 030             0

                        /TELETYPE OUTPUT ROUTINE

030 245 365  TTYOUT,    PUSHPSW
030 246 333             IN
030 247 021             021
030 250 346             ANI
030 251 004             004
030 252 312             JZ
030 253 246             TTYOUT + 1
030 254 030             0
030 255 361             POPPSW
030 256 323             OUT
030 257 020             020
030 260 311             RET
```

```
030 261 040  MSG1,    040
030 262 117           117
030 263 126           126
030 264 105           105
030 265 122           122
030 266 040           040
030 267 000           000
```

The four bits originally stored in bit positions D3 through D0 in the packed 8-bit word are not rotated, but are instead ANDed with 00001111_2 and then added to 10110000_2 to construct the ASCII equivalent. An example may be helpful in understanding this:

```
MMMMLLLL      M = Most significant digit, four bits
              L = Least significant digit, four bits
```

Once the data word has been rotated four bit positions to the right, it looks like this:

```
LLLLMMMM
```

This is then ANDed with the data word 00001111_2:

```
LLLLMMMM
00001111
--------
0000MMMM = Result of AND operation
```

Now, 10110000_2 is added to the result of the AND operation to yield the ASCII equivalent of the BCD character, MMMM:

```
 0000MMMM
+10110000
 --------
 1011MMMM = ASCII equivalent for the BCD digit, MMMM
```

DECIMAL-POINT DETECTION

The decimal point's position is determined by a one-out-of-three code output by the digital panel meter. Since the decimal point can be located in any one of three positions, i.e., 1234 or 1.234 or 12.34, and since the teletypewriter will print from left to right, it is necessary to check for the presence of the leftmost decimal point first. The decimal point's position is indicated by having one of the three decimal-point outputs at a logic zero. The output that is at a logic zero indicates this according to the convention previously described in the *Digital Outputs* section of *Digital Panel Meter Interfacing Example No. 2*. The decimal-point information is input by the computer at Input Port 202 and it has the following representation:

```
0  0  0  X  X  X  X  X          Status and Flag Word
↑  ↑  ↑
↑  ↑     ←19.99-volt range
↑     ←1.999-volt range
  ←.1999-volt range (199.9 millivolts)
```

The program stores this combined Status and Flag Word in the 8080's B register. The DECM subroutine is used in three places to test for a decimal point and to output a decimal point if it detects a logic zero in the current bit position that it is testing. The DECM subroutine rotates the decimal-flag bits into the 8080's carry flag and then tests them with a return-if-carry instruction, RC.

The RC instruction causes a return to the main program if no "decimal point" is found in the bit position undergoing the test. If, however, the carry is set to a logic zero as a result of the bit rotation, indicating that the "decimal point" has been detected, additional software steps output a decimal point to the teletypewriter so that it fills the next character position in the voltage being printed. After the decimal point is output, the computer returns control to the main program.

MESSAGE SOFTWARE

One of the requirements of the data logger's software is to have the computer type "OVER" on the printer if an overflow condition exists. The software at symbolic address OVER does this by pointing to the message storage area, MSG1. The data stored at MSG1, and in the five following memory locations are the ASCII characters:

```
MSG1,  040   Space
       117     O
       126     V
       105     E
       122     R
       040   Space
       000    Null
```

A typical "in-range" output will take six character positions for the sign, the four digits, and for a decimal point. The "OVER" message will also take six character positions since we have included the four letters and two spaces, one before the message and one after it. The last character is a null, or all zeros. This signals the software in the OVER portion of the program that it has reached the end of the ASCII message file which is to be output.

The OVER subroutine is a general-purpose one that you may find useful elsewhere in your programming. It is much more flexible than the teletypewriter output that would be generated by:

```
MVIA   /Load register A with data
 040   /ASCII for a space
```

```
CALL     /Call the teletypewriter
TTYOUT   /subroutine
  0
MVIA     /Do another character
 117
CALL
TTYOUT
  0
  •
  •
 etc
```

It is also much easier to change the message to be output by changing the ASCII "string" of characters stored in successive memory locations in a buffer area of memory rather than changing a portion of the main program, such as that just listed.

These and other software techniques will be the subject of future books, authored by Dr. Christopher A. Titus. Detailed and well-documented examples will be the rule in these forthcoming Sams books. Dr. Titus is with Tychon, Inc., Blacksburg, VA.

SOFTWARE MODIFICATIONS

The data-logging program presented in Example 3-2 works well for many applications, but some users may find that modifications are necessary before it can be used in a particular situation. Here are some of the typical modifications that may be made to the data logging program:

1. *More columns of data are required per printed page.* The number of columns of data is preset at the INIT symbolic address, with the data byte that immediately follows the MVIL instruction. For 10 columns of data, the new value that would follow the MVIL instruction would be 012. Remember that these are octal values.
2. *More spaces are required between data values.* The number of spaces between the columns has been preset to three in the COUNT portion of the software with an MVIH instruction. The number of spaces between columns may be changed by changing the data byte immediately following this MVIH instruction.

   To eliminate the spaces between the columns completely, you can substitute a jump address TIMER in the first three bytes of the software steps at symbolic address MRSP:

```
Old Program Steps          New Program Steps
MRSP,  MVIA                  MRSP,  JMP
        040                         TIMER
       CALL                           0
```

Only the first three bytes of MRSP are shown since these are the only ones that will need to be changed to eliminate the spaces between the output columns. Remember that since an unconditional jump has been inserted in the MRSP program, the following old bytes (instructions and data) do not have to be changed since the computer cannot get to them. This is often called a *patch* in a program. Some data-processing programs may require the elimination of the spaces between columns, particularly if you are generating a paper tape of the data for later use.

3. *Changing the output message.* If you do not like the OVER message that is output by the program to indicate an overflow condition, the ASCII message file may be changed by specifying new ASCII characters, starting at symbolic address MSG1. Remember to end your message file with a null character, 000, to terminate the printing of the message. Do not exceed six characters if you wish your columns to be properly aligned.

   Some computer programs, such as BASIC or FOCAL,* may be used to evaluate the data punched onto paper tape by the datalogger software. The format of the data punched onto the tape must be such that it is understood by these programs. This means that the "OVER" message may cause problems since the data processing program may try to evaluate the ASCII characters for each letter as a data value. The message may be changed to output a single character, say "$", for an overflow condition. The value of ±0.000 could also be output by default to indicate the overflow condition.

   Some data-processing programs may require the elimination of the carriage-return and the line-feed subroutine, since these may be recognized by the data-processing program as characters that represent the end of the data file. Removal of the carriage-return and line-feed subroutine is easily accomplished by placing a return instruction, RET, at the symbolic address CRLF. Other programs may require a carriage-return and a line-feed subroutine after each data point.

These changes, and others, are readily made in the software listings that we have provided. Please use it and make modifications to it as you wish.

In this unit we have provided you with information about the dual-slope type of analog-to-digital converters. Taking this a logical step further, we have introduced you to the concepts of using a digital panel meter as an A/D converter with binary-coded-decimal outputs. You will find that digital panel meters are particularly useful in those ap-

* FOCAL is a registered trademark of the Digital Equipment Corporation, Maynard, Massachusetts.

plications that will require a visual indication of the signal (or quantity) being measured by the computer. Outputs of this type can be particularly reassuring to human operators. We have provided two examples of how BCD data may be stored or used.

4

Miscellaneous Conversion Techniques

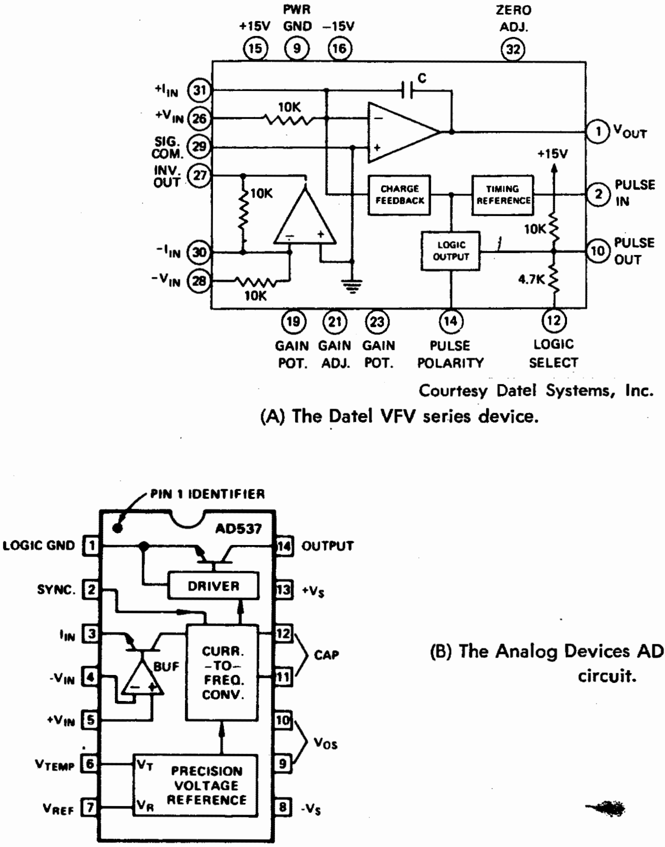
INTRODUCTION TO THIS UNIT

There are a number of conversion techniques used in analog-to-digital converters besides the ramp, successive-approximation, and dual-slope techniques previously described. These additional A/D conversion techniques are not in as widespread use, but they do find use in special situations. Therefore, we feel that it is important that you are at least familiar with their operation before we present new topics.

VOLTAGE-TO-FREQUENCY CONVERTERS

The voltage-to-frequency (v/f) converters are not as widely used as the other techniques discussed previously, i.e., the ramp, successive-approximation, and dual-slope methods. They find use in systems with a large dynamic range of input voltages, generally several decades, where the conversions can be slow and where sensors that generate voltage outputs may be some distance from the measuring device or computer.

The voltage-to-frequency converters operate by converting a voltage input to a frequency output, where the frequency is directly proportional to the input voltage. The frequency output is generally a standard logic-compatible voltage for transistor-transistor logic (TTL) circuits or complementary metal-oxide semiconductor (CMOS) devices. The output is either represented by a series of



Courtesy Datel Systems, Inc.

(A) The Datel VFV series device.

(B) The Analog Devices AD537 integrated circuit.

Courtesy Analog Devices, Inc.

Fig. 4-1. Typical voltage-to-frequency converter devices.

pulses or a square wave at the frequency generated by the particular voltage input. The linearity for v/f converters is generally 0.05% over the specified operating and temperature range of the particular device in use.

Typical voltage-to-frequency devices are the Datel Systems, Inc., VFV series and the Analog Devices, Inc., AD537. Each of these devices has a maximum frequency output that is between 100 kHz and 150 kHz. Both devices may also function as frequency-to-voltage converters to reconstruct the voltage being measured from the frequency originally generated by a voltage-to-frequency converter. The Analog Devices integrated circuit may also be operated in the bipolar mode, so that voltages with both positive and negative components can be measured. A split or dual power supply is required for this mode of operation. Both the VFV and AD537 devices are illustrated in Fig. 4-1.

Measurements of analog signals are made simply by counting the number of pulses or clock transitions per unit time, generally one second. A simple 4- or 5-digit decade counter may be used to display the result of a conversion. A binary counter may also be used if a binary count is more applicable, as would be the case in computer interfacing. Conversion times are generally long, since counts must be totalized. Conversion rates of 10 measurements per second are possible, with longer measurement times of one to 10 seconds being used if more resolution is required.

Even though the v/f counters are slow when compared to the fast conversion speeds of successive-approximation converters, the v/f converters are frequently used where it is necessary to integrate a signal over a relatively long period. The v/f converters are always performing conversions, so that integration amounts to totaling the counts produced over the time period of interest.

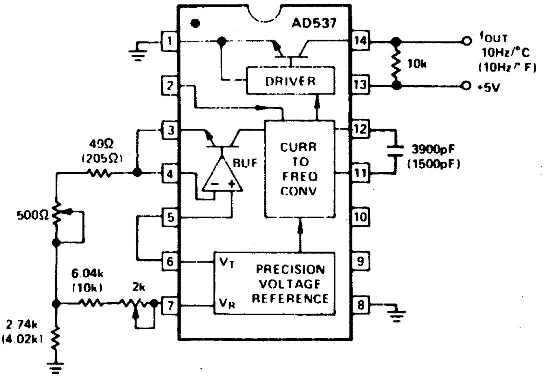
The use of v/f converters in integration schemes provides an easy way to measure the area under a peak output by an instrument. Since an integration sums the input over a period of time, periodic noise signals superimposed upon the voltage being measured may be integrated to zero.

Interfacing v/f converters to microcomputers is straightforward since only a counter is needed. The v/f conversion technique is particularly useful when the voltages to be measured or digitized are some distance from the computer. Only two wires are required to transmit the frequency information to the computer from a remote measuring site. The use of line drivers and line receivers makes this technique particularly insensitive to noise.

The Analog Devices AD537 voltage-to-frequency converter is a particularly interesting device since it is low-cost and since it has many applications. It can be interfaced with most of the common logic families, it can be multiplexed, and it can operate in a true two-wire configuration where power for the device is actually obtained from the wires that carry the frequency signal. This is an important consideration in remote monitoring applications since each sensor will not require an individual power source.

The AD537 v/f converter has a linear, temperature-proportional output available so that the AD537 integrated circuit alone can be used as a temperature sensor, converting temperature to frequency. A typical circuit is shown in Fig. 4-2. The output changes at a rate of 10 Hz per degree when the integrated circuit is used with the components shown. Component values are for Celsius or (Fahrenheit).

As mentioned previously, when a v/f converter is used in a computer interface, either a binary or a decade counter may be used to accumulate the frequency value. In some applications, both may be used. In the circuit shown in Fig. 4-2, a decade counter may be used for a direct readout of degrees, while a binary counter is used



Courtesy Analog Devices, Inc.

Fig. 4-2. The use of the Analog Devices AD537 v/f converter as a temperature sensor.

at the same time to provide a computer with a binary value of the temperature.

FLASH CONVERTERS

The *flash,* or *simultaneous conversion,* technique operates by comparing the unknown voltage signal to be digitized with a series of preset reference potentials. The comparisons take place simultaneously. Thus, for an *n*-bit conversion, 2^n-1 comparators are required. For an 8-bit conversion, 255 comparators and references would be required. A single, stable reference is used with a resistor ladder to provide the necessary comparison voltages. The reference input of the first comparator is set to one-half of the step voltages, with all of the remaining comparators' reference inputs set to be one voltage step above the previous one. The block diagram of a flash converter is shown in Fig. 4-3. The encoder section converts the outputs of the comparators into a 6-bit binary code.

The outputs provided by the comparators would have to be encoded in some way to provide a binary output that is representative of the unknown voltage being measured. The gating can be complex for converters having seven or more bits.

Since one comparator is used for each voltage step, this conversion technique is useful only in high-speed 4-, 5-, or 6-bit converters. The conversion time is limited by the comparators' response time and the speed of the logic circuits that perform the code conversion.

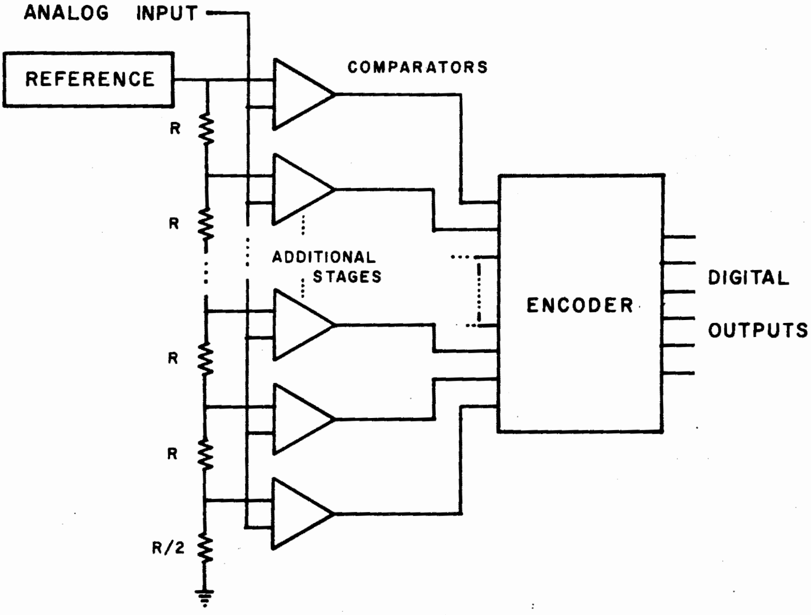


Fig. 4-3. Block diagram of a typical 6-bit flash converter.

Combinations of the flash technique and a feedback D/A converter are used to increase the resolution possible to more than eight bits, but the conversion speeds are slower and the costs are higher than they would be for a 6-bit flash converter. When a D/A converter is used for feedback in a flash converter, the converter is called a *simultaneous ripple converter* or a *simultaneous feedback converter*. The block diagram for this type of a converter is shown in Fig. 4-4.

The simultaneous feedback converter operates by using two 4-bit flash converters similar to the converters shown in Fig. 4-3. These are shown in Fig. 4-4 as 4-bit A/D #1 and #2. The unknown analog input voltage is first digitized to four bits of resolution using 4-bit flash A/D #1. For an input range of zero to one volt, this would be a resolution of one part in sixteen, or 15 voltage steps of 66.7 millivolts per step.

We will assume that the unknown voltage is 690 millivolts so that we may examine the operation of the simultaneous feedback converter shown in Fig. 4-4. The first 4-bit flash A/D converter (#1) can only resolve the unknown signal into sixteen possible outputs representing the fifteen voltage steps between zero and one volt. This is the 66.7-millivolt step discussed previously. The possible voltage

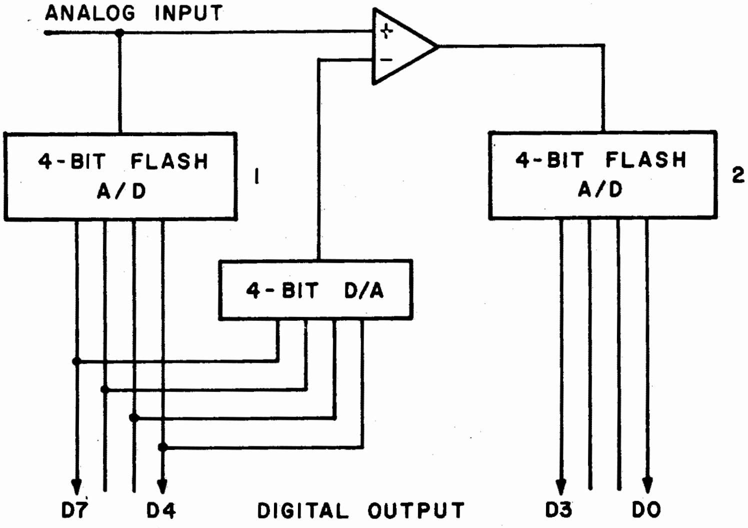


Fig. 4-4. Block diagram for a typical simultaneous feedback converter.

steps found within the 4-bit flash converter are listed in Table 4-1. *Remember that the first comparator's reference voltage is one-half of the step voltage and that all of the other comparators have inputs of one voltage step above the previous one.*

Table 4-1. Possible Voltage Steps Within the 4-Bit Flash Converter

Step	Voltage (mV)	Step	Voltage (mV)
15	967	7	433
14	900	6	366
13	833	5	300
12	767	4	233
11	700	3	166
10	633	2	100
9	566	1	33
8	500		

The unknown input of 690 millivolts falls between Step 10 and Step 11. Comparators 1 through 10 indicate that the unknown input voltage is greater than their reference inputs as noted in Table 4-1, so the code corresponding to the last comparator that senses the "overflow" condition becomes the code that represents the unknown voltage. In this case, $10 = 1010_2$. *Note, that there is no "comparatoı zero." If none of the comparators indicate an "overflow," the decoding logic will output the code for zero, 0000_2.*

The voltage used as the reference at the 10th comparator is only 633 millivolts, so the remaining difference of 57 millivolts between the comparator's reference and the unknown voltage being measured goes unresolved, so far.

The binary code output by the first 4-bit flash converter is applied to the inputs of a 4-bit digital-to-analog converter. The converter is set up to generate a voltage that is equal to the reference voltage for the last comparator that indicates an overflow condition. In this case that is comparator 10, so the 4-bit D/A converter outputs 633 millivolts. This voltage is subtracted from the unknown voltage by using a differential input amplifier. The amplifier outputs the difference between the unknown signal and the reference potential of the last comparator in the overflow state. In this case the difference is:

$$\begin{array}{rl} 690 & \text{millivolt unknown signal} \\ -633 & \text{millivolt reference to last overflow comparator} \\ \hline 57 & \text{millivolt difference} \end{array}$$

The 57-millivolt difference is equal to the "unresolved" portion of the unknown signal that we mentioned before. The second 4-bit flash converter is set up to resolve these small differences of up to 66 millivolts, the maximum voltage that would go unresolved by the first 4-bit flash converter.

In this type of a converter, a flash converter first resolves the unknown signal as best it can with four bits of resolution. The difference between the unknown signal and the closest approximation by the first A/D converter is passed along to another flash converter through a D/A conversion and voltage subtraction process.

Since flash converters are still used in this type of a converter, there are many comparators within the system. The general rule when using the simultaneous feedback converter is that Q comparators are required, where:

$$Q = 2^{[(n/2)+1]} - 2$$

$$n = \text{number of bits of resolution}$$

For the 8-bit converter system illustrated in Fig. 4-4, 30 comparators are required. It is important to note that even though a 4-bit D/A converter was used in this scheme, it must still be accurate to at least eight bits, or one part in 256, to preserve the accuracy of the system. All of the components in this type of a system must be at least as accurate as the total resolution required of the system. If this condition is not met, there will be significant errors in the digitized outputs.

A further refinement of this simultaneous feedback converter is the *variable threshold flash converter technique*. This conversion technique is beyond our present scope; it is thoroughly discussed in *Motorola Application Note AN-702,* Motorola Semiconductor Products, Inc., Phoenix, AZ 85036.

The only other type of converter that we will mention is the *tracking converter*. This type of converter may be thought of as a fast ramp A/D converter that can generate a positive and a negative ramp through the use of up-down counters. The converter continuously follows the input voltage by using the comparator's output, as per a normal ramp converter, to indicate whether the ramp should go up or down to reach the unknown voltage. The digital outputs are always active, indicating the current digital value of the unknown input.

There are probably other conversion techniques, but they are not directly applicable to computer interfacing. The discussion of the ramp, dual-slope, successive-approximation, and voltage-to-frequency converters probably covers 95% of the converter types in use today. Faster successive-approximation A/D converters are becoming available and they may soon match the speed of some flash-type converters.

5

Sample-and-Hold Circuits and Multiplexer Devices

INTRODUCTION TO THIS UNIT

The purpose of this unit is to introduce you to the use of sample-and-hold circuits and multiplexers in analog systems. You will find that sample-and-hold circuits are particularly useful when you are trying to use an analog-to-digital converter to measure varying signals. Multiplexers are quite often used in applications where a single A-D converter is being used to acquire data from a number of sensors.

SAMPLE-AND-HOLD CIRCUITS

Sample-and-hold devices (S/H) or sample-and-hold amplifiers (SHA) are analog circuit elements that are the analog equivalent of the digital latch. Sample-and-hold circuits are used when we wish to sample an analog signal and then hold it steady at a particular point so that a particular voltage of interest may be measured or used elsewhere in a system. The operation of an ideal sample-and-hold device is shown in Fig. 5-1. In this example, the S/H output follows, or tracks, the voltage input during the *SAMPLE* period and it holds the latest analog voltage when it switches to the *HOLD* mode. In Fig. 5-1, the input and output voltage lines have been offset slightly for clarity.

The simplest sample-and-hold circuit is a capacitor and a switch. The low-leakage types of capacitors (polystyrene or teflon) are preferable to the other types available. This first attempt at a sample-and-hold circuit is shown in Fig. 5-2.

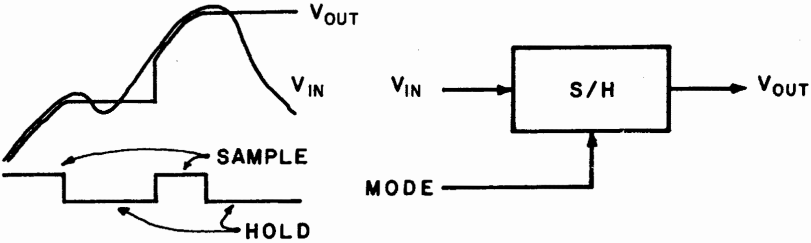


Fig. 5-1. Inputs and output for sample-and-hold device showing both the SAMPLE and HOLD modes of operation.

In this first attempt at a sample-and-hold circuit, the capacitor will start to charge or discharge to make V_{OUT} equal to V_{IN} once the switch is closed. The capacitor charges at a predetermined rate, so the voltage present at the output, V_{OUT}, will not instantaneously follow, or *track,* the input voltage, V_{IN}. In this case, we have also assumed that the unknown voltage source can supply the charging current required by the resistor-capacitor network. The resistor shown in the diagram may have a very low resistance; the resistance supplied by the switch and the interconnections. It must also be remembered that loads on the capacitor's output will also tend to discharge the voltage present.

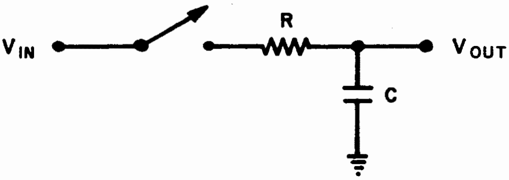


Fig. 5-2. The use of a capacitor and a switch to construct a sample-and-hold circuit.

Most sample-and-hold circuits are more complex, using high-impedance field-effect transistor (FET) input operational amplifiers. A typical circuit is shown in Fig. 5-3.

The mode-control switch shown in Fig. 5-3 is typically a *diode bridge switch* or a field-effect transistor switch. For a further discussion of sample-and-hold devices, including their design and operating characteristics, we recommend the Siliconix Incorporated book, *Analog Switches and their Applications*[4].

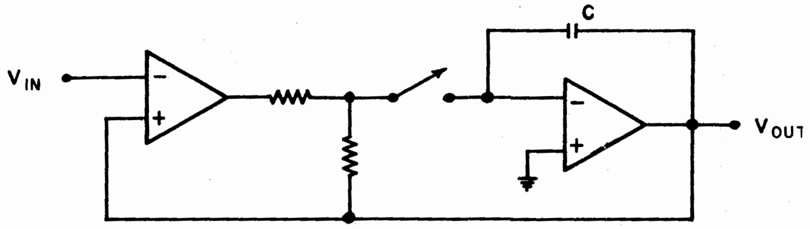


Fig. 5-3. A typical sample-and-hold circuit using operational ampliers.

USING SAMPLE-AND-HOLD DEVICES

Sample-and-hold devices have several purposes in digital-to-analog and analog-to-digital converter applications. Sample-and-hold devices may be used to:

- Hold an analog signal steady so that an A/D conversion may be performed.
- Simultaneously sample many analog inputs for later measurement (requires one S/H device per input).
- Deglitch a D/A converter's output to eliminate output voltage spikes or settling transients.
- Distribute one D/A converter's output to several points where voltages must be maintained constantly.

The second and the fourth uses listed are becoming less important than they were two or three years ago. This is because it is now probably less expensive to dedicate an analog-to-digital converter to each input to be measured and to have one digital-to-analog converter per output, depending upon the specific application.

Sample-and-hold devices are frequently used to *deglitch* the outputs from D/A converters by providing *analog double buffering.* Once the D/A converter has performed the digital-to-analog conversion and its output has settled to within the desired percentage error range, the D/A converter's output is sampled and held with an S/H module. The output of the S/H device is then presented to the voltage receiving device, or the device which would normally be connected to the D/A converter. This type of operation is particularly useful where fast D/A converter response is needed, but glitches can not be tolerated. Video displays that use D/A converters to position the electron beam are often deglitched in this way.

The more usual use of sample-and-hold devices is to sample and hold an analog signal at a particular point while it is measured by an A/D converter. A sample-and-hold device would be particularly useful in an instrumentation problem where an instrument's output changes too quickly to be measured by either a successive-approximation or a ramp A/D converter. You may recall that this was the case in Unit 2 in the examples in which the software controlled successive-approximation and ramp converters attempted to digitize a "peak" output.

When a sample-and-hold device is used prior to the input of the unknown signal to a comparator (in the case of a software-controlled A/D converter), or to an A/D converter module, if one is used, the digitization may proceed to give an accurate representation of the unknown peak voltage. The block diagram in Fig. 5-4 shows how this might be implemented. The SAMPLE signal from the computer allows the S/H module to sample the unknown signal. When the peak is reached, the instrument generates the PEAK signal, forcing the S/H device into the hold mode.

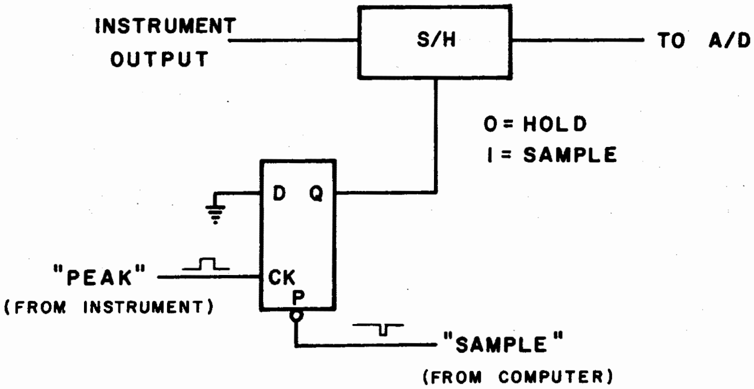


Fig. 5-4. Block diagram for a typical sample-and-hold computer interface.

The timing diagram shown in Fig. 5-5 shows the timing relationships between the signals used to control the S/H device shown in Fig. 5-4. The instrument's analog output is also shown, as is the output of the S/H device. The sample-and-hold input and output are offset for clarity.

Now, either a slow ramp A/D converter or a fast successive-approximation A/D converter could be used to provide the correct digital value for the voltage present at the "peak" output. The sample-and-hold device will maintain the voltage until it can be measured.

The use of sample-and-hold *front ends* with A/D converters greatly increases their capability to measure changing signals. Without using a sample-and-hold circuit, even fast converters can only be used to measure very slowly changing signals. For example, the maximum frequency of a sine wave which can be measured with a

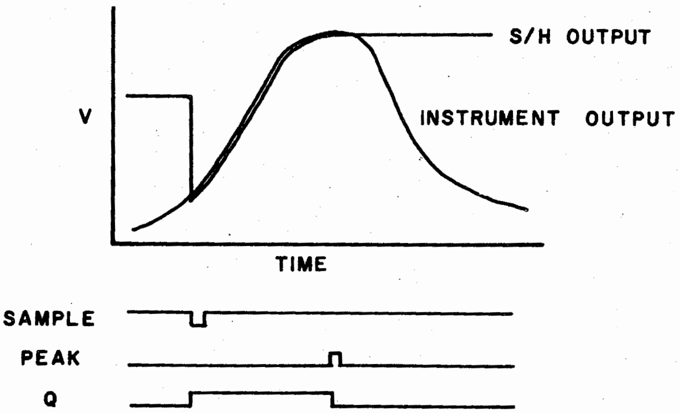


Fig. 5-5. A typical timing diagram for a sample-and-hold module under instrument and computer control.

10-bit, 10-microsecond A/D converter to an accuracy of ±1 LSB is 16 Hz[2]. This assumes that the A/D converter is attempting to measure a full-scale input signal.

The frequency/converter relationship may be expressed mathematically:[2]

For an accuracy of ±½ LSB, the rate of voltage change is:

$$\frac{\Delta V}{\Delta t} \leq \frac{1}{2} \frac{\text{Full Scale Voltage}}{2^n}$$

Which is equal to:

$$\frac{\Delta V}{\Delta t} \leq \frac{\text{Full Scale Voltage}}{2^{n+1}}$$

Thus, for a 10-volt full-scale A/D converter, this rate of change must be less than 5 millivolts per 10 microseconds, or 500 volts per second.

To calculate the maximum frequency for a particular voltage range, the following formula may be used:

$$\frac{\Delta V}{\Delta t} = (\text{Input Range}) \cdot (2\pi f)$$

In this formula, the input range is usually the full-scale range of the A/D converter and the frequency, f, is the highest frequency that will not exceed the $\Delta V/\Delta t$ rate of change.

These calculations should show you the utility of sample-and-hold devices and why they are often used with analog-to-digital converters. In the previously discussed data acquisition example, in which greenhouse temperatures were being measured, the temperature probably do not change quickly enough to require the use of a sample-and-hold device. In other situations, sample-and-hold front end are the general rule.

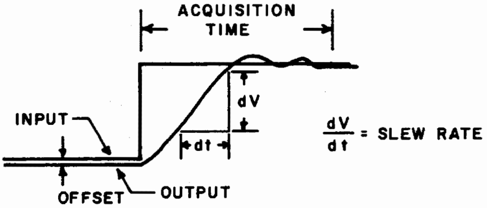


Fig. 5-6. Representation of acquisition time, offset, and slew rate.

Sample-and-hold devices are not ideal, and there are some term that will help you better understand their limitations and uses. Thes are listed as follows, keyed to Figs. 5-6, 5-7, and 5-8.

DC Offset

This is the difference between the input and the output voltages c the device when the input is grounded. It is usually expressed i millivolts. The offset may be adjusted to zero with external components, but the offset will generally change with changes in tim and temperature.

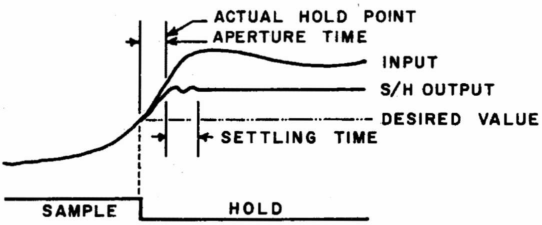


Fig. 5-7. Representation of aperture time, aperture uncertainty, and settling time.

Acquisition Time

The sample-and-hold is a nonideal device and it requires a defini period of time to actually acquire and track the analog input onc it is placed in the sample mode. The acquisition time is the time r quired to go from the hold state to the tracking state, where tl output remains within 0.01% of the input. The acquisition time generally a few microseconds. This may also be called the *settlir time* when the sample-and-hold is already in the sample mode.

Slew Rate

This defines the maximum rate of change for the output and it is expressed as volts per second. The slew rate is a limitation imposed by the charging rate of the capacitors used to store the voltage, and the actual slew rates of the operational amplifiers used in the circuit.

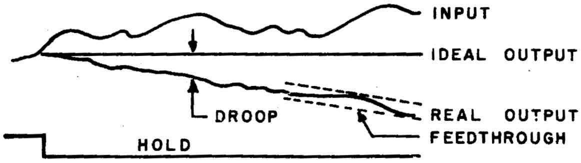


Fig. 5-8. Representation of linearity, decay rate, and feedthrough.

Aperture Time

This is the time period required by the device to go from the sample mode into the hold mode, once the hold command has been received. Sample-and-hold devices do not go from one mode to the other instantaneously. When going from the sample to the hold mode, sampling continues for a few nanoseconds.

Aperture Uncertainty

The aperture uncertainty is the variation in the aperture time from device to device. This uncertainty is generally limited to a few nanoseconds.

Linearity

The linearity describes the variation of the output from the expected output over the entire output voltage range of the sample-and-hold device. This is the difference between the INPUT and the IDEAL OUTPUT shown in Fig. 5-8. This may also be called a *gain error,* and it is expressed as a percentage, say, 0.01%. It may also be expressed as the deviation in the plot of input *vs.* real output from a "best straight line."

Decay Rate

Since the sample-and-hold circuit incorporates a capacitor for storing charge, the capacitor will tend to discharge to zero stored charge over a period of time. The discharge rate is a function of switch leakage current and the current required by the other circuit elements connected to the capacitor. The decay rate or *droop rate* is expressed as millivolts per second.

Feedthrough

The feedthrough of a sample-and-hold device is an indicatio of the level of the input reaching the output when the S/H devic is in the hold mode. This may be expressed as a percentage, or i *decibels,* db.

Small-Signal Bandwidth

This describes the frequency at which the output of the sample-and-hold is attenuated by 3 dB with respect to the input. This mean that the output is attenuated by about 30%. This property is no illustrated in the figures. Figs. 5-6, 5-7, and 5-8 are adapted fron those found in the *Analog-Digital Conversion Handbook.*[1]

There are limitations to the capabilities of sample-and-hold devices. Those devices that have short acquisition times use smal capacitors and, thus, the voltage droop rate will be large. The use o larger capacitors means that the acquisition time will be longer, bu the voltage droop rate will be less. When high acquisition speeds an long hold times are required in an application, two sample-and-hol modules may be used. The first sample-and-hold module will quickl acquire the analog signal at the point of interest and the second sample-and-hold module, connected to the output of the first, will the acquire and hold this stable signal. The second S/H requires longe to acquire the voltage presented by the first device, but since a larg capacitor is used, the droop rate of the output will be low. An example of this cascaded sample-and-hold is shown in Fig. 5-9. You should remember that in this type of a cascaded S/H configuration, the overall sampling rate will be determined by the sum of th sampling rates of both S/H devices.

There is a variety of commercially available sample-and-hold devices that will simplify analog circuit designs. The following device are representative of the many different types available:

Analog Devices, Inc., Norwood, MA 02062

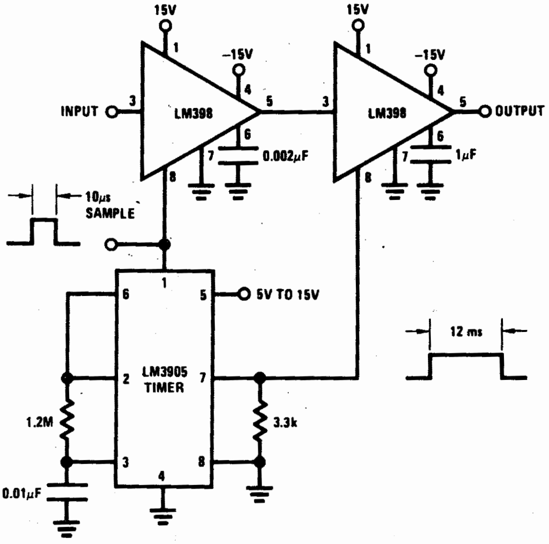
SHA-5	General Purpose
SHA-1A	General Purpose

Burr-Brown Research Corp., Tucson, AZ 85734

SHC80KP	Low Cost
SHM60	High Speed

Datel Systems, Inc., Canton, MA 02021

SHM-LM2	Low Cost, Integrated Circuit
SHM-CM	General Purpose, ±12-volt range



Courtesy National Semiconductor Corp.

Fig. 5-9. Using two sample-and-hold devices in a cascaded configuration for a high acquisition speed and low droop rate.

Hybrid Systems Corp., Bedford, MA 01730
SH703 — Low Cost

Intersil, Inc., Cupertino, CA 95014
IH5110 — Low Cost, Integrated Circuit

National Semiconductor Corp., Santa Clara, CA 95051
LF389 — Low Cost, Integrated Circuit

Teledyne-Philbrick, Dedham, MA 02026
Model 4853 — High Speed

MULTIPLEXERS

In many analog-to-digital converter applications, it becomes too expensive to dedicate one A/D converter to each sensor. An alternate approach is to share one A/D converter among several sensors. This is called *multiplexing,* since many signal sources share a common transmission path to a single receiving device, in this case the A/D converter.

A multiplexer may be as simple as a rotary switch having multiple *taps,* or positions, or as complex as a microwave communication

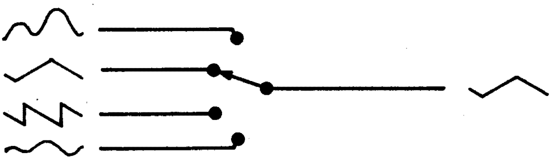


Fig. 5-10. A simple rotary switch multiplexer showing four possible input signals, any one of which may be switched to the output.

multiplexer used by the telephone companies. In each case, the communication path is dedicated to one signal for a short period of time before switching to the next signal to be transmitted. In each case, one and only one signal uses the transmission path at any one time. A multiplexer may be thought of as a simple switch, as shown in Fig. 5-10.

Multiplexers are a relevant topic when discussing A/D converters but our purpose in this section is not to provide a deep understanding of the multiplexer devices themselves. We are interested in some uses of analog switches or multiplexers, particularly those applications which lend themselves to switching a number of analog signals to a single A/D converter. For more details about the devices themselves and their operating characteristics, we suggest two references *Data Conversion Handbook*[2] and *Analog Switches and Their Applications*.[4] You will find that the data sheets for individual multiplexer and analog switch devices also contain useful information about specific applications.

Electromechanical Multiplexers

The simplest analog multiplexers are mechanical switches or patch cables which may be switched by hand to arrange the necessary signal paths. These switching schemes are not very convenient for data-acquisition applications where you may be interested in switching between channels or analog sensors at a rapid rate.

Relays provide a good compromise since they are small and are easy to turn on and turn off with standard transistor-transistor logic (TTL) levels or with standard TTL driver devices such as the SN-75451A dual peripheral driver integrated circuit. Some small-signal reed relays are available in dual in-line packages (DIPs), the size of 14- or 16-pin integrated-circuit devices.

Reed-type relays are generally the ones chosen for use with low-level signals because they are sealed and they have low contact resistance. Mechanical relays have contact bounce, which occurs during contact opening and contact closing and appears to the circuit as short, multiple openings and closings of the contacts. The contact bounce period is short, a few milliseconds, but this can introduce noise in the analog signal if the signal is measured during this time. Low-level currents cannot "clean" the metal contact surface the way higher currents can, so the contact resistance may increase over a period of time. Relays with mercury-wetted contacts are available for low-level signal switching, with the mercury acting to make a clean, low-resistance, metal-to-metal contact which is not appreciably affected by aging.

Reed relays are available in a variety of configurations, such as single-pole, single-throw (spst); single-pole, double-throw (spdt); and so on. Generally, the spst-type are used for single-ended analog signal multiplexing. Prices for reed relays range between $3 and $20, depending upon the quality and the contact configuration needed.

Mechanical devices do have limitations that should be considered. One of the main problems is mechanical wear to the contacts, since the contacts do come under stress during each contact closure. Thus, they have a definite lifetime. Consider a dry mechanical reed relay with a minimum life of 100 million operations. At a rate of 10 samples per second or 10 opening and closing actions per second, there would be a minimum of 3000 hours of operation, or 125 days of operation, before the probability of a contact failure is very great. Granted, this is the *minimum* life, and most relays will last longer. However, this "short" lifetime may be a problem, particularly in remote data-acquisition applications. Reed relays are acceptable for use in many analog multiplexing applications, but they should be considered carefully.

Electronic Multiplexers

There are many different types of semiconductor devices that may be used for signal switching, although metal-oxide semiconductor (MOS) devices have found the greatest use in low-level signal switching or multiplexer applications. The semiconductor fabrication technology is not a part of our discussion, but you should be aware of the terms *complementary-metal-oxide semiconductor* (CMOS) and *metal-oxide semiconductor, field-effect transistor* (MOSFET), two technologies used to manfacture semiconductor signal-switching and multiplexing devices.

Some of the advantages of semiconductor switches over other types of switches that make them practical for use in multiplexers are:

- Small size, standard dual-in-line packages (DIPs)
- Directly compatible with TTL signals for switching
- Built-in, on-board digital decoders for channel select
- Positive and negative signal input (bipolar signals)
- High-speed switching

- Long life, no mechanical wear
- Low contact resistance, less than 100 ohms
- High off-state resistance, typically 10^9 ohms

SEMICONDUCTOR ANALOG MULTIPLEXER CONSIDERATIONS

Semiconductor switches are not ideal devices and they, too, have some limitations or constraints which must be considered prior to their use in multiplexer circuits. Almost all analog switches that use semiconductor devices to perform the switching function require two power supplies, typically +15 volts and −15 volts. Signal inputs cannot exceed these power-supply potentials without damaging the device.

Early semiconductor switches were susceptible to a problem called *latch-up,* which caused them to act as though they were silicon controlled rectifiers (SCRs). Once they were turned on to pass a signal, they refused to turn off until the input signal reached zero volts. Since the signal input could not be counted upon to go to ground potential, the switches remained in the on state indefinitely, or until the power was removed from the system.

Many of the semiconductor devices, particularly the CMOS and MOSFET switches, are easily damaged by static electrical discharges such as those produced by synthetic fabrics, rugs, etc. Newer designs incorporate static-protection devices within the multiplexer integrated circuit.

There are some properties of semiconductor multiplexers which are best described by short definitions:

Crosstalk

This is the measure of the amount of a signal which is being input to an "off" channel, but which appears at the output of the multiplexer, superimposed upon the signal of interest that is passed through the "on" channel. This is a direct function of the frequencies of the signals being input to the multiplexer. Since the switches are capacitively coupled on the integrated circuit, some of the signals "leak" through to other channels. Crosstalk will increase with the frequencies of the signals input to the multiplexer. This is similar to the feedthrough problem associated with sample-and-hold devices.

Settling Time

The settling time measures the time necessary for the multiplexer's output to be within a certain error percentage of the input signal once the channel is selected, or turned on. This must be specified as either the semiconductor switch's switching time plus the analog output settling time, or as the analog output settling time alone.

Throughput Rate

This rate is a measure of the fastest channel-to-channel switching rate which may be used if the rated accuracy, generally 0.01%, is to be accepted. This takes into account the settling time of the device.

Bandwidth

This defines the ability of the multiplexer to pass a signal at a particular frequency once the multiplexer's channel is turned on. The bandwidth is the −3-dB point. This is equivalent to the small-signal bandwidth associated with sample-and-hold devices.

Switching Transients

When a multiplexer is switched from one channel to the next, and one of the channels is turned off, a switching transient occurs. These voltage transients or spikes appear at the multiplexer's output and they may cause inaccurate measurements if the output is sampled, digitized, or integrated during this time. The switching transients may be removed, if necessary, by using a sample-and-hold device between the output of the multiplexer and the input of the measuring device.

MULTIPLEXER SIGNAL INPUTS

There are a wide variety of signal sources that can provide outputs to be multiplexed. These outputs can include low-level thermocouple signals and high-level pressure transducer outputs, dc and ac outputs, high-frequency and low-frequency outputs. These types of signals may all be multiplexed successfully, although some premultiplexer and postmultiplexer signal conditioning may be required.

Low-Level Signals

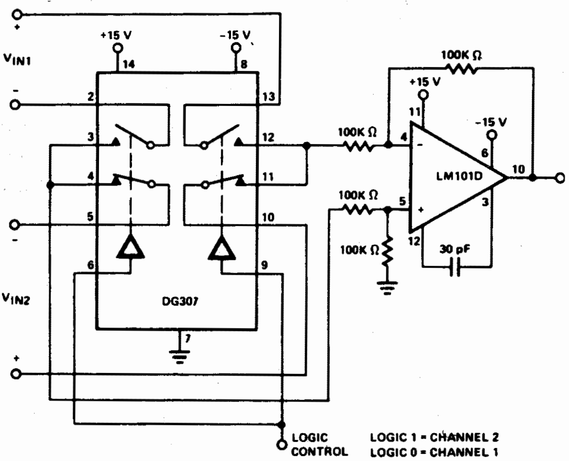
Low-level signals may require amplification before they are input to a multiplexer, since transient signals may be large enough to cause significant errors in the low-level multiplexer output. If necessary, the resulting multiplexed and amplified signals may be attenuated after being multiplexed. If amplification cannot be used due to a wide range of signal inputs, a postmultiplexer filter could be used to remove any unwanted noise generated by switching transients.

Differential Signals

There are often cases where the signal to be measured does not exist as a single voltage that is referenced to ground potential in the usual way. In these cases, the signal exists as the difference between two voltages and it is called a *differential signal*. The unknown signal may have a very low potential when it is compared to the actual voltages present on the two inputs. For example, you may have two signal voltages, 5.37 volts and 5.35 volts. The difference is 0.02 volt, providing the signal that is of interest to us. This signal is very small when compared to the two potentials, 5.37 and 5.35 volts.

A differential-input instrumentation amplifier may be used to recover the difference, yielding a single output, the voltage to be measured. The single output can then be multiplexed and routed to an A/D converter for digitization. When a large number of differential signals are to be measured in this way, it becomes expensive to provide each pair of differential-input lines with its own instrumentation amplifier. A good solution to this problem is to use a *double-pole* multiplexer, which has two independent sets of switches that open and close at the same time. The double-pole multiplexer is used to switch both lines of a differential input signal through the multiplexer to a common differential-input instrumentation amplifier. In this way, a single instrumentation amplifier may be "shared" among many differential signal sources.

An example of differential, or double-pole, multiplexing is shown in Fig. 5-11.[4] In this example, no channel decoding is provided since only two switches are used.



Courtesy Siliconix, Inc.

Fig. 5-11. A two-channel differential-input multiplexer circuit with a differential-input amplifier being used to provide a single-ended output.

The type of configuration shown in Fig. 5-11 may also be used when both differential and nondifferential, or single-ended, signals are being multiplexed and connected to the same signal receiver, whether it is an amplifier, A/D converter, or other device. The single-ended inputs have their second input referenced, or connected, to a local analog ground point.

MULTIPLEXER APPLICATIONS

Analog switches may be used in almost any circuit which requires a voltage switch. Typical applications for analog switches include their use in D/A converters, programmable gain amplifiers, filters, and integrators. Our main interest in these switches centers around their use in analog multiplexers used to switch multiple signal inputs to a common point for amplification and digitization. There are two types of switching devices that we will consider: those without decoders and those with decoders.

Switches Without Decoders

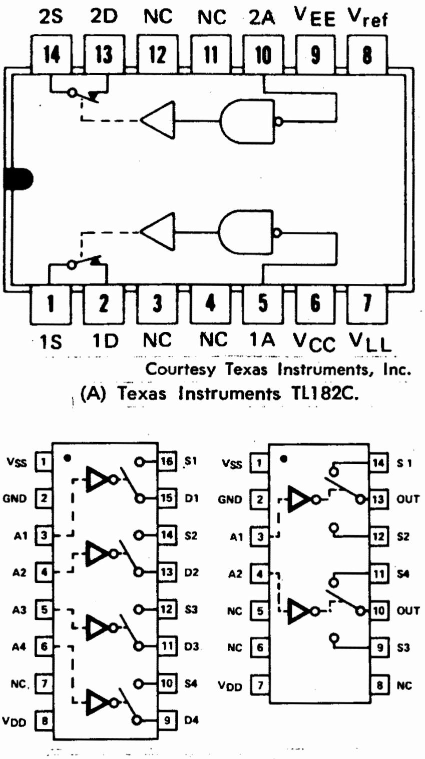
Some analog switches, such as the Texas Instruments TL182C and the Analog Devices 7510, 7511, and 7512 devices, have control inputs for each switch. The pin configurations for these integrated circuits are shown in Fig. 5-12. This type of analog switch requires a logic one or a logic zero to actuate each switch. These switches find use in applications where more than one switch is to be actuated at one time, or where individual switch control may be needed.

Another multiplexing example is shown in Fig. 5-13. In this circuit, four thermocouples are being multiplexed in a differential mode. The two differential multiplexer outputs are connected to a common differential-input amplifier. Each thermocouple is switched and connected to the amplifier by applying the appropriate level to the CK input for the selected thermocouple. The switches are individually controlled, so it is possible in this configuration to connect more than one thermocouple to the amplifier at one time. In this example you should note that although low-level thermocouple signals are being multiplexed, no preamplification or other signal conditioning is used.

Switches With Decoders

Switches used for analog signal multiplexing are generally more useful when they are equipped with built-in or on-chip decoder circuits. These decoder circuits accept a parallel binary input and then turn on, or actuate, the correct switch corresponding to the binary code applied. The binary code can only represent a single value at one time, so only one switch at a time is actuated. Some examples of switches with decoders are shown in Fig. 5-14. The truth tables for these devices are also shown. The AD7506 is a 16-channel device and the AD7507 is an 8-channel device.

When using analog multiplexers with on-chip decoders, it is still the user's responsibility to provide the correct code for the channel



Courtesy Texas Instruments, Inc.

(A) Texas Instruments TL182C.

Courtesy Analog Devices, Inc.

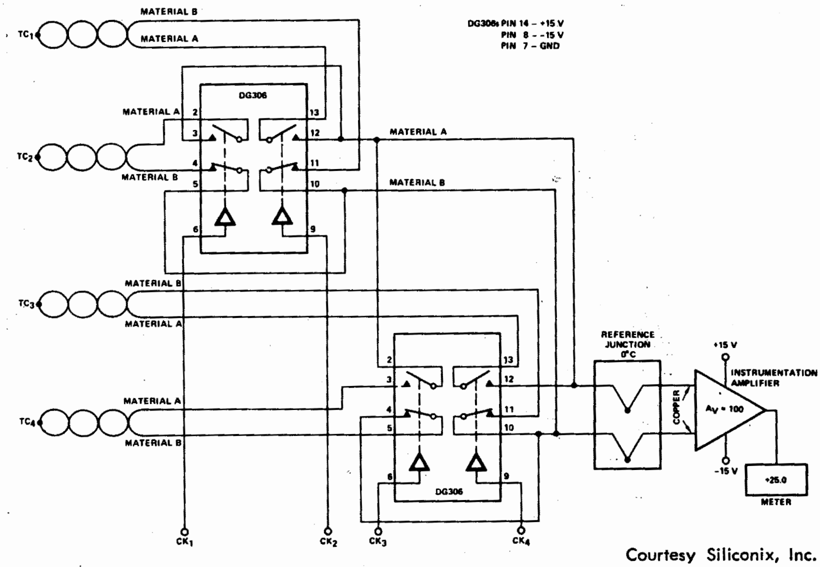
(B) Analog Devices AD7510 series.

Fig. 5-12. Pin configurations for typical multiplexer integrated circuits that do not contain decoding logic.

required. Many decoder circuits also provide an enable connection. This type of an input allows the multiplexer scheme to be expanded to include a larger number of selectable channels. A typical example is the 32-channel multiplexer circuit shown in Fig. 5-15. In this example, a Siliconix DG506 multiplexer is used. Note the use of the enable input, pin 18. This input is used to allow switching between the two multiplexer devices by enabling one and disabling the other. Actually, this is immaterial to the user, since the enable input may be thought of as simply another address input. By using the enable inputs and additional TTL decoder integrated-circuits, the multiplexer may be expanded almost indefinitely.

MULTIPLEXER INTERFACING

In many data-acquisition applications, where a number of analog inputs are to be digitized, multiplexers are used so that a single A/D converter may be shared among the inputs. In this type of a system,



Courtesy Siliconix, Inc.

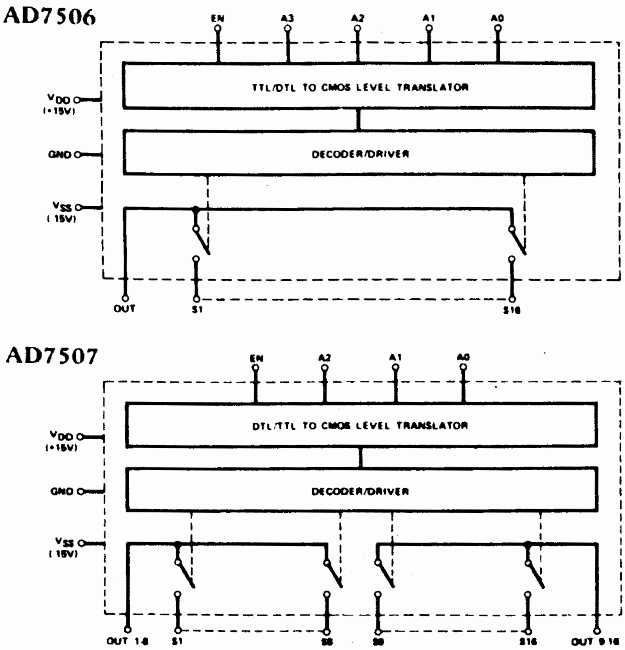
Fig. 5-13. A thermocouple multiplexing example using four thermocouples as the signal sources and an instrumentation amplifier for signal recovery.

it is the responsibility of the designer and programmer to allow the computer to have the ability to select the proper channel of the multiplexer.

The multiplexers with on-chip decoders are probably the wisest choice since we can avoid the need for additional logic for switch selection. It will be necessary, however, to have an output port or latch available so that the channel code may be output by the computer and latched for as long as it is needed. The latch outputs are simply connected to the channel address inputs on the multiplexer device. If the multiplexer scheme shown in Fig. 5-15 is to be used with a computer interface, the latch's outputs would be connected to the address inputs, A4 through A0. These would be connected through the latch to the computer's data bus lines, D4 through D0. The binary data word latched at the output port would control the channel selected.

While a standard latched output port will provide the multiplexer control necessary, there may be a better or more appropriate type of "latch" to use in this type of circuit. An SN74193 programmable, up-down binary counter will provide a number of useful functions not found in a latch alone. These functions are:

- Parallel latching of a count value
- Ability to increment the count
- Ability to decrement the count



(A) Block diagrams.

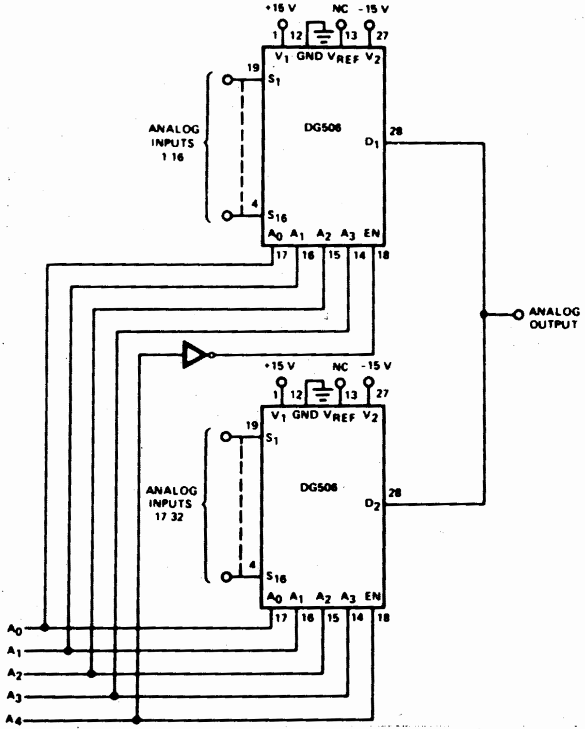
AD7506					
A_3	A_2	A_1	A_0	E_N	"ON"
0	0	0	0	1	1
0	0	0	1	1	2
0	0	1	0	1	3
0	0	1	1	1	4
0	1	0	0	1	5
0	1	0	1	1	6
0	1	1	0	1	7
0	1	1	1	1	8
1	0	0	0	1	9
1	0	0	1	1	10
1	0	1	0	1	11
1	0	1	1	1	12
1	1	0	0	1	13
1	1	0	1	1	14
1	1	1	0	1	15
1	1	1	1	1	16
X	X	X	X	0	None

AD7507				
A_2	A_1	A_0	E_N	"ON"
0	0	0	1	1 & 9
0	0	1	1	2 & 10
0	1	0	1	3 & 11
0	1	1	1	4 & 12
1	0	0	1	5 & 13
1	0	1	1	6 & 14
1	1	0	1	7 & 15
1	1	1	1	8 & 16
X	X	X	0	None

(B) Truth tables.

Courtesy Analog Devices, Inc.

Fig. 5-14. Block diagrams and truth tables for typical multiplexers with on-chip decoders.



Courtesy Siliconix, Inc.

Fig. 5-15. Circuit diagram of a 32-channel analog multiplexer using the Siliconix DG506 multiplexer integrated circuits and an inverter.

- Ability to clear the count to zero

With these capabilities, we may now load a multiplexer channel number in the SN74193 counter, and we may also increment or decrement the count to switch from channel to channel in a sequential fashion. The count may also be cleared to reset the multiplexer to Channel 0. A typical interface for a 16-channel *programmable* multiplexer is shown in Fig. 5-16.

Remember that the multiplexer is under software control and that we can write our programs to take advantage of this flexibility. Software to sequence through the 16 channels is fairly simple; we only need to generate a command to increment the count stored in the SN74193 counter. In some cases, it may be necessary to sample a number of analog channels, but they may not be contiguous or sequential. For example, it may be necessary to scan and digitize the signals present on Channels 9, 3, 5, 2, 9, 3, 5, 2, etc. Software to do this is shown in Example 5-1.

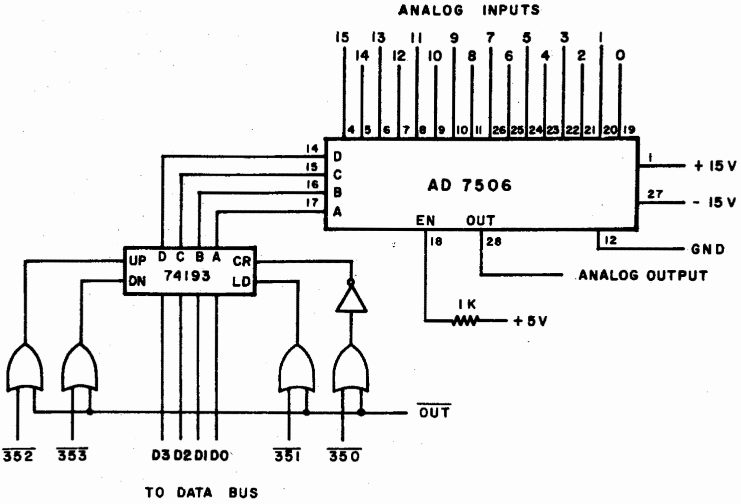


Fig. 5-16. A microcomputer-controlled analog multiplexer using software commands to select and control channel numbers.

```
        /EXAMPLE 5-1
MVIA    /LOAD REGISTER A WITH CHANNEL NUMBER
011     /011=DECIMAL 9
OUT     /OUTPUT IT TO THE MPX
351
CALL    /DO A CONVERSION
ADC
0
MVIA    /LOAD NEXT CHANNEL NUMBER
003
OUT     /OUTPUT IT TO THE MPX
351
 .
 .
ETC
```

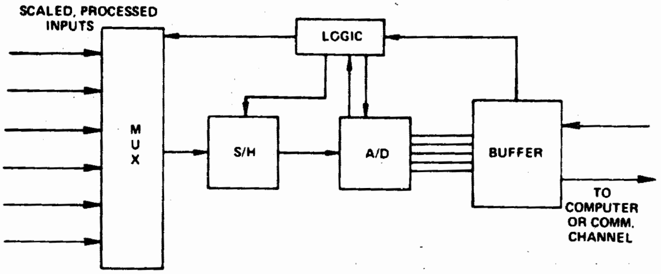
In the software shown in Example 5-1, all of the channels to be scanned are indicated in the main program. It will prove difficult to change the channels to be scanned and the total number of channels to be scanned on a day-by-day basis, particularly if the program sequence is stored in PROM or ROM. A more general approach is to use a *scan table* which indicates to a general-purpose scanning multiplexer subroutine which channels are to be scanned and digitized. A detailed explanation of this type of software is provided in Unit 6.

In the temperature data-acquisition example which was discussed previously, the system might be expanded to include additional temperature sensors plus sensors for humidity, water levels, etc. Each of these sensors would be connected to a multiplexer and the multiplexer's output would be connected to the A/D converter's input. Signal conditioning might be required for some of the sensors, depending upon the voltage or current levels produced by the sensor and the voltage input range of the A/D converter. For signals that might be changing rapidly, sample-and-hold modules might be required, too. In addition, multiple displays of data files for each sensor could be made available and control software could be added to test for specific conditions present at each sensor.

Future developments in the A/D converter field will probably lead to small, prepackaged data-acquisition systems that contain analog multiplexers, sample-and-hold circuits, and fast, accurate A/D converters. Prices for both discrete data-acquisition "black boxes" and monolithic single-chip converter devices continue to fall, so we may also see the advent of inexpensive converters, allowing one A/D converter to be used with a single sensor.

MULTIPLEXERS AND SAMPLE-AND-HOLD SYSTEMS

Multiplexers and sample-and-hold devices are particularly useful when they are used together in data-acquisition systems. Two configurations are possible, one in which a single sample-and-hold device is used between an analog multiplexer and an A/D converter and another in which each channel has its own sample-and-hold device. The first configuration is shown in Fig. 5-17.



Courtesy Analog Devices, Inc.

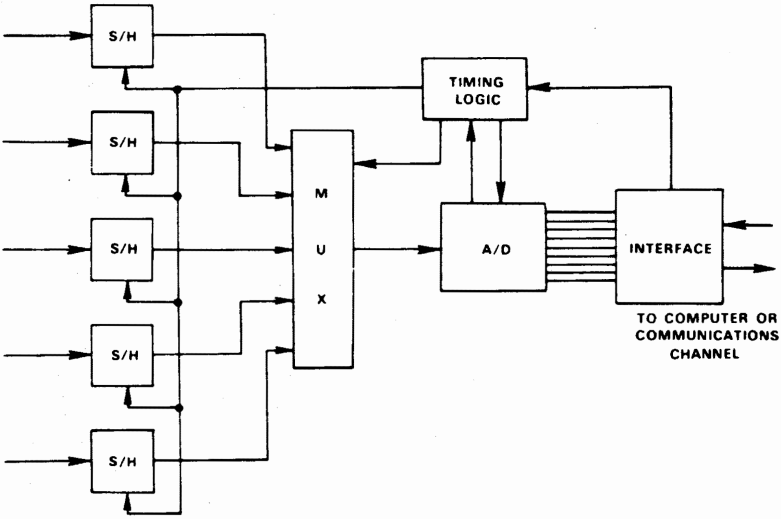
Fig. 5-17. A typical data-acquisition system using a single sample-and-hold module and a multiplexer.

The circuit shown in Fig. 5-17 is an economic solution to a multichannel data-acquisition task. It is easily expanded by adding more multiplexers. The system operates by switching to an active channel of interest, sampling the signal, and digitizing it. The process can, however, introduce tens of microseconds of delay between the actual measurements taking place. This delay is due to the switching time of the multiplexer, the acquisition time and the settling time of the sample-and-hold circuit, and the conversion time required by the A/D converter. The time between samples may be slightly reduced by having the multiplexer switch to the next channel of interest after the sample has been acquired and held by the sample-and-hold circuit.

In a system such as this, the inputs may be digitized sequentially or at random. The random mode of operation will find use in applications where it is necessary to input and digitize data from various channels at different rates.

The second data-acquisition system uses one sample-and-hold device per analog input. This type of configuration is more expensive than the one shown in Fig. 5-17, but it is useful in those applications which require the simultaneous measurement of a number of analog signals. This would be the case in the analysis of transient phenomena or frequency-dependent measurements taken during vibrational testing.

In this second mode of operation, all of the sample-and-hold devices are switched from the sample mode to the hold mode at the



Courtesy Analog Devices, Inc.

Fig. 5-18. A typical multiplexed simultaneous sampling system in which one sample-and-hold device is used with each input channel.

same time. The anaolg signals are thus maintained until they can be digitized in sequence, again using a single A/D converter with an analog multiplexer to switch the many analog inputs to the converter. A typical system is shown in Fig. 5-18.

If the digitizing takes some time to complete, low-droop-rate sample-and-hold devices must be used. The droop rate must be such that the output of the last channel to be digitized does not change by enough to seriously affect the data value associated with its sensor. For example, if 10 channels are to be measured in a simultaneous sampling system, and if the channels are sequenced at 20 microseconds per channel, it will take 180 microseconds until the digitization of the analog value at the last channel is started. If a 10-bit A/D converter is being used, the droop rate must be such that the output of the last sample-and-hold is still within one part in 1024 after 180 microseconds.

In cases such as this, where fast acquisition speeds and low droop rates are required, a cascaded sample-and-hold system such as that previously discussed would be useful. This is illustrated in Fig. 5-9. As you will see in the next unit, the single sample-and-hold device is the solution generally applied to data-acquisition problems.

SIGNAL PROCESSING

While most details of signal processing are beyond the scope of this book, which has focused upon the converter device interfacing and software interaction, it is important that you understand some of the terms associated with signal processing. We also provide some references for you so that you will have further information available if signal processing proves to be important in your particular application.

In this book we have assumed that you can provide a "clean," noise-free signal that is to be digitized with an A/D converter, or that you can use the voltage or the current output by a D/A converter. These are, however, ideal conditions and your application may require filtering to reduce noise, wider dynamic ranges for signals covering several decades, or *ratiometric* conversions where the ratio of one signal to another is important and not the actual values of the two signals.

Noise

Noise comes in many forms and it may be reduced by filtering, integration, signal correlation, etc., or it may be accepted and worked around. In any case, the subject of noise is a complex one, better handled by others. We recommend the *Instrumentation for Scientists Series, Module 4, Optimization of Electronic Measurements*[5] and *The*

Design of Active Filters,[6] both of which are noted among the references used in this unit.

Dynamic Range

The dynamic range, or the number of decades over which the signal varies, may be larger than can be handled with accuracy by many converters. Programmable-gain instrumentation amplifiers may be used to provide the necessary full-scale voltage required by an A/D converter. The gain may be programmed under software control, if necessary, or the gain may be autoranging, as is the case in some digital panel meters. Amplifiers with a logarithmic response could also be used to output, say, 1 volt per decade of input voltage. Thus, a change from 0.01 volt to 0.1 volt at the input would generate a 1-volt change in the logarithmic amplifier's output, as would the change from 0.1 volt to 1.0 volt.

Ratiometric Conversions

It is often necessary to measure the ratio of one signal to another, rather than the actual value of the signals. This can be done by digitizing each value and then performing the ratio or mathematics in software, or by using a ratiometric A/D converter. This type of an A/D converter accepts two inputs and outputs a value which represents the ratio between them.

REFERENCES

1. Sheingold, D. H., ed., *Analog-Digital Conversion Notes,* Analog Devices, Inc., Norwood, MA 02062, 1977.
2. Bruck, D. B., *Data Conversion Handbook,* Hybrid Systems Corporation, Burlington, MA 01803, 1974.
3. National Semiconductor Corporation, *LF-389 Data Sheet,* National Semiconductor Corporation, Santa Clara, CA 95051, 1976.
4. Siliconix, Inc., *Analog Switches and Their Applications,* Siliconix, Inc., Santa Clara, CA 95054, 1976.
5. Malmstadt, H. V., *et al., Instrumentation for Scientists Series, Module 4, Optimization of Electronic Measurements,* W. A. Benjamin, Inc., Menlo Park, CA 94025, 1974.
6. Berlin, H. M., *Design of Active Filters, With Experiments,* Howard W. Sams, & Co., Inc., Indianapolis, IN 46268, 1978.

6

. . . of Bits, Boards, and Black Boxes

INTRODUCTION TO THIS UNIT

In the past few years, analog module manufacturers have made the analog-to-digital converter interfacing task easier by incorporating interface-like devices such as three-state outputs and latched inputs, within the converter modules. With the introduction of sets of microcomputer modules from companies such as Control Logic and Pro-Log and the introduction of single board computers, such as those available from Intel, Motorola, and National Semiconductor, module manufacturers have also started to provide complete, ready-to-use analog interfaces that are compatible with the various bus signals. These boards or cards provide an easy way to add digital and analog converter capabilities to small computer systems.

BLACK BOXES—DATA-ACQUISITION MODULES

Modular data-acquisition packages or devices generally provide a complete analog-to-digital converter interface for microcomputer or minicomputer systems. Except in those instances where special needs are not easily met, you will find it less expensive to purchase a data-acquisition module than to attempt to construct one yourself.

Data-acquisition modules currently cost about $300 each, although the current trend indicates that the modules will become smaller and less expensive as semiconductor manufacturers start to integrate the necessary circuitry within a single integrated-circuit package. The ata-acquisition modules currently available have the following features:

- 16-channel, single-ended analog inputs or 8-channel differential analog inputs
- Complete multiplexer control for random or sequential channel selection
- Internal differential amplifier and sample-and-hold
- 12-bit binary A/D converter
- Internal logic for timing and control
- Typically a 7.5-by-12.5 centimeter package
- Conversion rates of up to 50 kHz

All of the necessary control circuitry is contained within the module, including the control for the sample-and-hold and A/D converter. The user must supply the connections to the microcomputer's data and control bus signals.

Data-acquisition modules are available from a number of companies. A representative sample of manufacturers and the modules that they produce is listed as follows:

ADAC Corporation, Woburn, MA 01801

ADAM-12	16-Channel, 12-bit A/D Converter

Analog Devices, Inc., Norwood, MA 02062

DAS1128	16-Channel, 12-bit A/D Converter

Analogic Corporation, Wakefield, MA 01880

MP6812	16-Channel, 12-bit A/D Converter

Burr-Brown Research Corp., Tucson, AZ 85734

SDM853	16-Channel, 12-bit A/D Converter
MP-20	16-Channel, 8-bit A/D Converter

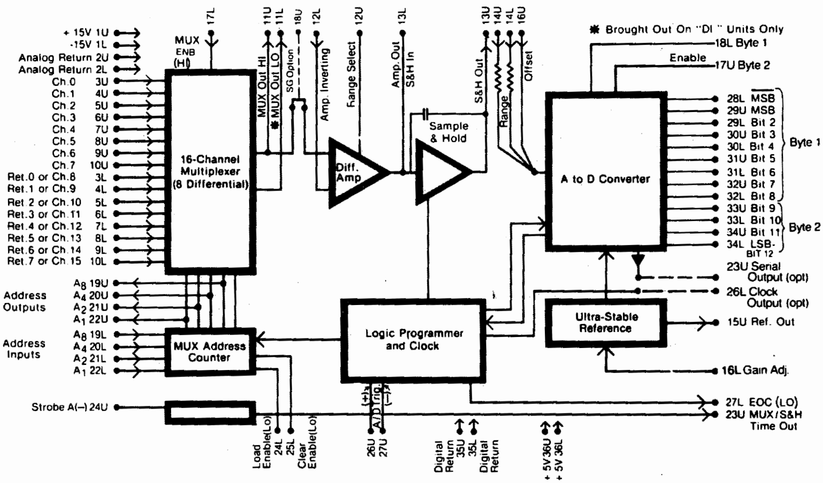
Data Translation, Inc., Framingham, MA 01701

DT5701	16-Channel, 12-bit A/D Converter
DT820	8-Channel, 8-bit A/D Converter

Datel Systems, Inc., Canton, MA 02021

MDAS-16	16-Channel, 12-bit A/D Converter
MDAS-8D	8-Channel differential, 12-bit A/D

There are a number of other data-acquisition-module manufacturers that we have not been able to list in the chart above. Many of the manufacturers listed have a variety of different modules with fea-



Courtesy Data Translation, Inc.

Fig. 6-1. A typical 16-channel data-acquisition module with a 12-bit A/D converter and control logic.

tures such as differential inputs, multiplexer expanders, etc. They are too numerous to list here.

A typical data-acquisition module is shown in Fig. 6-1. You should be able to quickly identify the analog multiplexer, the sample-and-hold, and the 12-bit analog-to-digital converter sections. You will also notice that a programmable counter has been used to supply the multiplexer's 4-bit address. A differential amplifier is also supplied so that this particular module may be used in either the single-ended or differential mode.

There are some important characteristics of data-acquisition modules that should be considered before one is chosen for use in an interface. Some questions, which you may find important, are listed as follows:

1. Are all of the internal devices connected and ready to use?

   Some modules may have uncommitted multiplexers, sample-and-hold devices, and A/D converters. In this case, you can connect the devices as they are needed, but this may make the module difficult to use, particularly if this is a first attempt at A/D converter interfacing. Most module manufacturers supply detailed application information that covers the set-up and use of their devices.
2. Are the inputs and outputs compatible with the signals provided and required by the microcomputer?

Some of the available modules have three-state parallel data output lines that are compatible with most microcomputer data buses. This is the case for the DATAX module shown in Fig. 6-1. This module also has individual byte control lines so that the 12-bit data from the A/D converter may be input in an 8-bit byte and a 4-bit byte.

Modules such as the Analog Devices DAS1128 and the Data Translation DT820 do not have three-state outputs. This makes them slightly more complicated to interface, since you must supply the three-state buffers that will enable the data-acquisition modules to be compatible with the microcomputer's three-state bus. Devices such as the 8212 and DM8095 (SN74365) are not expensive, but device decoders and other circuit elements will be needed, too, adding to the total interface circuitry required.

3. Are the power supply requirements met within your present microcomputer system?

   Many microcomputer and microprocessor vendors place a great deal of emphasis on the use of a single power supply, generally +5 volts, to power a small microcomputer system. If this is the case in your microcomputer system, you will have to add power supplies for the +15 volts and −15 volts generally required by data-acquisition modules. Almost all of the data-acquisition modules will require +5 volts, but to avoid power supply noise you may wish to include a separate power supply for the +5 volts needed by the module. Our experience has shown that open-frame power supplies often require additional filtering if they are to be used with a high-accuracy data-acquisition module.
4. Can the number of analog channels be expanded? Can the inputs be used in the single-ended mode or in the differential mode without having to change to a different module?

   Many modules can be expanded to include additional analog channels. This usually requires an additional multiplexer module. There is a limit of about 64 channels that may be accommodated by a data-acquisition module. Most modules may be used in either the single-ended or differential modes without changing to another device. Some modules do require additional circuitry if you wish to expand to more analog channels or if you wish to use differential inputs instead of single-ended inputs.

DATA-ACQUISITION MODULE INTERFACING

Interfacing data-acquisition modules to microcomputers is not difficult, particularly when a module with three-state data outputs has been selected for use. Modules without the three-state outputs will require additional three-state buffers so that they can be connected to the microcompupter's data bus. Fig. 6-2 shows a typical interface in which an ADAC Corporation ADAM-12 module has been used.

The ADAM-12 interface provides address decoding for accumulator I/O, although memory-mapped I/O could have been used. A 6-bit binary comparator integrated circuit, the DM8160, has been chosen to help simplify the address decoding shown in Fig. 6-2. The only connections to the microcomputer are the 8-bit data bus, the 8-bit low-address bus, and the two control signals: $\overline{\text{IN}}$ and $\overline{\text{OUT}}$.

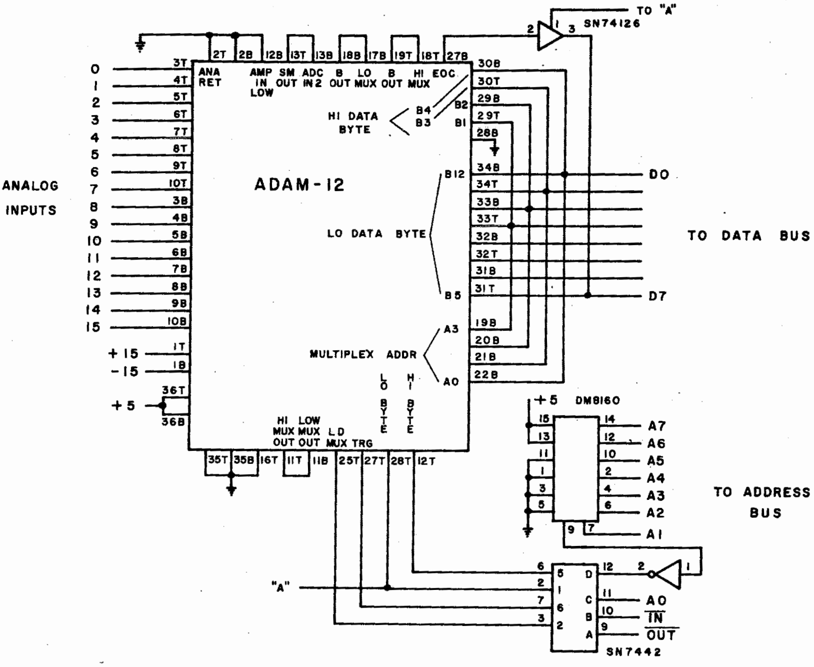


Fig. 6-2. An interface for an ADAM-12 data-acquisition module.

The four control signals and their actions are listed as follows:

OUT 300 Load the four least significant bits of data in the 8080's A register (accumulator) into the multiplexer channel select latch.

IN 300 Start a conversion process within the ADAM-12.

OUT 301 Input the eight least significant bits of the A/D converter's data.

IN 301 Input the four most significant bits of the A/D converter's data and the end-of-conversion flag (in position D7).

You should observe that only four additional integrated circuits are required in addition to the ADAM-12 module. These are an SN7442 decoder, an SN7404 inverter, an SN74126 three-state buffer, and the DM8160 comparator.

DATA-ACQUISITION SOFTWARE

The software necessary to control the ADAM-12 module is very similar to the program required to control a 10-bit A/D converter. The main difference is that a multiplexer address of 0 through 15 must be entered into the module prior to the start of a conversion. When only one of the 16 channels is to be used, the multiplexer channel address can be loaded into the module at the start of the program. There will be no need to change or update it as the program executes.

It is interesting to note that the data output by the ADAM-12 module is actually the inverse of what we would expect. Obviously, there is an inversion taking place somewhere in the system. This could be corrected by using additional circuitry to perform the data inversion, but a software command can perform the same function. We will use the complement-register-A instruction, CMA.

```
                             /EXAMPLE 6-1
                             /DATA ACQUISITION SOFTWARE FOR AN ADAC CORP.
                             /ADAM-12 MODULE

                             *030 000
030 000 061   START,         LXISP      /LOAD STACK POINTER
030 001 377                  377
030 002 030                  030
030 003 303                  JMP        /JUMP OVER DELAY FIRST TIME THRU
030 004 011                  LOOP+3
030 005 030                  0
030 006 315   LOOP,          CALL       /CALL A TIME DELAY
030 007 250                  DELAY
030 010 030                  0
030 011 016                  MVIC       /POINT COUNTER
030 012 020                  020        /020=16 DECIMAL
030 013 021                  LXID       /SET UP STORAGE ADDRESS
030 014 202                  STORE
030 015 030                  0
030 016 052                  LHLD       /GET STATUS WORDS TO H & L
```

```
030 017 200             STATUS
030 020 030             0
030 021 051   NEXT,     DADH      /THIS IS A 16 BIT ROTATION
030 022 322             JNC
030 023 031             OVER
030 024 030             0
030 025 043             INXH      /IF OVERFLOW, ROTATE A 1 INTO LSB
030 026 303             JMP       /IF STATUS IS A 1, GET DATA
030 027 042             CONVRT
030 030 030             0
030 031 023   OVER,     INXD      /IF NOT, MOVE ADDR POINTER
030 032 023             INXD      /FOR THE NEXT POINT
030 033 015             DCRC      /ALL POINTS DONE?
030 034 302             JNZ       /NO, DO NEXT ONE
030 035 021             NEXT
030 036 030             0
030 037 303             JMP       /YES, REINITIALIZE
030 040 006             LOOP
030 041 030             0

                        /ADC SOFTWARE FOR CONTROLLING
                        /THE ADAM-12 MODULE

030 042 015   CONVRT,   DCRC      /DECREMENT THE COUNT
030 043 171             MOVAC     /MOVE IT TO A
030 044 323             OUT       /OUTPUT IT AS THE MPX ADDR
030 045 300             300
030 046 323             OUT       /START A CONVERSION
030 047 301             301
030 050 333   EOC,      IN        /INPUT ADC STATUS AND 4 MSB'S
030 051 301             301
030 052 267             ORAA      /SET THE FLAGS
030 053 372             JM        /NOT DONE, CHECK AGAIN
030 054 050             EOC,
030 055 030             0
030 056 057             CMA       /DONE, COMPLEMENT DATA
030 057 346             ANI       /MASK OUT UNUSED BITS
030 060 017             017
030 061 107             MOVBA     /STORE DATA IN B
030 062 333             IN        /INPUT 8 LSB'S
030 063 300             300
030 064 057             CMA
030 065 022             STAXD     /STORE DATA
030 066 023             INXD
030 067 170             MOVAB
030 070 022             STAXD
030 071 023             INXD
030 072 171             MOVAC     /CHECK COUNT
030 073 247             ANAA      /SET FLAGS
030 074 312             JZ
030 075 006             LOOP
030 076 030             0
030 077 303             JMP
030 100 021             NEXT
030 101 030             0
```

```
                                 /DATA STORAGE AREA
                                 *030 200
020 200 000  STATUS,  0          /STATUS BITS FOR CHANNELS 7-0
030 201 000           0          /STATUS BITS FOR CHANNELS 15-8
030 202 000  STORE,   0          /DATA STORAGE AREA STARTS HERE
                                 /AND USES 32 LOCATIONS
                      /STANDARD TIME DELAY SUBROUTINE
                      *030 250
030 250 365  DELAY,   PUSHPSW
030 251 325           PUSHD
030 252 021           LXID
030 253 000           000        /TIMING BYTES
030 254 110           110
030 255 033  DEC,     DCXD
030 256 172           MOVAD
030 257 263           ORAE
030 260 302           JNZ
030 261 255           DEC
030 262 030           0
030 263 321           POPD
030 264 361           POPPSW
030 265 311           RET
```

The program shown in Example 6-1 shows how a microcomputer could be used to input and store data from up to 16 channels. Each of the analog channels may be active (on) or inactive (off), as required for a specific application. The on and off channels are represented by a 16-bit status word stored in two consecutive bytes of read/write memory. The status word may be established prior to the use of the data-acquisition program. Switch or teletypewriter inputs are an effective means of establishing the on and the off channels. Of course, software commands are used to set up the 16-bit status word. Software to do this is not shown in Example 6-1.

The 12-bit data obtained from the A/D converter will be stored in two consecutive read/write memory locations associated with a particular channel. If a channel is inactive, no new data is stored. The channels will be scanned in sequence, Channel 15 through Channel 0 and then back to Channel 15 again.

A call to a time-delay subroutine, DELAY, has been provided so that the active channels may be scanned once each delay period. The actual time-delay period is determined by the two timing bytes noted in the DELAY subroutine. These bytes may be changed for a particular application. If, for example, the time delay is set up for a one-second period, all active channels will be sampled quickly and then the computer will "wait" one second before sampling them again.

An external interrupt-based timer or a flag timer could also be used to indicate when the conversions and scans are to be performed, but these are not shown in the example.

OTHER DATA-ACQUISITION MODULES

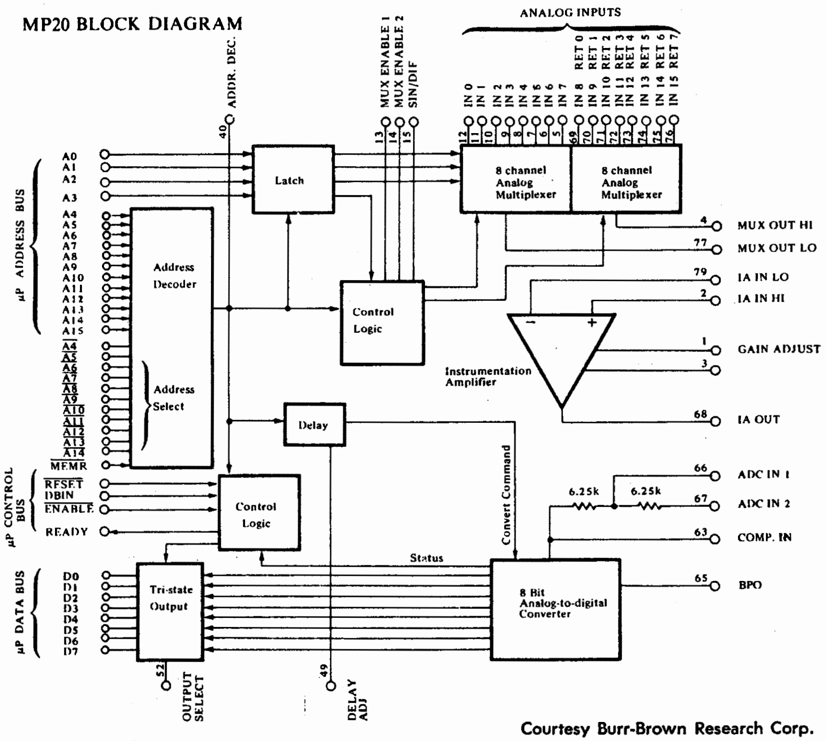
Another type of data-acquisition module is the Burr-Brown Research Corporation's MP-20 device. This is a small module designed specifically to interface with 8080-type microprocessor chips such as the 8080A, 8085, and Z80. The MP-20 is smaller than most other "black box" modules that have been mentioned so far. The MP-20 has some specific differences that are important to note:

- The MP-20 has internal device decoding logic so that it may be connected directly to the 8080's address bus. The actual device addresses used with the module are selected by the user by applying logic zero or logic one levels to 11 address-select comparator pins.
- The memory-mapped I/O technique is used to control the device. This allows all of the memory reference instructions to be used. The MP-20 could also be used with accumulator I/O if this was desired.
- The A/D converter has eight bits of resolution; a sample-and-hold circuit is not incorporated within the package. Sample-and-hold devices could be added to each channel, but this would require additional control circuitry.
- The converter's status is indicated by a single output, READY. This output may be connected directly to the 8080's READY input, thus placing the 8080 in a WAIT state until the conversion has been completed. This means that the 8080 cannot process any other software steps while waiting for the end of conversion. The MP-20's READY could also be used as a flag for software or interrupt control. The READY output is not three-state.

A block diagram of the MP-20 is shown in Fig. 6-3. Note that the multiplexer, amplifier, and A/D converter are not connected to each other. It is our opinion that the MP-20 will find application in many microcomputer systems where eight bits of resolution and the lack of a sample-and-hold circuit are acceptable. Burr-Brown has a detailed application note that covers the MP-20 device. An equivalent device, the MP-21, is available for 6800-type microcomputers.

BOARDS: PLUG-IN ANALOG INPUT AND OUTPUT

Many of the analog converter manufacturers and the microcomputer manufacturers now have completely interfaced analog input/output boards that are *plug compatible* with many of the existing microcomputer systems. This means that the analog/digital inter-



Courtesy Burr-Brown Research Corp.

Fig. 6-3. A block diagram of the MP-20 16-channel data-acquisition module.

facing task is often reduced to the selection, purchase, and plug-in of the necessary analog I/O board.

These converter interfaces are available in either memory-mapped or accumulator I/O configurations. Some are available with both types of I/O, the choice being made by the user through the selection of jumper options made with short jumper wires.

Some of the A/D and D/A converter interface board manufacturers are listed as follows. The boards listed are compatible with the single-board computer bus developed by Intel Corporation for their SBC-80/10 and SBC-80/20 microcomputers.

ADAC Corporation, Woburn, MA 01801

735 Series	16-Channel, 12-bit A/D, two 12-bit D/A converters, and an on-board clock

Analog Devices, Inc., Norwood, MA 02062

RTI-1200	16-Channel, 12-bit A/D, two 12-bit D/A converters, 2708 PROM socket, real-time clock, and control outputs

Burr-Brown Research Corporation, Tucson, AZ 85734

MP8616	16-Channel, 12-bit A/D Converter
MP8616AO	MP8616 with an 8-bit D/A

Data Translation, Inc., Framingham, MA 01701

DT1571	16-Channel, 12-bit A/D, two 12-bit D/A converters

Datel Systems, Inc., Canton, MA 02021

ST-80016D	16-Channel (differential), 12-bit A/D converter
ST-800DA4	Four 12-bit D/A converters

We believe that these analog/digital interface boards are representative of the many 8080-compatible boards available. Portions of the data sheets for the Analog Devices RTI-1200 and the Burr-Brown MP8600 series are provided in the appendices.

Most of the analog I/O boards will require extensive software to get them to operate effectively. As you are probably aware, software takes the greatest portion of the time devoted to interfacing. This is also true when interfaces are provided for you in plug-in form. Most of the user's manuals provided with analog I/O boards are seriously lacking in the area of software support. Some suppliers provide set-up and test routines, but few provide any additional software examples that might be useful when you attempt to use the interface. This is unfortunate.

One exception is the user's manual for the Analog Devices RTI-1200 interface. This manual contains over a dozen software examples which are useful during both set-up and calibration and during the actual application of the RTI-1200. The manual itself is over 50 pages.

The available analog I/O boards provide an easy means of interfacing analog signals to microcomputer systems. They simplify the hardware design, but their use must be carefully evaluated. Their drawback is that they may be too complex and too expensive for the problem at hand, thus slowing down the overall effort. The software to operate the board may also be complex, taking longer to develop than would the software for a simple A/D converter.

BITS . . . AND THINGS

We are frequently faced with microcomputer systems in which there is a variety of converter bit-lengths. Eight-, 10-, or 12-bit digital-to-analog and analog-to-digital converters may be used together in a system where different resolutions are required for both analog inputs and analog outputs. For example, data may be acquired using an 8-bit A/D converter and then displayed using a 10-bit D/A converter. The question quickly arises, what do we do with the bits? In the situation where the various converters have the same number of bits, there is not a problem. Where there are different bit-lengths, there are some special considerations.

Input Greater Than Output

In the case where the number of bits that are input from an A/D converter is greater than the number of data bits available at the D/A converter that will be used to output the data, there is no great difficulty in treating the data.

A typical example is posed by a system having a 10-bit A/D converter and an 8-bit D/A converter, which will be used to display or plot the data. In this case, it is easy for the computer to "ignore" the least significant bits of the 10-bit data word. The D/A converter's output will not have as great a resolution as the A/D converter that was used to acquire the data. Even if the two least significant bits of the 10-bit word are ignored, the signal output will still be in error by less than +1 least significant bit of the 8-bit word:

10-bit A/D converter's input value 0111110011
8-bit D/A converter's output value 01111100

The two least significant bits of the 10-bit data word, 11, have been ignored by the D/A converter. This bit manipulation was performed by the programs shown in Examples 6-2 and 6-3.

```
                        /EXAMPLE 6-2 ROUTINE TO CONVERT 10-BIT DATA
                        /TO 8-BIT DATA FOR D/A OUTPUT
                        *003 100
003 100 170             MOVAB   /GET 2 MSB'S
003 101 037             RAR     /ROTATE RIGHT INTO CARRY
003 102 107             MOVBA   /PUT RESULT BACK
003 103 171             MOVAC   /GET 8 LSB'S
003 104 037             RAR     /ROTATE PREVIOUS CARRY INTO A
003 105 117             MOVCA   /PUT RESULT BACK
003 106 170             MOVAB   /DO IT AGAIN FOR NEXT BIT
003 107 037             RAR
003 110 107             MOVBA
003 111 171             MOVAC
003 112 037             RAR
003 113 117             MOVCA
```

```
                        /EXAMPLE 6-3 ROUTINE TO CONVERT 10-BIT DATA
                        /TO 8-BIT DATA FOR D/A OUTPUT, USING ROTATE
                        /AND MASKING INSTRUCTIONS
                        *003 100
003 100 171             MOVAC   /GET THE 8 LSB'S
```

```
003 101 017        RRC       /ROTATE THE DATA RIGHT
003 102 017        RRC       /ROTATE IT AGAIN
003 103 346        ANI       /MASK OUT THE 2 MSB'S WHICH
003 104 077        077       /ARE OLD D0 AND D1
003 105 117        MOVCA     /STORE IT BACK IN REG C
003 106 170        MOVAB     /GET THE 2 MSB'S
003 107 017        RRC       /ROTATE IT, TOO
003 110 017        RRC
003 111 346        ANI       /MASK OUT ALL BUT THE
003 112 300        300       /2 MSB'S
003 113 261        ORAC      /OR IT WITH REG C
003 114 117        MOVCA     /STORE RESULT IN REG C
```

In Example 6-2, the 10 bits of data are simply rotated to the right, eliminating bits D1 and D0 of the original data word. The carry bit is used to transfer bits from the 8080's B register to the C register during the rotate instructions.

Example 6-3 works in a similar fashion, except that the two 8-bit portions of the 10-bit data word are rotated independently and then finally combined into one 8-bit word through the use of logical operations. Remember that even though only two bits of one 8-bit data word are used, the entire 8-bit word is processed by 8080 logical instructions. These two most significant bits are stored in the 8080's B register, bits D1 and D0. The remaining eight bits are stored in register C. You should be able to use paper and pencil to follow through the sequence of program steps in Examples 6-2 and 6-3.

A more complicated software routine is required if we wish to "round" the 8-bit result so that it more accurately reflects the value of the original 10-bit data word that is being truncated. Before we examine the software that may be used, we should decide exactly what needs to be done. Here are four 10-bit numbers, all of which have the same most significant eight bits:

$01111100\underline{00}$ $01111100\underline{01}$ $01111100\underline{10}$ $01111100\underline{11}$

As you can see, the two least significant bits can occur in any one of four possible conditions without affecting the eight most significant bits. We will use the following rules for rounding the most significant eight bits, based upon the value of the two least significant bits. The following rounding rules will apply:

Two LSBs	*Rounding*
0 0	None
0 1	None
1 0	Add 1 to the 8 MSBs
1 1	Add 1 to the 8 MSBs

Actually, the software to do this is fairly simple, since rounding will be accomplished simply by adding the value 0000000010_2 to the 10-bit data value. The addition performs the rounding in all four cases:

```
  0111110000    0111110001    0111110010    0111110011
 +0000000010   +0000000010   +0000000010   +0000000010
  ----------    ----------    ----------    ----------
  0111110010    0111110011    0111110100    0111110101
```

Only in the two cases where the two least significant bits are 10 and 11 does the rounding actually change the value of the eight most significant bits. After the addition is completed, the 10-bit result may be treated as per Examples 6-2 and 6-3. The rounding results in a better representation of the data values input from a 10-bit source and output to an 8-bit destination.

There is, however, a limiting case that must be tested for in the rounding program. Suppose that we had the values 1111111110 or 1111111111. Should they be rounded? If they are, the resulting 8-bit value will be 00000000 in both cases, with an overflow occurring as the result of the addition of 0000000010 to the data. This is unacceptable, since the rounding would actually change the two greatest values to zero.

This problem may be avoided by testing for the overflow that would be caused by the addition of 0000000010 to either 1111111110 or 1111111111. If an overflow occurs, it would be indicated by the 11th bit, or if we are still using registers B and C, by bit D2 in register B. The software shown in Example 6-4 performs the rounding, the rotations, and the rounding check. If an overflow is detected, the result is decremented by one to restore it to 11111111.

```
                       /EXAMPLE 6-4 ROUTINE TO ROUND A 10-BIT NUMBER
                       /TO BE OUTPUT TO AN 8-BIT D/A CONVERTER
                       *003 100
003 100 003            INXB       /ADD 2 TO THE 10-BIT NUMBER
003 101 003            INXB
003 102 170            MOVAB      /THIS IS THE SAME AS EX 6-2
003 103 037            RAR        /IT ROTATES THE NUMBER
003 104 107            MOVBA
003 105 171            MOVAC
003 106 037            RAR
003 107 117            MOVCA
003 110 170            MOVAB
003 111 037            RAR
003 112 107            MOVBA
003 113 171            MOVAC
003 114 037            RAR
003 115 117            MOVCA
003 116 170            MOVAB      /GET WHAT IS LEFT OF THE 2 MSB'S
003 117 037            RAR        /ROTATE ANY CARRY FROM ADD'N INTO
003 120 322            JNC        /INTO CARRY BIT
003 121 124            OK         /IF NO CARRY, ROUNDING IS OK
```

```
003 122 003          0
003 123 015          DCRC      /DECREMENT C IF ROUNDING WENT TOO FAR
003 124 000   OK,    NOP       /ROUTINE ENDS HERE
```

Output Greater Than Input

This case occurs when the input device has less resolution than the output device. Perhaps a signal is digitized to eight bits of resolution, but it is necessary to output the data to a 10-bit D/A converter. Data from a 10-bit A/D converter will also be output on the 10-bit D/A converter and we would like the data from the 10-bit and the 8-bit A/D converter to have the same full-scale output when they are displayed. This will make comparisons of data values easy.

The eight bits of data generated by the 8-bit A/D converter may be positioned in any of three places within a 10-bit data word. We will assume that the bits are not going to be "split" apart. This is shown in the following groups of 10-bit words:

00XXXXXXXX or 0XXXXXXXX0 or XXXXXXXX00

Here the Xs represent the eight bits of the 8-bit data word. If the data is output to the 10-bit D/A converter as shown in the left-most example, the D/A converter's output will only reach one-quarter of its full-scale output, even for the largest 8-bit data word, 11111111, since it will be output as 0011111111. This limitation could be overcome by installing an amplifier with a gain of four on the output of the D/A converter, but this would also amplify the output from the normal 10-bit values. This is not acceptable.

The middle case also has limitations, since the maximum value would be output as only one-half of the full-scale output available from the 10-bit D/A converter. Moving the eight bits two positions to the left makes the most sense. This approaches the full-scale output of the 10-bit D/A converter. In fact, it comes to within three voltage steps of it.

Moving the eight bits to the left to form a 10-bit data word should not be a problem for you. Rotate instructions such as those used in Example 6-2 may be used with the rotation direction changed from right to left. The 8-bit data output is still accurate to one part in 256 and it is now readily compared to signals that have equal full-scale voltages, but that have been digitized to 10-bit resoluion.

7

Experiments With Digital-to-Analog and Analog-to-Digital Converters

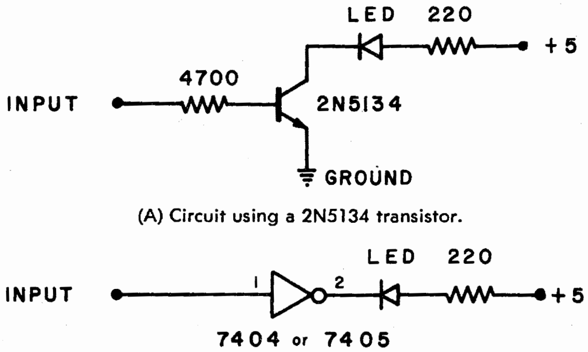
INTRODUCTION TO THE EXPERIMENTS

We have chosen to set aside a special unit for the D/A converter and A/D converter experiments. The main reason for doing this is that it will save you time when performing a series of the experiments. Many of the circuits and software steps are used in slightly different configurations in sequential experiments. With the experiments located in one section of this book, it is easy to go from one experiment to the next without having to dismantle your equipment only to rewire it again in the next unit.

Please heed the notes at the end of some of the experiments. These notes will tell you what should be saved and not removed from your breadboard as you go on to the next experiment. If portions of programs are to be saved, remember to leave the power applied to your computer.

Unlike some of the experiments in the other books of the *Blacksburg Continuing Education Series,* the experiments in this book will require some extensive circuit breadboarding and some long programs to operate correctly. Please take extra care when loading programs and wiring the interface circuits. This will decrease the amount of time spent debugging your program and circuit when they do not work as they should. To simplify some of the circuitry, we have developed the LR-35 Outboard® which provides you with a double-buffered 10-bit digital-to-analog converter with all of the necessary support circuitry. This unit is particularly useful with the popular breadboarding sockets such as the SK-10 socket. Both the breadboarding sockets and the LR-35 Outboard are available from E&L Instruments, Inc., Derby, CT 06418.

If you would rather build or breadboard the 10-bit A/D converter yourself, the complete circuit for the LR-35 is shown in Fig. 7-1. We have used an Analog Devices AD7522 10-bit D/A converter device on the LR-35. An equivalent double-buffered, 10-bit D/A converter may be substituted in the experiments if this is preferable to you.



(A) Circuit using a 2N5134 transistor.

(B) Circuit using an SN7404 or SN7405 integrated circuit.

Fig. 7-1. Two typical lamp monitor circuits.

Many of the experiments will reference specific hardware that is used in the interface circuit. Please feel free to implement equivalent circuits if you cannot obtain the parts specified. The examples and figures provided in previous units will provide you with good design examples that you should be able to use.

Some of the experiments use two output ports, each of which is equipped with eight light-emitting diodes (LEDSs) so that binary data may be output under software control. If this type of output port is not available on your computer, we suggest that you implement some latch circuits using one of the many schemes outlined in *The 8080A Bugbook* or *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 2,* published by Howard W. Sams & Co., Inc. The two ports that we used were assigned device addresses of 002 and 000 (02 and 00 hexadecimal).

The lamp monitor specified in the schematic for the circuit used in Experiment 3 may be one section of an LR-6 Lamp Monitor Outboard, or it may be a simple LED with an appropriate driver. Many such circuits are shown in Unit 6 *Logic & Memory Experiments Using TTL Integrated Circuits, Book 2,* published by Howard W. Sams & Co., Inc. Two typical lamp monitor circuits are shown in Fig. 7-1. Either may be used. If you use an SN7404 or an SN7405, pin 14 must be connected to +5 volts and pin 7 must be connected to ground to supply power to the integrated circuit.

The experiments have been configured so that they are readily implemented on most 8080-based microcomputer/microprocessor systems. We used an E&L Instruments Mini-Micro Designer (MMD-1) computer, which has an 8-bit bidirectional data bus (uninverted) and a 16-bit unidirectional address bus. Some computers, such as the Intel SBC-80/10, have inverted data and address buses. Some systems will also require additional handshaking circuitry that is not implemented in our experiments.

We have used the notation $\overline{\text{IN}}$ for the computer's I/O read signal and the notation $\overline{\text{OUT}}$ for the computer's I/O write signal. These signals are only generated by accumulator I/O transfer commands. They are equivalent to the $\overline{\text{I/OR}}$ and $\overline{\text{I/OW}}$ signals present in other computer systems. The experiments use device codes 003, 004, 005, and 006. The MMD-1 computer that we used had an I/O decoder section that provided ready access to these device addresses. If your computer does not have a device decoder that will provide you with these device codes, you may wish to implement the simple device decoder shown in Fig. 7-2. This provides device codes 000 through 007. If these device codes are already in use in your microcomputer system, you can readily change the device addresses used in the programs that are provided with each experiment.

In Experiments 7, 8, and 9, a Burr-Brown MP-10 module may be used. This device provides two 8-bit D/A converters in one package. Device decoding is provided for within the package. We have used device codes 360, 361, and 363 in Experiments 7, 8, and 9. If you choose to use two other 8-bit converters in place of the MP-10 module, you may require a device decoder for these device addresses. Substituting the circuit in Fig. 7-3 for the equivalent five-input "gate" shown previously in Fig. 7-2 for the device decoding scheme for device addresses 000 through 007 will provide you with device addresses 360 through 367. These addresses are shown in parentheses in the schematic circuit diagram in Fig. 7-2.

If your computer's read/write memory, which is available for your use, does not have a section located between addresses 002 000 and 003 377 (0200 and 0FFF hexadecimal) you have two choices.

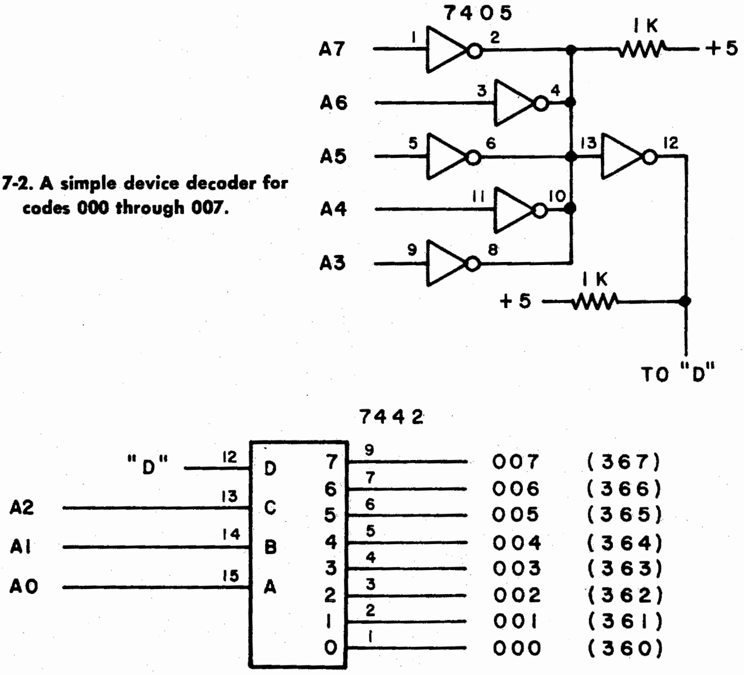


Fig. 7-2. A simple device decoder for codes 000 through 007.

You may either rejumper your read/write memory's address-select logic to "move" a portion of the memory so that it is moved to within this area, or you may convert the addresses in the program listings to "move" the programs to available areas of read/write memory. Check your programs carefully if you choose to "relocate" them.

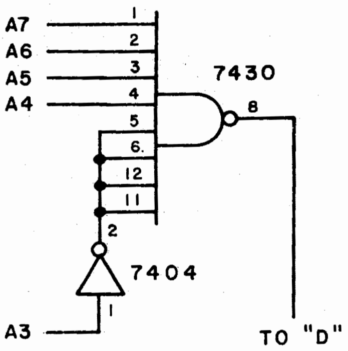


Fig. 7-3. Alternate five-input "gate."

We have assumed that you have already established a stack pointer in the 8080 CPU. This may be done with a monitor program. If this is not the case, you will have to load a stack pointer into the 8080. We suggest a stack pointer of 003 377 so that the stack will be placed in the upper portion of read/write memory that you will be using in the experiments. If you wish to load a stack pointer, the following method may be used:

1. Load the following program in your computer:

```
003 000 061  LXISP  /LOAD THE STACK POINTER
003 001 377  377    /LOW ADDRESS OF STACK
003 002 003  003    /HIGH ADDRESS OF STACK
003 003 166  HLT    /HALT
```

2. Run the program and then start your experiments. Remember to repeat this process if you turn the power off and on.

The MMD-1 computer that we used for the experiments establishes the stack for us through the use of a Keyboard Executive program (KEX).

The experiments use +12 and −12 volts. These voltages are not particularly dangerous, but their application to the wrong portion of a circuit may be disastrous. We recommend that you make all power connections before any other wiring is done. After the power connections are made, check them again. You may notice some noise on the analog outputs provided by the D/A converters used in experiments. This noise will be particularly noticeable if you are using an oscilloscope. Most noise is due to the transfer of digital noise to the analog circuit through the +5-, +12-, and −12-volt power connections. You may wish to filter these voltage connections with some 0.1-μF capacitors between the power connection and ground.

If you are an instructor, you may be interested in placing portions of the programs in programmable read-only-memory (PROM). This is particularly useful in large classes, since it will decrease the time that is spent debugging student's circuits and programs.

Please feel free to make use of the circuits and programs as you wish. We encourage you to devise your own experiments. Perhaps you can come up with some interesting adaptations of the hardware and software that we have provided.

If you are fairly new to breadboarding, and you require some additional help, we refer you to Unit 9 of *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 1*. You may also find that the LR-25 Breadboarding Station Outboard is a useful addition to your breadboarding aids. This unit contains eight lamp monitors, four logic switches, an RC clock, and two debounced logic push buttons.

The following experiments illustrate the use of digital-to-analog and analog-to-digital converters.

Experiment No.	*Purpose*
1	A 10-bit D/A converter is interfaced to the 8080-based computer and it is used to generate voltage ramp outputs with variable periods.
2	A 10-bit D/A converter is used to generate triangular voltage outputs with variable periods.*
3	A ramp A/D converter is constructed by using a 10-bit D/A converter and a control program.*
4	A successive-approximation A/D converter is constructed by using a 10-bit D/A converter and a control program.*
5	A small data-acquisition and display system is constructed by using the successive-approximation A/D converter that was constructed in Experiment 4.*
6	The threshold voltages of a NAND gate are measured by using a 10-bit D/A converter to output test voltages to the device under test.*
7	The Burr-Brown MP-10 dual D/A converter module is interfaced to the computer and used in a variety of ways.
8	Two 8-bit D/A converters are used to generate the signals required for a Y *vs.* T display on an oscilloscope.
9	Two 8-bit D/A converters are used to generate the signals required for a true X-Y display on an oscilloscope.*

* These experiments use hardware and/or software that was used in the previous experiment.

EXPERIMENT NO. 1
INTERFACING A 10-BIT DIGITAL-TO-ANALOG CONVERTER

Purpose

The purpose of this experiment is to interface the 10-bit Analog Devices AD7522 digital-to-analog converter to an 8080-based microcomputer. A parallel output program will be used to test the interface.

Discussion

The first program in this experiment uses the AD7522 10-bit D/A converter, or an equivalent double-buffered 10-bit D/A converter, to generate a slow linear ramp that may be observed with the aid of a volt-ohm-milliammeter (vom) or an oscilloscope. The changing information that is output to the D/A converter will also be displayed at two output ports equipped with LEDs.

Since the AD7522 D/A converter is a 10-bit device, the output voltage is divided into 1024 discreet voltages, each one being approximately 5.5 millivolts greater or less than the adjacent voltage steps. The time for a complete "sweep" of all 1024 voltage steps is determined by the time required to execute a time delay subroutine within the linear ramp generating software. Initially, the ramp's period will be about 10 seconds.

The D/A converter has 10 data inputs (one per bit) but the 8080's data bus has only eight bits and, therefore, can only supply the D/A converter with eight bits of data at a time. The AD7522 D/A converter has three internal registers that are used for data storage. The eight least significant bits of data are strobed into an 8-bit holding register when the AD7522's Low Byte Strobe (LBS) input is pulsed. The two most significant bits are strobed into a 2-bit holding register when the High Byte Strobe (HBS) input is pulsed. Loading data into these two registers will not affect the voltage that is being output by the D/A converter portion of the device. Only when a third input, Load D/A Converter (LDAC) is pulsed will the entire 10-bit word be transferred from the two holding registers to the D/A converter register and applied to the inputs of the 10-bit D/A converter. This has been discussed previously. It is called *double buffering.* If you are not using the AD7522 D/A converter, you will have to construct a double-buffered D/A converter interface. A detailed double-buffered scheme is shown in Unit 1.

Pin Configurations of the Integrated Circuits (Fig. 7-4)

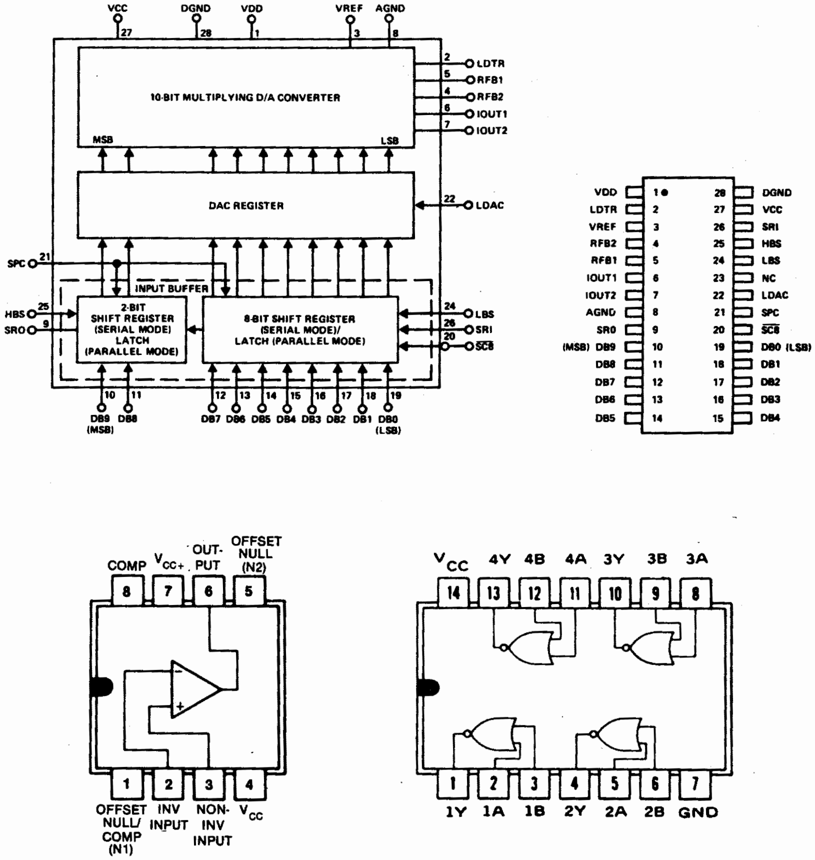
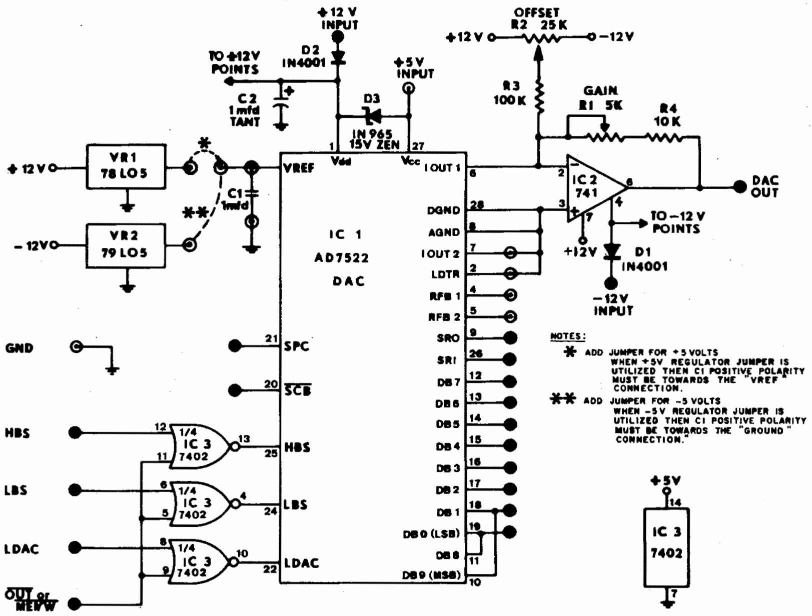


Fig. 7-4.

Schematic Diagram of the Circuit (Fig. 7-5)



NOTES:

* ADD JUMPER FOR +5 VOLTS
WHEN +5V REGULATOR JUMPER IS UTILIZED THEN C1 POSITIVE POLARITY MUST BE TOWARDS THE "VREF" CONNECTION.

** ADD JUMPER FOR -5 VOLTS
WHEN -5V REGULATOR JUMPER IS UTILIZED THEN C1 POSITIVE POLARITY MUST BE TOWARDS THE "GROUND" CONNECTION."

Fig. 7-5. Schematic diagram for the 10-bit AD7522D/A converter interface. Use a 79L05 voltage regulator (VR-2) for a 0–5-volt output range.

The schematic diagram for a completely interfaced AD7522 10-bit D/A converter was shown previously in Unit No. 1. It is repeated again in Fig. 7-5.

If an LR-35 Outboard is available, you may wish to use it rather than construct the interface circuit for the AD7522 integrated circuit. You may also substitute any other double-buffered 10-bit D/A converter.

Step 1

Wire the circuit shown in Fig. 7-5 or use the LR-35 Outboard as shown in Fig. 7-6. If you have wired the circuit shown in Fig. 7-5, be sure to make the additional connections shown in Fig. 7-6.

To observe the voltage output by the D/A converter, connect a volt-ohm-milliammeter (vom) to the DAC OUT connection on your breadboard. Since accumulator I/O will be used, be sure that the 8080's $\overline{\text{OUT}}$ signal is properly connected to the point labeled $\overline{\text{OUT}}$ $\overline{\text{MEMW}}$ on the interface, or on the LR-35 Outboard.

Address codes 003, 004, and 005 will be used. You will have to wire an appropriate device address decoder if these decoded de-

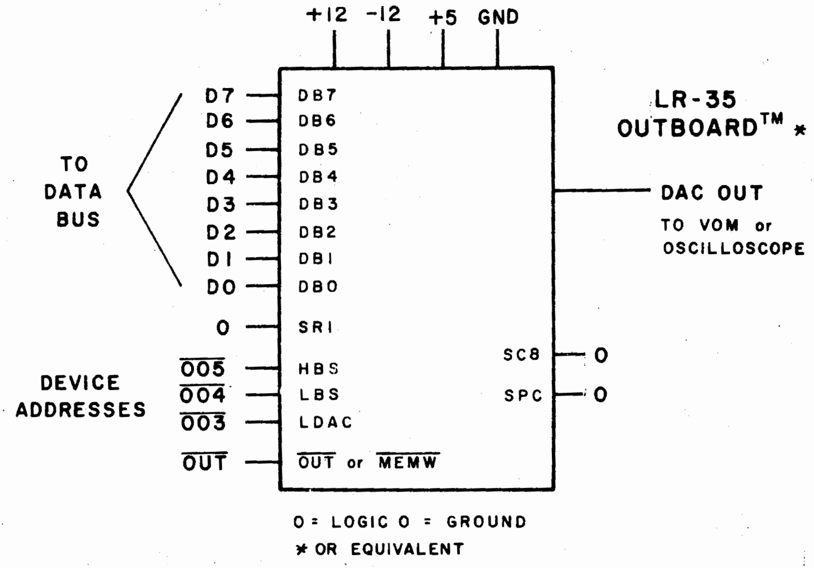


Fig. 7-6. Connections between the AD7522 interface and the 8080 microcomputer system.

vice addresses are not readily available on your computer. A useful circuit is shown in Fig. 7-2 of this unit. If you are using an MMD-1 computer, you will find that device codes 003 through 007 are provided on breadboarding pins adjacent to the SN74L42 decoder integrated circuit in the I/O DECODER section of the main printed-circuit board.

If you are in doubt about the generation of device select pulses and I/O device addresses, we refer you to Unit 17 in *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 2.*

Step 2

Apply power to your computer system and load the following program into the computer's read/write memory. A time-delay subroutine that starts at address 003 200 will be used in this experiment. It is also shown in the program.

```
                        /THIS PROGRAM USES A 10-BIT DIGITAL
                        /TO ANALOG CONVERTER TO GENERATE A
                        /LINEAR RAMP.
                        *003 000
003 000 175   START,    MOVAL     /GET THE LOW 8 BITS OF DATA
003 001 323             OUT       /AND OUTPUT IT TO THE DAC LOW
003 002 004             004       /DATA BYTE HOLDING REGISTER
003 003 323             OUT       /ALSO OUTPUT IT TO ONE OF THE
```

```
003 004 002           002      /OUTPUT PORTS ON THE COMPUTER
003 005 174           MOVAH    /THEN GET THE HIGH 8 BITS OF DATA
003 006 323           OUT      /AND OUTPUT IT TO THE DAC
003 007 005           005
003 010 323           OUT      /ALSO OUTPUT IT TO AN OUTPUT PORT
003 011 000           000      /EQUIPPED WITH LEDS
003 012 323           OUT      /USE AN OUTPUT PULSE TO TRANSFER THE
003 013 003           003      /DATA FROM THE DAC HOLDING REGISTER
003 014 315           CALL     /INTO THE DAC. THEN CALL A DELAY SUB-
003 015 200           DELAY    /ROUTINE
003 016 003           0
003 017 043           INXH     /NOW INCREMENT THE DATA VALUE BY 1
003 020 303           JMP      /AND OUTPUT THE NEW VALUE
003 021 000           START
003 022 003           0
                      /THIS IS THE DELAY SUBROUTINE
                      *003 200
003 200 365  DELAY,   PUSHPSW  /SAVE REGISTERS
003 201 325           PUSHD
003 202 021           LXID     /SET TIMING BYTES IN REGISTERS
003 203 046           046      /D AND E
003 204 001           001
003 205 033  DEC,     DCXD     /DECREMENT THE REGISTER PAIR
003 206 172           MOVAD
003 207 263           ORAE     /CHECK FOR REG PAIR = 000
003 210 302           JNZ      /IF NOT ZERO, DO IT AGAIN
003 211 205           DEC
003 212 003           0
003 213 321           POPD     /DONE, RESTORE REGISTERS
003 214 361           POPPSW
003 215 311           RET
```

Step 3

Begin execution of the program at address 003 000. Be sure that the vom is connected to the voltage output from the D/A converter interface, DAC OUT. What do you observe?

We observed a slow but steady increase in the voltage that was output by the D/A converter until a final reading of about +5 volts was reached. At this point the vom needle returned to zero volts and the swing of the needle started again. The process was repeated again and again. The time period required for the voltage to swing from zero volts to +5 volts was about 10 seconds on our computer (750-kHz clock, 1.3-μsec period).

Step 4

Change the contents of memory location 003 203 from 046 to 047 and start the program. Is there any difference between the period now required for a complete voltage swing between zero volts and +5 volts and the period measured previously?

We could not see any difference. Do you know why?

The timing byte in the delay subroutine was changed, but only the least significant bit was changed. The change was, therefore, very small.

Step 5

Change the contents of memory location 003 204 from 001 to 002. Start the program again and note the time that it takes for the vom's needle to make a complete excursion from zero to +5 volts.

We observed that it took about 18 seconds for the D/A converter's output to go from zero to +5 volts. Do you know why the time for a complete zero- to +5-volt swing has increased so much over the period observed in Steps 3 and 4?

The large difference in the period of the voltage *ramp* is due to the large change that we introduced in the timing bytes used by the time-delay subroutine. The change was made in the most significant byte of data in the three-byte LXID instruction. The first change, in Step 4, changed the timing byte from 001 046 to 001 047.

The second change, in Step 5, increased the timing byte from 001 047 to 002 047. *Remember, we did not change the least significant byte back to 046 once it was set to 047 in Step 4.* The latest change almost doubles the time spent in the time-delay subroutine.

Step 6

You have used a 10-bit D/A converter and software to generate a positive ramp. What would be the easiest way to generate a negative ramp with the present interface circuit?

Probably the easiest modification could be made to the software. We would suggest substituting a decrement instruction for the increment instruction now used in location 003 017.

Change the instruction in location 003 017 from an INXH (043) to a DCXH (053) instruction. Change the two timing bytes in the DELAY subroutine so that they are the same as those indicated in the listing of the DELAY program, i.e., location 003 203 should contain 046 and location 003 204 should contain 001.

Begin the program at address 003 000 and note your observations of the vom's voltage changes in the space below:

You should observe that the needle now starts a slow, downward swing from approximately +5 volts to zero volts. Once the needle reaches zero volts, it quickly returns to the +5 volt position for another downward excursion. The ramp period should be about 10 seconds since the DELAY subroutine has been reinitialized to contain the original timing bytes.

Step 7

The call to the DELAY subroutine may be removed from the linear ramp program by changing the contents of locations 003 014, 003 015, and 003 016 all to 000. What does this do to the program?

Thse substitutions replace all three bytes of the call instruction with no-operation instructions (NOP = 000). Why did we have to fill all three bytes with NOP instructions? Could the call instruction be "removed" from the normal program flow by simply replacing the first byte with a NOP instruction?

All three bytes must be "removed" from the program. If only the first byte of a multibyte instruction is replaced with a NOP instruction, the remaining bytes will be treated as instructions by the computer. In the linear ramp program, replacing only the first byte of the call instruction with a NOP instruction would result in the following:

```
 .   .   .        .
003 012 323      OUT
003 013 003      003
003 014 000      NOP
003 015 200       ?
003 016 003       ?
 .   .   .        .
 .   .   .        .
```

The computer has no way of knowing that it is not supposed to execute the instructions that are represented by the codes 200 and 003, the address portion of the call instruction. What operations do these codes represent?

The 200 is an ADDB instruction and the 003 is an INXB instruction. *When you wish to "remove" an instruction from a program, remember to replace all of the bytes with NOP instructions.*

Replace the contents of locations 003 014, 003 015, and 003 016 all with 000. Start the program at address 003 000. What do you observe?

The vom needle seemed to oscillate at about 2.5 volts. The oscillations are small in terms of voltage changes. Since the ramp is now generated without the DELAY subroutine, its period is very short. The meter indicates only the average voltage present. You can observe the ramps with the aid of an oscilloscope. Are the ramps positive or negative?

We observed negative ramps. Remember that a DCXH instruction has been substituted in the program.

DO NOT DISCONNECT THE INTERFACE OR TURN OFF THE POWER. The hardware and some of the software used in this experiment will be used in the next experiment.

EXPERIMENT NO. 2
COMPLEX DIGITAL-TO-ANALOG CONVERTER OUTPUTS

Purpose

The purpose of this experiment is to generate a triangular voltage output using the D/A converter that was interfaced to the microcomputer in the previous experiment.

Discussion

In the previous experiment, the Analog Devices AD7522 10-bit D/A converter integrated circuit was interfaced to the 8080 microcomputer. Software was used to generate both a positive and a negative ramp. The change between positive and negative ramps was simple, requiring only the substitution of increment (INXH) or decrement (DCXH) instructions. A triangular output is one in which a positive ramp is followed by a negative ramp, which is in turn followed by a positive ramp, and so on.

In this experiment a single 8-bit register will be incremented. The incremented information will be output to the eight LSBs of the D/A converter and also to an output port so that the values may be observed with lamp monitors or seven-segment displays, if desired. When the contents of the register being incremented reaches 377, the largest possible 8-bit value, the software will switch to a series of steps to decrement the contents of the register and to output these new values to the D/A converter and to the output port. When the contents of the register is finally decremented to 000, the incrementing program steps will again start to generate the positive portion of the signal. The DELAY subroutine will again be used.

Pin Configuration of the Integrated Circuits

Schematic Diagram of the Circuit

Both the pin configurations for the integrated circuits and the schematic diagram of the circuit have been given in Experiment No. 1 (Figs. 7-4 and 7-5). Refer to Experiment No. 1 for the necessary information.

Step 1

If the D/A converter interface developed in Experiment No. 1 is not wired to your computer, refer to the circuit details in Experiment No. 1, Step 1, and rewire the D/A converter as shown. If the circuit is intact, go ahead to Step 2.

Step 2

Enter the following program into the computer's read/write memory starting at address 003 000. This program will use the time-delay subroutine that was entered into the read/write memory in Experiment 1. If this subroutine is not present in your computer's memory, it has been included in the following program. Enter it starting at address 003 200.

```
                        /THIS PROGRAM WILL GENERATE A TRIANGULAR
                        /WAVEFORM USING A DIGITAL-TO-ANALOG CONVERTER.
                        *003 000
003 000 074   UP,       INRA     /INCREMENT THE CONTENTS OF REG A
003 001 312             JZ       /IF THE VALUE IS 000, WE INCREMENTED
003 002 020             DOWN     /PAST 377, SO WE HAVE TO SET A=376
003 003 003             0        /AND THEN BEGIN DECREMENTING.
003 004 323             OUT      /LATCH THE DATA INTO THE DAC
003 005 004             004      /HOLDING REGISTER.
003 006 323             OUT      /AND ALSO LATCH OUT TO ONE
003 007 002             002      /OF THE COMPUTER'S LED OUTPUT PORTS
```

```
003 010 323            OUT       /THIS PULSE TRANSFERS THE CON-
003 011 003            003       /TENTS OF THE HOLDING REGISTER TO THE DAC
003 012 315            CALL      /CALL THE DELAY ROUTINE SO THAT WE
003 013 200            DELAY     /CAN SEE THE CHANGE ON THE VOM
003 014 003            0         /AND THE LEDS
003 015 303            JMP
003 016 000            UP        /NOW INCREMENT A AND
003 017 003            0         /REPEAT THE PROCESS
003 020 076  DOWN,     MVIA      /A 377 WAS OUTPUT LAST, SO
003 021 376            376       /OUTPUT 376 FOR THE RAMP DOWN
003 022 323            OUT       /LATCH THE DATA OUT
003 023 004            004       /TO THE DAC HOLDING REGISTER
003 024 323            OUT       /ALSO LATCH IT TO THE COMPUTER'S
003 025 002            002       /LED EQUIPPED OUTPUT PORT.
003 026 323            OUT       /STROBE THE CONTENTS OF THE HOLDING
003 027 003            003       /REGISTER INTO THE DAC.
003 030 075            DCRA      /DECREMENT THE CONTENTS OF A
003 031 315            CALL      /NOW WAIT FOR THE VOM TO CATCH UP
003 032 200            DELAY     /BECAUSE OF ITS SLOW RESPONSE TIME.
003 033 003            0
003 034 302            JNZ       /DID WE INCREMENT A TO 000 ?
003 035 022            DOWN+2    /NO, KEEP DISPLAYING AND DECREMENTING
003 036 003            0
003 037 303            JMP       /YES, SO BEGIN THE RAMP UP AGAIN
003 040 000            UP
003 041 003            0
                       /THIS IS THE DELAY SUBROUTINE
                       *003 200
003 200 365  DELAY,    PUSHPSW   /SAVE REGISTERS
003 201 325            PUSHD
003 202 021            LXID      /SET TIMING BYTES IN REGISTERS
003 203 046            046       /D AND E
003 204 001            001
003 205 033  DEC,      DCXD      /DECREMENT THE REGISTER PAIR
003 206 172            MOVAD
003 207 263            ORAE      /CHECK FOR REG PAIR = 000
003 210 302            JNZ       /IF NOT ZERO, DO IT AGAIN
003 211 205            DEC
003 212 003            0
003 213 321            POPD      /DONE, RESTORE REGISTERS
003 214 361            POPPSW
003 215 311            RET
```

Step 3

Start the program at address 003 000 and note the highest and the lowest voltage readings on the vom as the program is executed:

We observed that the vom needle varied between 0 and 1.3 volts. Your observation may be different. Why doesn't the needle make a full excursion from 0 to +5 volts, as was observed in the linear ramp generating experiment?

Remember that the AD7522 is a 10-bit D/A converter. We are only using the eight least significant bits in this experiment. It is impossible for us to know the state of the two most significant bits in your converter. The software has not changed them; therefore, there are four possible voltage ranges which may be observed by people doing this experiment:

		Eight LSBs	
Range	*Two MSBs*	*Minimum*	*Maximum*
0.00–1.25 Volts	00	00000000	11111111
1.25–2.50 Volts	01	00000000	11111111
2.50–3.75 Volts	10	00000000	11111111
3.75–5.00 Volts	11	00000000	11111111

You should observe that your vom's needle swings with a difference of about 1.2 to 1.4 volts, in close approximation to one of the ranges noted above.

Step 4

You will now change the time-delay period of the DELAY subroutine. Change the value contained in memory location 003 204 from 001 to 000. Start the program at address 003 000 and note any difference in the frequency of the triangular voltage output as indicated on the vom and on the output port's LEDs.

We found that the frequency was higher (shorter period). The reason for this is that different data values have been used by the DELAY subroutine. You have changed the timing data from 001 046 to 000 046 and you should observe about 23 complete cycles of the triangular output in 15 seconds.

Step 5

Another way to make the frequency of the triangular voltage output reasonable fast is to change the PUSHPSW instruction in the DELAY subroutine to a RET (311). What will this do?

By changing the PUSHPSW (365) instruction to a RET (311), or return, instruction, you will cause the computer to immediately return from the subroutine once the call to DELAY is executed. The remaining steps in DELAY are, therefore, not executed.

Step 6

Change the instruction contained in memory location 003 200 from a 365 to a 311. This inserts the return instruction at the start of the DELAY subroutine. Execute the program. What do you observe?

We needed an oscilloscope to observe the ramp since it had a frequency of about 20 Hz. It was difficult to see such fast voltage changes on the vom or at the output port.

Replace the return instruction in memory location 003 200 with the PUSHPSW instruction (365) that was there previously. Also replace the original timing byte (001) in memory location 003 204.

DO NOT DISCONNECT THE INTERFACE OR TURN OFF THE POWER. The hardware and some of the software used in this experiment will be used in the next experiment.

EXPERIMENT NO. 3
A SOFTWARE-CONTROLLED RAMP A/D CONVERTER

Purpose

The purpose fo this experiment is to construct an analog-to-digital converter by using the 10-bit digital-to-analog converter that was previously interfaced to the computer along with an analog comparator, the LM311. An input port will also be used. This experiment will use the linear ramp conversion technique to determine the value of unknown analog voltages.

Discussion

This experiment uses a slow linear ramp, similar to the one generated in Experiment 1. An unknown voltage input, $V_{UNKNOWN}$, is continuously compared to the linear voltage ramp that is being output by the D/A converter under software control. If the output of the D/A converter, V_{DAC}, or DACOUT, is less than the unknown voltage input, then the state of the comparator's output will be a logic 0. When $V_{DAC} > V_{UNKNOWN}$, the comparator's output will be a logic 1. The state of the comparator's output is input into the 8080's A register through a three-state input port. It is tested by using a conditional jump instruction, JNZ. If the comparator's output is logic 0, another voltage step in the ramp is generated. If the comparator's output is a logic 1, the unknown voltage has been matched and the process repeats itself, starting the ramp at zero volts.

The 8080's H and L registers hold the binary values that are used to generate the ramp. The data is output to the 10-bit D/A converter and to two output ports where the binary values may be monitored. The output of the comparator is observed with a lamp monitor. A typical lamp monitor circuit is described in the introduction to these experiments.

Pin Configurations of the Integrated Circuits (Fig. 7-7)

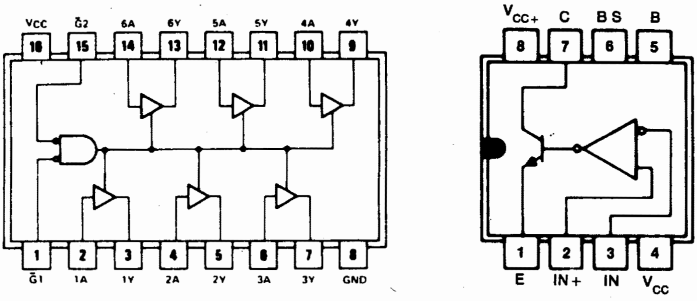


Fig. 7-7.

Schematic Diagram of the Circuit (Fig. 7-8)

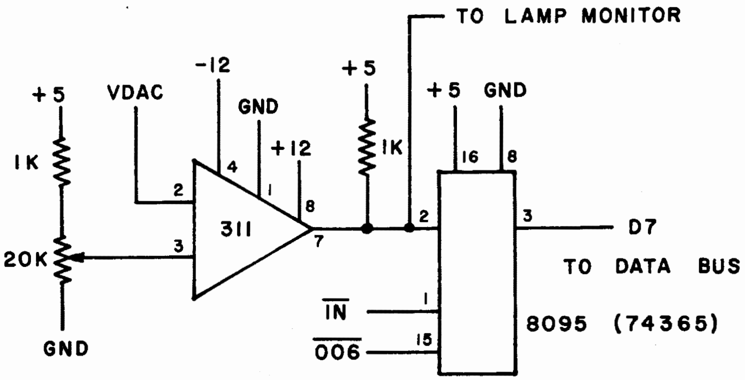


Fig. 7-8.

Step 1

Wire the additional circuitry shown in Fig. 7-8. The AD7522 10-bit D/A converter that was interfaced to the computer in Experiment 1 will be used again in this experiment. If this converter is not interfaced to your computer, refer to Experiment 1, Step 1 for the interfacing details.

Remember to connect the output of the D/A converter, V_{DAC}, or DACOUT, to the V_{DAC} connection noted in the schematic diagram (Fig. 7-8). The vom may be used to observe the D/A converter's output voltage, but this is optional.

Step 2

Enter the program which follows into the computer's memory starting at address 003 000. The DELAY subroutine that was used in previous experiments will also be used in this experiment. Be sure that it is loaded correctly, starting at address 003 200. It is presented again in the program listing. The data-display subroutine, DISPLA, is located "above" the DELAY subroutine, starting at address 003 200. Be careful when loading these program steps.

```
                          /THIS PROGRAM USES A DAC AND A COMPARATOR
                          /TO FORM A RAMP ANALOG-TO-DIGITAL CONVERTER.
                          *003 000
003 000 041  ADC,         LXIH      /H&L ARE USED TO HOLD THE CURRENT
003 001 000               000       /DIGITAL "RAMP" VALUE.
003 002 000               000       /SET BOTH OF THEM TO 000 INITIALLY
003 003 323               OUT       /THE OUT INSTRUCTION IS USED TO GEN-
003 004 007               007       /ERATE A SYNCH PULSE FOR A SCOPE.
003 005 175  DACOUT,      MOVAL     /GET THE LOW EIGHT BITS OF "RAMP"
003 006 323               OUT       /OUTPUT THEM TO DAC
003 007 004               004
003 010 174               MOVAH     /GET THE HIGH TWO BITS OF "RAMP"
003 011 323               OUT       /OUTPUT THEM TO DAC
003 012 005               005
003 013 323               OUT       /STROBE THE DAC'S HOLDING REG
003 014 003               003
003 015 315               CALL      /CALL THE H&L DISPLAY SUBROUTINE
003 016 220               DISPLA    /AND THE DELAY ROUTINE SO THAT WE
003 017 003               0         /CAN SEE THE CONTENTS OF H&L
003 020 333               IN        /NOW CHECK THE COMPARATOR'S OUTPUT
003 021 006               006       /TO SEE IF H&L ARE BIG ENOUGH TO
003 022 346               ANI       /APPROXIMATE THE UNKNOWN ANALOG
003 023 200               200       /VOLTAGE
003 024 302               JNZ       /H&L ARE BIG ENOUGH BECAUSE THE COM-
003 025 000               ADC       /PARATOR'S OUTPUT IS A 1. THEREFORE
003 026 003               0         /START ANOTHER CONVERSION.
003 027 043               INXH      /H&L ARE NOT BIG ENOUGH, SO INCREMENT
003 030 303               JMP       /THEM BY 1 AND TRY AGAIN.
003 031 005               DACOUT
```

```
003 032 003             0

                        /THIS IS THE DISPLAY OUTPUT SUBROUTINE
                        *003 220
003 220 174   DISPLA,   MOVAH     /GET THE HIGH DATA BYTE
003 221 323             OUT       /AND LATCH THE VALUE OUT TO ONE OF
003 222 000             000       /THE LED OUTPUT PORTS ON THE COMPUTER
003 223 175             MOVAL     /GET THE LO DATA BYTE AND
003 224 323             OUT       /ALSO LATCH IT OUT.
003 225 002             002
003 226 315             CALL      /AFTER LATCHING H&L, DELAY FOR
003 227 200             DELAY     /A FEW MSEC OR SEC
003 230 003             0
003 231 311             RET       /THEN RETURN TO THE MAIN PROGRAM

                        /THIS IS THE DELAY SUBROUTINE
                        *003 200
003 200 365   DELAY,    PUSHPSW   /SAVE REGISTERS
003 201 325             PUSHD
003 202 021             LXID      /SET TIMING BYTES IN REGISTERS
003 203 046             046       /D AND E
003 204 001             001
003 205 033   DEC,      DCXD      /DECREMENT THE REGISTER PAIR
003 206 172             MOVAD
003 207 263             ORAE      /CHECK FOR REG PAIR = 000
003 210 302             JNZ       /IF NOT ZERO, DO IT AGAIN
003 211 205             DEC
003 212 003             0
003 213 321             POPD      /DONE, RESTORE REGISTERS
003 214 361             POPPSW
003 215 311             RET
```

Step 3

Start the program at address 003 000. What do you observe taking place at the LEDs connected to output ports 000 and 002? What is taking place at the vom? Vary the potentiometer setting and note your observations below:

We observed that as the potentiometer was rotated to one extreme, fewer and fewer of the LEDs at the output ports were lit. The voltage indicated by the vom also became lower, and the lamp monitor connected to the comparator's output flickered.

As the potentiometer was turned toward the other extreme, more and more of the LEDs at output ports 000 and 002 became lit. The vom reading increased to higher voltages. As higher voltages were reached, the LEDs at output ports 000 and 002 could be seen to increment and flicker. Do you know what causes this "flickering?"

It takes the computer some time to increment and output the values contained in registers H and L, particularly since a time delay is used in the program. The flickering is produced as a direct result of the longer time that it takes the computer to increment registers H and L to higher and higher values.

Will it take the computer longer to match a high voltage than it will take to match a low voltage?

Yes, since the count required by the D/A converter to generate a high voltage requires more software time than does a relatively low count for lower voltages.

Step 4

Remove the time-delay program from the normal program flow. Do this by substituting a return instruction (311) for the instruction in memory location 003 200. Start the program again. Vary the potentiometer settings and note the effect on the voltages measured by the vom. Start at a low voltage (few LEDs lit) and proceed to higher voltages. What do you observe?

We observed that when the potentiometer presents the comparator with relatively low voltages, the vom needle is fairly steady. As the voltage is increased past about 2 volts, the needle starts to oscillate or quiver. When the highest voltage is applied to the comparator by the potentiometer, the vom needle swings over a range of one-half volt. Should we expect the vom to act like this?

Yes. Again, the oscillations are caused by the slow response of the vom to changing voltages. The computer can "ramp" the voltage output by the D/A converter faster than the vom can measure it. Remember, there is no time delay in the program.

Replace the return instruction that you stored in location 003 200 with a PUSHPSW instruction (365) before going on. This restores the operation of the time-delay subroutine, DELAY.

Step 5

You will now modify the DELAY subroutine to slow down the conversion process. This is done by changing the timing bytes in locations 003 203 and 003 204. Make the following changes to the DELAY subroutine: Change the value in memory location 003 203 to 000 and change the value in memory location 003 304 to

050. Start the program at address 003 000 and note your observations.

The computer now increments the ramp value at the rate of about three counts per second. As noted previously, the values displayed on the LEDs become smaller and smaller as the potentiometer presents lower and lower voltages to the comparator for measurement. The values displayed on the LEDs become larger as the voltage is increased by turning the potentiometer in the opposite direction.

You should also be able to see the lamp monitor's output change state when the comparator's output indicates that the ramp voltage is equal to, or exceeds, the unknown input, $V_{UNKNOWN}$. Note that the comparator's output will return to logic 0 when the ramp is reset to start at its lowest value, 0000000000_2.

Step 6

If a dual-trace oscilloscope is available, you may wish to perform this step. If not, read through this section and continue to Step 7.

Connect one of the oscilloscope's inputs to the D/A converter's output, V_{DAC}, or DACOUT. Connect the other oscilloscope input to $V_{UNKNOWN}$.

Remove the call to the DELAY subroutine by substituting three NOP instructions (000) in locations 003 226, 003 227, and 003 230. This will speed up the ramp generation. Set the potentiometer at about its midpoint, start the program, and observe the ramp on the oscilloscope. Using the second trace, position the line produced by $V_{UNKNOWN}$ at the top of the ramp. Be sure that both input channels are set to the same gain levels and that both are set to accept dc input levels.

What do you observe as you vary the potentiometer settings?

You should observe two important things. First, the top of the ramp will be equal to the unknown voltage over the full range of the D/A converter's output. Second, as the unknown voltage is lowered, more conversions take place per unit of time. Higher voltages require longer conversion times.

Step 7

The program that you are using for the computer-controlled ramp A/D converter constantly displays the various test values that are also output to the 10-bit D/A converter. The program may be modified so that only the final result of the conversion process is displayed at output ports 000 and 002.

Could you suggest the necessary software changes that would be needed to accomplish this? Remember, the binary data is to be output only after a conversion has been completed.

Here are the changes that we suggest:

a) Remove the call to the DISPLA subroutine. Substitute NOP instructions in locations 003 015, 003 016, and 003 017.

b) The JNZ instruction at address 003 024 is only executed at the end of the conversion process. Change this:

```
from:  JNZ               to:  JNZ      302
       ADC                    DISPLA   220
        0                       0      003
```

This branches the program to display the data at the end of the conversion.

c) The DISPLA software is no longer called as a subroutine, thus the return instruction is no longer needed. Replace it with a jump to symbolic address ADC. This will restart another conversion after the data is displayed:

```
003 231   JMP   303
003 232   ADC   000
003 233    0    003
```

Make these changes and run the program. You should observe only the final value of the digitized unknown voltage and not the ramping or incrementing test values at output ports 000 and 002.

Remember to reinsert the return instruction (311) in location 003 231 prior to going to the next experiment.

Conclusions

You have seen that an analog-to-digital converter can be constructed by using a digital-to-analog converter, a comparator, and microcomputer software for control and decision making. When using the ramp conversion technique, however, we have observed that the conversion time is directly proportional to the voltage that is to be measured or converted.

The next experiment will show you how a successive-approximation converter may be constructed by using the identical hardware used in this experiment. Only the software will be changed.

DO NOT DISCONNECT THE INTERFACE OR TURN OFF THE POWER. The hardware and some of the software used in this experiment will be used in the next experiment.

EXPERIMENT NO. 4
A SOFTWARE-CONTROLLED SUCCESSIVE-APPROXIMATION A/D CONVERTER

Purpose

The purpose of this experiment is to use the successive-approximation analog-to-digital conversion technique to digitize an analog signal.

Discussion

The successive-approximation analog-to-digital conversion technique tests each of the individual binary bit positions in sequence, from the most significant bit (MSB) to the least significant bit (LSB). The process starts when a logic one is applied to the D/A converter in bit-position D9 (MSB) and the resulting test voltage compared to the unknown voltage at the comparator. If the unknown voltage is exceeded by the voltage in this test, the bit is returned to the logic-zero state and the next least significant bit is tested. If the unknown voltage is not exceeded by this test, the bit remains a logic one and the next least significant bit position is tested, using the same procedure.

In this way, the voltages are tested from the largest voltage step to the smallest. The individual bit positions and their corresponding test voltages for a 0- to +5.115-volt full scale, 10-bit D/A converter are listed below.

Bit Position	Weighing Voltage
D0	0.005 Volts
D1	0.010
D2	0.020
D3	0.040
D4	0.080
D5	0.160
D6	0.320
D7	0.640
D8	1.280
D9	2.560

Schematic Diagram of the Circuit

The circuit that is used in Experiment 4 is the same as the one used in Experiment 3 (Fig. 7-8). We refer you to Experiment 3 for the details. A 10-bit D/A converter interface is also used. This is documented in Experiment 1, Step 1.

Step 1

Load the following program into the computer's memory starting at address 003 000. The DELAY and DISPLA subroutines used in the previous experiment will also be used by this program. If they are still available in the computer's read/write memory you may use them, but we suggest that you check them first. They are included in the program if you need to re-enter them.

```
                             /THIS PROGRAM USES THE SUCCESSIVE APPROXIMATION
                             /TECHNIQUE TO DETERMINE THE DIGITAL VALUE
                             /OF AN UNKNOWN ANALOG VOLTAGE USING A
                             /AD7522 DAC AND A COMPARATOR.
                             *003 000
003 000 041  ADC,     LXIH     /H&L CONTAIN THE CURRENT APPROXIMATION
003 001 000           000
003 002 002           002      /WHICH IS XX XXX X10 00 000 000
003 003 021           LXID     /D&E CONTAIN THE CURRENT BIT
003 004 000           000      /THAT IS ADDED OR SUBTRACTED
003 005 002           002      /FROM THE CURRENT APPROXIMATION
003 006 006           MVIB     /B IS USED AS A BIT COUNTER
003 007 012           012      /012 OCTAL = 10 DECIMAL (10 BIT DAC)
003 010 175  DACOUT,  MOVAL    /GET EIGHT LSB'S OF DATA
003 011 323           OUT      /OUTPUT THE M TO DAC
003 012 004           004
003 013 174           MOVAH    /GET TWO MSB'S OF DATA
003 014 323           OUT      /OUTPUT THEM TO DAC
003 015 005           005
003 016 323           OUT      /STROBE THE DAC HOLDING REG
003 017 003           003
003 020 315           CALL     /DISPLAY THE CURRENT APPROXIMATION
003 021 220           DISPLA   /AND THEN DELAY FOR A SHORT
003 022 003           0        /PERIOD OF TIME
003 023 333           IN       /TEST THE COMPARATORS OUTPUT. IF IT'S
003 024 006           006      /A 1, WE'RE TOO HI, IF 0, TOO LOW.
003 025 346           ANI
003 026 200           200
003 027 302           JNZ      /IT'S A 1, WE'VE APPROXIMATED TOO HIGH
003 030 050           TOOHI
003 031 003           0
003 032 172  DIV2,    MOVAD    /WE'RE TOO LOW, INCREASE THE APPROX
003 033 037           RAR      /SO MOVE THE TEST BIT TO ONE OF THE
003 034 127           MOVDA    /LEAST SIGNIFICANT BITS.
003 035 173           MOVAE    /WE ALSO ROTATE THE LEAST SIGNIFICANT
003 036 037           RAR      /BYTE, ROTATING ANY CARRY FROM D
003 037 137           MOVEA    /INTO E.
003 040 031           DADD     /ADD D&E TO H&L, RESULT IS IN H&L
003 041 005           DCRB     /TRIED ALL 10 BITS YET ?
003 042 302           JNZ      /NO, TRY THE NEXT APPROXIMATION
003 043 010           DACOUT   /BY LATCHING IT OUT AND TESTING THE
003 044 003           0        /COMPARATOR'S OUTPUT.
003 045 303           JMP      /YES, WE HAVE TRIED ALL 10 BITS,
003 046 000           ADC      /SO START THE APPROXIMATION AGAIN
```

```
003 047 003             0
003 050 175   TOOHI,    MOVAL     /THE APPROXIMATION WAS TOO BIG, SO
003 051 223             SUBE      /SUBTRACT THE CURRENT "BIT" FROM BOTH
003 052 157             MOVLA     /H&L, THEN ROTATE THE TEST BIT, ADD IT
003 053 174             MOVAH     /WITH THE DADD AND TRY AGAIN
003 054 232             SBBD      /(THIS IS IN CASE OF A BORROW)
003 055 147             MOVHA
003 056 303             JMP       /NOW TRY THE NEXT TEST BIT IN THE WORD
003 057 032             DIV2
003 060 003             0

                        /THIS IS THE DELAY SUBROUTINE
                        *003 200
003 200 365   DELAY,    PUSHPSW   /SAVE REGISTERS
003 201 325             PUSHD
003 202 021             LXID      /SET TIMING BYTES IN REGISTERS
003 203 046             046       /D AND E
003 204 001             001
003 205 033   DEC,      DCXD      /DECREMENT THE REGISTER PAIR
003 206 172             MOVAD
003 207 263             ORAE      /CHECK FOR REG PAIR = 000
003 210 302             JNZ       /IF NOT ZERO, DO IT AGAIN
003 211 205             DEC
003 212 003             0
003 213 321             POPD      /DONE, RESTORE REGISTERS
003 214 361             POPPSW
003 215 311             RET

                        *003 220
003 220 174,  DISPLA,   MOVAH     /GET THE HIGH DATA BYTE
003 221 323             OUT       /AND LATCH THE VALUE OUT TO ONE
003 222 000             000       /OF THE OUTPUT PORTS WITH LEDS
003 223 175             MOVAL     /GET THE LO DATA BYTE AND ALSO
003 224 323             OUT       /LATCH IT OUT
003 225 002             002
003 226 315             CALL      /AFTER LATCH H&L OUT, DELAY FOR
003 227 200             DELAY     /A FEW MSEC. OR SEC.
003 230 003             0
003 231 311             RET
```

Step 2

Change the contents of memory location 003 204 to 200. This will increase the length of time that the computer will spend executing the time-delay subroutine.

Step 3

Begin execution of the program at address 003 000. What do you observe at the two output ports, 000 and 002? With the vom connected to the D/A converter's output, what do you observe on the meter as the program is executing?

If the program is operating properly, you should see that some of the LEDs associated with output ports 000 and 002 are flashing on and off at a rate of about one per second. Depending upon the setting of your potentiometer, some of the LEDs will come on and remain on, while others will come on and then go off.

You should also observe that the vom's needle appears to "home-in" on a particular voltage.

Step 4

Rotate the potentiometer until it is completely counterclockwise. Let the computer run for at least 10 seconds. What do you observe?

Now rotate the potentiometer to its other extreme (clockwise). Again, let the computer run for at least 10 seconds before noting your observations below:

Your actual observations will depend upon the configuration of the potentiometer, i.e., whether the clockwise position supplies the highest or the lowest potential to the comparator.

We found that in one position all but the two least significant bits at the two output ports were at logic zero. The vom needle seems to home-in on a voltage that is close to ground potential. At the other extreme, almost all of the LEDs remained on and the vom needle seemed to home-in on a high potential, about +5 volts.

Step 5

Again, rotate the potentiometer to its extreme positions and note the various states of the LEDs as the computer homes in on the high and the low voltages. What happens at the highest voltage input? What happens at the lowest voltage input?

We observed that the LEDs came on, one at a time, from the most significant bit to the least significant bit in each case. When the potentiometer was set to its highest setting, the LEDs came on and stayed on. When the potentiometer was set to provide a low voltage to the comparator, the LEDs came on and then turned off and remained off. One or two of the least significant bit positions may have its LED lit.

Step 6

Set the potentiometer at its midpoint. With the program running, what do you observe at the two output ports?

We observed that the same testing sequence of LEDs on and LEDs off was again performed by the computer. Now some of the LEDs remained on while others were turned off. Again, the vom's needle appeared to home in on a voltage, toward the center of the 0- to 5-volt range.

We have not yet commented about the lamp monitor connected to the comparator's output. Watch this indicator and try and determine the relationship between the state of the comparator's output and whether or not the individual LED that is indicating the current bit position being tested will remain on or go off. Is there a relationship?

We observed that there is a definite relationship between the state of the comparator's output, as indicated by the lamp monitor, and the on/off states of the LED associated with the current bit position being tested.

If a new bit position is being tested and the lamp monitor remains on, the LED for that bit position will be turned off and the next bit position will be tested. If, however, the lamp monitor remains off as a bit position is tested, the LED associated with that bit position will remain on.

Step 7

Change the contents of memory location 003 204 from 100 to 020. Again, start the program at address 003 000. What happens to the LEDs at output ports 000 and 002 now?

Although the speed of the conversions is faster, the same LEDs were found to be on or off at the end of the successive approximation. The speed of the conversion does not alter the final, digitized value for a steady voltage input.

Step 8

Now change the contents of memory location 003 204 from 020 to 004 and start the program at address 003 000. This will further increase the conversion speed. What do you observe now at the LEDs associated with output ports 000 and 002?

We found that the conversion was so fast that we could not follow it.

If an oscilloscope is available, you may wish to try Steps 9 and 10. In any case, we suggest that you read through them.

Step 9

In this step you will remove the time-delay subroutine from the sequence of instructions that are executed by the successive-approximation conversion program. Change the contents of location 003 200 from 365 to 311. What does this substitution do? Connect the oscilloscope to the D/A converter's voltage output.

The 311 that you have substituted into your program is a return instruction, RET. Placing it at the start of the DELAY subroutine means that the subroutine will immediately return control back to the successive-approximation program.

With the oscilloscope connected to the D/A converter's output, you should be able to observe the various voltage steps as they are output by the D/A converter (under computer control). In the space below, sketch one cycle of the oscilloscope's trace as the test voltages attempt to match the unknown voltage:

Your sketch should be similar to the drawing in Fig. 2-6 in Unit 2. The largest voltage steps are always tested first.

Step 10

Connect the other oscilloscope input to the connection between the comparator's input and the potentiometer's wiper terminal. This will let you examine the unknown voltage that is being digitized. With the two traces' gains set equally, position the unknown voltage trace on the oscilloscope so that it is coincident with the last test voltage test level (smallest step). Now vary the potentiometer's setting. Does the D/A converter's output still seek or home in on the unknown voltage? Are the conversion times equal for high and low voltages?

The conversion speed is independent of the voltage level being measured. You should be able to measure ten voltage levels or tests on the oscilloscope. It will always take *N* steps for an *N*-bit, successive-approximation A/D converter to perform a conversion. We observed that for an 8080 with a 1.33-microsecond clock period, the computer took about three milliseconds to perform a conversion.

Step 11

You may wish to make the following changes to the program. These changes will allow the program to update the display of the data at output ports 000 and 002 only upon completion of a 10-bit conversion.

a) Change:	003 045	from	JMP 303	to	JMP 303
	003 046		ADC 000		DISPLA 220
	003 047		003		003
b) Change:	003 020	from	CALL 315	to	NOP 000
	003 021		DISPLA 220		NOP 000
	003 022		003		NOP 000
c) Change:	003 231	from	RET 311	to	JMP 303
	003 232		---		ADC 000
	003 233		---		003

Run the successive-approximation conversion program, starting a address 003 000. You will only see the result of a conversio at the LEDs at output ports 000 and 002.

Return the program steps listed above to their original content before going on to the next experiment. It is important to be sur to do this since some of the software used in this experiment wil be used again.

DO NOT DISCONNECT THE INTERFACE OR TURN OF THE POWER. The hardware and some of the software used i this experiment will be used in the next experiment.

EXPERIMENT NO. 5
A SMALL DATA-ACQUISITION SYSTEM

Purpose

The purpose of this experiment is to use a 10-bit D/A converte for both data display and for successive-approximation A/D con versions. The program used in this experiment will acquire 12 10-bit data points, displaying them as they are acquired.

Discussion

Data acquisition is generally accomplished by using an A/D converter that is controlled by a computer. The data may be output to a teletypewriter or to a D/A converter for display. In this experiment, one 10-bit D/A converter will be used to perform both tasks. The program has been written to take advantage of the D/A converter's output for the output of data values (to an oscilloscope) when the converter is not being used to generate the outputs that are necessary for the software-controlled successive-approximation conversion.

A main data-display loop has been used to provide a fixed time delay, and register pair D (registers D and E) are used to count the number of times that the computer is to execute the display loop before another analog input is digitized and placed in the display file. Thus, the computer will spend most of its time displaying the data. The data buffer that is being displayed is 256 memory locations in length. Each data point is 10 bits, so the data buffer can contain 128 data values—two memory locations used per value.

Schematic Diagram of the Circuit

The circuit used in this experiment is exactly the same as the circuit that was used in Experiments 3 and 4 (Fig. 7-8). We refer you to Experiment 3 for the necessary circuit details. The 10-bit D/A converter interface uses the Analog Devices AD7522 integrated circuit, along with some external components. Details of this interface are found in Experiment 1, Step 1. Any other double-buffered 10-bit D/A converter may be used in place of the AD7522.

Step 1

The circuitry that was used in Experiment 3 or Eperiment 4 must be available for use in this experiment. If this complete circuit is not interfaced to your computer, refer to Experiment 1, Step 1 and to Experiment 3 for the interfacing details.

Some of the software steps used in the software-controlled successive-approximation A/D converter experiment, Experiment 4, will be used here. These are the program steps that are located at address 003 000 through, and including, address 003 056. They are provided for you in the software listing of the program for this experiment. If you have just completed Experiment 4, you might want to recheck these steps. If they are already in the computer's read/write memory, it will save you some time.

If you do not have the following program steps in your computer's memory, enter them. DO NOT attempt to run this portion of the program.

```
                         /THIS PROGRAM USES THE SUCCESSIVE APPROXIMATION
                         /TECHNIQUE TO DETERMINE THE DIGITAL VALUE
                         /OF AN UNKNOWN ANALOG VOLTAGE USING A
                         /AD4522 DAC AND A COMPARATOR.
                         *003 000
003 000 041  ADC,        LXIH      /H&L CONTAIN THE CURRENT APPROXIMATION
003 001 000              000
003 002 002              002       /WHICH IS XX XXX X10 00 000 000
003 003 021              LXID      /D&E CONTAIN THE CURRENT BIT
003 004 000              000       /THAT IS ADDED OR SUBTRACTED
003 005 002              002       /FROM THE CURRENT APPROXIMATION
003 006 006              MVIB      /B IS USED AS A BIT COUNTER
003 007 012              012       /012 OCTAL = 10 DECIMAL (10 BIT DAC)
003 010 175  DACOUT,     MOVAL     /GET EIGHT LSB'S OF DATA
003 011 323              OUT       /OUTPUT THEM TO DAC
003 012 004              004
003 013 174              MOVAH     /GET TWO MSB'S OF DATA
003 014 323              OUT       /OUTPUT THEM TO DAC
003 015 005              005
003 016 323              OUT       /STROBE THE DAC HOLDING REG
003 017 003              003
003 020 315              CALL      /DISPLAY THE CURRENT APPROXIMATION
003 021 220              DISPLA    /AND THEN DELAY FOR A SHORT
003 022 003              0         /PERIOD OF TIME
003 023 333              IN        /TEST THE COMPARATORS OUTPUT. IF IT'S
003 024 006              006       /A 1, WE'RE TOO HI, IF 0, TOO LOW.
003 025 346              ANI
003 026 200              200
003 027 302              JNZ       /IT'S A 1, WE'VE APPROXIMATED TOO HIGH
003 030 050              TOOHI
003 031 003              0
003 032 172  DIV2,       MOVAD     /WE'RE TOO LOW, INCREASE THE
003 033 037              RAR          APPROXIMATION
003 034 127              MOVDA     /SO MOVE THE TEST BIT TO ONE OF THE
003 035 173              MOVAE     /LEAST SIGNIFICANT BITS.
003 036 037              RAR       /WE ALSO ROTATE THE LEAST SIGNIFICANT
003 037 137              MOVEA     /BYTE, ROTATING ANY CARRY FROM D
003 040 031              DADD      /INTO E.
003 041 005              DCRB      /ADD D&E TO H&L, RESULT IS IN H&L
003 042 302              JNZ       /TRIED ALL 10 BITS YET ?
003 043 010              DACOUT    /NO, TRY THE NEXT APPROXIMATION
003 044 003              0         /BY LATCHING IT OUT AND TESTING THE
003 045 303              JMP       /COMPARATOR'S  OUTPUT.
003 046 000              ADC       /YES, WE HAVE TRIED ALL 10 BITS,
003 047 003              0         /SO START THE APPROXIMATION AGAIN
003 050 175  TOOHI,      MOVAL     /THE APPROXIMATION WAS TOO BIG, SO
003 051 223              SUBE      /SUBTRACT THE CURRENT "BIT" FROM BOTH
003 052 157              MOVLA     /H&L, THEN ROTATE THE TEST BIT, ADD IT
003 053 174              MOVAH     /WITH THE DADE AND TRY AGAIN.
003 054 232              SBBD      /(THIS IS IN CASE OF A BORROW)
003 055 147              MOVHA
003 056 303              JMP       /NOW TRY THE NEXT TEST BIT IN THE WORD
```

Step 2

Make the following changes to the successive-approximation software which has been included in Step 1:

1. Remove the call to the DISPLA Subroutine present at addresses 003 020, 003 021, and 003 022, by substituting NOP (no-operation) instructions for all three bytes. A NOP is equal to 000.
2. Replace the JMP instruction byte at address 003 056 with a return instruction, RET = 311.

What will these changes do to the program?

We have removed the CALL to the DISPLA subroutine so that the result of the conversion will no longer be displayed in the LEDs at output ports 000 and 002. This also removes the time-delay subroutine, DELAY, since it is used only by the DISPLA subroutine.

We have also placed a return instruction in the memory location that was used previously to hold a jump instruction. Normally, the jump instruction would cause the computer to continually execute the successive-approximation A/D converter software. The use of a return instruction in place of a jump turns the ADC portion of the program into an ADC subroutine, so that only one conversion will be performed whenever the ADC subroutine is called.

Step 3

Load the following data acquisition and control program steps into the computer's memory. *NOTE: This program starts at address 003 100. Be careful not to load it "on top of" the ADC subroutine!*

```
                              /THIS IS THE DATA ACQUISITION CONTROL AND
                              /DISPLAY PROGRAM. THE SUCCESSIVE APPROXI-
                              /MATION SOFTWARE FROM EXP 4 IS USED.
                              *003 100
003 100 061   START,          LXISP      /SET UP A STACK AREA
003 101 377                   377
003 102 003                   003
003 103 041                   LXIH       /SET UP A POINTER ADDRESS
003 104 000                   000
003 105 002                   002
003 106 345   LOOP,           PUSHH      /STORE THE MEMORY ADDRESS
003 107 021                   LXID       /SET UP TIMING BYTES
```

```
003 110 300           300
003 111 000           000
003 112 41   INIT,    LXIH      /SET UP DISPLAY BUFFER ADDRESS
003 113 000           000
003 114 002           002
003 115 174  DISPLA,  MOVAH     /GET 8 LSB'S OF DATA
003 116 323           OUT       /OUTPUT THEM TO D/A
003 117 004           004
003 120 054           INRL      /INCREMENT MEMORY ADDR
003 121 176           MOVAM     /GET 2 MSB'S OF DATA
003 122 323           OUT       /OUTPUT TO THE D/A
003 123 005           005
003 124 323           OUT       /STROBE THE D/A TO LOAD 10 BITS
003 125 003           003
003 126 054           INRL      /INCREMENT MEMORY ADDR AGAIN
003 127 302           JNZ       /GONE THRU FILE YET?
003 130 115           DISPLA    /NO, JUMP TO DISPLA TO DISPLAY
003 131 003           0         /THE NEXT POINT
003 132 033           DCXD      /YES, DECREMENT LOOP COUNT (TIMING BYTE)
003 133 172           MOVAD     /ARE ALL 16 BITS OF COUNT = 0?
003 134 263           ORAE
003 135 302  ALFA,    JNZ       /NO, INITIALIZE FOR ANOTHER DISPLAY
003 136 112           INIT      /OF THE BUFFER CONTENTS
003 137 003           0
003 140 315           CALL      /YES, WE "TIMED OUT" SO DO AN A/D
003 141 000           ADC       /CONVERSION
003 142 003           0
003 143 353           XCHG      /MOVE DATA FROM H&L TO D&E
003 144 341           POPH      /GET POINTER ADDR FROM STACK
003 145 163           MOVME     /STORE 8 LSB'S
003 146 054           INRL
003 147 162           MOVMD     /STORE 2 MSB'S
003 150 054           INRL
003 151 302           JNZ       /ALL 128 NEW POINTS ACQUIRED?
003 152 106           LOOP      /NO, GO BACK AND DO ANOTHER SET OF
003 153 003           0         /BUFFER DISPLAYS
003 154 076           MVIA      /YES, LOAD A WITH 303
003 155 303           303
003 156 062           STA       /MOVE IT TO ADDRESS ALFA
003 157 135           ALFA
003 160 003           0
003 161 303           JMP       /NOW INITIALIZE AND DISPLAY DATA
003 162 112           INIT
003 163 003           0
```

The major portion of the computer's time will be devoted to executing the DISPLA portion of the program, down to the JNZ instruction located at address 003 127. How may clock cycles will be required by the computer to execute these steps, including the JNZ instruction? You may need to reference either *Introductory Experiments in Digital Electronics and 8080A Microcomputer Programming and Interfacing, Book 2* or the *Intel 8080 System User's Manual* for this information.

We found that 73 machine cycles are required. At 1.33 microseconds per cycle (our computer's clock period), approximately 97 microseconds are required to display one 10-bit point. If your computer's clock period is not the same as ours, calculate the time required by your computer to display one 10-bit point. Note it below:

Step 4

Using the result calculated in Step 3, how long will it take to go through the DISPLA loop 128 times to display all of the data points one time?

We calculated the time to be (97 microseconds) × (128 points) = 12.4 milliseconds. Your time may be different if your computer's clock period is different from ours.

If register-pair D is set to a value of 000 300, how many times will the display loop be executed, displaying the 128-point files?

The DISPLA loop will be executed 192 times for each block of 128 10-bit points. How long will the computer spend in the DISPLA loop if we ignore the additional time that is required by the few instructions after the JNZ DISPLA instruction.

Our computer would take about 2.4 seconds .Thus, the data to be acquired by the ADC subroutine, the computer-controlled successive-approximation A/D converter, will be sampled once every 2.4 seconds, updating the data stored in memory.

Step 5

Connect an oscilloscope to the D/A converter's voltage output, V_{DAC} or DACOUT. You should have read/write memory at addresses 002 000 through 002 377 for data storage.

Start the program at address 003 100. Note your observations below:

We observed that random data is displayed on the oscilloscope. Remember that this is not an X *vs.* Y display, but a Y *vs.* t display, the time axis or t axis being supplied by the oscilloscope's time-base generator circuitry.

Step 6

As the program is executing, slowly rotate the potentiometer through to its complete limits of rotation. Take about 20 to 30 seconds to do this. Do you notice any effect upon the data that is being displayed?

We noticed that new data values were entered and displayed on the oscilloscope, their vertical position being in direct proportion to the movement of the potentiometer. The display seemed to flash briefly as new data points were acquired.

Continue to acquire data points until all 128 data points have been acquired. You may wish to adjust the oscilloscope's time base so that only one complete display of the data points is shown. What happens at the end of the data-acquisition period?

We found that the computer continued to display the 128 points in the data file. No new points are added to the file.

Step 7

Without changing the potentiometer setting, or disturbing the interface, restart the program at address 003 100. Does the data-acquisition sequence that you observed in Step 6 repeat itself?

We observed only the display of the data file, consisting of the data that was acquired during Step 6. Can you explain this? Examine the software steps from memory address 003 154 through 003 163. If you need an additional hint, check the contents of location 003 135. What has happened?

A software "trick" has been used to make the program alter itself after it has acquired 128 data points. The program contains steps to substitute an unconditional jump for the conditional, jump-if-not-zero instruction (JNZ = 302) at symbolic addresses ALFA. After one complete pass through the data-acquisition program, the modification takes place allowing only for the display of the data file.

Step 8

Replace the instruction code now stored at symbolic address ALFA with the op code for a JNZ instruction. Restart the program at address 003 100. What happens now?

The program runs as expected, acquiring new data points and displaying them. The data acquired previously in Step 6 is still contained in the buffer memory so it will be displayed until all of the old data points are "displaced" by new ones.

Step 9

The software trick that was noted in Step 7 provides a "lock" that prevents additional data points from being acquired once the complete 128-point data file has been established. Could you add software steps to initialize the JNZ instruction at ALFA at the start of the program? What instructions would you add? Try to add them so that they do not disturb or "move" instructions that are already a part of the program.

We added the following steps:

```
003 073  076      MVIA        /Load A with 302
003 074  302       302
003 075  062       STA        /Store it ALFA
003 076  135      ALFA
003 077  003        0
```

You may wish to try these steps. They eliminate the need to reinitialize the JNZ instruction as outlined in Step 8. If you add these program steps, be sure to start your new program at address 003 073. You might also wish to add program steps that would initialize the data file to one value, say, 000.

The hardware used in this experiment will be used in the next experiment. Do not disconnect it if you plan to do the next experiment. The software will not be used, so you may turn off the power, if you wish.

EXPERIMENT NO. 6
GATE THRESHOLD MEASUREMENTS USING A D/A CONVERTER

Purpose

The purpose of this experiment is to measure the voltages at which the output of a standard 7400-series gate changes from a logic zero to a logic one and also from a logic one to a logic zero.

Discussion

Voltages may be measured indirectly, without the use of an A/D converter or other measuring device. This experiment uses a microcomputer and a 10-bit D/A converter to measure the threshold or switching voltages of a gate. An SN74L00 or SN74LS00 will be used. The D/A converter will be used to generate a ramp which will be applied to the inputs of the gate. The gate's output will be monitored for a change of state. The voltage at which this change occurs will be noted as its binary value, i.e., the binary value applied to the 10-bit D/A converter.

Schematic Diagram of the Circuit

The 10-bit D/A converter that has been used in the previous experiments (Fig. 7-8) will also be used in this experiment. If this interface is not connected to your computer, we refer you to Step 1 of Experiment 1 for the necessary details.

If you have completed Experiment 3, 4, or 5, the necessary input port has already been constructed and should be available on your breadboard. If an input port is not available, you will be instructed to implement it in a later step.

Individual schematic diagrams will be provided in the individual steps in which they will be used.

Step 1

With the 10-bit D/A converter interfaced to the computer, load the following program into the computer's memory. Connect an oscilloscope or a vom to the D/A converter's voltage output, DACOUT or V_{DAC}. Run this program by starting it at address 003 000.

```
                    /THIS IS A SHORT TEST PROGRAM FOR THE
                    /CHECK-OUT OF THE GAIN AND OFFSET
                    *003 000
003 000 041         LXIH       /SET UP A VALUE TO BE OUTPUT
003 001 000         000        /TO THE D/A CONVERTER
```

```
003 002 000         000
003 003 175         MOVAL     /LOAD THE VALUE TO THE D/A
003 004 323         OUT
003 005 004         004
003 006 174         MOVAH
003 007 323         OUT
003 010 005         005
003 011 323         OUT       /STROBE THE D/A
003 012 003         003
003 013 166         HLT       /STOP
```

When the program has been run, what is the voltage that appears at the D/A converter's output?

We found that the output was very close to ground (zero volts). If you are using the AD7522 10-bit D/A converter as shown in Experiment 1, or if you are using the LR-35 Outboard, adjust the OFFSET potentiometer until the D/A converter's output is as close to zero volts as possible. If you are using an equivalent D/A converter interface, an offset control may be present. If so, make the necessary adjustment so that the D/A converter's output is zero volts. If an offset control is not present, go on to the next step.

Step 2

Substitute the data value 377 for the present values stored in the memory locations with addresses 003 001 and 003 002. Again, start the program at address 003 000 and note the voltage output by the D/A converter in the space below:

We observed an output of about 4.6 volts when we performed this step. If your D/A converter has a gain control, adjust it so that the voltage that is output by the D/A converter is between 5.0 and 5.1 volts. If you do not have a gain control on your interface, go on to the next step.

Step 3

Repeat the steps in Step 1 to be sure that the D/A converter is still "zeroed" and that it will output zero volts when the input code is all zeros.

Step 4

If you have just completed Experiment 3, 4, or 5, remove all of the connections to the LM311 comparator integrated circuit.

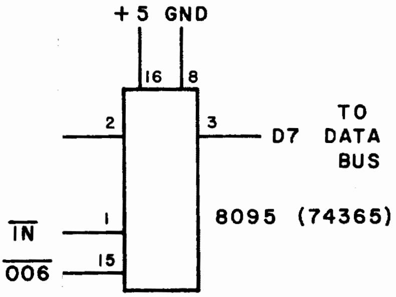


Fig. 7-9. The DM8095 or SN74365 integrated circuit connected as a three-state input port, IN 006.

Leave the DM8095 (SN74365) three-state input port connected to the computer.

If you do not have an input port breadboarded, wire the DM8095 integrated circuit as shown in Fig. 7-9. Be sure to check for the proper connection of the control signals $\overline{\text{IN}}$ and $\overline{006}$ to the DM8095 integrated circuit. Power and ground must also be connected as shown in Fig. 7-9. The SN74365 is equivalent to the DM8095 device.

Step 5

Connect a quad NAND gate integrated circuit as is shown in Fig. 7-10. Only one of the four gate devices will be used. We recommend an SN74L00 or an SN74LS00 for this. Remember to connect the power and ground pins to the proper power buses.

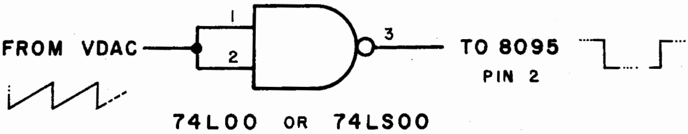


Fig. 7-10.

Step 6

Enter the following program into the computer's memory. The program will generate a linear ramp output that is applied to the inputs of the SN7400-series gate.

```
                      *003 000

003 000 041           LXIH      /SET UP STARTING VALUE FOR RAMP
003 001 000           000
003 002 000           000
003 003 175   RAMP,   MOVAL     /OUTPUT VALUE TO THE D/A
003 004 323           OUT
003 005 004           004
```

```
003 006 174          MOVAH
003 007 323          OUT
003 010 005          005
003 011 323          OUT      /STROBE THE D/A
003 012 003          003
003 013 333          IN       /INPUT GATE'S STATUS
003 014 006          006
003 015 346          ANI      /MASK OUT OTHER BITS
003 016 200          200
003 017 312          JZ       /WHEN IT CHANGES, WE'RE DONE
003 020 026          DONE
003 021 003          0
003 022 043          INXH     /INCREASE RAMP BY ONE
003 023 303          JMP      /OUTPUT NEW VALUE
003 024 003          RAMP
003 025 003          0
003 026 174  DONE,   MOVAH    /GET THE 2 MSB'S
003 027 323          OUT      /OUTPUT THEM TO PORT 0
003 030 000          000
003 031 175          MOVAL    /GET THE 8 LSB'S
003 032 323          OUT      /DISPLAY THEM, TOO
003 033 002          002
003 034 166          HLT      /STOP
```

Step 7

Start the program at memory address 003 000 and note the 10-bit result for each of four trials in the spaces provided. The results will be observed at the LEDs connected to output ports 000 and 002.

TRIAL 1 ______________ TRIAL 2 ______________

TRIAL 3 ______________ TRIAL 4 ______________

For an SN74L00 gate, we observed values of 001 021, 001 014, 001 013, and 001 014 (in octal).

Can these readings be converted to voltages? How?

Each step for a 10-bit D/A converter is about 5 millivolts. Our readings were about 1.34 volts.

What were the voltages that you measured? The actual voltages that you have measured will depend upon the offset and gain of the converter that you are using, power supply noise, etc. Various types of gates from different manufacturers may also cause slight differences in the threshold voltages that you are able to measure.

With a positive-going linear voltage ramp you have measured the voltage at which the NAND-gate switches its output from a logic one to a logic zero.

Step 8

In this step you will measure the voltage at which the NAND-gate output switches from a logic zero to a logic one. What type of a ramp would be required for this?

A negative-going ramp is required. What software changes are needed so that a negative ramp may be generated with the same basic program that has been provided for you?

We made the following changes to the program so that a negative ramp would be generated:

1. Change the data bytes that are loaded into the H and L registers with the LXIH instruction so that register L is loaded with 377 and register H is loaded with 003. You should be able to do this without further help, based upon your previous software experience.
2. Change the INXH instruction to a DCXH instruction (053).

Are any other program changes necessary?

YES. The instruction that is used to test the status of the gate's output must be changed to reflect the change in the gate's initial output when the program is started.

Change the JZ instruction to a JNZ instruction (302).

Step 9

Start the program at memory address 003 000 and again note your observations in the spaces provided below:

TRIAL 1 ______________ TRIAL 2 ______________

TRIAL 3 ______________ TRIAL 4 ______________

We observed outputs of 001 051, 001 050, 001 050, and 001 051. These corresponded to voltages of about 1.5 volts in our system. The difference between the two threshold voltages that we measured was about 160 millivolts.

Step 10

This step is optional. You may wish to use a variable power supply to supply the power to the NAND gate that is being tested. The variable voltage should be kept to between 4.75 and 7.0 volts to prevent damage to the gate. Be sure that there is a good ground connection between the variable power supply and the computer system. Is there any effect on the threshold voltages when the gate's supply voltage is varied? Are the thresholds the same for different gates?

This experiment is based upon one developed by Dr. Mike Daugherty at California State College, Dominguez Hills, CA 90747.

EXPERIMENT NO. 7
USING TWO 8-BIT D/A CONVERTERS; THE BURR-BROWN MP-10 MODULE

Purpose

The purpose of this experiment is to explore the use of the Burr-Brown MP-10 dual D/A converter interface and how it may be used as either a memory-mapped or as an accumulator I/O device.

Discussion

The Burr-Brown MP-10 dual D/A converter interface incorporates all of the logic necessary to interface two 8-bit D/A converters to a microcomputer. The necessary device decoding, gating, latching, and converting circuits are provided in the module.

The MP-10 module incorporates a peripheral interface integrated circuit similar to the Intel 8255 device. This "chip" is under the control of the programmer and it can actually be programmed to change its internal functions through program commands. In the MP-10, it is used as a bus-compatible dual latch, latching eight bits for each of the D/A converters.

When using programmable interface devices, some control of the circuit is needed, generally at the start of a program. You will see that this is the case with the MP-10. An oscilloscope is required in this experiment.

Pin Configuration of the MP-10 Module

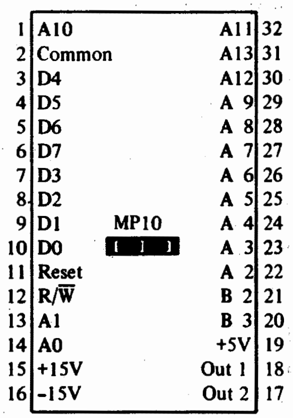


Fig. 7-11.

Schematic Diagram of the Circuit (Fig. 7-12)

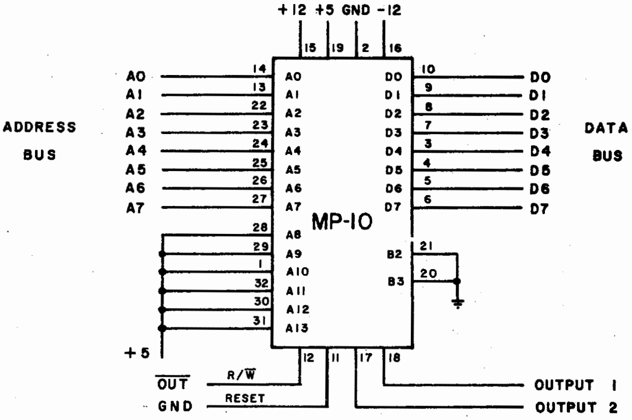


Fig. 7-12.

Step 1

Wire the MP-10 module as shown in the schematic diagram (Fig. 7-12). Take care to be sure that the +12- and −12-volt power inputs are correctly wired. Reverse polarity may cause extensive damage to this module.

Step 2

Once the circuit has been wired, turn on the power and momentarily break the connection between the MP-10's RESET input, pin 11, and the ground connection. This is best done by lifting the wire connection between the two points. This resets the MP-10's internal logic. This must be done each time that power is applied to the MP-10. Generally, a reset push button will perform this function.

Step 3

Enter the following program into the computer's memory and then start it at address 003 000.

```
                          *003 000
003 000 076               MVIA     /SET UP A WITH INIT PATTERN
003 001 200               200
003 002 323               OUT      /OUTPUT IT TO THE D/A'S
003 003 363               363
003 004 043     LOOP,     INXH     /INCR REG PAIR H
003 005 000               NOP
003 006 174               MOVAH    /GET DATA FROM REG H
003 007 000               NOP
003 010 323               OUT      /OUTPUT IT TO A D/A
003 011 360               360
003 012 175               MOVAL    /GET DATA FROM REG L
003 013 000               NOP
003 014 323               OUT      /OUTPUT IT TO A D/A
003 015 361               361
003 016 303               JMP
003 017 004               LOOP
003 020 003               000
```

Alternately connect an oscilloscope to the MP-10's OUTPUT 1 and OUTPUT 2 connections. What do you observe?

You should observe that two varying signals are present. If this is not the case, recheck your interface and the software listed in the program. We found that two negative ramps are produced by the MP-10. One had a short period while the other had a long period.

What is the voltage range of the two analog outputs?

We observed an output from about −10 volts to +8 volts when using the +12- and −12-volt power supply. Burr-Brown recommends a +15- and −15-volt supply. With this type of a supply, the full +10- to −10-volt output range is observed. *You may use a separate ±15-volt supply if there is a good common-ground connection between the computer and the power supply.*

Step 4

Since an increment instruction is used in the short program that was entered into the computer in Step 3 and the MP-10 outputs a negative ramp, what does this indicate is happening to either the analog data or the digital data?

There is a net inversion of the data, either analog or digital.

Is the interface set up for memory-mapped or accumulator I/O? How do you know?

Accumulator I/O is being used. The $\overline{\text{OUT}}$ signal is being used to strobe the MP-10, and an OUT instruction is used to transfer the data from the 8080 to the MP-10's D/A converters.

Step 5

Substitute a DCXH instruction (053) for the INXH instruction that is presently in the program at memory location 003 004. What is its effect when the program is again run?

The ramps are now positive-going. Now substitute the INXH instruction back at memory location 003 004. Can you suggest another type of software modification that would have the same effect of generating positive-going ramps? The decrement instruction may not be used.

We substituted two complement-register-A instructions (CMA = 057) for the two no-operation instructions (NOP = 000) in locations 003 007 and 003 013.

Step 6

Substitute the two CMA instructions in the program as noted in Step 5. Execute the program. Does this produce the positive-going ramp outputs?

We found that it did. You should realize by now that there are generally a number of possible solutions to a given problem. Hardware and software solutions are possible. Can you suggest some hardware solutions to invert the ramps generated in the original program that was presented in Step 3?

Invert the data-bus connections to the MP-10 module or invert the analog output signal with a unity-gain operational amplifier.

Step 7

Examine the schematic diagram (Fig. 7-12) and the pin diagram (Fig. 7-11) for the MP-10 module. You have connected its address inputs A13 through A8 to +5 volts (logic 1). Since the MP-10 is designed to be used primarily as a memory-mapped I/O device, we have hardwired these address pins to simulate the presence of these high address bits in the proper state for operation. We have also wired input pins B2 and B3 to ground (logic 0). These are the address-option pins that have been described in the text of Unit No. 1.

With the present connections to the MP-10, the following 8-bit address codes are used to address the device:

11110000	D/A Converter #1
11110001	D/A Converter #2
11110011	Internal Control Register

Address code 11110010 is not used in the MP-10.

Step 8

Turn off the computer and any separate power supply for the MP-10, and remove the connection between the MP-10's A13 address pin (pin 31) and the +5-volt supply. Make a connection between the MP-10's A13 address pin (pin 31) and the 8080's A15 address pin (pin 36). A small clip lead or jumper may be used for this connection to the 8080 integrated circuit if this address connection is not readily available.

Since the MP-10 requires that all address inputs, A13 through A4, be at a logic 1 to operate, what is the implication of this new connection to the 8080's A15 address pin?

This connection means that the module will only operate when one of the device addresses listed in Step 7 is present, AND when the 8080's A15 address output is a logic one.

Step 9

Remove the connection between the MP-10 module and the 8080's $\overline{\text{OUT}}$ signal. Reconnect the MP-10's pin 12 and the 8080's memory-write signal, $\overline{\text{MW}}$ or $\overline{\text{MEMW}}$. This connection and the connection between the MP-10 and the 8080's A15 address pin have reconfigured the MP-10 as a memory-mapped I/O device. *CAUTION: This configuration is not absolutely decoded. The*

MP-10 will have many, many addresses within the 64K that are possible, since not all of the 8080's 16 address lines have been used to decode the addresses assigned to the MP-10.

Step 10

Enter the following program into the computer's memory. This is a ramp generator program that uses memory-mapped I/O devices, in this case the MP-10. Remember to turn on the power to your system.

```
                        *003 000
003 000 041   START,    LXIH        /SET UP REGS H & L TO MEMORY ADDR
003 001 363             363         /OF D/A'S CONTROL REG
003 002 200             200
003 003 066             MVIM        /LOAD IT WITH CONTROL WORD
003 004 200             200         /CONTROL WORD = 200
003 005 043   AGAIN,    INXH        /INCREMENT REG PAIR H
003 006 042             SHLD        /SAVE H & L AT ADDRESS
003 007 360             360         /200 360 AND 200 361, THE ADDRESSES
003 010 200             200         /OF THE D/A CONVERTERS
003 011 303             JMP         /DO IT AGAIN
003 012 005             AGAIN
003 013 003             0
```

This program outputs the octal control word 200 to the Internal Control Register of the MP-10 (address 200 363) to initialize it. The two 8-bit D/A converters are "located" at addresses 200 360 and 200 361.

Once the program has been entered, momentarily remove the ground connection to the MP-10's RESET input, pin 11. Now start the program. What do you observe?

We observed that negative-going ramps are again being generated.

Step 11

If you wish to use the memory-mapped I/O interface to again generate positive-going ramps, you will need to make some hardware or software changes. Assume that all changes are to be made in the software. Can you suggest some typical software routines that could be used? Try not to use the decrement instruction.

We found that the following program works well. Register pair D is used to hold an incremented count. The values in registers D and E are complemented and then moved to register pair H

Once the data is in registers H and L, it is "output" to the D/A converters with the SHLD instruction.

```
                          *003 000
003 000 041               LXIH       /YOU'VE SEEN THIS INITIALIZATION
003 001 363               363        /BEFORE
003 002 200               200
003 003 066               MVIM
003 004 200               200
003 005 023    RAMP,      INXD       /INCREMENT REG PAIR D
003 006 172               MOVAD      /GET DATA FROM REG D
003 007 057               CMA        /COMPLEMENT IT
003 010 147               MOVHA      /STORE IT IN REG H
003 011 173               MOVAE      /DO THE SAME FOR REG E
003 012 057               CMA
003 013 157               MOVLA
003 014 042               SHLD       /AGAIN, STORE REGS H & L IN MEMORY
003 015 360               360        /ADDR 200 360 AND 200 361
003 016 200               200
003 017 303               JMP        /KEEP GOING
003 020 005               RAMP
003 021 003               0
```

You have probably noticed that there is little advantage to using memory-mapped I/O in this example. Actually, when compared to the accumulator I/O technique, the memory-mapped I/O technique requires more program steps to perform an equivalent function. Many people treat memory-mapped I/O as a panacea. It is not, and its use should be carefully evaluated.

Step 12

Turn off your computer. If you do not plan to do any of the following experiments, the interface may be dismantled. If you will be doing any of the following experiments, make the following changes to again return the MP-10 D/A converter interface to the accumulator I/O configuration.

Remove the connection between the MP-10 and the $\overline{\text{MEMW}}$ output of the 8080. Connect the MP-10's pin 12 to the $\overline{\text{OUT}}$ signal instead. Remove the connection between the MP-10 and the 8080's A15 address output (8080 pin 36). Connect this MP-10 input (pin 31) to +5 volts, instead.

EXPERIMENT NO. 8
USING TWO D/A CONVERTERS FOR A Y-vs.-T

Purpose

The purpose of this experiment is to use two 8-bit D/A converters and appropriate software to provide a Y-*vs.*-T type of display for data. Various data files will be displayed on an oscilloscope.

Discussion

In many cases it is necessary to output a file of data as analog voltages, which are displayed on an oscilloscope or output to a plotter to form a permanent record of the data. In either case, the use of two D/A converters greatly simplifies the output procedure.

One of the D/A converters is used to output the analog data while the other is used to output a steadily increasing voltage so that points are plotted, or displayed, in a continuous left-to-right form. In this experiment, either the Burr-Brown MP-10 D/A converter module or two independent 8-bit D/A converters may be used. We have found that the Signetics NE5018 8-bit D/A converter will work well in this experiment. The interfacing details are presented in the appropriate steps.

Schematic Diagram of the Circuit

The actual schematic diagram of the circuit will vary, depending upon the type of D/A converters that you are using. We recommend that you use either the Burr-Brown MP-10 module or two of the Signetics NE5018 devices.

Step 1

If you will be using the Burr-Brown MP-10 module, refer to Experiment 7 for the interfacing circuit (Fig. 7-12) that is required and then proceed to Step 3. If you will be using the Signetics NE5018 8-bit D/A converter integrated circuit, continue to Step 2. If you have decided to substitute another type of 8-bit D/A converter, wire it, test it, and then go on to Step 3.

Step 2

The Signetics NE5018 8-bit D/A converter is inexpensive and easy to interface to 8080-like microcomputers. Fig. 7-13 is the block diagram of the NE5018 integrated circuit. The pin numbers for the 22-pin package is shown in Fig. 7-14.

You should note that the NE5018 device incorporates an 8-bit latched D/A converter, buffer output amplifier, and reference, all on the same integrated circuit. This makes the interfacing job very easy as shown in the schematic diagram in Fig. 7-15.

Wire two of these interfaces to your computer. You will have to provide a device address decoder for addresses 360 and 361, the addresses to be used as the input “XYZ” for each of the interfaces, respectively. An additional output instruction, OUT 363, is used to initialize the Burr-Brown MP-10, if it is used. If you are using the NE5018, you may ignore the operation of this step in the program.

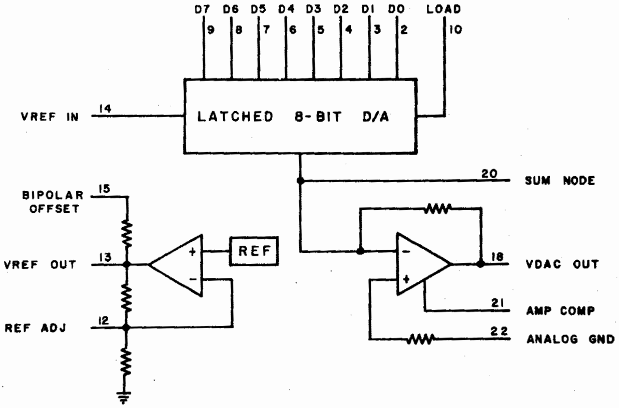


Fig. 7-13. Block diagram for the NE5018 8-bit D/A converter.

Step 3

This experiment requires the use of either a dual-trace oscilloscope with X-Y display capability, or a single-trace oscilloscope with provision for an external time base.

Connect the oscilloscope so that the output from the D/A converter with device-code 361 provides the time base or X-axis input and the output from the D/A converter with device-code 360 provides the data for the Y-axis input.

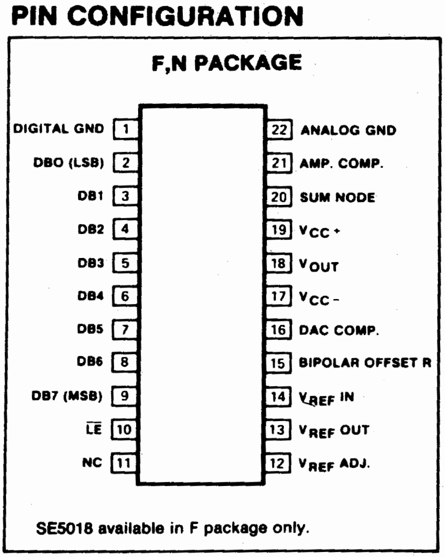


Fig. 7-14.

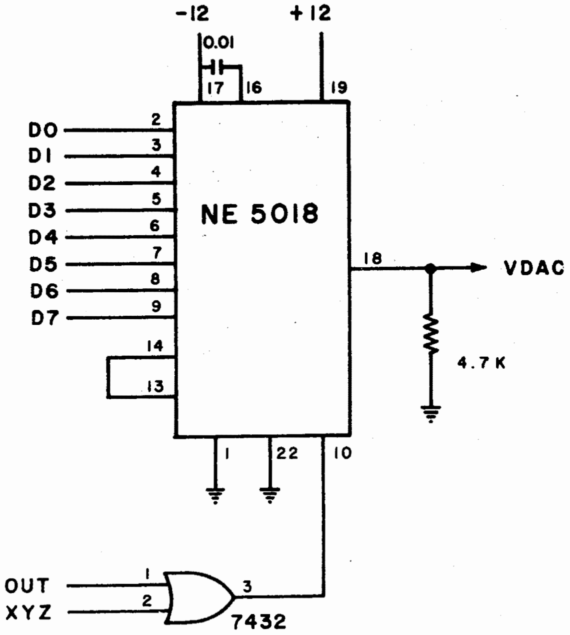


Fig. 7-15. Interface circuit for the NE5018 8-bit D/A converter.

Step 4

With both D/A converters interfaced to the computer and also connected to the oscilloscope, enter the following program into the computer's memory.

```
                        *003 000
003 000 061             LXISP     /SET UP A STACK AREA
003 001 377             377
003 002 003             003
003 003 076             MVIA      /INITIALIZE THE MP-10 MODULE
003 004 200             200       /IF IT IS BEING USED
003 005 323             OUT
003 006 363             363
003 007 041   NEW,      LXIH      /SET UP POINTERS TO DATA BUFFER
003 010 000             000
003 011 002             002
003 012 176   LOOP,     MOVAM     /GET A DATA POINT FROM THE BUFFER
003 013 000             NOP
003 014 323             OUT       /OUTPUT IT TO A D/A CONVERTER
003 015 360             360
003 016 175             MOVAL     /GET THE LOW ADDR OF THE POINT
003 017 000             NOP
003 020 323             OUT       /OUTPUT IT TO THE OTHER D/A
003 021 361             361
```

```
003 022 000           NOP       /OPEN AREA
003 023 000           NOP
003 024 000           NOP
003 025 054           INRL      /INCREMENT LOW ADDR
003 026 303           JMP
003 027 012           LOOP
003 030 003           0
```

This program will output 256 8-bit data words that are stored in memory locations 002 000 through 002 377. The data will be represented by the D/A converter's output voltages that correspond to inputs of 000 through 377. (Minimum scale to full scale.)

Step 5

With the internal time base of the oscilloscope connected or properly switched to provide the timing, start the program. What do you observe? You may have to adjust the oscilloscope trigger controls to "hold" the displayed information steady.

We observed a random, but repetitive, pattern on the oscilloscope screen. Be sure that you have read/write memory available with addresses 002 000 through 002 377.

If the trace is somewhat blurred, you may wish to add a time-delay subroutine to the program so that each point or "dot" will be displayed for a longer period. The following time-delay subroutine, previously listed in other experiments, will work well.

```
                      /THIS IS THE DELAY SUBROUTINE
                      *003 200
003 200 365  DELAY,   PUSHPSW  /SAVE REGISTERS
003 201 325           PUSHD
003 202 021           LXID     /SET TIMING BYTES IN REGISTERS
003 203 046           046      /D & E
003 204 001           001
003 205 033  DEC,     DCXD     /DECREMENT THE REGISTER PAIR
003 206 172           MOVAD
003 207 263           ORAE     /CHECK FOR REG PAIR = 000
003 210 302           JNZ      /IF NOT ZERO, DO IT AGAIN
003 211 205           DEC
003 212 003           0
003 213 321           POPD     /DONE, RESTORE REGISTERS
003 214 361           POPPSW
003 215 311           RET
```

Remember to add the necessary CALL instruction and two address bytes to the main program. These may be substituted for the three NOP instructions at addresses 003 022 through 003 024.

NOTE: If you are using a D/A converter that inverts the analog or digital data, you may wish to use complement instructions (CMA = 057) in place of the two NOP instructions at memory locations 003 013 and 003 017.

If you use the time-delay subroutine, we suggest that you start with timing bytes of 000 002. These seemed to work well for us.

Step 6

Is it possible to examine the displayed data and determine the start and the end of the data file? Which point corresponds to the data value stored at 002 000?

We could not tell. The display was relatively continuous and we had no way of distinguishing between the various data points. You could always "cheat" and load some data value, say 000 or 377, into the first few locations in the data-storage area of memory. These would be displayed to indicate the start of the data file.

Step 7

Load the value 377 into memory locations 002 000 through 002 012. Restart the program and note your observations below. Are these points distinguishable from the others?

Yes, we found them in the data that we were displaying. They are also found one or more times, depending upon the oscilloscope time base and trigger settings.

Step 8

We mentioned that one of the D/A converters may be used to supply an external time base to the oscilloscope. How is this done? Can you examine the software and establish a relationship between the data that is being displayed and the value that is being output to the other D/A converter?

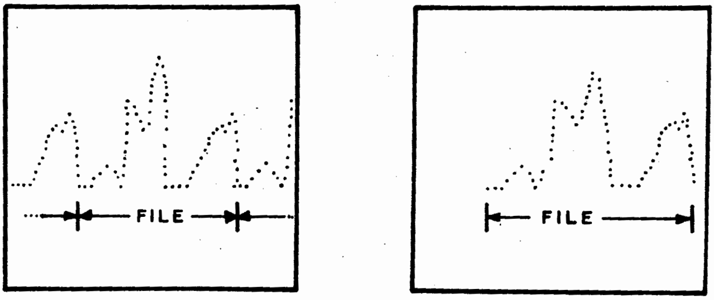
An examination of the program indicates that the D/A converter with device code 361 will output a voltage that is proportional to the low address of each data point. This will always be constant for a given data point. This provides the increasing ramp voltage that is used to "sweep" the electron beam across the oscilloscope's screen, from left to right.

Step 9

Reconfigure the oscilloscope to use the other D/A converter (device 361) to provide the external time base. Start the software. What do you observe?

We observed that a set of data points were displayed. If the D/A converter that is supplying the ramp is connected to an external-time-base input, the display will be a series of data displays, one after the other as was seen previously. If the D/A converter is supplying the ramp to an X-axis input, a single display of the data file will be displayed. There will be no continuous display of one data file "image" after another. The two types of displays are shown in Fig. 7-16.

The oscilloscope that we used was a Heath IO-4510, used in the X-Y mode. The display that we observed was similar to that illustration shown for the Ramp Input to X-Axis Input in Fig. 7-16B. In this mode, it was easy to use the X-axis and Y-axis gain controls to move the data so that it completely filled the space on the oscilloscope's screen.



(A) Ramp input to external-time-base input.

(B) Ramp input to X-axis input.

Fig. 7-16. Data display using two D/A converters.

Step 10

Enter the following data values into the computer's memory at the addresses shown:

```
002 000 000  DATA,  000          002 012 040          040
002 001 001         001          002 013 050          050
002 002 002         002          002 014 060          060
002 003 003         003          002 015 070          070
002 004 004         004          002 016 120          120
002 005 005         005          002 017 140          140
002 006 006         006          002 020 160          160
002 007 007         007          002 021 200          200
002 010 020         020          002 022 220          220
002 011 030         030          002 023 240          240
```

Start the computer. Can you observe the "data" from the 18-point file that you entered? What does it look like?

We observed a series of three ramps with increasing slopes. Each ramp is represented by a series of points. Other data values that were also in the 256-point file are displayed, too.

Step 11

Could the program be modified so that fewer than 256 points are displayed? Can you suggest some software steps that would implement this?

We made the following changes to the program. These modifications monitor the low address of the data point that is being displayed. When a limiting address is reached (set in the program), the computer will again display the file, starting with the first point.

```
                        *003 025
003 025 054             INRL        /INCREMENT LOW ADDR
003 026 175             MOVAL       /MOVE IT TO REG A
003 027 376             CPI         /COMPARE IT TO THE UPPER LIMIT
003 030 000             000         /ADDRESS STORED HERE
003 031 302             JNZ         /IF NOT EQUAL, KEEP DISPLAYING
003 032 012             LOOP        /POINTS IN THE BUFFER
003 033 003             0
003 034 303             JMP         /IF UPPER ADDR IS REACHED,
003 035 007             NEW         /REINITIALIZE THE ADDR POINTERS
003 036 003             0           /AND DISPLAY THE BUFFER AGAIN
```

Enter these new steps and start the program. You may also wish to try the program steps that you devised as part of this step. What do you observe?

We observed that the data file was again displayed, as before. Now, substitute the value 025 for the value 000 in memory location 003 030. This changes the upper limit of the display. Restart the program. What happens now?

Only the first 26_8 data points are displayed. The comparison value or the low address of the last data point to be displayed may be changed to include more or fewer data points in the displayed data.

Would it be possible to output something other that a continuous line or point plot using the Y-*vs.*-T display method? Could shapes or patterns be displayed?

No. Only one point may be displayed for each point on the time axis. Thus, the output cannot have two analog values at the same "time."

Leave the D/A converters interfaced to your computer. They will be used in the next experiment. Power may be turned off at this time.

EXPERIMENT NO. 9
USING TWO D/A CONVERTERS FOR AN X-vs.-Y DISPLAY

Purpose

This experiment explores the use of two D/A converters for output of X- and Y-axis data to a display device for a true X-*vs.*-Y display.

Discussion

The Y-*vs.*-T display technique allows you to display data that has only one value for each point on the X-axis or time axis. The X-*vs.*-Y display technique divides the display area into a matrix of points, any one of which may be "addressed" by one voltage coordinate from one of the D/A converters and by another voltage coordinate from the other D/A converter. Since two 8-bit D/A converters will be used, there will be 256 addressable points on each axis, for a total of 65,536 points which may be addressed independently.

This experiment will use an oscilloscope to present a graphic display of the data.

Schematic Diagram of the Circuit

The interface used in Experiment 8 will be used in this experiment. We refer you to Experiment 8 for the interfacing details.

Step 1

Be sure that the two 8-bit D/A converters that were used in Experiment 8 are still interfaced to your computer. If they are not, refer to the interfacing details in Experiment 8. We used a Burr-Brown MP-10 module and two of the Signetics NE5018 integrated circuits when we tested this experiment.

Step 2

Enter the following program into your computer's memory:

```
003 000 061        LXISP      /SET UP A STACK AREA
003 001 377        377
003 002 003        003
003 003 076        MVIA       /INITIALIZE THE MP-10
```

```
003 004 200          200
003 005 323          OUT
003 006 363          363
003 007 041  NEW,    LXIH      /SET UP X AXIS DATA FILE ADDR
003 010 000          000
003 011 002          002
003 012 021          LXID      /SET UP Y AXIS DATA FILE ADDR
003 013 200          200
003 014 002          002
003 015 176  LOOP,   MOVAM     /GET X AXIS POINT
003 016 000          NOP
003 017 323          OUT       /OUTPUT IT
003 020 360          360
003 021 032          LDAXD     /GET THE Y AXIS DATA
003 022 000          NOP
003 023 323          OUT       /OUTPUT IT, TOO
003 024 361          361
003 025 000          NOP       /SPARE
003 026 000          NOP
003 027 000          NOP
003 030 054          INRL      /INCREMENT X AXIS ADDR POINTER
003 031 034          INRE      /INCREMENT Y AXIS POINTER, TOO
003 032 302          JNZ       /IF E IS NOT = 0, DISPLAY NEXT
003 033 015          LOOP      /POINT IN THE BUFFER
003 034 003          0
003 035 303          JMP       /IF BUFFER DONE, DO IT AGAIN, SO
003 036 007          NEW       /REINITIALIZE THE POINTER ADDRESSES
003 037 003          0
```

After you have loaded this program into the computer's memory, you may wish to go back and check it to correct any errors that you have made during its entry.

Step 3

Connect an oscilloscope which has X-*vs.*-Y display capability so that the Y-axis input is connected to the D/A converter with device address 361 and the X-axis input is connected to the D/A converter that has device address 360. This is the same configuration that was used in Experiment 8.

Start the program at address 003 000. You may have to adjust the gain control for each input so that the points displayed fill the oscilloscope's screen. Each input's position controls may also need adjustment.

Briefly describe your observations:

We observed the display of many points on the oscilloscope screen, much like those points that were displayed in the previous experiment.

You may observe some streaking or smearing between the data points. This is caused by the timing of the display software. The computer spends little time in displaying each point so the points are not much brighter than some of the streaks. Three no-operation instructions (NOP = 000) have been placed in the program so that a call to a delay subroutine may be added. This will cause the computer to display each point for a somewhat longer time.

To make this change, replace the three NOP instructions with the following three-byte call instruction:

```
003 025 315     CALL
003 026 200     Low address of the DELAY subroutine
003 003 003     High address of the DELAY subroutine
```

This step calls the delay subroutine that you have used in many of the previous experiments. It is listed again for clarity, in the space below. Timing bytes of 000 and 002 in locations 003 204 and 003 205, respectively, worked well for us.

```
                         /THIS IS THE DELAY SUBROUTINE
                         *003 200
003 200 365   DELAY,     PUSHPSW  /SAVE REGISTERS
003 201 325              PUSHD
003 202 021              LXID     /SET TIMING BYTES IN REGISTERS
003 203 046              046      /D AND E
003 204 001              001
003 205 033   DEC,       DCXD     /DECREMENT THE REGISTER PAIR
003 206 172              MOVAD
003 207 263              ORAE     /CHECK FOR REG PAIR = 000
003 210 302              JNZ      /IF NOT ZERO, DO IT AGAIN
003 211 205              DEC
003 212 003              0
003 213 321              POPD     /DONE, RESTORE REGISTERS
003 214 361              POPPSW
003 215 311              RET
```

Step 4

Enter the following data points into the computer's memory. Each point has an X and a Y coordinate value.

Address	Data	Address	Data
002 000	000	002 200	000
002 001	040	002 201	000
002 002	140	002 202	000
002 003	200	002 203	377
002 004	300	002 204	377

Once the X-axis and Y-axis data are entered for each of the five points, start the program again. Do you observe any difference? Can you distinguish these five points from the other points being displayed.

We observed no difference in the display and we could not distinguish the five new points. There are just too many other points from which they must be distinguished.

It would be better if the software could be modified to display only those points of interest to us. This is the same type of change that was made to the program in Experiment 8.

Step 5

Make the following changes to the program. Change only the steps that are listed below:

```
003 030 034          INRE     /INCREMENT Y AXIS DATA ADDR
003 031 054          INRL     /INCREMENT X AXIS POINTER, TOO
003 032 175          MOVAL    /GET CONTENTS OF REG L
003 033 376          CPI      /COMPARE IT TO A LIMIT ADDRESS
003 034 010          010      /STORED HERE
003 035 302          JNZ      /IF LIMIT NOT REACHED, DISPLAY
003 036 015          LOOP     /THE NEXT POINT
003 037 003          0
003 040 303          JMP      /IF LIMIT REACHED, REINITIALIZE
003 041 007          NEW      /AND START AGAIN
003 042 003          0
```

Once these new steps have been entered and checked, again start the software. You should observe that fewer points are now displayed. You may wish to experiment with the timing bytes in the DELAY subroutine to decrease any smear between points. If you are using the DELAY subroutine, be sure that the call instruction is still in the main program.

Step 6

Enter the following data into the computer's memory.

Address	Data	Address	Data
002 000	000	002 200	000
002 001	040	002 201	040
002 002	100	002 202	100
002 003	040	002 203	140
002 004	020	002 204	200
002 005	010	002 205	240
002 006	000	002 206	300
002 007	000	002 207	340

Now start the program. What do you observe?

We observed a series of eight data points forming a peak.

Step 7

Enter the following data into the computer's memory.

Address	Data	Address	Data
002 000	000	002 200	000
002 001	040	002 201	200
002 002	000	002 202	377
002 003	200	002 203	340
002 004	377	002 204	377
002 005	340	002 205	200
002 006	377	002 206	000
002 007	200	002 207	040

Again start the program and note your observations of the display in the following space.

We observed a square with a point in each side.

Could a true square be displayed in the Y-*vs*.-T type of display?

No. It is impossible to have two points with the same "time" coordinate. A close approximation could be attempted, but more complex graphics would not be possible.

How could you increase the number of points that are being displayed by the program?

You could go back to the "old" software that was first described in this experiment (for a display of 128 points) or modify the portion of the program that was entered in Step 5 so that the limiting address is changed to reflect your display needs. The address is stored in the memory location with address 003 034. This program has the capability to display up to 128 data points, each with two 8-bit coordinate values.

PART REQUIREMENTS FOR THE EXPERIMENTS

All of the experiments will require an 8080-based microcomputer that provides access to the control signals, data bus signals, and address bus signals for breadboarding. We used an E&L Instruments Mini-Micro Designer System (MMD-1) while preparing and testing these experiments. In addition, we recommend at least one additional solderless breadboarding socket.

The experiments will also require various lengths of #24 or #26 hook-up wire and any tools that students may require, such as small pliers or a small screwdriver for removal of the integrated circuits from the breadboards. You may have to wire device decoders and a lamp monitor, if they are not available. Several schematic diagrams for these devices are provided in the *Introduction to the Experiments* in this unit.

An inexpensive volt-ohm-milliammeter (vom) should be available and an oscilloscope with either two input channels, or a single input and provision for an external time base, should be provided. You will not need one oscilloscope per student; they are readily shared.

If you choose to build the circuit that is the equivalent of the LR-35 Outboard, we suggest that you build it on a piece of perfboard or on a solderless breadboarding socket. The complete schematic is provided in Experiment 1. If you construct this circuit, the following parts will be needed:

1	AD7522 10-bit D/A converter, double-buffered (Analog Devices, Inc., Nordwood, MA 02062)
1	SN7402 quad NOR gate integrated circuit
1	741 operational amplifier integrated circuit
1	79L05 negative 5-volt voltage regulator
2	1N4001 diodes
1	25K, 10-turn trimmer potentiometer
1	5K, 10-turn trimmer potentiometer
1	100K, ¼-watt carbon resistor
1	10K, ¼-watt carbon resistor
1	1N965 15-volt zener diode
2	1-μF tantalum capacitors

The following parts are required for the experiments, exclusive of the parts listed previously for the LR-35 Outboard equivalent D/A converter module:

1	LR-35 Outboard or equivalent circuitry[1]
1	DM8095 or SN74365 three-state buffer integrated circuit (IC)
1	LM311 comparator IC
1	SN74L00 or SN74LS00 quad NAND gate IC
1	SN7432 quad OR gate IC
2	NE5018 latched 8-bit D/A converter IC[2] or MP-10 dual 8-bit D/A converter module,[3] or equivalent circuitry
1	LR-6 quad lamp monitor outboard or equivalent circuit. See Introduction.
2	1000-ohm, ¼-watt carbon resistors

2	4700-ohm, ¼-watt carbon resistors
1	20K, one-turn trimmer potentiometer
2	0.01-μF ceramic disc capacitors
1	Small screwdriver for potentiometer adjustments

NOTES:

1. Available from: E&L Instruments, Inc.
61 First Street
Derby, CT 06418

2. Available from: Signetics Corporation
811 East Arques Avenue
Sunnyvale, CA 94086

3. Available from: Burr-Brown Research Corp.
P. O. Box 11400
Tucson, AZ 85734

Consult each manufacturer for the address of a local distributor.

Appendix A

This appendix contains the National Semiconductor Corporation' Application Note, *Specifying A/D and D/A Converters* (AN-156) It contains many useful definitions and illustrations that will amplif the definitions that have been presented in this book.

SPECIFYING A/D AND D/A CONVERTERS

The specification or selection of analog-to-digital (A/D) or digital-to-analog (D/A) converters can be a chancey thing unless the specifications are understood by the person making the selection. Of course, you know you want an accurate converter of specific resolution; but how do you insure that you get what you want? For example, 12 switches, 12 arbitrarily valued resistors, and a reference will produce a 12-bit DAC exhibiting 12 quantum steps of output voltage. In all probability, the user wants something better than the expected performance of such a DAC. Specifying a 12-bit DAC or an ADC must be made with a full understanding of accuracy, linearity, differential linearity, monotonicity, scale, gain, offset, and hysteresis errors.

This note explains the meanings of and the relationships between the various specifications encountered in A/D and D/A converter descriptions. It is intended that the meanings be presented in the simplest and clearest practical terms. Included are transfer curves showing the several types of errors discussed. Timing and control signals and several binary codes are described as they relate to A/D and D/A converters.

MEANING OF PERFORMANCE SPECS

Resolution describes the smallest standard incremental change in output voltage of a DAC or the amount of input voltage change required to increment the output of an ADC between one code change and the next adjacent code change. A converter with n switches can resolve 1 part in 2^n. The least significant increment is then 2^{-n}, or one least significant bit (LSB). In contrast, the most significant bit (MSB) carries a weight of 2^{-1}. Resolution applies to DACs and ADCs, and may be expressed in percent of full scale or in binary bits. For example, an ADC with 12-bit resolution could resolve 1 part in 2^{12} (1 part in 4096) or 0.0245% of full scale. A converter with 10V full scale could resolve a 2.45mV input change. Likewise, a 12-bit DAC would exhibit an output voltage change of 0.0245% of full scale when the binary input code is incremented one binary bit (1 LSB). Resolution is a design parameter rather than a performance specification; it says nothing about accuracy or linearity.

Accuracy is sometimes considered to be a non-specific term when applied to D/A or A/D converters. A linearity spec is generally considered as more descriptive. An accuracy specification describes the worst case deviation of the DAC output voltage from a straight line drawn between zero and full scale; it includes all errors. A 12-bit DAC could not have a conversion accuracy better than ±½ LSB or ±1 part in 2^{12+1} (±0.0122% of full scale due to finite resolution). This would be the case in figure 1 if there were no errors. Actually, ±0.0122% FS represents a deviation from 100% accuracy; therefore accuracy should be specified as 99.9878%. However, convention would dictate 0.0122% as being an accuracy spec rather than an inaccuracy (tolerance or error) spec.

Accuracy as applied to an ADC would describe the difference between the actual input voltage and the full-scale weighted equivalent of the binary output code; included are quantizing and all other errors. If a 12-bit ADC is stated to be ±1 LSB accurate, this is equivalent to ±0.0245% or twice the minimum possible quantizing error of 0.0122%. An accuracy spec describes the maximum sum of all errors including quantizing error, but is rarely provided on data sheets as the several errors are listed separately.

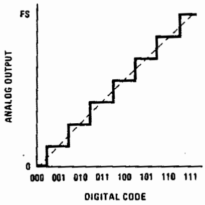


FIGURE 1. Linear DAC Transfer Curve Showing Minimum Resolution Error and Best Possible Accuracy

DA-B15M26/Printed in U.S.A.

Courtesy National Semiconductor Corp.

Quantizing Error is the maximum deviation from a straight line transfer function of a perfect ADC. As, by its very nature, an ADC quantizes the analog input into a finite number of output codes, only an infinite resolution ADC would exhibit zero quantizing error. A perfect ADC, suitably offset ½ LSB at zero scale as shown in figure 2, exhibits only ±½ LSB maximum output error. If not offset, the error will be $^{+1}_{-0}$ LSB as shown in figure 3. For example, a perfect 12-bit ADC will show a ±½ LSB error of ±0.0122% while the quantizing error of an 8-bit ADC is ±½ part in 2^8 or ±0.195% of full scale. Quantizing error is not strictly applicable to a DAC; the equivalent effect is more properly a resolution error.

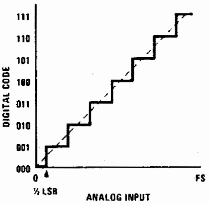


FIGURE 2. ADC Transfer Curve, ½ LSB Offset at Zero

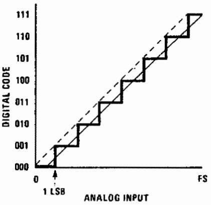


FIGURE 3. ADC Transfer Curve, No Offset

Scale Error (full scale error) is the departure from design output voltage of a DAC for a given input code, usually full-scale code. (See figure 4.) In an ADC it is the departure of actual input voltage from design input voltage for a full-scale output code. Scale errors can be caused by errors in reference voltage, ladder resistor values, or amplifier gain, *et. al.* (See **Temperature Coefficient.**) Scale errors may be corrected by adjusting output amplifier gain or reference voltage. If the transfer curve resembles that of figure 7, a scale adjustment at ¾ scale could improve the overall ± accuracy compared to an adjustment at full scale.

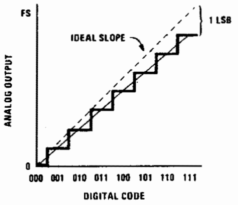


FIGURE 4. Linear, 1 LSB Scale Error

Gain Error is essentially the same as scale error for an ADC. In the case of a DAC with current and voltage mode outputs, the current output could be to scale while the voltage output could exhibit a gain error. The amplifier feedback resistors would be trimmed to correct the gain error.

Offset Error (zero error) is the output voltage of a DAC with zero code input, or it is the required mean value of input voltage of an ADC to set zero code out. (See figure 5.) Offset error is usually caused by amplifier or comparator input offset voltage or current; it can usually be trimmed to zero with an offset zero adjust potentiometer external to the DAC or ADC. Offset error may be expressed in % FS or in fractional LSB.

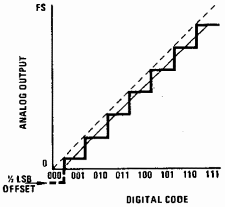


FIGURE 5. Linear, ½ LSB Offset Error

Hysteresis Error in an ADC causes the voltage at which a code transition occurs to be dependent upon the direction from which the transition is approached. This is usually caused by hysteresis in the comparator inside an ADC. Excessive hysteresis may be reduced by design; however, some slight hysteresis is inevitable and may be objectionable in converters if hysteresis approaches ½ LSB.

Linearity, or, more accurately, non-linearity specifications describe the departure from a linear transfer curve for either an ADC or a DAC. Linearity error does not include quantizing, zero, or scale errors. Thus, a specifi-

Courtesy National Semiconductor Corp.

cation of ±½ LSB linearity implies error in addition to the inherent ±½ LSB quantizing or resolution error. In reference to figure 2, showing no errors other than quantizing error, a linearity error allows for one or more of the steps being greater or less than the ideal shown.

Figure 6 shows a 3-bit DAC transfer curve with no more than ±½ LSB non-linearity, yet one step shown is of zero amplitude. This is within the specification, as the maximum deviation from the ideal straight line is ±1 LSB (½ LSB resolution error plus ½ LSB non-linearity). With any linearity error, there is a differential non-linearity (see below). A ±½ LSB linearity spec guarantees monotonicity (see below) and ⩽ ±1 LSB differential non-linearity (see below). In the example of figure 6, the code transition from 100 to 101 is the worst possible non-linearity, being the transition from 1 LSB high at code 100 to 1 LSB low at 110. Any fractional non-linearity beyond ±½ LSB will allow for a non-monotonic transfer curve. Figure 7 shows a typical non-linear curve; non-linearity is 1¼ LSB yet the curve is smooth and monotonic.

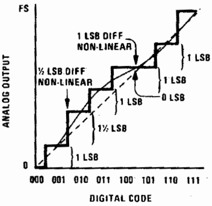


FIGURE 6. ±½ LSB Non-Linearity (Implies 1 LSB Possible Error), 1 LSB Differential Non-Linearity (Implies Monotonicity)

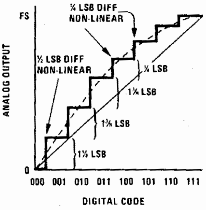


FIGURE 7. 1¼ LSB Non-Linear, ½ LSB Differential Non-Linearity

Linearity specs refer to either ADCs or to DACs, and do not include quantizing, gain, offset, or scale errors. Linearity errors are of prime importance along with differential linearity in either ADC or DAC specs, as all other errors (except quantizing, and temperature and long-term drifts) may be adjusted to zero. Linearity errors may be expressed in % FS or fractional LSB.

Differential Non-Linearity indicates the difference between actual analog voltage change and the ideal (1 LSB) voltage change at any code change of a DAC. For example, a DAC with a 1.5 LSB step at a code change would be said to exhibit ½ LSB differential non-linearity (see figures 6 and 7). Differential non-linearity may be expressed in fractional bits or in % FS.

Differential linearity specs are just as important as linearity specs because the apparent quality of a converter curve can be significantly affected by differential non-linearity even though the linearity spec is good. Figure 6 shows a curve with a ±½ LSB linearity and ±1 LSB differential non-linearity while figure 7 shows a curve with +1¼ LSB linearity and ±½ LSB differential non-linearity. In many user applications, the curve of figure 7 would be preferred over that of figure 6 because the curve is smoother. The differential non-linearity spec describes the smoothness of a curve; therefore it is of great importance to the user. A gross example of differential non-linearity is shown in figure 8 where the linearity spec is ±1 LSB and the differential linearity spec is ±2 LSB. The effect is to allow a transfer curve with grossly degraded resolution; the normal 8-step curve is reduced to 3 steps in figure 8. Similarly, a 16-step curve (4-bit converter) with only 2 LSB differential non-linearity could be reduced to 6 steps (a 2.6-bit converter?). The real message is, "Beware of the specs." Do not ignore or omit differential linearity characteristics on a converter unless the linearity spec is tight enough to guarantee the desired differential linearity. As this characteristic is impractical to measure on a production basis, it is rarely, if ever, specified, and linearity is the primary specified parameter. Differential non-linearity can always be as much as twice the non-linearity, but no more.

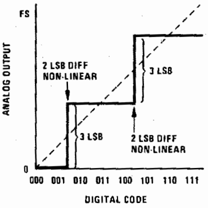


FIGURE 8. ±1 LSB Linear, ±2 LSB Differential Non-Linear

Monotonicity. A monotonic curve has no change in sign of the slope; thus all incremental elements of a monotonically increasing curve will have positive or zero, but never negative slope. The converse is true for decreasing curves. The transfer curve of a monotonic DAC will contain steps of only positive or zero height; and no negative steps. Thus a smooth line connecting all output voltage points will contain no peaks or dips. The transfer function of a monotonic ADC will provide no decreasing output code for increasing input voltage.

Courtesy National Semiconductor Corp.

Figure 9 shows a non-monotonic DAC transfer curve. For the curve to be non-monotonic, the linearity error must exceed ±½ LSB no matter by how little. The greater the linearity error, the more significant the negative step might be. A non-monotonic curve may not be a special disadvantage in some systems; however, it is a disaster in closed-loop servo systems of any type (including a DAC-controlled ADC). A ±½ LSB maximum linearity spec on an n-bit converter guarantees monotonicity to n bits. A converter exhibiting more than ±½ LSB non-linearity may be monotonic, but is not necessarily monotonic. For example, a 12-bit DAC with ±½ bit linearity to 10 bits (not ±½ LSB) will be monotonic at 10 bits but may or may not be monotonic at 12 bits unless tested and guaranteed to be 12-bit monotonic.

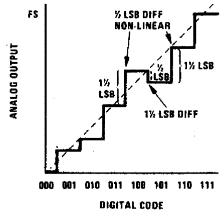
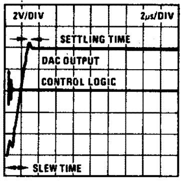
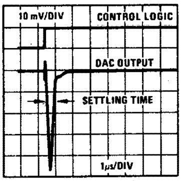


FIGURE 9. Non-Monotonic (Must be > ±½ LSB Non-Linear)



(a) Full-Scale Step



(b) 1 LSB Step

FIGURE 10. DAC Slew and Settling Time

Settling Time is the elapsed time after a code transition for DAC output to reach final value within specified limits, usually ±½ LSB. (See also **Conversion Rate** below.) Settling time is often listed along with a slew rate specification; if so, it may not include slew time. If no slew rate spec is included, the settling time spec must be expected to include slew time. Settling time is usually summed with slew time to obtain total elapsed time for the output to settle to final value. Figure 10 delineates that part of the total elapsed time which is considered to be slew and that part which is settling time. It is apparent from this figure that the total time is greater for a major than for a minor code change due to amplifier slew limitations, but settling time may also be different depending upon amplifier overload recovery characteristics.

Slew Rate is an inherent limitation of the output amplifier in a DAC which limits the rate of change of output voltage after code transitions. Slew rate is usually anywhere from 0.2 to several hundred volts/µs. Delay in reaching final value of DAC output voltage is the sum of slew time and settling time as shown in figure 10.

Overshoot and Glitches occur whenever a code transition occurs in a DAC. There are two causes. The current output of a DAC contains switching glitches due to possible asynchronous switching of the bit currents (expected to be worst at half-scale transition when all bits are switched). These glitches are normally of extremely short duration but could be of ½ scale amplitude. The current switching glitches are generally somewhat attenuated at the voltage output of the DAC because the output amplifier is unable to slew at a very high rate; they are, however, partially coupled around the amplifier via the amplifier feedback network and seen at the output. The output amplifier introduces overshoot and some non-critically damped ringing which may be minimized but not entirely eliminated except at the expense of slew rate and settling time.

Temperature Coefficient of the various components of a DAC or ADC can produce or increase any of the several errors as the operating temperature varies. Zero scale offset error can change due to the TC of the amplifier and comparator input offset voltages and currents. Scale error can occur due to shifts in the reference, changes in ladder resistance or non-compensating RC product shifts in dual-slope ADCs, changes in beta or reference current in current switches, changes in amplifier bias current, or drift in amplifier gain-set resistors. Linearity and monotonicity of the DAC can be affected by differential temperature drifts of the ladder resistors and switches. Overshoot, settling time, and slew rate can be affected by temperature due to internal change in amplifier gain and bandwidth. In short, every specification except resolution and quantizing error can be affected by temperature changes.

Courtesy National Semiconductor Corp.

Long-Term Drift, due mainly to resistor and semiconductor aging can affect all those characteristics which temperature change can affect. Characteristics most commonly affected are linearity, monotonicity, scale, and offset. Scale change due to reference aging is usually the most important change.

Supply Rejection relates to the ability of a DAC or ADC to maintain scale, offset, TC, slew rate, and linearity when the supply voltage is varied. The reference must, of course, remain constant unless considering a multiplying DAC. Most affected are current sources (affecting linearity and scale) and amplifiers or comparators (affecting offset and slew rate). Supply rejection is usually specified only as a % FS change at or near full scale at 25°C.

Conversion Rate is the speed at which an ADC or DAC can make repetitive data conversions. It is affected by propagation delay in counting circuits, ladder switches and comparators; ladder RC and amplifier settling times; amplifier and comparator slew rates; and integrating time of dual-slope converters. Conversion rate is specified as a number of conversions per second, or conversion time is specified as a number of microseconds to complete one conversion (including the effects of settling time). Sometimes, conversion rate is specified for less than full resolution, thus showing a misleading (high) rate.

Clock Rate is the minimum or maximum pulse rate at which ADC counters may be driven. There is a fixed relationship between the minimum conversion rate and the clock rate depending upon the converter accuracy and type. All factors which affect conversion rate of an ADC limit the clock rate.

Input Impedance of an ADC describes the load placed on the analog source.

Output Drive Capability describes the digital load driving capability of an ADC or the analog load driving capacity of a DAC; it is usually given as a current level or a voltage output into a given load.

CODES

Several types of DAC input or ADC output codes are in common use. Each has its advantages depending upon the system interfacing the converter. Most codes are binary in form; each is described and compared below.

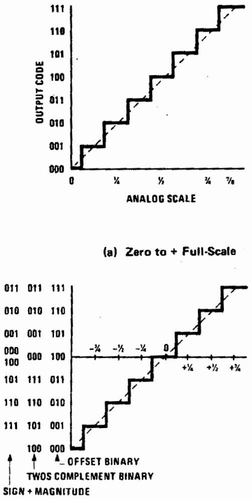
Natural Binary (or simply Binary) is the usual 2^n code with 2, 4, 8, 16, . . . , 2^n progression. An input or output high or "1" is considered a signal, whereas a "0" is considered an absence of signal. This is a positive true binary signal. Zero scale is then all "zeros" while full scale is all "ones."

Complementary Binary (or Inverted Binary) is the negative true binary system. It is identical to the binary code except that all binary bits are inverted. Thus, zero scale is all "ones" while full scale is all "zeros."

Binary Coded Decimal (BCD) is the representation of decimal numbers in binary form. It is useful in ADC systems intended to drive decimal displays. Its advantage over decimal is that only 4 lines are needed to represent 10 digits. The disadvantage of coding DACs or ADCs in BCD is that a full 4 bits could represent 16 digits while only 10 are represented in BCD. The full-scale resolution of a BCD coded system is less than that of a binary coded system. For example, a 12-bit BCD system has a resolution of only 1 part in 1000 compared to 1 part in 4096 for a binary system. This represents a loss in resolution of over 4:1.

Offset Binary is a natural binary code except that it is offset (usually ½ scale) in order to represent negative and positive values. Maximum negative scale is represented to be all "zeros" while maximum positive scale is represented as all "ones." Zero scale (actually center scale) is then represented as a leading "one" and all remaining "zeros." The comparison with binary is shown in figure 11.

Twos Complement Binary is an alternate and more widely used code to represent negative values. With this code, zero and positive values are represented as in natural binary while all negative values are represented in a twos complement form. That is, the twos complement of a number represents a negative value so that interface to a computer or microprocessor is simplified. The twos complement is formed by complementing each bit and then adding a 1; any overflow is neglected. The decimal number −8 is represented in twos complement as follows: start with binary code of decimal 8 (off scale for ± representation in 4 bits so not a valid code in the ± scale of 4 bits) which is 1000; complement it to 0111; add 0001 to get 1000. The comparison with offset binary is shown in figure 11. Note that the offset binary representation of the ± scale differs from the twos complement representation only in that the MSB is complemented. The conversion from offset binary to twos complement only requires that the MSB be inverted.



(a) Zero to + Full-Scale

(b) ± Full-Scale

FIGURE 11. ADC Codes

Courtesy National Semiconductor Corp.

Sign Plus Magnitude coding contains polarity information in the MSB (MSB = 1 indicates a negative sign); all other bits represent magnitude only. This code is compared to offset binary and twos complement in figure 11. Note that one code is used up in providing a double code for zero. Sign plus magnitude code is used in certain instrument and audio systems; its advantage is that only one bit need be changed for small scale changes in the vicinity of zero, and plus and minus scales are symmetrical. A DVM might be an example of its use.

CONTROL

Each ADC must accept and/or provide digital control signals telling it and/or the external system what to do and when to do it. Control signals should be compatible with one or more types of logic in common use. Control signal timing must be such that the converter or connected system will accept the signals. Common control signals are listed below.

Start Conversion (SC) is a digital signal to an ADC which initiates a single conversion cycle. Typically, an SC signal must be present at the fall (or rise) of the clock waveform to initiate the cycle. A DAC needs no SC signal; however, such could be provided to gate digital inputs to a DAC.

End of Conversion (EOC) is a digital signal from an ADC which informs the external system that the digital output data is valid. Typically, an EOC output can be connected to an SC input to cause the ADC to operate in continuous conversion mode. In non-continuous conversion systems, the SC signal is a command from the system to the ADC. A DAC does not supply an EOC signal.

Clock signals are required or must be generated within an ADC to control counting or successive approximation registers. The clock controls the conversion speed within the limitations of the ADC. DACs do not require clock signals.

CONCLUSION

Once the user has a working knowledge of DAC or ADC characteristics and specifications, he should be able to select a converter to suit a specific system need. The likelihood of overspecification, and therefore an unnecessarily high cost, is likewise reduced. The user will also be aware that specific parameters, test conditions, test circuits, and even definitions may vary from manufacturer to manufacturer. For practical production reasons, parameters may not be tested in the same manner for all converter types, even those supplied by the same manufacturer. Using information in this note, the user should, however, be able to sort out and understand those specifications (from any manufacturer) pertinent to his needs.

Manufactured under one or more of the following U.S. patents: 3083262, 3189758, 3231797, 3303356, 3317671, 3323071, 3381071, 3408542, 3421025, 3426423, 3440498, 3518750, 3519897, 3557431, 3560765, 3566218, 3571630, 3575609, 3579059, 3593069, 3597640, 3607469, 3617859, 3631312, 3633052, 3638131, 3648071, 3651565, 3693248.

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 737-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 05-27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632

NATIONAL

National does not assume any responsibility for use of any circuitry described; no circuit patent licenses are implied; and National reserves the right, at any time without notice, to change said circuitry.

Courtesy National Semiconductor Corp.

Appendix B

This appendix contains some data sheets for analog/digital converter products, digital panel meters, and data-acquisition systems. They are provided to give you a representative idea of what information is contained in data sheets and to provide additional information about some of the devices that have been used and mentioned in the text.

Some of the data sheets have been abridged to keep the material as concise as possible and due to publication limitations. We suggest that you contact manufacturers directly for complete data sheets and catalogs of their products before you make a decision to purchase a particular device.

The following data sheets have been provided:

CMOS 10-Bit, Buffered Multiplying D/A Converter AD7522
Courtesy of Analog Devices, Inc., Norwood, MA 02062
8-Bit μP-Compatible D-to-A Converter NE5018
Courtesy of Signetics Corporation, Sunnyvale, CA 94086
MP-10, MP-11 Microprocessor Analog Output Components
Courtesy of Burr-Brown Research Corporation, Tucson, AZ 85734
3½ Digit AC Line Powered DPM AD2009
Courtesy of Analog Devices, Inc., Norwood, MA 02062
World's First Automatic Ranging Digital Panel Meter Model DM-2000AR
Courtesy of Datel Systems, Inc., Canton, MA 02021
Real-Time Analog I/O Interface Model RTI-1200
Courtesy of Analog Devices, Inc., Norwood, MA 02062

MP8600 Series, Burr-Brown Analog I/O Systems (for Intel Microcomputers)
Courtesy of Burr-Brown Research Corporation, Tucson, AZ 85734

ADC0816/ADC0817 Single-Chip Data-Acquisition System
Courtesy of National Semiconductor Corporation, Santa Clara, CA 95051

There are many other module manufacturers and we suggest that you consult the current engineering periodicals for advertisements and new product releases. The latest issue of Electronic Design's *Gold Book* buyers' guide listed over 80 manufacturers of analog/digital converters from integrated circuits to complete systems.

ANALOG DEVICES

CMOS 10-Bit, Buffered Multiplying D/A Converter

AD7522

FEATURES

10-Bit Resolution
8, 9, & 10-Bit Linearity
Microprocessor Compatible
Double Buffered Inputs
Serial or Parallel Loading
DTL/TTL/CMOS Direct Interface
Nonlinearity Tempco: 2ppM of FSR/°C
Gain Tempco: 10ppM of FSR/°C
Very Low Power Dissipation
Very Low Feedthrough

GENERAL DESCRIPTION

The AD7522 is a monolithic CMOS 10-bit multiplying D/A converter, with an input buffer and a holding register, allowing direct interface with microprocessors. Most applications require the addition of only an operational amplifier and a reference voltage.

The key to easy interface to a data bus is the AD7522's ability to load the input buffer in two bytes (an 8-bit and a 2-bit byte), and subsequently move this data to a holding register, where the digital word is converted into an analog current or voltage (with external operational amplifier). The input loading of either 8 or 10 bits can be done in a parallel or serial mode.

The AD7522 is packaged in a 28-pin DIP, and operates with a +15V main supply at 2mA max, and a logic supply of +5V for TTL interface, or +10 to +15V for CMOS interface.

A thin film on high density CMOS process, using silicon nitride passivation, ensures high reliability and excellent stability.

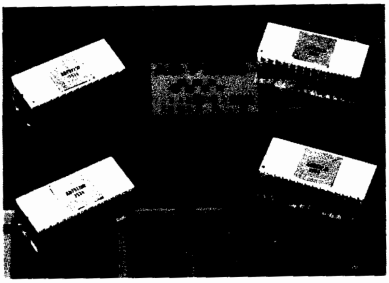
ORDERING INFORMATION

Nonlinearity	Temperature Range		
	0°C to +70°C	-25°C to +85°C	-55°C to +125°C
0.2% (8-Bit)	AD7522JN	AD7522JD	AD7522SD
0.1% (9-Bit)	AD7522KN	AD7522KD	AD7522TD
0.05% (10-Bit)	AD7522LN	AD7522LD	AD7522UD

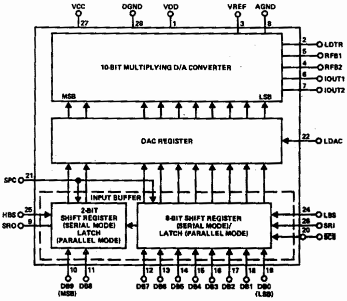
PACKAGE IDENTIFICATION

Suffix "D": Ceramic DIP Package
Suffix "N": Plastic DIP Package

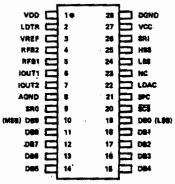
Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.



FUNCTIONAL DIAGRAM



PIN CONFIGURATION



Route 1 Industrial Park; P.O.Box 280; Norwood,Mass. 02062
Tel: 617/329-4700 TWX: 710/394-6577

West Coast	Mid-West	Texas
213/595-1783	312/894-3300	214/231-5094

Courtesy Analog Devices, Inc.

SPECIFICATIONS (VDD = +15V, VCC = +5V, VREF = ±10V, TA = +25°C unless otherwise noted)

PARAMETER[1]	VERSION	TA = 25°C			OVER SPECIFIED TEMP. RANGE		UNITS	TEST CONDITIONS
		MIN	TYP	MAX	MIN	MAX		
STATIC ACCURACY								$\overline{SC8}$ = "1"
Resolution	All	10			10		Bits	
Nonlinearity	J			±0.2			% FSR	−10V ≤ VREF ≤ +10V
	S			±0.2		±0.2	% FSR	
	K			±0.1			% FSR	
	T			±0.1		±0.1	% FSR	
	L			±0.05			% FSR	
	U			±0.05		±0.05	% FSR	
Nonlinearity Tempco[2]	J, K, L		±1			±2	ppm FSR/°C	
	S, T, U					±2	ppm FSR/°C	
Gain Error	J, K, L		±0.3				% Reading	
Gain Error Tempco[2]	J, K, L		±5			±10	ppm of Reading/°C	
	S, T, U					±10	ppm of Reading/°C	
Output Leakage Current at IOUT1 or IOUT2	All					200	nA	IOUT1: DB0 through DB9 = 0 IOUT2: DB0 through DB9 = 1
Power Supply Rejection	J, K, L		50				ppm of Reading/%	
AC ACCURACY								
Feedthrough Error[2]	All		1	10			mV p-p	VREF = 20V p-p; 10 kHz
Output Current Settling Time	J, K, L		500				ns	To 0.05% of FSR for a FSR Step. HBS and LBS Low to High LDAC = 1
REFERENCE INPUT								
Input Resistance	All	5		20			kΩ	
ANALOG OUTPUT								
Output Capacitance								
COUT1	J, K, L		120				pF	All Data Inputs High
COUT2	J, K, L		40				pF	
COUT1	J, K, L		40				pF	All Data Inputs Low
COUT2	J, K, L		120				pF	
DIGITAL INPUTS								
Low State Threshold	All			0.8		0.8	V	VCC = +5V
	All			1.5		1.5	V	VCC = +15V
High State Threshold	All	2.4			2.4		V	VCC = +5V
	All	13.5			13.5		V	VCC = +15V
Input Current	J, K, L		1				µA	
LDAC Pulse Width[2]	All	500			500		ns	LDAC: 0 to +3V
HBS, LBS Pulse Width[2]	All	500			500		ns	HBS, LBS: 0 to +3V
Serial Clock Frequency[2]	All			1		1	MHz	
HBS, LBS Data Set Up[3]	All	250			250		ns	
Data Hold Time[4]	All	500	200		500		ns	

Notes

[1] Specifications subject to change without notice.

[2] Guaranteed by design. Not tested.

[3] Data setup time is the minimum amount of time required for DB0 - DB9 to be stable prior to strobing HBS, LBS.

[4] Data hold time is the minimum amount of time required for DB0 - DB9 to be stable after strobing HBS, LBS.

Courtesy Analog Devices, Inc.

SPECIFICATIONS (continued)

PARAMETER[1]	VERSION	TA = +25°C			OVER SPECIFIED TEMP. RANGE		UNITS	TEST CONDITIONS
		MIN	TYP	MAX	MIN	MAX		
POWER REQUIREMENTS								
IDD	All			2			mA	In Quiescent State
ICC	All			2			mA	

ABSOLUTE MAXIMUM RATINGS

VREF to GND	±25V
VDD to GND	+17V
VCC to GND	+17V
VCC to VDD	+0.4V
IOUT1, IOUT2	±5 mA
Operating Temperature	
JN, KN, LN versions	0°C to +70°C
JD, KD, LD versions	–25°C to +85°C
SD, TD, UD versions	–55°C to +125°C

Storage Temperature	–65°C to +150°C
Power Dissipation (Package)	
Up to +50°C:	
Plastic (Suffix N)	1200 mW
Ceramic (Suffix D)	1000 mW
Derate Above +50°C by	
Plastic (Suffix N)	12 mW/°C
Ceramic (Suffix D)	10 mW/°C
Digital Input Voltage Range	VDD to GND

CAUTION:

1. Do not apply voltages higher than VCC to SRO.
2. Do not apply voltages higher than VDD or less than GND to any other input/output terminal except VREF, RFB1 or RFB2.
3. The digital control inputs are zener protected, however permanent damage may occur on unconnected units under high energy electrostatic fields. Keep unused units in conductive foam at all times.
4. VCC should never exceed VDD by more than 0.4V, especially during power ON or OFF sequencing.

Terminology

RESOLUTION

Value of the LSB. For example, a unipolar n-bit converter has a resolution of (2^{-n}) (VREF). A bipolar n-bit converter has a resolution of $[2^{-(n-1)}]$ [VREF]. Resolution in no way implies linearity.

NONLINEARITY

Error contributed by deviation of the DAC transfer function from a best straight line function. For a multiplying DAC, the nonlinearity should be independent of the sign or magnitude of VREF. Nonlinearity is normally expressed as a percentage of full scale range (% FSR).

GAIN

The "gain" of a converter is that analog scale factor setting that establishes the nominal conversion relationship, e.g., 10V full scale. It is a linear error which can be externally adjusted. (See gain adjustment on page 6.)

OUTPUT LEAKAGE CURRENT

Current which appears on the OUT1 terminal when the DAC register is loaded with all "0's" or on the OUT2 terminal when the DAC register is loaded with all "1's."

Courtesy Analog Devices, Inc.

Pin Function Description

PIN	MNEMONIC	DESCRIPTION
1	VDD	+15V (nominal) Main Supply.
2	LDTR	R-2R Ladder Termination Resistor. Normally grounded for unipolar operation or terminated at IOUT2 for bipolar operation.
3	VREF	Reference Voltage Input. Since the AD7522 is a multiplying DAC, VREF may vary over the range of ±10V.
4	RFB2	Rfeedback ÷ 2; gives full scale equal to VREF/2.
5	RFB1	Rfeedback, used for normal unity gain (at full scale) D/A conversion.
6	IOUT1	DAC Current OUT-1 Bus. Normally terminated at virtual ground of output amplifier.
7	IOUT2	DAC Current OUT-2 Bus, terminated at ground for unipolar operation, or virtual ground of op amp for bipolar operation.
8	AGND	Analog Ground. Back gate of DAC N-channel SPDT current steering switches.
9	SRO	Serial Output. An auxiliary output for recovering data in the input buffer.
10	DB9	Data Bit 9. Most significant parallel data input.
11	DB8	Data Bit 8.
12	DB7	Data Bit 7.
13	DB6	Data Bit 6.
14	DB5	Data Bit 5.
15	DB4 Note 1	Data Bit 4.
16	DB3	Data Bit 3.
17	DB2	Data Bit 2.
18	DB1	Data Bit 1.
19	DB0	Data Bit 0. Least significant parallel data input.
20	$\overline{SC8}$	8-Bit Short Cycle Control. When in serial mode, if $\overline{SC8}$ is held to logic "0," the two least significant input latches in the input buffer are bypassed to provide proper serial loading of 8-bit serial words. If $\overline{SC8}$ is held to logic "1," the AD7522 will accept a 10-bit serial word. Data bits 0(LSB) and DB1 are in a parallel load mode when $\overline{SC8}$ = 0, and should be tied to a logic low state to prevent false data from being loaded.

PIN	MNEMONIC	DESCRIPTION
21	SPC	Serial/Parallel Control. If SPC is a logic "0," the AD7522 will load parallel data appearing on DB0 through DB9 into the input buffer when the appropriate strobe inputs are exercised (see HBS and LBS). If SPC is a logic "1," the AD7522 will load serial data appearing on Pin 26 into the input buffers. Each serial data bit must be "strobed" into the buffer with the HBS and LBS.
22	LDAC	Load DAC: When LDAC is a logic "0," the AD7522 is in the "hold" mode, and digital activity in the input buffer is locked out. When LDAC is a logic "1," the AD7522 is in the "load" mode, and data in the input buffer loads the DAC register.
23	NC	No Connection.
24	LBS	Low Byte Strobe. When in "parallel load" mode (SPC = 0), parallel data appearing on the DB0 (LSB) through DB7 inputs will be "clocked" into the input buffer on the positive going edge of the LBS. When in "serial load" mode (SPC = 1), serial data bits appearing at the serial input terminal, Pin 26, will be "clocked" into the input buffer on the positive going edge of HBS and LBS. (HBS and LBS must be clocked simultaneously when in "serial load" mode.)
25	HBS	High Byte Strobe. When in "parallel load" mode (SPC = 0), parallel data appearing on the DB9 (MSB) and DB8 data inputs will be "clocked" into the input buffer on the positive going edge of HBS. When in "serial load" mode (SPC = 1), serial data bits appearing at the serial input terminal, Pin 26, will be "clocked" into the input buffer on the positive going edges of HBS and LBS. (HBS and LBS must be clocked simultaneously when in "serial load" mode.)
26	SRI	Serial Input.

Courtesy Analog Devices, Inc.

Pin Function Description cont'd

PIN	MNEMONIC	DESCRIPTION
27	VCC	Logic Supply. If +5V is applied, all digital inputs/outputs are TTL compatible. If +10V to +15V is applied, digital inputs/outputs are CMOS compatible.

PIN	MNEMONIC	DESCRIPTION
28	DGND	Digital Ground.

Note 1: Logic "1" applied to a data bit steers that bit's current to the IOUT1 terminal.

DAC Circuit Description

GENERAL CIRCUIT INFORMATION

The AD7522's DAC functional block consists of a highly stable Silicon Chromium thin film R-2R ladder, and ten SPDT N-channel current steering switches. Most applications require the addition of only an output operational amplifier and a voltage or current reference.

The simplified D/A circuit is shown in Figure 1. An inverted R-2R ladder structure is used — that is, the binarily weighted currents are switched between the IOUT1 and IOUT2 bus lines, thus maintaining a constant current in each ladder leg independent of the switch state.

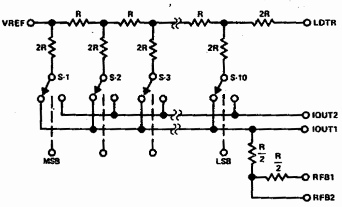


Figure 1. DAC Functional Diagram

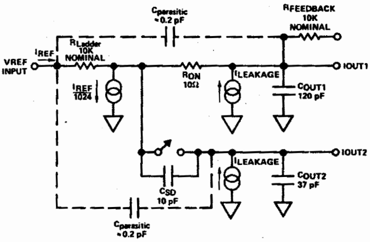


Figure 2. Equivalent Circuit
(Shown for all Digital Inputs High)

EQUIVALENT CIRCUIT

The DAC equivalent circuit is shown in Figure 2. The current source $I_{LEAKAGE}$ is composed of surface and junction leakages to the substrate, while the $I_{REF}/1024$ current source represents the 1LSB of current lost through the ladder termination resistor to ground. The C_{OUT1} and C_{OUT2} output capacitances are as shown when the DAC latches feed the DAC with all "1's." If the DAC latches are loaded with all "0's," C_{OUT1} is 37 pF, while C_{OUT2} is 120 pF. In addition, C_{SD} is shunted by 10 ohms, and the 10 ohm R_{ON} in IOUT1 is replaced by a C_{SD} of 10 pF. When fast amplifiers are used, it will be necessary to provide phase compensation (in the form of feedback capacitance) to cancel the pole formed by $R_{FEEDBACK}$ and C_{OUT} if stability is to be maintained.

Courtesy Analog Devices, Inc.

UNIPOLAR OPERATION

Figure 3 shows the analog circuit connections required for unipolar operation. The input code/output voltage relationship is shown in Table 1.

Zero Offset Adjustment

1. Adjust the op amp's offset potentiometer for < 1 mV on the amplifier junction. (Each millivolt of amplifier V_{OS} causes ±0.66 mV of differential nonlinearity which adds to the ladder nonlinearity.)

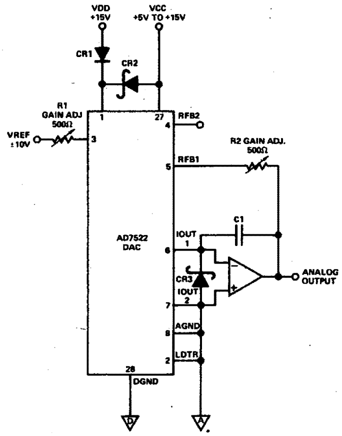


Figure 3. Unipolar Binary Operation (2-Quadrant Multiplication)

Gain Adjustment

1. Set R1 and R2 to 0Ω. Load the DAC register with all "1's."
2. If analog out is greater than −VREF, increase R1 for required full scale output. If analog out is less than −VREF, increase R2 for required full scale output.

TABLE 1
UNIPOLAR CODE TABLE

DIGITAL INPUT	ANALOG OUTPUT
1111111111	−VREF (1 − 2^{-10})
1000000001	−VREF (1/2 + 2^{-10})
1000060000	−VREF/2
0111111111	−VREF (1/2 − 2^{-10})
0000000001	−VREF (2^{-10})
0000000000	0

TABLE 2
BIPOLAR CODE TABLE

DIGITAL INPUT	ANALOG OUTPUT
1111111111	−VREF (1 − 2^{-9})
1000000001	−VREF (2^{-9})
1000000000	0
0111111111	VREF (2^{-9})
0000000001	VREF (1 − 2^{-9})
0000000000	VREF

BIPOLAR OPERATION

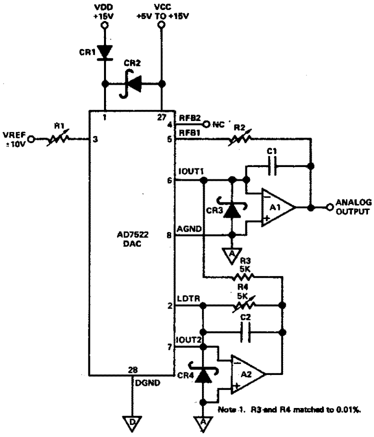
Figure 4 shows the analog circuit connections required for bipolar operation. The input code/output voltage relationship is shown in Table 2.

Zero Offset Adjustment

1. Adjust the offset potentiometer of amplifier A1 and A2 for < 1 mV on the respective summing junctions. If the analog out for code 1000000000 is not zero, sum current into or out of the summing junction of A1 for 0V at analog out.

Gain Adjustment

1. Load the DAC register with all "0's." Set R1 and R2 to 0Ω.
2. If analog out is greater than +VREF, increase R2 until it reads precisely +VREF. If analog out is less than +VREF, increase R1 until it reads precisely VREF.



Note 1. R3 and R4 matched to 0.01%.

Figure 4. Bipolar Operation

Courtesy Analog Devices, Inc.

SINGLE BYTE PARALLEL LOADING

Figure 5 illustrates the logic connections for loading single byte parallel data into the input buffer. DB0 should be grounded on "K" and "T" versions, and DB0 and DB1 should be grounded on "J" and "S" versions for monotonic operation of the DAC. DB9 is always the MSB, whether 8-bit, 9-bit, or 10-bit linear AD7522's are used.

When data is stable on the parallel inputs (DB0-DB9), it can be transferred into the input buffer on the positive edge of the strobe pulse.

Data is transferred from the input buffer to the DAC register when LDAC is a logic "1." LDAC is a level-actuated (versus edge-triggered) function, and must be held "high" at least 0.5μs for data transfer to occur.

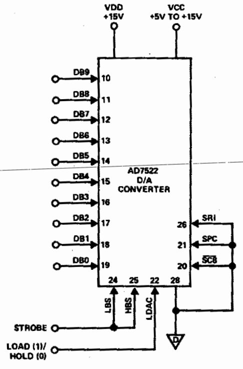


Figure 5. Single Byte Parallel Loading

TWO BYTE PARALLEL LOADING

Figures 6 and 7 show the logic connections and timing requirements for interfacing the AD7522 to an 8-bit data bus for two byte loading of a 10-bit word.

First, the least significant data byte (DB0 through DB7) is loaded into the input buffer on the positive edge of LBS. Subsequently, the data bus is used for status indication and instruction fetching by the CPU. When the most significant data byte (DB8 and DB9) is available on the bus, the input buffer is loaded on the positive edge of HBS. The DAC register updates to the new 10-bit word when LDAC is "high." LDAC may be exercised coincident with, or at any time after HBS loads the second byte of data into the input buffer.

SERIAL LOADING

Figure 8 and Figure 9 show the connections and timing diagram for serial loading.

To load a 10-bit word ($\overline{SC8}$ = 1), HBS and LBS must be strobed simultaneously with exactly 10 positive edges to clock the serial data into the input buffer. For 8-bit words (SC8 = 0), only 8 positive edges are required.

The DAC register can now be loaded by holding LDAC "high."

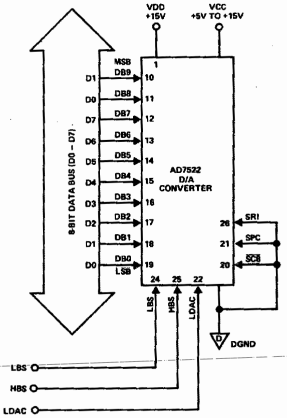


Figure 6. Two Byte Parallel Loading

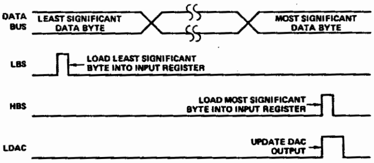


Figure 7. Timing Diagram

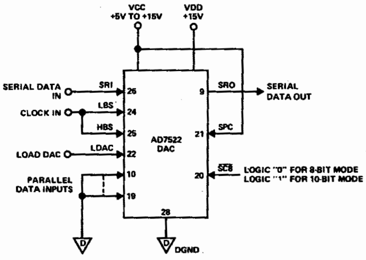


Figure 8. Serial 8- and 10-Bit Loading
(Analog Outputs Not Shown for Clarity)

Courtesy Analog Devices, Inc.

Applications cont'd

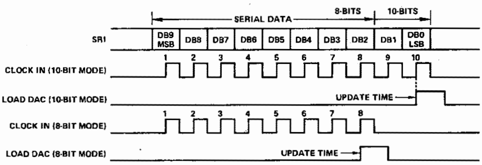


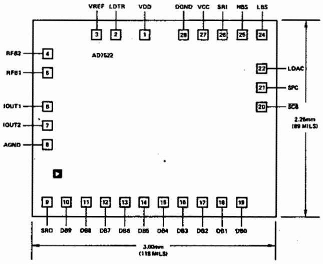
Figure 9. Timing Diagram for Serial 8- and 10-Bit Loading

APPLICATION HINTS

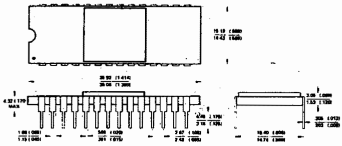
1. CR1 and CR2 on Figures 3 and 4 protect the AD7522 against latch-up if VCC exceeds VDD, and may be omitted if VDD and VCC are driven from the same voltage.
2. Diodes CR3 on Figure 3 and CR3 and CR4 on Figure 4 clamp the amplifier junction to −300 mV if they attempt to swing negative during power up or power down. The input structures of some high-speed op amps can supply substantial current under the transient conditions encountered during power sequencing. It is recommended that the PC layout be able to accommodate the diodes.
3. Fast op amps will require phase compensation for stability due to the pole formed by C_{OUT1} or C_{OUT2} and $R_{FEEDBACK}$.
4. During serial loading, all data inputs (DB0 through BD9), should be grounded.

Mechanical Information

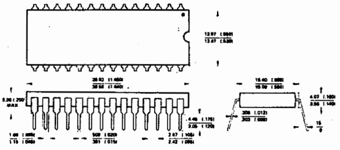
BONDING DIAGRAM



28 PIN CERAMIC DIP



28 PIN PLASTIC DIP



1. Lead no. 1 identified by dot or notch.
2. Dimensions in millimeters (inches).

Courtesy Analog Devices, Inc.

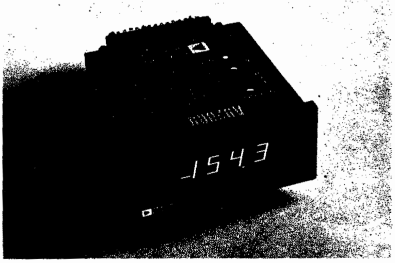
ANALOG DEVICES

3½ Digit AC Line Powered DPM

AD2009

FEATURES
- AC Line Powered
- Bright, Seven Segment Gas Discharge Display
- BCD Data Outputs Standard
- Hold and Trigger Control Signals
- Full Scale Ranges of ±1.999V or ±199.9mV
- Display Blanking Control
- Industry Standard Panel Cutout

APPLICATIONS
- General Purpose DPM Applications Requiring AC Power and a High Visibility Display
- Data Logging and Digital Feedback Control Systems



GENERAL DESCRIPTION

The AD2009 is a low cost 3½ digit, AC line powered DPM designed for general purpose DPM applications. The AD2009 measures bipolar input voltages over full scale ranges of either ±1.999V or ±199.9mV, with an accuracy of ±0.1% reading ±1 digit and displays the readings on large, bright 0.55" (14mm) Beckman gas discharge displays.

LARGE, BRIGHT DISPLAY

For display only applications, the Beckman display offers excellent appearance and visibility. The AD2009 display is easily read up to 50 feet (15m) away and over all ambient lighting conditions. The non-glare lens allows a choice of either red or amber display colors, and is easily silk-screened with company logo or measurement units. External control of decimal points and display blanking is provided.

SIMPLE DATA INTERFACING

Since the AD2009 is designed around TTL logic circuits, parallel BCD data, TTL/DTL compatible, is a standard feature, allowing easy interfacing to a variety of data peripherals, such as digital comparators and line printers. Under internal control, the AD2009 converts at a nominal rate of six conversions per second. Using the Hold and Trigger controls, up to 100 conversions per second can be externally triggered.

INDUSTRY STANDARD CASE DESIGN

In response to industry's urgent need for DPM standardization, Analog Devices has adopted the most popular AC powered DPM panel cutout size for the AD2009 and all future AC line powered DPM's. Since this 3.924" x 1.682" (99.67 x 42.72mm) panel cutout is used by so many AC powered panel meters, the potential DPM customers can be assured that second-sources will be available and future new products will be usable without mechanical changes to their instruments or systems.

DESIGNED AND BUILT FOR RELIABILITY

Design and manufacturing techniques are chosen to insure reliability in the AD2009. Conservative design techniques and thorough component evaluation are only the beginning. Manufacturing processes are monitored by continuous quality assurance inspections to insure proper workmanship and testing. Like every other Analog Devices' DPM, each AD2009 is fully tested for electrical specifications, calibrated, and given one full week of failure free burn-in before shipment.

THEORY OF OPERATION

The AD2009 uses a dual slope conversion technique with an absolute value voltage to current converter input. The entire conversion cycle takes less than 10 milliseconds, allowing a complete conversion to be done during the negative half cycle of the AC line, and the resulting reading is displayed during the positive half cycle of the AC line. This scheme not only insures a flicker free display, but also allows externally triggered conversions at rates up to 100/second for data interfacing applications. In order to insure a bright display even during operation at low line voltages and to help insure the reliability of the Beckman displays, a separate power supply is provided to continually illuminate two "keep-alives" in the Beckman display.

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

Route 1 Industrial Park; P.O. Box 280; Norwood, Mass. 02062
Tel: 617/329-4700 TWX: 710/394-6577
West Coast 213/595-1783
Mid-West 312/894-3300
Texas 214/231-5094

Courtesy Analog Devices, Inc.

SPECIFICATIONS (typical @ +25°C and nominal line voltage)

DISPLAY OUTPUT

- Beckman Seven Segment Gas Discharge Display, 0.55" High (14mm) for Three Data Digits, 100% Overrange and Negative Polarity Indication. Overload indicated by blanking the three data digits and displaying the "1" overrange. The polarity remains valid.
- Decimal Points Selectable at Input.
- Display Blanking

ANALOG INPUT

- Configuration: Bipolar, Single Ended
- Full Scale Range: ±1.999V or ±199.9mV (see S option)
- Automatic Polarity
- Input Impedance: 100MΩDC
- Bias Current, Both Ranges: 3nA @ 2V FS, 20nA @ 200mV FS
- Overvoltage Protection, Both Ranges: 200VDC Sustained

ACCURACY

- ±0.1% ±1 Digit[1]
- Resolution: 1mV or 100μV (S option)
- Temperature Range[2]: 0 to +50°C Operating
  -25°C to +85°C Storage
- Temperature Coefficient:
  Gain (both ranges) – ±60ppm/°C
  Zero Offset (2V Input) – ±30μV/°C
  (200mV Input) – ±10μV/°C
- Warm-Up Time to Rated Accuracy: 15 minutes
- Settling Time to Rated Accuracy: 0.3 sec

NORMAL MODE REJECTION

- 18dB @ 60Hz

COMMON MODE REJECTION (1kΩ source imbalance @ 50-60Hz, with standard shielded transformer)

- 2V Input – 100dB
- 200mV Input – 80dB

COMMON MODE VOLTAGE

- ±300VDC (600VAC p/p) (floated on power supply transformer when BCD outputs and control signals are not used)

CONVERSION TIME

- 10msec

CONVERSION RATE

- Internal Trigger: 6 conversions per second
- External Trigger: 0-100 conversions per second

DIGITAL CONTROL SIGNALS

- DTL/TTL Compatible

	In	Out
Logic "0"	<0.8V	<0.4V
	>2.0V	>2.4V

CONTROL INPUTS[3]

- Display Blank (1TTL Load). Logic "0" or grounding blanks the entire display, not including the decimal points. Logic "1" or open circuit for normal operation. Display blanking has no effect on output data and the display reading is valid immediately upon removal of a blanking signal.
- Hold (1TTL Load). Logic "0" or grounding disables either the external or internal trigger and the last conversion is held and displayed.
- External Trigger (1TTL Load). Positive pulse (500μsec max width) will initiate conversion.
- Decimal Points (Not TTL Compatible). Grounding will illuminate the desired decimal point. External drive circuitry must be capable of withstanding 100V when the decimal points are turned off.

DATA OUTPUTS[3]

- 3BCD Digits (Drives 6TTL Loads). Positive true, unlatched
- Overrange (Drives 6TTL Loads). Unlatched, Logic "0" indicates overrange (≥1000).
- Overload (Drives 6TTL Loads). Unlatched, Logic "0" indicates overload (≥2000).
- Polarity (Drives 6TTL Loads). Latched, Logic "1" indicates positive polarity.
- Status (Drives 10TTL Loads). All digital outputs are valid when status is at Logic "0". Logic "1" indicates conversion is in progress.
- Internal Trigger Output (Not TTL Compatible). When connected to External Trigger Input will cause the AD2009 to convert at 6 conversions per second. This output can only be used for triggering the AD2009.

POWER INPUT

- AC line, 50–60Hz, 4.2 Watts at 60Hz; 4.7 Watts at 50Hz (at nominal line voltages).

CALIBRATION ADJUSTMENTS

- Gain
- Zero
- Recommended recalibration interval – 6 months

SIZE

- 4.18"W x 1.93"H x 4.15"L (106 x 49 x 112mm)
- 4.77"L (121mm) to rear of card edge connector
- Panel cutout required: 1.682 x 3.924" (42.72 x 99.67mm)

WEIGHT

- 15 ounces (425 grams)

OPTIONS[4] – ORDERING GUIDE

- AC Power Inputs (50–60Hz)
  AD2009 – 117VAC
  AD2009/E – 220VAC
  AD2009/F – 100VAC
  AD2009/H – 240VAC
  } ±10%
-
  AD2009 – 1.999VDC Full Scale
  AD2009/S – 199.9mVDC Full Scale
-
  Lens 7 – Red with ADI Logo
  Lens 8 – Red without ADI Logo
  Lens 13 – Amber with ADI Logo
  Lens 14 – Amber without ADI Logo

CONNECTOR

- 30 Pin, 0.156" Spacing Card Edge Connector, Amphenol 225-21524-601 (117) or Equivalent

[1] Guaranteed @ +25°C.
[2] Guaranteed.
[3] Not to be used when the AD2009 is floating on common mode voltages.
[4] Only one input range and AC power input may be specified.
[5] Lens 7 is supplied if no lens option is specified.
Specifications subject to change without notice.

Courtesy Analog Devices, Inc.

INTERFACING THE AD2009

Input Connections

The AD2009 has a single ended input with common analog and digital grounds. When digital control lines and BCD data outputs are not used, the entire DPM can be floated on the power supply transformer at up to 300VDC common mode voltages. If these signals are used, care should be taken to insure against ground loops within the system causing erratic and/or erroneous readings.

Decimal Points

Grounding the proper pin will illuminate the desired decimal point. If external logic drives are used to control the decimal points, drive circuitry must be able to withstand 100V when the decimal points are turned off.

Display Blanking

The entire display (excluding decimal points) may be blanked by applying logic "0" or grounding the proper control input (pin 13). Blanking the display has no effect on the output data or the conversion process. The data remains valid during blanking and the DPM reading is correct immediately upon removal of the blanking signal.

Interfacing Digital Data Outputs

The digital data outputs of the AD2009 are unlatched, positive true, parallel BCD, at DTL/TTL logic levels. As shown in the timing diagram (Figure 1), all data outputs are valid when the STATUS line is low. The STATUS line is high during conversion when erroneous data will be present on the outputs.

TRIGGERING CONVERSIONS

The AD2009 may be triggered internally at six conversions per second, or externally at rates of up to 100 conversions per second. For internal triggering, the Internal Trigger Output (Pin 1) should be connected to the Trigger Input (Pin B). For external triggering, a positive trigger pulse (<500μs width) should be applied to the Trigger Input (Pin B). Whether internal or external triggering is used, the last reading can be held and displayed by grounding or applying logic "0" to the Hold Input. At high conversion rates, the display may flicker unless synchronized to the AC line input, but data outputs will remain valid.

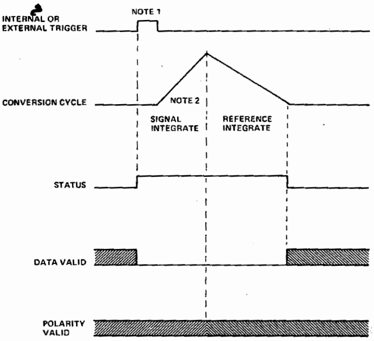
CALIBRATION PROCEDURE

"WARNING: For the safety of personnel and interconnected equipment, all calibration should be done using a plastic trimming tool only."

A precision voltage reference is needed for calibration of the AD2009. The location of calibration potentiometers is shown in Figure 2. Before calibrating the AD2009, allow the unit to warmup to normal operating temperature. Always adjust the zero offset first then the gain.

Zero adjustment: Short the signal input (Pin 2) to the signal ground (Pin 10) and adjust the zero adjustment pot until the meter reads 000.

Gain adjustment: Apply an input of +1.900V (+190.0mV on AD2009/S) and adjust the gain pot until the meter reads 1900 exactly.



NOTES:
1. Minimum pulse width 100ns, maximum 500μs.
2. Approximately 1ms before conversion begins.

Figure 1. AD2009 Timing Diagram

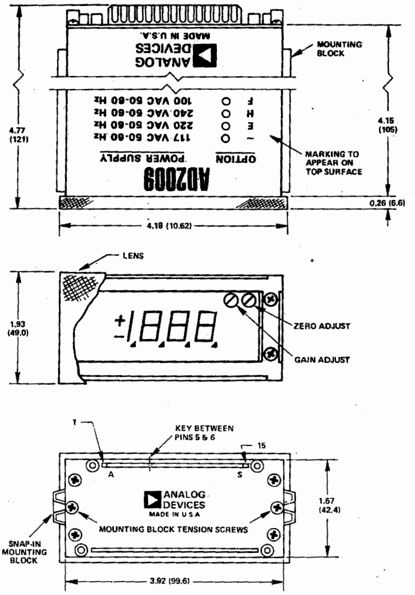
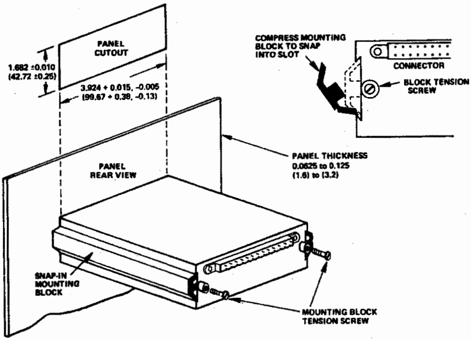


Figure 2. AD2009 Mechanical Outline (Dimensions shown in inches and (mm))

Courtesy Analog Devices, Inc.



MOUNTING INSTRUCTIONS:

1. SLIDE DPM THROUGH PANEL CUTOUT FROM FRONT OF PANEL.
2. SNAP MOUNTING BLOCK INTO SLOT ON DPM SIDES.
3. TIGHTEN MOUNTING BLOCK TENSION SCREWS SNUGLY TO SECURE DPM TO PANEL (DO NOT OVERTIGHTEN!)
4. SNAP LENS ONTO FRONT OF DPM.

Figure 3. AD2009 Mounting Instructions (Dimensions shown in inches and (mm))

PIN REF	PIN FUNCTION		PIN REF	PIN FUNCTION
1	INTERNAL TRIGGER OUT[1]		A	NO CONNECTION
2	SIGNAL INPUT		B	EXTERNAL TRIGGER IN[1]
3	STATUS (PRINT)		C	OVERLOAD
4	POLARITY		D	HOLD
5	BCD 8		E	BCD 1
6	BCD 2	KEY	F	BCD 4
7	BCD 80		H	BCD 10
8	BCD 20		J	BCD 40
9	BCD 800		K	BCD 100
10	SIGNAL GROUND		L	DP3/XX.X
11	BCD 400		M	DP2/X.XX
12	BCD 200		N	DIGITAL GROUND
13	DISPLAY BLANK		P	DP1/.XXX
14	OVERRANGE		R	SHIELD (EARTH GROUND)
15	AC LINE HI		S	AC LINE LO

[1] Pin 1 and Pin B must be connected for operation with internal trigger.

Figure 4. AD2009 Signal and Pin Designations

Courtesy Analog Devices, Inc.

DATEL SYSTEMS, INC.

WORLD'S FIRST AUTOMATIC RANGING DIGITAL PANEL METER

MODEL DM-2000AR

Covered by GSA Contract No. GS-00S-27959

FEATURES

- 3 Full-Auto Ranging-Scales: ±19.99V, ±1.999V, ±199.9mV
- 3-1/2 Digit Solid State LED Display
- Automatic Polarity and Overflow Display
- Operates From Single +5V DC Supply
- Optional Optically Isolated Input
- Optional External Ranging Control

DESCRIPTION

The Datel DM-2000AR is the first digital panel meter to feature automatic ranging over three full scale input ranges. It measures readings from 20 volts down to 100 microvolts without external scaling and increases the dynamic range of the instrument to 103 db over conventional DPM's. The unique autoranging circuitry operates by first sampling an input on the most sensitive range (±199.9mV). If an overange condition is indicated, the circuitry is automatically switched to the next higher range and recycled, and again to the highest range if the second is exceeded. The worst case conversion time is 33 milliseconds (30 conversions per second) if the highest range is selected and somewhat faster on the lower ranges.

The automatic ranging feature, with three full scale input voltage ranges of ±19.99 volts, ±1.999 volts and ±199.9mV offers the user a number of advantages. The cost savings over the parts, installation and test of a front panel range switch alone would justify the use of DM-2000AR. Another advantage is the savings in operator time and the associated errors incurred with the use of a range switch. Also, the versatility of a 20 volt to 100 microvolt dynamic range cannot be discounted in bench testing situations.

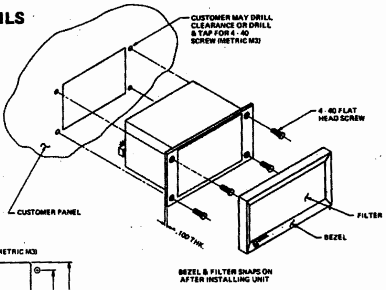
The DM-2000AR has a single ended input with an input impedance of 1 Megohm and overvoltage protection to ±100 volts. Accuracy is ±.1% of full scale, ±1 count with a maximum conversion time of 33 milliseconds. The temperature coefficient of the DM-2000AR is ±100 PPM/°C.

The display of the DM-2000AR is a 3-1/2 digit, seven-segment LED display with automatic polarity indication and automatic overflow indication. A digital input allows for the testing of all readout segments by displaying "+1888". All segments are viewed through a red filter which sharpens contrast and eliminates internal reflections.

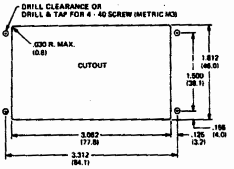
All displayed information is available at the I/O connector in BCD form. Digital inputs and outputs also allow for start-stop control and external clocking. Also, the automatic ranging can be overriden for external (manual) control. In addition, an optical isolation option is available which decouples the analog input section from the digital logic so that ground disturbances caused by the fast-switching digital circuits will not affect the analog input.

The DM-2000AR is packaged in a rugged 3"W x 1-3/4"H x 2-1/4"D LEXAN case with a total weight of less than 6 oz. Construction is entirely modular with snap-apart PC boards. The unit can easily be panel mounted with access to calibration controls obtained by snapping off the front bezel. The entire unit can be removed from its mounting panel and disassembled with just a screw driver in less than one minute.

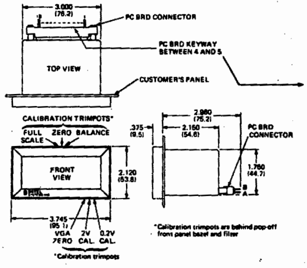
MOUNTING DETAILS


CUSTOMER MAY DRILL CLEARANCE OR DRILL & TAP FOR 4-40 SCREW (METRIC M3)
4-40 FLAT HEAD SCREW
CUSTOMER PANEL
FILTER
BEZEL
.100 THK.
BEZEL & FILTER SNAPS ON AFTER INSTALLING UNIT

PANEL MOUNTING



MECHANICAL DIMENSIONS INCHES (MM)



INPUT/OUTPUT CONNECTIONS

PC BRD CONNECTOR BOTTOM TOP

	A	B		
ANALOG GROUND	1	1	ANALOG INPUT (HI)	
SHIELD GROUND	2	2	SHIELD GROUND	
NOT USED	3	3	NOT USED	
RANGE MODE	4	4	LOGIC GROUND	
$\overline{\text{DECIMAL POINT 100}}$	5	5	BIT 1	OUT
$\overline{\text{DECIMAL POINT 10}}$	6	6	BIT 2	OUT
$\overline{\text{DECIMAL POINT 1}}$	7	7	BIT 4	OUT
$\overline{\text{E.O.C.}}$ (STATUS)	8	8	BIT 8	OUT
$\overline{\text{OVERLOAD SCALE OUT}}$	9	9	BIT 10	OUT
INT. START GATE	10	10	BIT 20	OUT
INT. START ADJ.	11	11	BIT 40	OUT
INT. START OUT	12	12	BIT 80	OUT
START INPUT	13	13	BIT 100	OUT
$\overline{\text{LAMP TEST}}$	14	14	BIT 200	OUT
1000 OUT	15	15	BIT 400	OUT
SIGN OUT	16	16	BIT 800	OUT
EXTERNAL RANGING	17	17	EXTERNAL RANGING	
POWER COMMON	18	18	POWER INPUT, +5VDC	

Edge board PC contacts are on 0.1" centers

Ground A 4 for normal autoranging operation. Jumper A12 to A13 unless using ext. clock. SEE APPLICATIONS

Courtesy Datel Systems, Inc.

SPECIFICATIONS (Typical @ 25 C, 15 minutes warm-up)

ANALOG INPUT

Note: Display reads in volts on high ranges, millivolts on low range.

(single ended)	
Full Scale Input Ranges (automatic ranging)	±19.99V ±1.999V ±199.9mV
Input Bias Current	2nA (all ranges)
Input Overvoltage	±100V max.
Input Impedance	1 MEGOHM (Note 5)

Additional specifications (available only with optical isolation option):

Input Configuration	Single ended Bipolar Floating
Common Mode Voltage	±100V_{CM} max. to digital output common
Common Mode Rejection	70dB @ 60 Hz

PERFORMANCE

Accuracy	±.1% of F.S. ±1 count (5)
Resolution	±1 count (±100μvolts)
Temperature Coefficient of Full Scale	±100 PPM/°C
Zero Drift (referred to the input)	30μV/°C (199.9mV and 1.999V range) 150μV/°C (19.99V range)
Conversion Time	33 msec max (see timing diagram)
Input Settling Time	1 m sec for F.S. change
Operating Temperature Range	0°C to +50°C
Storage Temperature Range	-20°C to +85°C
Warm-up Time	15 min to specified accuracy
Input Power (See Note 4)	+5 ±.25VDC at 800mA (max), <50mV spikes

DISPLAY OUTPUT

Display Type	red LED seven segment 0.3" high
Data	3-1/2 digits (1999 max count)
Polarity	±automatically displayed
Decimal Point	automatically displayed
Overflow	"OF" automatically displayed

DIGITAL OUTPUTS

BCD (3LSD's) Data Outputs **(pins B5 thru B16)**	3 digits (8421) 12 lines positive true Loading: 2 TTL loads (2) Except 800 output: 1 TTL load
1/2 Digit (MSD) Data Output **(pin A15)**	1/2 digit-1 line-positive true Loading: 2 TTL loads (2)
Polarity **(pin A16)**	1 line, "1" = positive, "0" = negative Loading: 2 TTL loads
$\overline{\text{Overload}}$ $\overline{\text{Scale}}$ $\overline{\text{Out}}$ **(pin A9)** **Note: Negative True**	1 line, HIGH - input signal within range LOW - input signal outside range Loading: 2 TTL loads
$\overline{\text{Decimal}}$ $\overline{\text{Point}}$ $\overline{\text{Outputs}}$ Note: Negative True, Decimal points are controlled by auto-ranging logic.	3 lines, negative true Range 19.99V @ Pin 6 Range 1.999V @ Pin 5 Range 199.9mV @ Pin 7 Loading: 1 TTL load (2)
$\overline{\text{End}}$ $\overline{\text{of}}$ $\overline{\text{Conversion}}$ **(pin A8, Busy)**	1 line, HIGH - during the conversion period (display blanked) LOW - conversion complete Loading: 2 TTL loads (2)
Internal Start Output **(pin A12)**	1 line, positive true pulse Loading: 1 TTL load (2)

DIGITAL INPUTS (Dynamic inputs should have TTL rise times)

Ranging Mode Control **(pin A4)**	1 line, "1" = external ranging, "0" = automatic ranging Loading: 3 TTL loads (1)
External Ranging Control **(pins B17 and A17)** **Shown with pin A4 at HI. A17 & B17 are disabled if A4 is LO.**	2 lines - Loading: 1 TTL load (1)

Range	Pin A17	Pin B17
19.99V	"0"	"1"
1.999V	"1"	"0"
199.9mV	"0"	"0"

"11" Range code lights 2 D.P. Don't use.

External Start Convert Command **(Pin A13)** **Start Trigger Clock, may be supplied from int. start, A12**	1 line-min pulse width-100 nsec "0" to "1" (↑) resets output register and blanks display, "1" to "0" (↓) initiates conversion process. Loading: 1 TTL load (1)
Internal Start Gate **(Pin A10)**	1 line, gates internal clock "1" = run, "0" = stop Loading: 1 TTL load (1)
Internal Start Adjust **(Pin A11)**	Controls rate of Internal Start Clock (see application section)
$\overline{\text{Lamp}}$ $\overline{\text{Test}}$ **(Pin A14)**	1 line, negative true, displays +1888 to test all display segments Loading: sink 35mA.

NOTE: Internal start clock is 2 samples/second and is externally adjustabl

ADJUSTMENTS

Zero, balance, full scale, VGA zero, 2V cal and 0.2V cal trimpots accessible behind snap-on front bezel

Recalibration Interval	(Normal conditions) 90 days

PHYSICAL (3)

Case Size	3"W x 1.75"H x 2.25"D
Case Material	Black polycarbonate plastic
Weight	6 oz. approx.
Mounting	Panel mounted through a 1.812" x 3.062" cutout with four 4-40 screws.
Connector **(not included with DPM, add to order)**	Dual 18-pin, PC Edgeboard Type, 0.1" centers (Viking #3VH18/IJN-5 or equal) with key between pins 4 and 5

NOTES:

(1) Low ("0") ≤ + 0.8V, High ("1") ≥ + 2.0V } Inputs

(2) Low ("0") ≤ + 0.4V, High ("1") ≥ + 2.4V } Outputs

(3) Module is fully repairable and features snap together PC Boards

(4) Avoid logic spikes entering DPM on the +5V power input. Use external filtering if required. Recommended power supply is a Datel UPM-5/1000B or equivalent highly regulated type. Power current is 400 to 800mA max depending on digits displayed and sample rate

(5) 1 Megohm input resistor is in series with input summing junc- tion. Accuracy specification wil degrade with significant externı source impedance (> 1KΩ)

GROUND A4 FOR AUTORANGING. JUMPER A12/A13 FOR INTERNAL START CLOCK

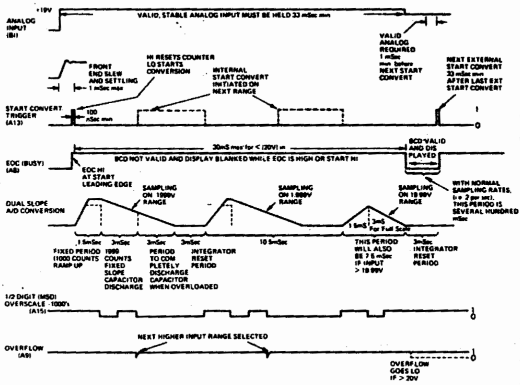
Courtesy Datel Systems, Inc.

DM-2000AR DPM TIMING DIAGRAM

AUTORANGING MODE (PIN A4 GROUNDED)
19.0 VOLTS ANALOG INPUT

NOTES:

1. TIMING IS NOT TO SCALE
2. TIMING IS TYPICAL EXCEPT WHERE INDICATED MIN OR MAX
3. EOC WILL FALL 4.5 TO 33mSec TYP AFTER START CONVERT DEPENDING ON ANALOG INPUT LEVEL
4. DUTY CYCLE OF DISPLAY BLANKING DURING EOC HIGH VARIES WITH SAMPLE RATE AND INPUT RANGE. MAY CAUSE DISPLAY FLICKER IN SOME APPLICATIONS. IF AN EXTERNAL START TRIGGER CLOCK IS USED, KEEP DUTY CYCLE LOW TO AVOID DISPLAY BLANKING AND FLICKER.


+19V
ANALOG INPUT (B1)
VALID, STABLE ANALOG INPUT MUST BE HELD 33 mSec min
VALID ANALOG REQUIRED 1 mSec min before NEXT START CONVERT
NEXT EXTERNAL START CONVERT 33 mSec min AFTER LAST EXT START CONVERT
FRONT END SLEW AND SETTLING
1 mSec max
HI RESETS COUNTER LO STARTS CONVERSION
INTERNAL START CONVERT INITIATED ON NEXT RANGE
START CONVERT TRIGGER (A13)
100 nSec min
1
0
30mS max for < (20V) in
BCD VALID AND DISPLAYED
EOC (BUSY) (A8)
BCD NOT VALID AND DISPLAY BLANKED WHILE EOC IS HIGH OR START HI
EOC HI AT START LEADING EDGE
SAMPLING ON 1999V RANGE
SAMPLING ON 1.999V RANGE
SAMPLING ON 19.99V RANGE
WITH NORMAL SAMPLING RATES, (i.e. 2 per sec), THIS PERIOD IS SEVERAL HUNDRED mSec
DUAL SLOPE A/D CONVERSION
1.5mS
3mS For Full Scale
1.5mSec
3mSec
3mSec
3mSec
10.5mSec
FIXED PERIOD (1000 COUNTS RAMP UP
1999 COUNTS FIXED SLOPE CAPACITOR DISCHARGE
PERIOD TO COMPLETELY DISCHARGE CAPACITOR WHEN OVERLOADED
INTEGRATOR RESET PERIOD
THIS PERIOD WILL ALSO BE 7.5 mSec IF INPUT > 19.99V
3mSec INTEGRATOR RESET PERIOD
1/2 DIGIT (MSD) OVERSCALE -1000's (A15)
1
0
NEXT HIGHER INPUT RANGE SELECTED
OVERFLOW (A9)
1
0
OVERFLOW GOES LO IF > 20V

COMPLETE MODULAR CONSTRUCTION

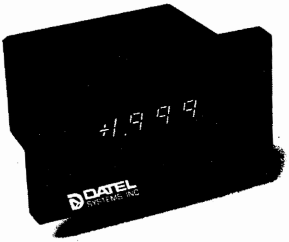
Total modular construction is another plus for the DM-2000 AR.

Servicing is simple and straightforward.

The unit can be removed through the front panel without opening the users' instrument. Once the snap-on front bezel and four mounting screws are removed, the meter slides out of its Lexan case. The procedure is uncomplicated and takes less than one minute. Once removed, modular service is possible due to the five plug-in interconnected PC boards — no wiring or soldering is required.

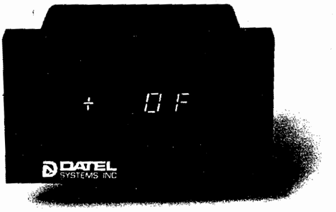
Troubleshooting is accomplished through board substitution and servicing can be completed within five minutes.

A full complement of replacement boards are readily available from Datel's Service Department.



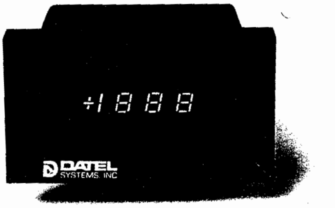
ALPHA NUMERIC INDICATION OF OVERFLOW

When the voltage input exceeds full scale by a minimum of one least significant digit, the characters "OF" are displayed. All data digits are blanked. An example of this would be when full scale is +19.99V, then +20.00V would be the smallest possible overload.



BUILT IN DISPLAY TEST FOR PERIODIC TESTING

Testing for faulty display segments can be achieved in a matter of moments guarding against erroneous readings. Grounding pin 14 at the rear connector will display +1888 to test all possible segment combinations.



Courtesy Datel Systems, Inc.

OPTICALLY ISOLATED INPUT

The DM-2000AR autoranging Digital Panel Meter normally uses a single-ended input with 1 megohm impedance to the input amplifier summing junction. A common bus, (pins A1, A2, B2, B4 and A18) reference analog input ground, digital output ground and +5V power common.

The DM-2000AR-2 version includes optoisolators and a DC/DC converter to give transformer isolation of the analog input up to ±100 Volts to power common. However, the digital outputs are still referenced to the power common/logic ground bus. Differential input impedance of the optoisolated DM-2000AR-2 is 1 megohm.

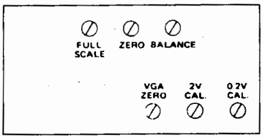
The DM-2000AR-2 isolated DPM reduces false readings in common mode voltage applications.

CALIBRATION PROCEDURE

1. Select manual mode (tie A4 to +5V) and the 20V range. Adjust the "balance" pot to obtain a flickering ± sign on the display.
2. Select the 0.2V range, and adjust the "V.G.A. zero" pot to obtain a flickering sign.
3. Repeat steps 1 and 2 until a flickering sign is obtained on both the 0.2V and 20V ranges (takes 2-3 adjustments).
4. On the 0.2V range, apply an input of ±300μV (be careful of noise on such a small input). Adjust the "zero" pot to obtain a reading of ±003 ±1 digit.
5. Select auto mode (ground A4) and apply input of +18.00V from a precision voltage reference source. Adjust "full scale" to obtain correct reading.
6. Apply input of +1.800V. Adjust "2VCAL" to obtain correct reading.
7. Apply input of +.1800V. Adjust "0.2VCAL" to obtain correct reading.

This completes calibration. Small drifts in the zero can be adjusted with the "V.G.A. zero" pot only which will not require selection of manual mode.

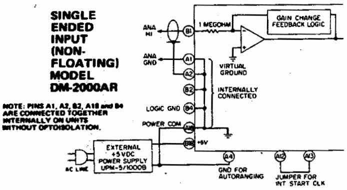
Location of trim pots



View behind front Bezel and Filter

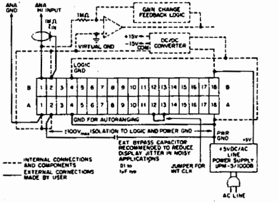
DON'T FLOAT A4. GROUND A4 FOR AUTORANGING, JUMPER A12/A13 FOR INT. START CLOCK

SINGLE ENDED INPUT (NON-FLOATING) MODEL DM-2000AR

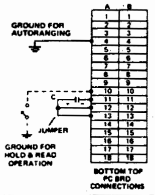


NOTE: PINS A1, A2, B2, A18 and B4 ARE CONNECTED TOGETHER INTERNALLY ON UNITS WITHOUT OPTOISOLATION.

SINGLE ENDED FLOATING INPUT (Optoisolated DM-2000AR-2 Only)


ANA GND
ANA HI INPUT
1MΩ
1MΩ Z_IN
GAIN CHANGE FEEDBACK LOGIC
+15V
-15V
COM
DC/DC CONVERTER
VIRTUAL GND
LOGIC GND
B
A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
GND FOR AUTORANGING
±100V max ISOLATION TO LOGIC AND POWER GND
PWR GND
+5V
EXT BYPASS CAPACITOR RECOMMENDED TO REDUCE DISPLAY JITTER IN NOISY APPLICATIONS
.01 to 1μF typ
JUMPER FOR INT CLK
+5VDC/AC LINE POWER SUPPLY UPM-5/1000B
AC LINE
INTERNAL CONNECTIONS AND COMPONENTS
EXTERNAL CONNECTIONS MADE BY USER

USING THE METER WITH THE INTERNAL "START" CLOCK



NOTE: JUMPER A12/A13 MUST BE USED IF EXT. START TRIGGER IS NOT SUPPLIED

1. CONNECT JUMPER A12 TO A13
   LEAVE PIN A10 OPEN
   CONNECT "C" FROM A11 TO A12, AS SHOWN

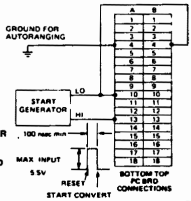
   "C" = OPTIONAL TIMING CAPACITOR TO SLOW DOWN INTERNAL "START" CLOCK. A 1.0μF 50V CAPACITOR WILL HALF RATE. WITHOUT EXTERNAL CAPACITOR THE RATE IS APPROX. 2 SAMPLES PER SECOND

2. FOR HOLD & READ OPERATION GROUND PIN A10

$$C\,(\mu F) \approx \frac{2}{\text{CONV/SEC}} - 1$$

USING THE METER WITH AN EXTERNAL "START"

USE LOW DUTY CYCLE START TRIGGER TO AVOID DISPLAY BLANKING AND FLICKER



APPLY AN EXTERNAL START PULSE TO A13 INITIATE CONVERSION (100 nsec MIN. WIDTH) TRANSITION FROM "LOW" TO "HIGH" RESETS OUTPUT REGISTER READOUT. BLANKS WHILE EOC IS HIGH. THE CONVERSION PROCESS IS INITIATED UPON RETURN FROM "HIGH" TO "LOW".

ORDERING INFORMATION

Model DM-2000AR

Add -2 suffix for optoisolated inputs

DM-2000AR (no optoisolation, less connectors)

DM-2000AR-2 (with optoisolation, less connectors)

Covered by GSA Contract No. GS-00S-27959

Connectors (not included with DPM. Be sure to add to your DPM order)
Solder Tab, Datel #2335-1 (Viking 3VH18/1JN-5)
Wire Wrap, Datel #2335-2 (Viking 3VH18/1JHD-5)

Suggested AC power supply:
UPM-5/1000 B 5V, 1A, 115VAC input
UPM-5/1000 BE 5V, 1A, 230VAC input
Power supply Socket, MS-7

Printed in U.S.A. Copyright © 1977 Datel Systems Inc. All rights reser

DATEL SYSTEMS, INC. 1020 TURNPIKE STREET, CANTON, MASS. 02021 / TEL. (617) 828-8000 / TWX: 710-348-0135 / TELEX: 924461
(714) 835-2751
(408) 733-2424
(213) 933-7256
(301) 840-9490

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

3/77 BULLETIN MARGJ107(

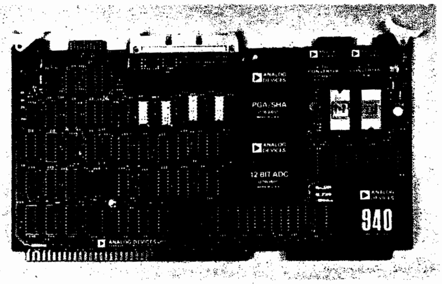
Courtesy Datel Systems, In

ANALOG DEVICES

Real-Time Analog I/O Interface

MODEL RTI-1200

FEATURES
- **Complete Analog I/O Subsystem**
- **Intel SBC-80/10 Compatible**
- **Memory Mapped I/O Interface**
- **Data Acquisition:**
  - **Up to 32 Input Channels**
  - **Sample and Hold Amplifier**
  - **Programmable Gain Amplifier**
  - **12 Bit A/D Converter**
  - **Input Fault Protection**
- **Real-Time Pacer Clock System**
- **On-Board PROM Socket**
- **Two Optional 12 Bit DAC's**
- **Optional 4-20mA Current Outputs**
- **Optional Single +5V Power**



GENERAL DESCRIPTION

The RTI-1200 is a complete analog input/output subsystem that greatly simplifies the task of interfacing analog signals to an Intel SBC-80/10 Single Board Computer, or other 8080-based microcomputers. It is functionally, electrically, and mechanically compatible with the SBC-80/10, and all connections to it are made simply by plugging the RTI-1200 into a slot in a card cage that also contains an SBC-80/10. The RTI-1200 can also be readily interfaced to other 8080-based microcomputers whose address, data, and control busses are accessible.

The RTI-1200 is interfaced to an SBC-80/10 or other 8080 based microcomputer as a block of contiguous memory locations. It combines on a single printed circuit card many features and capabilities which reduce the hardware required to interface analog signals to a microcomputer, and significantly ease the programming effort associated with inputting and outputting analog signals.

DATA ACQUISITION

The RTI-1200's most basic function is data acquisition. This is accomplished with an analog input multiplexer, a programmable gain amplifier, a sample-and-hold amplifier, and a 12 bit A/D converter. These components are shown in the block diagram (Figure 1). The standard RTI-1200 offers either 16 single ended or 8 differential input channels (user selected). An optional multiplexer expander allows for up to 32 single ended or 16 differential input channels. All of the analog inputs are fully protected up to ±28 volts, and additional protection against larger, potentially destructive overloads is afforded by fusing resistors located at the inputs.

The RTI-1200 can be configured by the user to accept 0 to +10V, ±5V, or ±10V full scale input signals. A programmable gain amplifier preceding the A/D converter has software selectable gains of 1, 2, 4 and 8. This expands the dynamic range of the A/D converter to 15 bits, and results in greater input sensitivity. For example, when operating on the 0 to +10V input range with a programmable gain amplifier gain of 8, the actual input range is 0 to +1.25V. The programmable gain amplifier allows the user to program different gains for different input channels, or to have different gains for varying input levels on the same channel. It is even possible to write software to implement automatic gain ranging operation.

Eight of the input channels have provisions for resistors provided by the user that allow the inputs to accept 4—20mA current loop signals. Output data from the A/D converter is in natural binary code for unipolar input ranges, and at the user's option can be either offset binary of two's complement coding when using bipolar input ranges. A special feature of the RTI-1200's data acquisition operation is that the controlling microcomputer's CPU (i.e., the 8080) is not tied up while a conversion is taking place. This significantly enhances system throughput capability, as the CPU is free to pursue other tasks while an A/D conversion is in progress.

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.

Route 1 Industrial Park; P.O. Box 280; Norwood, Mass. 02062
Tel: 617/329-4700 TWX: 710/394-6577

West Coast	Mid-West	Texas
213/595-1783	312/894-3300	214/231-5094

Courtesy Analog Devices, Inc.

SPECIFICATIONS (typical @ +25°C and with +5V and ±15V, unless otherwise noted)

DATA ACQUISITION	
Number of Analog Inputs	
Standard	16 Single-Ended or 8 Diff.
With Multiplexer Expander [1]	32 Single-Ended or 16 Diff.
Multiplexer Switching Characteristics	Break-Before-Make. All Switches Open When Power is Off.
Input Voltage Ranges[2]	0 to +10V, ±5V, ±10V
Programmable Gains[3]	1, 2, 4, 8 Software Selectable
Input Impedance	$>10^9$ Ohms
Input Bias Current	
at +25°C	5nA
over 0 to +70°C	50nA
Diff. Input Bias Current	
at +25°C	3nA
over 0 to +70°C	3.5nA
Input Offset Voltage	Adjustable to Zero
Input Overvoltage Protection	
Continuous Overvoltage	±28 Volts maximum
Overvoltage >±28V	Fusing Resistors
Accuracy	
Resolution	12 Bits
Nonlinearity Error[4]	±1/2LSB typ, ±1LSB max
Diff. Nonlinearity Error	±1/2LSB typ, ±1LSB max
Quantization Error	±1/2LSB max
Gain Error[5]	Adjustable to Zero
Noise Error[6]	±1/2LSB max
Temperature Coefficients	
Gain	±15ppm/°C typ, ±25ppm/°C max
Offset	±25μV/°C Referred to Input
Diff. Nonlinearity	±3ppm/°C max
Settling Time to ±0.01%[7]	10μs max at any Gain
SHA Aperture Time	90ns
SHA Aperture Width	20ns
SHA Aperture Uncertainty	±5ns
Conversion Time	25μs max
Maximum Throughput Rate[8]	28kHz
ANALOG OUTPUTS	
Number of DAC Channels[9]	2
Accuracy	
Resolution	12 Bits
Nonlinearity Error[4]	±1/2LSB
Diff. Nonlinearity Error	±1/2LSB
Voltage Output Characteristics	
Voltage Output Ranges[2]	±2.5V, 0 to +5V, ±5V, 0 to +10V, ±10V
Output Current	5mA min @ ±10V
Settling Time [10]	10μs max
Gain TC	±8ppm/°C typ, ±15ppm/°C max
Offset TC	±5μV/°C typ, ±20μV/°C max
Current Loop Characteristics[11]	
Current Output Range	4 to 20mA
Load Resistance Range	0 to 500Ω
Loop Supply Voltage	+15V to +30V
Settling Time [12]	50μs max
Gain TC	±10ppm/°C typ, ±25ppm/°C max
Offset TC	±0.4μA/°C
Reference Voltage Output	+5.00V ±0.01% @ 5mA max

NOTES

[1] The multiplexer expander is an option, and is shown in the ordering guide as MUX EXP.

[2] The desired range is user selectable with straps.

[3] The input gain of a channel is multiplied by the gain setting of the programmable gain amplifier (e.g., the input range of the 0 to +10V range when using a gain of 8 is 0 to +1.25V).

[4] Defined as deviation from a straight line passing through the end points of the range.

[5] For any one software programmable gain setting. Maximum error of 0.02% when using a programmable gain setting other than the one used during gain calibration.

[6] When using a programmable gain setting of 1. It is ±1.5LSB max when using a programmable gain setting of 8.

[7] For a 20V step. This specification is valid for a step change on one input, or following a channel change, or following a programmable gain change, or simultaneous changes involving any combination of these changes.

[8] Based on a 10μs settling time, followed by a 25μs A/D conversion time. Overall system throughput rate is enhanced because the CPU is not held up during conversions.

[9] Both DAC channels are optional, and are shown in the ordering guide as DAC-1 and DAC-2. Current loop outputs are also optional, and are shown in the ordering guide as CL-1 and CL-2.

[10] To ±0.01% of full scale range following a 20V step.

[11] The current loop characteristics include the effects of the driving D/A converter.

[12] To ±0.02% of full scale current following a full scale step.

[13] Space provided for HC-18/U crystal cut for a frequency of up to 50MHz. User can select to divide crystal frequency by 10^3 or 10^4 on-board the RTI-1200.

[14] The memory map on page 4 shows in detail where these data and control functions appear in memory.

[15] Power requirements shown are for an RTI-1200 with no DAC or current loop options.

[16] The DC/DC power converter is an option that converts +5VDC power to ±15V. It is shown in the ordering guide as DC/DC.

[17] +12V and -5V power is required only if optional PROM is used, but if the PROM is used, it's needed even if the optional DC/DC power converter is selected.

Specificatons subject to change without notice.

REAL-TIME PACER CLOCK SYSTEM	
Modes of Operation	Pacer-Timed Conversion Trigger, Pacer-Timed Interrupt, Pacer Off
Types of Clocks	Variable Frequency R-C, Fixed Frequency Crystal, External
Crystal Clock Freq. [13]	Determined by User Supplied Crystal
Variable Freq. Clock Range	30Hz to 30kHz, User Selectable
LOGIC DRIVER OUTPUTS	
Number Available	2
Characteristics	Open Collector, 30V max, 300mA max Continuous Sink Current per Output
MICROCOMPUTER INTERFACE	
Compatibility	Completely Compatible with Intel SBC-80/10 Bus System
Type of Interface	Interfaces as a Block of Memory Locations, Using Address, Data and Control Busses.
Position in Memory	User Selectable Among any of 14 Possible Locations.
On-Board PROM	Socket for Intel 2708 or Equivalent 1024 Byte x 8 Bit PROM, of which 1008 Bytes are Usable.
Data and Control Functions [14]	Card Select ADC Data End of Conversion Busy Conversion Delayed Time Mark Convert Command Multiplexer Address Gain Select DAC Data Logic Driver Control Pacer Clock Control End of Conversion Interrupt
POWER REQUIREMENTS [15]	
Without DC/DC Option [16]	+15V ±3% @ 40mA -15V ±3% @ 40mA +5V ±5% @ 1.2A
With DC/DC Option	+5V ±5% @ 1.7A
With Optional PROM [17]	+12V ±5% @ 50mA -5V ±5% @ 30mA
TEMPERATURE RANGE	
Operating	0 to +70°C
Storage	-55°C to +85°C
MECHANICAL	6.75" x 12.00" with 0.6" Board-to-Board Spacing

RTI-1200 ORDERING GUIDE

	OPTIONS					
MODEL NUMBER	DAC-1	CL-1	DAC-2	CL-2	DC/DC	MUX EXP.
RTI 1200-001						
-002	X					
-003	X	X				
-004	X		X			
-005	X	X	X			
-006	X	X	X	X		
-011						X
-012	X					X
-013	X	X				X
-014	X		X			X
-015	X	X	X			X
-016	X	X	X	X		X
-101					X	
-102	X				X	
-103	X	X			X	
-104	X		X		X	
-105	X	X	X		X	
-106	X	X	X	X	X	
-111					X	X
-112	X				X	X
-113	X	X			X	X
-114	X		X		X	X
-115	X	X	X		X	X
-116	X	X	X	X	X	X

X DENOTES OPTIONS INCLUDED WITH THE CORRESPONDING MODEL NUMBER. THE OPTIONS ARE DESCRIBED IN NOTES 1, 9, AND 16.

Courtesy Analog Devices, Inc.

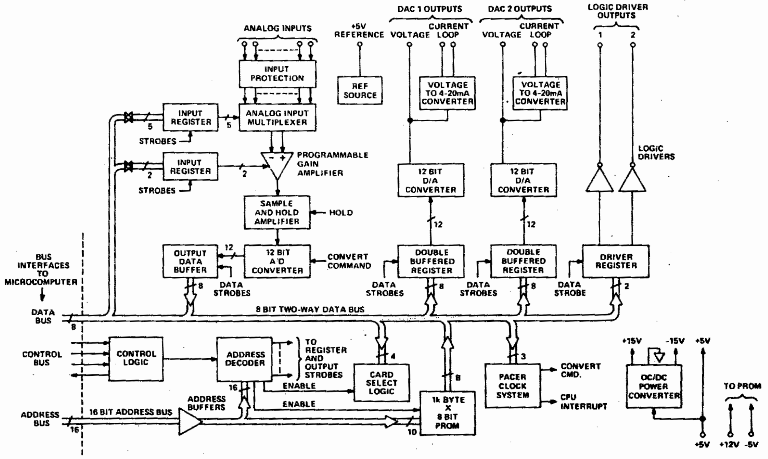


Figure 1. RTI-1200 Functional Block Diagram

ANALOG OUTPUTS

The RTI-1200 has provisions for two optional 12 bit D/A converters which are software driven via double buffered registers. They can be used for such functions as driving an analog recorder, or generating analog control signals. Both D/A converters can be user set to any of five voltage output ranges. The D/A input data is natural binary for unipolar output ranges, and at the user's choice can be offset binary or two's complement for bipolar output ranges. One or both analog output channels can also be optionally equipped with 4–20mA current loop outputs. This permits them to drive directly the 4–20mA control loops often used in process and industrial controls.

ON-BOARD MEMORY

The RTI-1200 contains a socket which can accommodate a 1024 byte x 8 bit PROM, such as the Intel 2708. The user can store programs in such a PROM that would establish setpoints, perform data linearization, execute testing subroutines, or perform other RTI-1200 related tasks. This can be of significance in easing programming effort, particularly when more than one RTI-1200 is used with a single SBC-80/10. Alternatively, the PROM socket can act simply as an extension of the PROM space available on the SBC-80/10, or other microcomputer.

REAL TIME PACER CLOCK SYSTEM

Most real world microcomputer applications requiring interfacing to analog data also require that many operations be referenced to real time. The RTI-1200 is equipped with a highly versatile real time pacer clock system that can provide real time operation without resorting to cumbersome and grossly inefficient software timing loops. Two pacer clocks are provided. One clock is of the R-C variety, and can be set by the user to any frequency between approximately 30Hz and 30kHz. The other is a crystal controlled clock, in which a user supplied crystal generates very precisely timed pulses. These pulses could be used to generate accurately spaced A/D conversions, as is required in Fourier transform analysis, or in generating a highly accurate time-of-day clock. A pulse from either of the clocks, or an externally supplied pacer signal, can either trigger A/D conversions directly, or signal interrupts to the controlling microcomputer.

OTHER FEATURES

Two software driven open-collector logic driver outputs are available for system control functions, such as providing pen lift commands in an analog data recording application. In addition, a precision 5 volt reference is a standard feature of every RTI-1200 for use in calibration and testing (e.g., as a test input on one of the anlaog input channels). Finally, the RTI-1200 can be ordered with an optional DC/DC power converter. In those instances where +15V and -15V power is not readily available, this option allows the RTI-1200 to be operated solely from the same +5V supply that powers the microcomputer used with the RTI-1200.

MEMORY MAP INTERFACE

The RTI-1200 interfaces to the SBC-80/10 as a 1K block (1024 bytes) of memory. The SBC-80/10 can address 65,536 bytes of memory, which can be envisioned as 64 blocks of 1024 bytes each. The RTI-1200 can be configured by the user to occupy one of 14 selected blocks. The 14 possible blocks are spread throughout the 65K address space, so the user should have no trouble positioning the RTI-1200 in a block that does not interfere with already committed address space.

Courtesy Analog Devices, Inc.

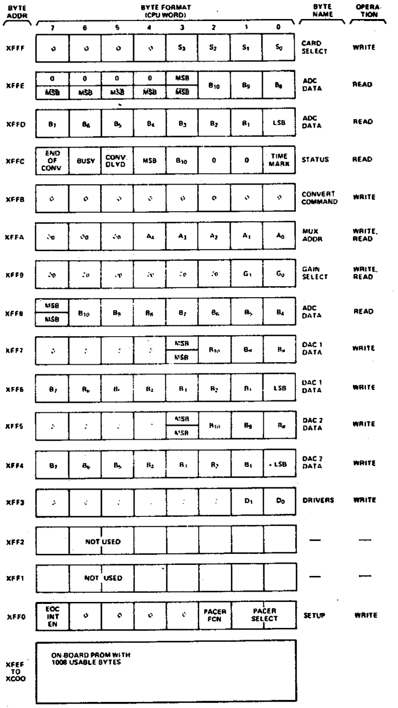
The top 16 addresses (i.e., highest numbered) in the 1K block occupied by the RTI-1200 are devoted to the data and control functions of the RTI-1200. The bottom 1008 addresses are reserved for the on-board PROM. This structure can be seen by referring to Figure 2, the memory map. The byte addresses are shown in hexadecimal notation, with the most significant digit on the left. The addresses from XFFO to XFFF are those associated with the RTI-1200 operation itself (the actual value of hexadecimal digit X is determined by where in the 8080's memory address space the user wishes to have the RTI-1200 appear. All of the memory bytes associated with a particular RTI-1200 will have the same value for X.) The bottom 1008 memory addresses, from XCOO to XFEF, are reserved for the use of an on-board PROM.

Since the RTI-1200 interfaces as memory, any of the 8080's memory reference instructions can be used. The memory map has been carefully thought out so as to make programming as easy as possible. A complete discussion of the memory map is included in the RTI-1200's User's Guide, and only a summary is included here.

When acquiring data, the desired channel is written into address XFFA. The number of the selected channel can also be read back, allowing the use of an increment memory instruction to advance the input multiplexer to the next channel. The desired gain of the programmable gain amplifier is written into address XFF9. A conversion is commanded either by a pulse from one of the pacer clocks, or by writing a convert command into address XFFB. The end of a conversion can be determined either by checking the EOC bit in the status word, or by directing the A/D converter's EOC signal to trigger an interrupt. The A/D converter's output data can be read as 12 bit data in two byte format at addresses XFFD and XFFE. This can be performed with a single instruction by using the 8080's LHLD instruction. If only 8 bit data is required, the 8 most significant bits can be read as a single byte at address XFF8.

Analog output data is loaded into the two 12 bit D/A converters at addresses XFF4 through XFF7. The data is in two byte form, and the data for a single D/A converter can be loaded with a single SHLD instruction. The use of double buffers on the RTI-1200 permits the two data bytes to be loaded simultaneously into the D/A converter. This allows the D/A converter's analog output to move directly from an old analog value to a new analog value without first going to an intermediate value.

Address XFFF is the address of the byte used to select one RTI-1200 from among two or more when multiple RTI-1200's time share the same 1k block of memory. Address XFF3 contains the two bits that control the two logic driver outputs. The exact nature and use of the status and setup bytes, as well as hints on maximizing program efficiency when using the RTI-1200, are covered in the RTI-1200 User's Guide.



BYTE ADDR	BYTE FORMAT (CPU WORD)								BYTE NAME	OPERATION
	7	6	5	4	3	2	1	0		
XFFF	0	0	0	0	S_3	S_2	S_1	S_0	CARD SELECT	WRITE
XFFE	0 MSB	0 MSB	0 MSB	0 MSB	MSB MSB	B_{10}	B_9	B_8	ADC DATA	READ
XFFD	B_7	B_6	B_5	B_4	B_3	B_2	B_1	LSB	ADC DATA	READ
XFFC	END OF CONV	BUSY	CONV. DLYD	MSB	B_{10}	0	0	TIME MARK	STATUS	READ
XFFB	0	0	0	0	0	0	0	0	CONVERT COMMAND	WRITE
XFFA	·0	·0	·0	A_4	A_3	A_2	A_1	A_0	MUX ADDR	WRITE, READ
XFF9	·0	·0	·0	·0	·0	·0	G_1	G_0	GAIN SELECT	WRITE, READ
XFF8	MSB MSB	B_{10}	B_9	B_8	B_7	B_6	B_5	B_4	ADC DATA	READ
XFF7	·	·	·	·	MSB MSB	B_{10}	B_9	B_8	DAC 1 DATA	WRITE
XFF6	B_7	B_6	B_5	B_4	B_3	B_2	B_1	LSB	DAC 1 DATA	WRITE
XFF5	·	·	·	·	MSB MSB	B_{10}	B_9	B_8	DAC 2 DATA	WRITE
XFF4	B_7	B_6	B_5	B_4	B_3	B_2	B_1	LSB	DAC 2 DATA	WRITE
XFF3	·	·	·	·	·	·	D_1	D_0	DRIVERS	WRITE
XFF2		NOT USED							—	—
XFF1		NOT USED							—	—
XFF0	EOC INT EN	0	0	0	0	PACER FCN	PACER SELECT		SETUP	WRITE
XFEF TO XCOO	ON-BOARD PROM WITH 1008 USABLE BYTES									

NOTES:

1. THE SYMBOL · MEANS THE BIT IS IGNORED.
2. BITS SHOWN AS ⌐ HAVE THE UPPER VALUE FOR UNIPOLAR CODES, AND THE LOWER VALUE FOR BIPOLAR CODING
3. THE SYMBOL ·0 MEANS THAT THE BIT IS IGNORED DURING A WRITE, AND READ AS A 0 DURING A READ

Figure 2. RTI-1200 Memory Map

CARD SELECT FEATURE

The RTI-1200 contains a card select feature, which if enabled by the user, allows up to 15 RTI-1200's to share a single 1k block of memory locations in the SBC-80/10. This feature (which is somewhat analogous to memory paging, or memory bank selection) allows one RTI-1200 to be active while the others are in a standby or wait state. This feature can be very useful in simplifying software when it is desirable to use the same subroutines with more than one RTI-1200 in a given system. It also conserves the use of memory space.

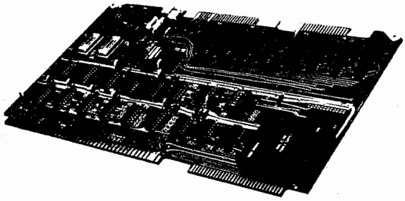
RTI-1200 USER'S GUIDE

Detailed installation and operating information, along with programming hints and a technical explanation of the operation of the RTI-1200 are contained within the RTI-1200 User's Guide. A copy of this manual is shipped with each RTI-1200.

Courtesy Analog Devices, Inc.

microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters

MP8600 SERIES



microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers

BURR-BROWN ANALOG I/O SYSTEMS FOR INTEL MICROCOMPUTERS

- INTEL - SBC80 Compatible
- INTEL - Intellec MDS Compatible

- LOW COST
- EASY TO PROGRAM
  Systems are treated as memory
- REDUCES SYSTEM DEVELOPMENT TIME
  System engineered and specified
  Operates from Computer's +5VDC
  Power Supply if desired
- EASY TO USE
  8 to 64 input channels on one board
  Analog input and output on one board

microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
micrcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters

BURR-BROWN

BB

International Airport Industrial Park•P.O. Box 11400•Tucson,Arizona 85734
Tel: 602-294-1431 • Twx: 910-952-1111 •Cable:BBRCORP • Telex: 66-6491

microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies

Courtesy Burr-Brown Research Corp.

DESCRIPTION

These microcomputer peripherals provide two functions that interface directly to Intel's SBC80 and Intellec MDS microcomputers. The functions are: (1) Analog Data Acquisition and (2) Analog Output. The devices are electrically and mechanically compatible with any SBC80 and Intellec MDS. Both analog input and output systems are contained on a single printed circuit board that is treated as memory input or output by the CPU. The cards will mate to any memory or I/O slot. They are compatible with the 0.6" spacing of the SBC80 or the 0.75" spacing of the Intellec MDS. The analog interface for each system is a connector at the opposite edge of the board from the bus connector.

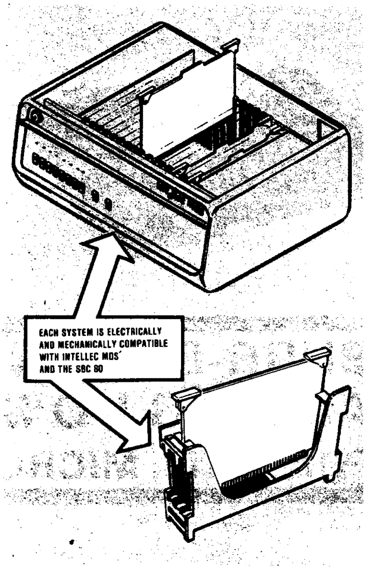
The Data Acquisition system is available with up to 64 channels single-ended on one board. It includes an input multiplexer, high gain instrumentation amplifier, 8-bit A/D converter along with all the necessary timing, decoding and control logic. A DC/DC converter (+5V to ±15V) is also available so that only the computer's power supply is required. The Data Acquisition System is available with two optional 8-bit D/A converters to provide analog input and output on the same board.

THEORY OF OPERATION

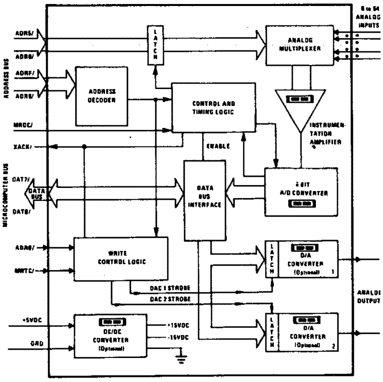
When programming with these peripherals, they are treated as memory locations. Any memory reference instruction can be used. Both the A/D converter output and the D/A converter input are 8-bit words so one memory location is needed for each channel. Because the address block occupied by each peripheral is user selectable, it can be placed anywhere in memory.

Because these units are treated as memory, a minimum of instructions are needed to read an input channel or to set the input of a D/A converter. For instance, the LHLD (load) instruction followed by the proper address can be used to read data from two successive analog input channels. It will automatically select the desired channel, initiate conversion and when conversion is complete, transfer the A/D converter output for the first channel to the 8080's L register and the second channel to the H register. Likewise a single LDA instruction can be used to read one analog input channel.

All of these systems are jumpered at the factory with the first channel at address $F700_{16}$. Each subsequent channel is one memory location past the start of the last channel so that the second channel is at location $F701_{16}$.


EACH SYSTEM IS ELECTRICALLY AND MECHANICALLY COMPATIBLE WITH INTELLEC MDS' AND THE SBC 80

ANALOG INPUT/ OUTPUT SYSTEM



Courtesy Burr-Brown Research Cor

SPECIFICATIONS

(All specifications typical at 25°C unless otherwise noted.)
Prices and specifications subject to change without notice.

ANALOG INPUT/OUTPUT SYSTEM	
ANALOG INPUT	
Number of analog inputs	
8 differential	MP8608
16 single-ended	MP8616
32 differential or 64 single-ended [5]	MP8632
Input voltage range[1]	±10mV to ±5V
ADC gain ranges[1]	±10V, 0 to 10V, 0 to 5V
(strap selectable)	±5V, ±2.5V
Amplifier gain range[1]	1 to 1000 V/V
(resistor programmable)	G = 100kΩ/R_{EXT}
Amplifier gain equation	(Resistor programmable)
	±15V
Input overvoltage protection	100 megohms
Input impedance	
Bias current	
25°C (max)	+300nA
0°C to 70°C	-2nA/°C
Amplifier input offset voltage	±2mV
Amplifier input offset voltage drift	$\pm(5 + \frac{1000}{G})$ µV/°C
ANALOG INPUT TRANSFER CHARACTERISTICS	
Resolution	8 bit binary
Throughput accuracy ±5V range (max)	±0.4% FSR[2]
±10mV range	±0.5% FSR
Temperature coefficient of accuracy	
±5V range (max)	±0.02% FSR/°C
±10mV range	±0.07% FSR/°C
Conversion time ±5V range	44 microseconds
±10mV range	84 microseconds
CMRR (for differential inputs)[3]	66 dB (Gain = 2)
	86 dB (Gain = 100)
ANALOG OUTPUT	
Number of analog outputs	2
Output voltage range[4]	±10V, 0 to 10V, ±5V, 0 to 5V, ±2.5V at 5mA (strap selectable)
Output impedance	1Ω
Output settling time (max)	< 5 microseconds
ANALOG OUTPUT TRANSFER CHARACTERISTICS	
Resolution	8 bits binary
Throughput accuracy (max)	±0.4% FSR
Temperature coefficient of accuracy	
Unipolar	±0.005% FSR/°C
Bipolar	±0.01% FSR/°C
DIGITAL INPUT/OUTPUT	
All signals are compatible with Microcomputer Bus	
Output coding	Bipolar, two's complement; Unipolar, straight binary
An analog input channel is selected by:	ADR0 through ADR5
An analog output channel is selected by:	ADR0
The input/output data bits are read through:	DAT0 through DAT 7
POWER REQUIREMENTS	
MP8608, MP8616, MP8632.	+5VDC ±5% at 1 amp, 25mV ripple
MP8608-NS, MP8616-NS, MP8632-NS	+5VDC ±5% at 500mA, 25mV ripple +15VDC ±3% at 40mA, 5mV ripple -15VDC ±3% at 40mA, 5mV ripple
With analog output MP8608-AO, MP8616-AO, MP8632-AO	+5VDC ±5% at 2 amp, 25mV ripple
MP8608-AO-NS, MP8616-AO-NS, MP8632-AO-NS.	+5VDC ±5% at 500mA, 25mV ripple +15VDC ±3% at 100mA, 5mV ripple -15VDC ±3% at 100mA, 5mV ripple
TEMPERATURE RANGE	0°C to 70°C

OPERATING INSTRUCTIONS

INSTALLATION

These units are shipped from the factory calibrated and ready for immediate use. Installation requires only plugging the card into any empty slot in the computer and wiring the analog connector.

PROGRAMMING

Programming of this analog I/O board is easily accomplished since all channels are treated as memory locations. Any memory reference instruction can be used. A single STA instruction may be used to load the accumulator contents to one of the D/A converters. Likewise a single LDA instruction can be used to read an analog input channel.

Single instructions can also be used to set the inputs of both D/A converters and read two adjacent analog input channels. An SHLD instruction referenced to DAC 1 will load the contents of the L register into DAC 1 and the contents of the H register into DAC 2. An LHLD instruction will read the channel addressed and the next higher channel. The channel addressed will be transferred to the L register and the next higher channel to the H register. Of course, any MOV instruction may also be used if direct addressing is not desired.

The normal operation of this board halts the CPU during the conversion time of the analog input system. This is because the software in this mode is simpler than in any other (i.e., only one instruction required!). If the halt feature is not desirable, it may be disabled. Figure 1 shows the jumpers required. The jumpers shown with an asterisk are plated-through holes and must be drilled out before installation of the other jumpers. A 0.055" (No. 54) drill should be used for this purpose. Caution must be exercised to prevent damage to the board (see Figure 2).

MECHANICAL CHARACTERISTICS

Compatible with Intellec MDS and SBC-604/614 card spacing.

Minimum card spacing: 12.7mm (0.5").

Microcomputer bus connector required: 86 pin PC edge connector with 0.156" contact centers.

50 pin analog edge connector on board.

Mating connector available from Burr-Brown: 2350MC (Viking # 3VH25/1JN5, solder tab); from 3M: 3415-0001 (Scotchflex).

[1] Connected at the factory for ±5V range (ADC range = ±10V, Gain = 2).
[2] FSR is Full Scale Range (i.e., 10V for ±5V range).
[3] DC to 60 Hz with 1 kΩ source unbalance.
[4] Connected at the factory for ±10V range.
[5] Connected at the factory as 32 differential.

Courtesy Burr-Brown Research Corp.

OPERATION WITH HALT. JUMPERS REQUIRED	OPERATION WITHOUT HALT. JUMPERS REQUIRED
W55*, W56*, W57*	W53, W54, W58

FIGURE 1. Halt Selection Jumpers

For operation without halting the CPU, the conversion should be started by using a single channel memory reference instruction (LDA or MOV). Then the CPU should execute a routine which will take longer than the conversion time (44 to 84 microseconds). When the CPU now uses an LDA or MOV referenced to the same memory location, the converted data will be transferred to the CPU.

The voltage data for these boards is represented by an 8-bit two's complement binary number. With a ±5V range, each bit has a value of 39.1mV, with the polarity of the voltage indicated by the sign of the binary number.

Each board is set at the factory for a block of addresses beginning at F700. Any analog data channel requires one memory location. Thus the first analog channel is located at F700 while the second analog channel is located at F701.

ADDRESS MODIFICATION

The base address of a board can be set to any value by properly jumpering its address selector. The most significant 8 bits of the address (ADR/8-F) are jumpered to read F7 by plated through connections on all boards. These addresses can be changed by first drilling out the hole that makes the connection (Figure 2) and then soldering a wire jumper between the bit and logical zero or one. A 0.055″ (No. 54) drill should be used for this purpose. Caution must be exercised to prevent damage to the board and the scattering of metal particles over its surface.

The remaining lower ordered bits have been connected by wire jumpers at the factory. To change the sense of a bit simply reverse the connection of its jumper.

ADDRESS	HIGH	LOW
4	W37	W38
5	W35	W36
6	W33	W34
7	W31	W32
8	W29*	W30
9	W27*	W28
A	W25*	W26
B	W23	W24*
C	W21*	W22
D	W19*	W20
E	W17*	W18
F	W15*	W16

* Plated through jumpers

ADDRESS JUMPERS

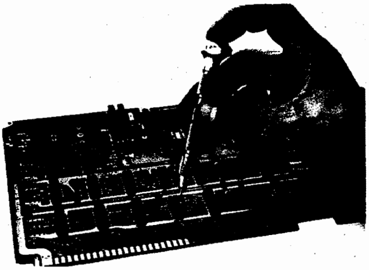


FIGURE 2. Drilling Out Plated Through Holes

ANALOG OUTPUT RANGE SELECTION

When included, each DAC is jumpered at the factory for ±10 volt operation (two's complement coding). However, it is possible to alter these jumpers as shown in Figure 3 for other output voltages and coding. Jumpers indicated by an asterisk are plated through holes on the board and should be removed by drilling as described in the section on address modification. When making a change, first remove those jumpers indicated for the present range and replace them with those jumpers required for the desired range.

Range	DAC 1	DAC 2
±10 ±5 ±2.5 0 to +10 0 to +5	W66*, W67* W65, W67* W65, W67*, W74 W65 W65, W74	W70*, W72* W69, W72* W69, W72*, W73 W69 W69, W73
Coding		
Two's Complement Straight Binary	W61* W62	W63* W64

FIGURE 3. Analog Output Range Selection

Two's complement coding is typically used for bipolar ranges and straight binary for unipolar ranges, but either coding can be used for any range.

BIPOLAR - TWO'S COMPLEMENT			
Digital Input/Output	±10V	±5V	±2.5V
01111111 (7F$_{16}$) 10000000 (80$_{16}$)	+9.922V -10.000V	+4.961V -5.000V	+2.480V -2.500V

UNIPOLAR - STRAIGHT BINARY		
Digital	Input/Output	0 to +10V
11111111 (FF$_{16}$) 00000000 (00$_{16}$)	9.961V 0.000V	4.980V 0.000V

TABLE I. Analog Full Scale Range Values.

ANALOG INPUT RANGE SELECTION

The analog input system can be set for any range between ±5V and ±2.5mV. It is set for ±5V (two's complement

Courtesy Burr-Brown Research Corp.

coding) from the factory. There are two gain determining elements in this system: the A/D converter and the instrumentation amplifier (IA). The A/D converter is set for a ±10V range and the IA for a gain of 2 at the factory. The A/D converter can be set for other ranges simply by changing jumpers as shown in Figure 4. Before adding new jumpers, remove those indicated for the present range. The input voltage presented to the analog multiplexer must not exceed 5.25VDC for proper operation.

RANGE	JUMPERS
±10V	W1*, W2*
±5V	W2*, W4
±2.5V	W2*, W4, W5
0 to +10V	W3, W4
0 to +5V	W3, W4, W5

*Plated through jumpers

FIGURE 4. A/D Converter Range Setting Jumpers

As configured at the factory, this board is jumpered for two's complement operation (see Table I above) with jumper W59* inserted and W60 open. For operation in the straight binary mode (any range) jumper W59* is open and W60 is inserted.

ANALOG INPUT LOW LEVEL OPERATION

Pads for external gain setting resistors (see Figure 7) have been provided so that the instrumentation amplifier can be user set for gains to 1000. The following formula can be used to calculate the value of the resistance: Gain = 100 kΩ/R$_{EXT}$, where R$_{EXT}$ is the resistance between pins 1 and 12 of the IA (R15, 94 in parallel form R$_{EXT}$ in Figure 7). The gain adjustment potentiometer on the board will give an adjustment range of ±1%. Therefore, if an R$_{EXT}$ with an accuracy of ±0.5% is used, the on-board potentiometer will have sufficient range for adjustment. Stable (50ppm) resistors should be used in this application. As shipped from the factory, R15 = 49.9kΩ. The settling time of the amplifier increases as the gain increases. A delay time of 41 microseconds is set at the factory to allow for multiplexer and amplifier settling times. This delay time is sufficient for amplifier gains of up to 50. For gains larger than 50, a longer delay time is required. A delay time of 81 microseconds will be obtained by removing R17. This delay time is sufficient for gains of up to 1000.

For lowest system noise, the ADC range should be set on the ±10V or 0 to 10V ranges with the amplifier providing all the system gain.

A 64/32 channel input board can be converted from single-ended operation to differential operation or vice versa by simply changing a few board jumpers (MP8632 from/to MP8664). Figure 5 indicates those jumpers that must be present for a given mode of operation. To convert from one mode to the other remove those jumpers indicated for the present type of operation and install those necessary for the desired mode of operation.

Required Jumpers for Differential	Required Jumpers for Single-ended
32 Channels	64 Channels
W8	W6
W9	W7
	W10

FIGURE 5. MP8632 Channel Conversion

The differential mode of operation should be used for analog signals in noisy environments. The differential mode is particularly useful for low level signals since they are more prone to noise than high level signals. This board can also operate in a pseudo-differential mode. In this mode, the system has the number of channels of the single-ended mode, but the minus input of the IA is connected to a remote common rather than grounded on the board. This mode of operation is useful if there is a remote ground common to all the input signals. In this way the advantages of single-ended operation (maximum number of channels) and differential operation (better noise rejection) are combined. Jumpers W10 and W52 are installed and W7 is removed for this mode of operation.

ANALOG OUTPUT CHECKOUT

A static check of the two analog outputs is very simple. Load the L register with the output 1 data word and the H register with the output 2 data word. An SHLD instruction can then be used to transfer the data to the DAC's. The addresses of the analog outputs are set at the factory to values of F700$_{16}$ and F701$_{16}$. The ideal values for plus and minus full scale are shown in Figure 6.

DATA WORD	RANGE				
	±2.5V	±5.0V	±10V	0 to +5V	0 to +10
80$_{16}$	-2.500	-5.000	-10.000		
7F$_{16}$	+2.480	+4.961	+9.922		
00$_{16}$				0.000	0.000
FF$_{16}$				+4.980	+9.961

FIGURE 6. DAC Full Scale Values.

Two's complement coding is shown for bipolar ranges; straight binary for unipolar ranges.

If the SBC80/10 monitor is available, the Insert (I) and Substitute (S) commands can also be used to accomplish an output write.

To check the dynamic characteristics of the analog outputs the following program can be used.

```
              ORG     3C40H
              LXI   H, 0F700H ;.Set H & L to DAC 1 Address
              MVI     A, 7FH  ;Initialize Data Word
LOOP:         MOV      M,A    ;Load DAC 1
              INX       H
              MOV      M, A   ;Load DAC 2
              DCX       H
              CMA             ;Compliment Data Word
              JMP     LOOP
              END
```

A 27 kHz square-wave will be present on both DAC outputs.

Courtesy Burr-Brown Research Corp.

ANALOG INPUT CALIBRATION

These systems are set at the factory for a ±5V input range. If the input system range is to be changed, the following program may be used to adjust gain and offset.

```
REF     EQU   80H        :Offset Ref = 80H, Full Scale Ref = 07FH
CO      EQU   01E8H      :Monitor routines
CROUT EQU   01F3H
NMOUT EQU   02C2H
        :
        ORG   3C50H
        :
        LXI   H, 0F700H :Initialize
        LXI   SP, 3FFFH
BEG1:   MVI   E, 10H
BEG2:   LXI   B, 0
CLP:    MOV   A, M       :Read data from board
        SUI   REF        :Increment data count if data = REF
        JNZ   NEQ
        INR   B
NEQ:    INR   C          :Have 100 conversions been made?
        MVI   A, 64H
        SUB   C
        JNZ   CLP
        MOV   A, B       :Yes, Print data count
        CALL  NMOUT
        MVI   C, 20H     :Print à space
        CALL  CO
        DCR   E          :Full line been printed?
        JNZ   BEG2
        CALL  CROUT      :Yes, Print CR & LF
        JMP   'BEG1
        END
```

The program assumes that the system is under the control of the SBC80/10 prototype package monitor (M80P, version 1.0, March 1, 1976). It may be used for both offset and gain calibration. The system offset should be adjusted first, followed by the gain adjustment.

If the address of channel zero on the board has been changed from F700$_{16}$ then the LXI H instruction should reflect that change.

A G3C50 monitor command will begin program execution. After 100 conversions have been made, the value (in hex) of the B register will be printed. This value represents the number of times the data read from the board was equal to "REF" (80 for offset; 7F for gain).

RANGE	OFFSET	GAIN
±5V	-4.980	+4.941
0 to +10	+19.53mV	+9.941
0 to +5	+9.766mV	+4.971

FIGURE 8. Analog Input Calibration Values

Calibration is performed by connecting a voltage source capable of 0.01% accuracy to input channel zero (this could also be a DC voltage source of less absolute accuracy whose output is monitored by a 0.01% DVM).

The offset and gain adjustments are made while applying the voltage shown in Figure 8. For other ranges, the offset voltage adjustment is made at the most negative value of the range less one half least significant bit (LSB). An LSB is equal to the span (full scale range) divided by 256 for 8 bit resolution. The gain adjustment is made at the most positive value of the range less 1 1/2 LSB. Thus for a range of ±50mV, an LSB is 100mV/256 = 391μV. The offset adjustment is made at -50mV + 195μV = -49.80mV and the gain adjustment at +50mV - 586μV = 49.41mV. Before making these adjustments, however, the unit should be allowed to reach thermal equilibrium (about 30 minutes under power).

The offset adjustment is made first by using the appropriate offset calibration voltage. Run the calibration program and adjust the on board offset potentiometer until the B register contains a value between 1E$_{16}$ and 46$_{16}$ (30$_{10}$ and 70$_{10}$).

To perform the gain adjustment change the data labeled "REF" in the calibration program from 80 to 7F, set the input voltage to the correct value as shown in Figure 8 and adjust the on board gain potentiometer in the same manner as described for offset.

If the SBC80 monitor is available, the substitute (S) command can be used to interrogate an input channel.

CONNECTOR PINOUT

P-3 ANALOG CONNECTOR PINOUT Pin No.				P-4 ANALOG CONNECTOR PINOUT Pin No.			
GND	1	2	GND	GND	1	2	GND
GND	3	4	GND		3	4	
-15VDC	5	6	-15VDC		5	6	
+15VDC	7	8	+15VDC		7	8	
AO RET 1	9	10	AO RET 2		9	10	
Analog Out 1	11	12	Analog Out 2		11	12	
GND	13	14	GND		13	14	
GND	15	16	GND		15	16	
Remote Common	17	18	GND		17	18	
IN0	19	20	IN32/RET0	IN16	19	20	IN48/RET16
IN1	21	22	IN33/RET1	IN17	21	22	IN49/RET17
IN2	23	24	IN34/RET2	IN18	23	24	IN50/RET18
IN3	25	26	IN35/RET3	IN19	25	26	IN51/RET19
IN4	27	28	IN36/RET4	IN20	27	28	IN52/RET20
IN5	29	30	IN37/RET5	IN21	29	30	IN53/RET21
IN6	31	32	IN38/RET6	IN22	31	32	IN54/RET22
IN7	33	34	IN39/RET7	IN23	33	34	IN55/RET23
IN8	35	36	IN40/RET8	IN24	35	36	IN56/RET24
IN9	37	38	IN41/RET9	IN25	37	38	IN57/RET25
IN10	39	40	IN42/RET10	IN26	39	40	IN58/RET26
IN11	41	42	IN43/RET11	IN27	41	42	IN59/RET27
IN12	43	44	IN44/RET12	IN28	43	44	IN60/RET28
IN13	45	46	IN45/RET13	IN29	45	46	IN61/RET29
IN14	47	48	IN46/RET14	IN30	47	48	IN62/RET30
IN15	49	50	IN47/RET15	IN31	49	50	IN63/RET31

Courtesy Burr-Brown Research Corp.

8-BIT µP-COMPATIBLE D/A CONVERTER — NE5018

NE5018-F,N

DESCRIPTION

The NE5018 is a complete 8-bit digital to analog converter subsystem on one monolithic chip. The data inputs have input latches,controlled by a latch enable pin. The data and latch enable inputs are ultra-low loading for easy interfacing with all logic systems. The latches appear transparent when the $\overline{LE}$ input is in the low state. When $\overline{LE}$ goes high, the input data present at the moment of transition is latched and retained until $\overline{LE}$ again goes low. This feature allows easy compatibility with most microprocessors.

The chip also comprises a stable voltage reference (5V nominal) and a high slew rate buffer amplifier. The voltage reference may be externally trimmed with a potentiometer for easy adjustment of full scale, while maintaining a low temperature co-efficient.

The output of the buffer amplifier may be offset so as to provide bipolar as well as unipolar operation.

FEATURES

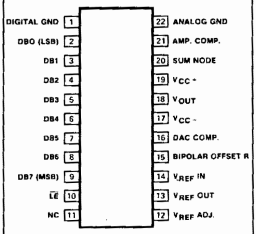
- **8-bit resolution**
- **Input latches**
- **Low-loading data inputs**
- **On-chip voltage reference**
- **Output buffer amplifier**
- **Accurate to ± 1/2 LSB**
- **Monotonic to 8 bits**
- **Amplifier and reference both short-circuit protected**
- **Compatible with 2650, 8080 and many other µP's.**

APPLICATIONS

- **Precision 8-bit D/A converters**
- **A/D converters**
- **Programmable power supplies**
- **Test equipment**
- **Measuring instruments**
- **Analog-digital multiplication**

PIN CONFIGURATION

F,N PACKAGE



SE5018 available in F package only.

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
$V_{CC}+$	Positive supply voltage	18	V
$V_{CC}-$	Negative supply voltage	-18	V
V_{IN}	Logic input voltage	0 to 18	V
$V_{REF}IN$	Voltage at V_{REF} input	12	V
$V_{REF}ADJ$	Voltage at V_{REF} adjust	0 to V_{REF}	V
V_{SUM}	Voltage at sum node	12	V
I_{REFSC}	Short-circuit current to ground at V_{REF} OUT	Continuous	
I_{OUTSC}	Short-circuit current to ground or either supply at V_{OUT}	Continuous	
I_{REF}	Reference input current	5	mA
P_D	Power dissipation*		
	-N package	800	mW
	-F package	1000	mW
T_A	Operating temperature range		
	SE5018	-55 to +125	°C
	NE5018	0 to +70	°C
T_{STG}	Storage temperature range	-65 to +150	°C
T_{SOLD}	Lead soldering temperature (10 seconds)	300	°C

*NOTE

For N package, derate at 120°C/W above 35°C
For F package, derate at 75°C/W above 75°C

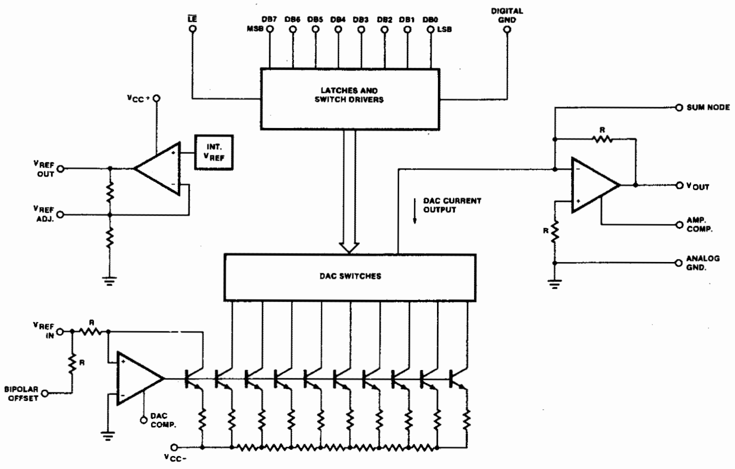
Courtesy Signetics Corp.

DC ELECTRICAL CHARACTERISTICS

$V_{CC}+ = 15V$, $V_{CC}- = -15V$, SE5018. $-55°C \leq T_A \leq 125°C$, NE5018. $0°C \leq T_A \leq 70°C$ unless otherwise specified.

PARAMETER		TEST CONDITIONS	SE5018			NE5018			UNITS
			Min	Typ	Max	Min	Typ	Max	
$V_{CC}+$	Positive supply voltage			15			15		V
$V_{CC}-$	Negative supply voltage			-15			-15		V
	Resolution			8			8		bits
	Relative accuracy				±0.19			±0.19	%
T_S	Settling time	To ± 1/2LSB, 10V step		2			2		µs
PSRR	Power supply Rejection ratio	$V_{CC}+$ +12 to +18V $V_{CC}-$ -12 to -18V		±1			±1		mV/V
$I_{CC}+$	Positive supply current	$V_{CC}+ = 15V$		8			8		mA
$I_{CC}-$	Negative supply current	$V_{CC}- = -15V$		-10			-10		mA
$I_{IN}(0)$	Logic "0" input current	$V_{IN} = 0V$		5			5		µA
$V_{IN}(0)$	Logic "0" input voltage				0.8			0.8	V
$V_{IN}(1)$	Logic "1" input voltage		2.0			2.0			V
T_{PWLE}	Latch enable pulse width			400			400		ns

BLOCK DIAGRAM

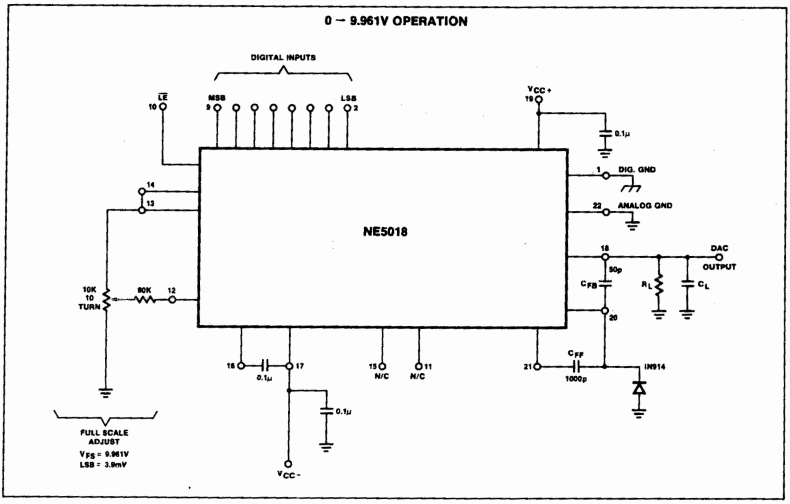


All R valves equal 5KΩ and are thermally matched.

Courtesy Signetics Corp.

EQUIVALENT SCHEMATIC

0 → 9.961V OPERATION



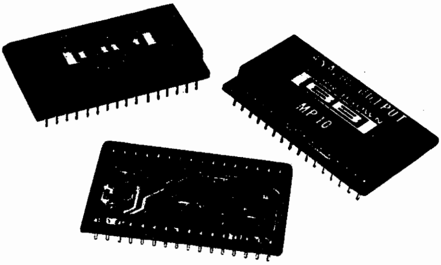
a subsidiary of U.S. Philips Corporation

Signetics Corporation
P.O. Box 9052
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700

Printed in USA Aug. 1977

Courtesy Signetics Corp.

data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers



MP10, MP11

data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers

MICROPROCESSOR ANALOG OUTPUT COMPONENTS

COMPATIBLE WITH:

8080 (Intel)
9080A (AMD)
Z-80 (Zilog)
6800 (Motorola)
8008 (Intel)
F-8 (Fairchild)
SC/MP (National)
650X (MOS Technology)

USE AS ANALOG INPUT AND OUTPUT
EASY TO USE
- **Completely compatible with most Microprocessors**
- **No external logic required**
- **Timing compatible**
- **Memory mapped**

SAVES DEVELOPMENT MONEY AND TIME
COMPLETELY SELF-CONTAINED

signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
micrcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters
microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters

BURR-BROWN

BB

International Airport Industrial Park•P.O. Box 11400•Tucson, Arizona 85734
Tel: 602-294-1431 • Twx: 910-952-1111 • Cable: BBRCORP • Telex: 66-6491

microcomputer peripherals/power supplies
signal conditioning components/amplifiers
data acquisition systems/data converters

Courtesy Burr-Brown Research Corp.

DESCRIPTION

These microprocessor peripherals provide an analog interface compatible with most microprocessors. The MP10 and MP11 are electrically and functionally microprocessor compatible in static or dynamic situations.

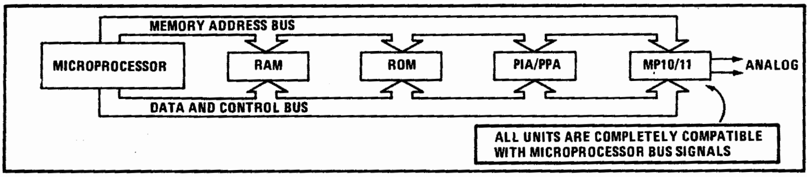
These units are complete analog systems packaged in 32 pin triple wide dual-in-line packages. They contain two 8 bit D/A converters which are internally trimmed for gain and offset so that no external trimming is required. All necessary interface, timing and address decoding logic is also included.

The MP10 is designed to be used with 8080A and 8008 type microprocessors. It can be used with SC/MP if pull-up resistors are added to the address bus, with the F-8 Dynamic or Static memory interface chip if the RAM WRITE signal is a minimum of 430nsec and with the Z-80 if t_w (ϕH) = t_w (ϕL) = 500ns. The MP11 is designed to be used with 6800 and 650X type microprocessors.

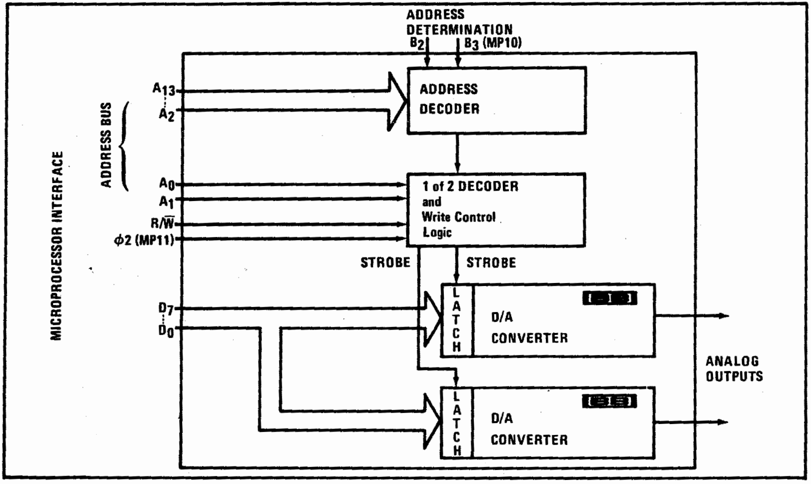
The address lines A_2 through A_{13}, B_2 and B_3 of the MP10 are CMOS compatible so that they can be directly connected to the address bus of an 8080 or 8008. All other input lines require standard TTL voltages. The address lines A_2 through A_{13} and B_2 of the MP11 are LSTTL compatible so they can be directly connected to the address bus of a 6800 or 650X. All other input lines require standard TTL voltages but are high impedance requiring only microamp drive currents.

THEORY OF OPERATION

When programming these peripherals, the user treats them as memory. Because the D/A converter input is an 8 bit word, one 8 bit memory location is required for each channel. Since these units are treated as memory, a single instruction is all that's needed to write to an output channel. For instance, when the MP10 is used with an 8080, a single instruction, SHLD, can be used to output data to both D/A converter channels from the H and L register pair. Likewise, when the MP11 is used with the 6800 or 650X, a single STX instruction can be used to output data to both D/A converter channels from the index register. The MP10 and the MP11 require an initialization as would any programmable peripheral.



MP10, MP11 BLOCK DIAGRAM



Courtesy Burr-Brown Research Corp.

ELECTRICAL SPECIFICATIONS

(Typical at 25°C and rated supplies unless otherwise noted.)

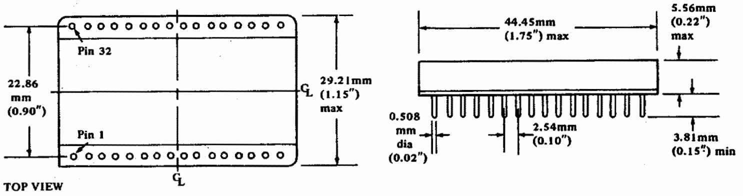
	MP10/MP11
ANALOG OUTPUT	
Number of analog outputs	2
Output voltage range	±10V
Output impedance	1Ω
Output settling time	25 μsec
TRANSFER CHARACTERISTICS	
Resolution	8 bit binary (complementary binary)
One LSB	78.1mV
Throughput accuracy (max)	±0.4% FSR[1, 2]
Throughput accuracy (typical)	±0.25% FSR
Temperature coefficient of accuracy	±0.008% FSR/°C

	MP10/MP11
DIGITAL INPUT/OUTPUT	
All signals compatible with the microprocessor bus	
An analog output channel selected by:	A0
Input data bits read by:	D0 - D7
POWER REQUIREMENTS	
	+5VDC ±5% at 90 mA +15V ±3% at 30 mA -15V ±3% at 30 mA
TEMPERATURE RANGE	
Operating temperature range	0 - 70°C
Storage temperature range	-55°C to +85°C

[1] FSR is Full Scale Range = 20V.

[2] Accuracy components are: Linearity Error ≐ ±0.2% FSR; Gain Error = ±0.1% FSR, Offset Error = ±0.1% FSR.

MECHANICAL SPECIFICATIONS



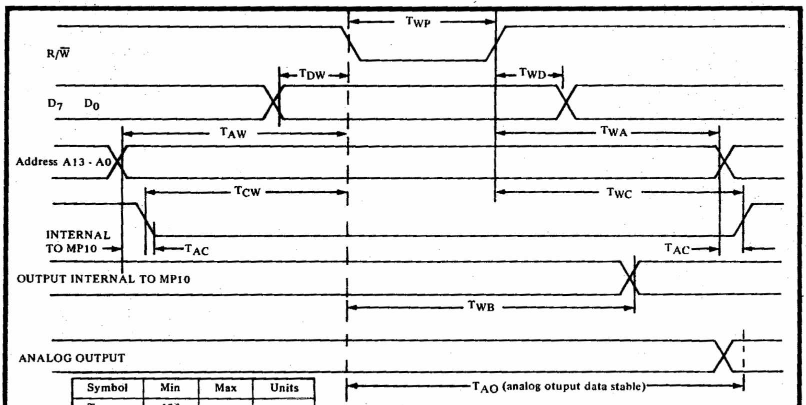
MATERIAL: Alumina
WEIGHT: 14 grams (0.5 oz)
PINS: Pin material and plating composition conform to Method 2003 (solderability) of Mil-Std-883 (except paragraph 3.2).
MATING CONNECTOR: 2302MC

PIN CONNECTIONS

8080 Pin Connections		MP10			8080 Pin Connections
1	1	A10	A11	32	40
2	2	Common	A13	31	38
3	3	D4	A12	30	37
4	4	D5	A 9	29	35
5	5	D6	A 8	28	34
6	6	D7	A 7	27	33
7	7	D3	A 6	26	32
8	8	D2	A 5	25	31
9	9	D1	A 4	24	30
10	10	D0	A 3	23	29
12	11	Reset	A 2	22	27
18	12	R/$\overline{W}$	B 2	21	—
26	13	A1	B 3	20	—
25	14	A0	+5V	19	20
—	15	+15V	Out 1	18	—
—	16	-15V	Out 2	17	—

6800 Pin Connections		MP11			6800 Pin Connections
—	1	Output 1	-15V	32	—
—	2	Output 2	+15V	31	—
8	3	+5V	R/$\overline{W}$	30	34
37	4	Enable	$\overline{\text{Reset}}$	29	40
9	5	A0	D0	28	33
10	6	A1	D1	27	32
11	7	A2	D2	26	31
12	8	A3	D3	25	30
13	9	A4	D4	24	29
14	10	A5	D5	23	28
15	11	A6	D6	22	27
16	12	A7	D7	21	26
17	13	A8	Common	20	21
18	14	A9	B2	19	—
19	15	A10	A13	18	23
20	16	A11	A12	17	22

Courtesy Burr-Brown Research Corp.



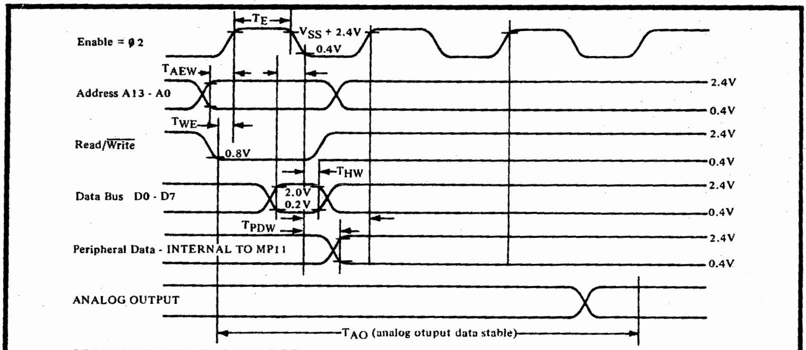
Symbol	Min	Max	Units
T_{WP}	430	—	ns
T_{DW}	50	—	ns
T_{WD}	65	—	ns
T_{AW}	620	—	ns
T_{WA}	35	—	ns
T_{WD}	65	—	ns
T_{WC}	35	—	ns
T_{WB}	—	635	ns
T_{AO}	16	25	μs
T_{AC}	—	600	ns

FOR THE 8080 ITSELF

$T_{AW} = 2tcy - t_{D3} - t_{R42} - 140ns$

$T_{AW} = 670ns$ Min

FIGURE 1. MP10 Timing Diagram.



Symbol	Min	Max	Unit
T_E	0.450	25	μs
T_{AEW}	180	—	ns
T_{DSU}	300	—	ns
T_{WE}	130	—	ns
T_{HW}	10	—	ns
T_{PDW}	—	1.0	μs
T_{AO}	17	25	μs

FIGURE 2. MP11 Timing Diagram.

Courtesy Burr-Brown Research Corp.

PROGRAMMING

These units are easily programmed since all are treated as memory locations. They use any memory reference instruction that can write data from internal registers or the accumulator. A single instruction can be used to write data to one or both channels. When the MP10 is used with an 8080, a single SHLD instruction referenced to the lower of the two addresses will automatically transfer the data in the H register to DAC1 and the data in the L register to DAC2. An STA instruction will transfer the data in the accumulator to either DAC. When the MP11 is used with a 6800, a single STX instruction referenced to the lower of the two addresses will automatically transfer the eight upper bits of the index register to DAC1 and the eight lower bits to DAC2. An STAA instruction will transfer the contents of the accumulator to either DAC. Of course, if direct addressing is not desired, MOV instructions may be used to transfer data from internal registers to a specific DAC memory location. As with any programmable peripheral, the MP10 and MP11 must be initialized.

MP10 INITIALIZATION

The RESET input controls the status of the control register of the MP10. An active high on this line will reset the control register to all "zeros".

The MP10 will require initialization every time RESET is activated. If RESET is connected to ground, the MP10 must be initialized only once before output of the data.

MP10 INITIALIZATION SEQUENCE:

1. Load initialization address
2. Load initialization data

MP10 INITIALIZATION ADDRESS:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
X	X	1	1	1	1	1	1	1	1	1	1	a	a	1	1
												User Defined			

X = don't care, not connected to MP10
1 = True

MP10 INITIALIZATION DATA

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	0	0	0	0	0	0	0	$= 80_{16}$

For 8080 the sequence may look as follows:

LXI H, ADDR;	ADDR = Initialization address Loads H & L registers with initialization address
MVI M, DATA;	DATA = 80 Loads initialization data (80_{16}) to initialization address

The initialization sequence assigns internal registers to function as input registers for the D/A converters. Now data can be written into the MP10. This is accomplished by outputing the correct MP10 address:

A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	1	1	1	1	1	1	1	1	a	a	0	0
				OUTPUT 1						User Defined			
1	1	1	1	1	1	1	1	1	1	a	a	0	1
				OUTPUT 2						User Defined			

The B_2 and B_3 inputs determine the address to which the MP10 will respond. The four memory locations which are possible are outlined below:

B_2	B_3	A_2	A_3
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

At the time that the address appears on the address bus, data will appear on the data bus and a $R/\overline{W}$ pulse will be generated by the microprocessor. 25μs later, the analog voltage will be stable at the selected output. Timing requirements shown in Figure 1 must be satisfied in order for the MP10 to be initialized and operate correctly. These timing requirements are completely compatible with the 8080.

MP11 INITIALIZATION

The $\overline{\text{RESET}}$ input controls the status of the control and peripheral registers of the MP11. The initialization sequence will differ if $\overline{\text{RESET}}$ is connected to a master reset line of a microprocessor or if it is hard-wired to V_{cc}. The MP11 will require initialization every time the $\overline{\text{RESET}}$ line is activated low. If the $\overline{\text{RESET}}$ line is hard wired to V_{cc}, the MP11 must be initialized only once before output of the data is attempted.

MP11 ADDRESS STRUCTURE

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
X	X	1	1	1	1	1	1	1	1	1	1	0	a	Y	Y

A_{15}, A_{14} - don't care, not connected to MP11
A_2 - Address is user selectable
A_0, A_1 - Addresses control the initialization sequence

Courtesy Burr-Brown Research Corp.

Initialization sequence when $\overline{\text{RESET}}$ is hard wired to V_{cc}:

1. Load accumulator with "zeros"
2. Store accumulator at memory locations:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
X	X	1	1	1	1	1	1	1	1	1	1	0	a	1	0	Address of Control register A

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
X	X	1	1	1	1	1	1	1	1	1	1	0	a	1	1	Address of Control register B

3. Load accumulator with "ones"
4. Store accumulator at memory locations:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
X	X	1	1	1	1	1	1	1	1	1	1	0	a	0	0	Address of Peripheral register A
X	X	1	1	1	1	1	1	1	1	1	1	0	a	0	1	Address of Peripheral register B
X	X	1	1	1	1	1	1	1	1	1	1	0	a	1	0	Address of Control register A
X	X	1	1	1	1	1	1	1	1	1	1	0	a	1	1	Address of Control register B

For the 6800 this sequence can be written as follows:

```
LDAA    "zeros"                              Loads Zeros in accumulator
STAA    Address of control register A        Stores zero's in C.R.A.
STAA    Address of control register B        Stores zero's in C.R. B
LDAA    "ones"                               Loads one's in accumulator
STAA    Address of peripheral register A     Stores one's in P.R.A
STAA    Address of peripheral register B     Stores one's in C.R.A
STAA    Address of control register B        Stores one's in C.R.B

LDX     # $0000                              Loads zero's in index register
STX     $ Address control register A         Stores zero's in C.R. A and B
LDX     # $1111                              Loads one's in index register
STX     $ Address peripheral register A      Stores one's in P.R. A and B
STX     $ Address control register A         Stores one's in C.R. A and B
```

Initialization sequence when $\overline{\text{RESET}}$ line is connected to master reset (control registers A and B are always set to zero after master reset and only ones need to be stored in the registers):

```
LDAA    "ones"
STAA    Address Peripheral register A
STAA    Address Peripheral register B
STAA    Address Control register A
STAA    Address Control register B
```

or as:

```
LDXX    # $1111
STX     $ Address Peripheral register A
STX     $ Address Control register A
```

Now data can be written into MP11. This is accomplished by outputing the correct MP11 address:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
X	X	1	1	1	1	1	1	1	1	1	1	0	a	0	0	OUTPUT 1
X	X	1	1	1	1	1	1	1	1	1	1	0	a	0	1	OUTPUT 2

At the time that the address appears on the address bus, data will appear on the data bus, and if the R/W and Enable pulses are correctly timed, 25μs from the true address the analog voltage will be stable at the selected output.

Timing requirements shown in Figure 1 must be satisfied for the MP11 to be initialized and operate correctly. All timing requirements are completely compatible with 6800 microprocessors. User definable address line A_2 used in conjunction with the B_2 input allows the user to place the MP11 in two different memory locations or use two different MP11's in order to expand the analog system to four outputs. When B_2 is wired to logical 1, the MP11 responds to an A_2 address of 0 and when B_2 is wired to a logical 0, the MP11 responds to an A_2 address of 1.

Courtesy Burr-Brown Research Corp.

TEST PROGRAMS

The test circuit and test programs following allow the user to test the operation of the MP10 or MP11. The test may be conducted by setting up the MP10/MP11 as shown in Figure 3. The microprocessor system should have a teletype/CRT terminal interface. The programs will step through several output voltage levels for each DAC output (see Figure 4). Notice how the software is different for the two test programs to illustrate two software approaches.

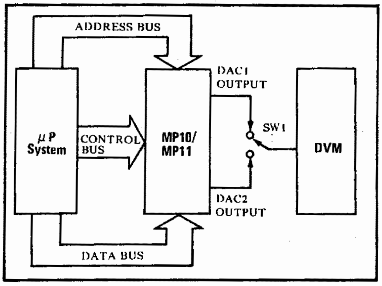


FIGURE 3. Test Circuit for MP10/MP11.

MP10 Test Program

```
                 Initialize MP10
              LXI H ADDR X       Address of the first
                                 byte of data.
LOOP 1     ┌─►MOV A, M       Load ACC with first byte
           │                 of data.
           │  STA ADDR2      Output to MP10 DAC1
           │  INX H          Increment ADDR1
           │  CALL CI    )   Call Input routine
           │  CPI        }   Wait for any character
           │  8D         )   except carriage return
           └──JNZ LOOP1
              LXI ADDR X
LOOP 2     ┌─►MOV A, M
           │  STA ADDR3      Output to MP10 DAC2
           │  INX H          Increment ADDR1
           │  CALL CI    )
           │  CPI        }   Wait for any character
           │  8D         )   except carriage return
           └──JNZ LOOP2
              RET
```

The MP10 test program will output five different voltages from DAC1 and then from DAC2 (see Figure 4). DAC1 will initially output -10V. To step through the other values for DAC1 enter any character other than carriage return (CR). To transfer control to DAC2, enter CR. DAC2 will output -10V. To step through the other values for DAC2 enter any character except CR. To exit the test program, enter CR.

Store the following codes in memory beginning with location ADDR X:

```
ADDR X     ← FF
ADDR X + 1 ← BF
ADDR X + 2 ← 7F
ADDR X + 3 ← 3F
ADDR X + 4 ← 00
```

ADDR 2 is the address of output 1, ADDR 3 is the address of output 2:

MP11 Test Program

```
                                 Initialize MP11
     LDX    # $ FFFF;            Load index register
     STX    ADDR 1;              Store FF in each DAC
     JSR    INP
     LDX    # $ BFBF;            Load index register
     STX    ADDR 1;              Store BF in each DAC
     JSR    INP
     LDX    # $ 7F7F;            Load index register
     STX    ADDR 1;              Store 7F in each DAC
     JSR    INP
     LDX    # $ 3F3F;            Load index register
     STX    ADDR 1;              Store 3F in each DAC
     JSR    INP
     LDX    # $ 0000;            Load index register
     STX    ADDR 1;              Store 00 in each DAC
     JSR    INP
INP  LDAA    ADDR X              Load Status )  Wait for
                                 of ACIA     }  TTY
     Bit A   #01                             )  input
     BEQ    INP
     LDA A   ADDR X + 1          Load Data   )
                                 From ACIA   )
     CMP    A                                )  Jump back
     8D                                      }  to test
     BNE    Back                             )  program or
     JMP    Return                           )  return to
BACK RTS                                     )  main program
```

The MP11 test program will output -10V from both DAC1 and DAC2 then wait for an input from the TTY. Any character except CR will advance both DAC's of the MP11 to the next value as defined in Figure 4. CR terminates test program by jumping to RETURN.

ADDR 1 is the address of output 1, ADDR X is the address of the ACIA.

Step	Ideal Output	Actual Output Limits
1	-10V	-9.922V to -10.078V
2	-5.0V	-4.922 to -5.078
3	0.000V	-0.078 to +0.078
4	+5.0V	+4.972 to +5.078
5	+9.922V	+9.844 to +10.000

FIGURE 4. Output Voltages for Test Programs.

Courtesy Burr-Brown Research Corp.

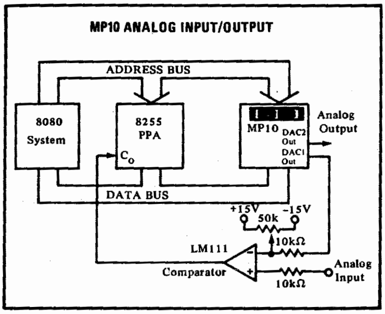
APPLICATIONS

ANALOG INPUT AND ANALOG OUTPUT

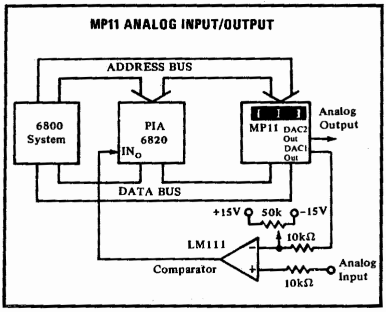
Although the MP10 and MP11 are analog output peripherals, they can be easily adapted to provide both analog inputs and outputs.

With the addition of a few external components, these units can each provide one analog input and one analog output for your system as shown below:

MP10 ANALOG INPUT/OUTPUT



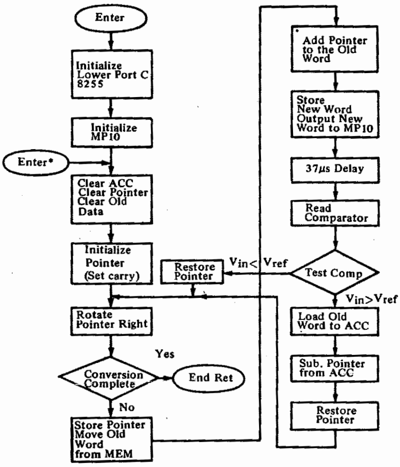
MP11 ANALOG INPUT/OUTPUT



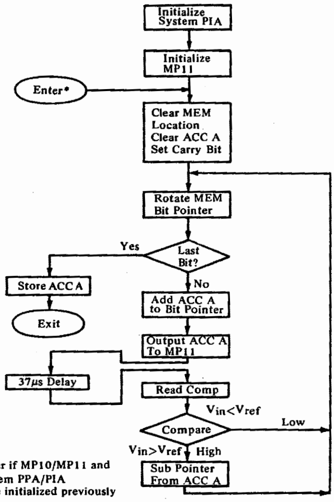
These systems use the microcomputer system to perform the logic of a successive approximation A/D converter, using one channel of the MP10 or MP11 to provide the D/A converter reference function required. In a successive approximation converter, the analog input is compared to known outputs of a D/A converter. First, the microcomputer turns the MSB on, waits for the settling time of the MP10 or MP11, and the switching time of the comparator, then reads the status. If the comparator indicates that the MSB voltage is smaller than the analog input, the MSB input to the MP10/MP11 stays "on" and the next most significant bit is turned on. If the comparator indicates that the MSB value is larger than the analog input, the microcomputer will turn the MSB "off" and turn "on" the next most significant bit. In this way all 8 bits of the D/A converter are tested. When the conversion is complete, the input of the D/A converter will be a digital representation of the analog input. This value will also be stored in the microprocessor's accumulator (complementary binary).

The A/D conversion will require approximately 900 microseconds when performed in this manner. Burr-Brown will shortly have available a detailed application note describing this process including all software required.

FLOWCHART USING 8080 and MP10



FLOWCHART USING 6800 and MP11



* Enter if MP10/MP11 and system PPA/PIA were initialized previously

Courtesy Burr-Brown Research Corp.

National
Semiconductor

SEPTEMBER 1977

ADC0816/ADC0817 Single Chip Data Acquisition System

General Description

The ADC0816, ADC0817 (MM74C948) data acquisition components are monolithic CMOS devices with an 8-bit analog-to-digital converter, a 16-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 16-channel multiplexer can directly access any one of 16 single-ended analog signals and provides the logic for additional channel expansion. Signal conditioning of any analog input signal is eased by direct access to the input of the 8-bit A/D converter.

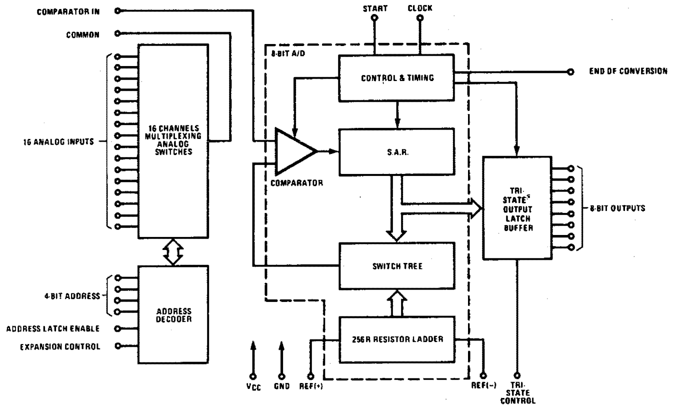
The device eliminates the need for external zero and full-scale adjustments and featurзs an absolute accuracy ≤ 1 LSB including quantitizing error. Easy interfacing to microprocessors is provided by the latched and decoded address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0816, ADC0817 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0816, ADC0817 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications such as process control, industrial control, and machine control.

Features

- Total unadjusted error < ±1/2 LSB
- Linearity error < ±1/2 LSB
- No missing codes
- Guaranteed monotonicity
- No offset adjust required
- No scale adjust required
- Conversion time of 100 μs
- Easy microprocessor interface
- Latched TRI-STATE output
- Latched address input
- Ratiometric conversion
- Single 5V supply
- Low power consumption—15 mW

Block Diagram



IM-B20M97/Printed in U.S.A.

Courtesy National Semiconductor Corp.

Absolute Maximum Ratings (Notes 1 and 2)

Voltage at Any Pin Except Control Inputs	−0.3V to V_{CC} + 0.3V
Voltage at Control Inputs (Start, TRI-STATE, Clock, ALE, ADD A, ADD B, ADD C, ADD D, Expansion Control)	−0.3V to + 15V
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Package Dissipation (at 25°C)	500 mW
Operating V_{CC} Range	4.5V to 6V
Absolute Maximum V_{CC}	6.5V
Lead Temperature (Soldering, 10 seconds)	300°C

DC Electrical Characteristics

4.75V ≤ V_{CC} ≤ 5.25V, −40°C ≤ T_A ≤ +85°C unless otherwise noted, (Note 2)

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
$V_{IN(1)}$	Logical "1" Input Voltage	V_{CC} = 5V	V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage	V_{CC} = 5V			1.5	V
$V_{OUT(1)}$	Logical "1" Output Voltage	I_O = −360 µA	V_{CC}−0.4			V
$V_{OUT(0)}$	Logical "0" Output Voltage	I_O = 1.6 mA			0.45	V
$V_{OUT(0)}$	Logical "0" Output Voltage EOC	I_O = 1.2 mA			0.45	V
$I_{IN(1)}$	Logical "1" Input Current (The Control Inputs)	V_{IN} = 15V			1.0	µA
$I_{IN(0)}$	Logical "0" Input Current (The Control Inputs)	V_{IN} = 0	−1.0			µA
I_{CC}	Supply Current	Clock Frequency = 500 kHz		300	1000	µA
I_{OUT}	TRI-STATE Output Current	V_O = 5V			3	µA
		V_O = 0	−3			µA

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All voltages measured with respect to GND unless otherwise specified.

Note 3: Non-linearity error is the maximum deviation from a straight line through the end points of the A/D transfer characteristic, *(Figure 2)*.

Note 4: Zero error is the difference between the output of an ideal and the actual A/D for zero input voltage, *(Figure 2)*.

Note 5: Full-scale error is the difference between the output of an ideal and the actual A/D for full-scale input voltage, *(Figure 2)*.

Note 6: Total unadjusted error is the maximum sum of non-linearity, zero and full-scale errors, *(Figure 3)*.

Note 7: Quantization error is the ±1/2 LSB uncertainty caused by the converter's finite resolution, *(Figure 3)*.

Note 8: Absolute Accuracy describes the difference between the actual input voltage and the full-scale weighted equivalent of the binary output code; included are quantizing and all other errors. Although rarely provided on data sheets, it is the best indication of a converter's true performance, *(Figure 3)*.

Note 9: Supply rejection relates to the ability of an ADC to maintain accuracy as the supply voltage varies. The supply and $V_{REF(+)}$ are varied together and the change in accuracy is measured-with respect to full-scale.

Note 10: Comparator input current is a bias current into or out of the chopper stabilized comparator. The bias current varies directly with clock frequency and has little temperature dependence, *(Figure 5)*.

Courtesy National Semiconductor Corp.

DC Electrical Characteristics (Continued)

ANALOG MULTIPLEXER

ADC0816, ADC0817 −40°C ≤ T_A ≤ +85°C unless otherwise noted

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
R_{ON}	Analog Multiplexer ON Resistance	(Any Selected Channel) T_A = 25°C, R_L = 10k		1.5	3	kΩ
					6	kΩ
ΔR_{ON}	Δ ON Resistance Between Any 2 Channels	(Any Selected Channel) R_L = 10k		75		Ω
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC} = 5V, V_{IN} = 5V, T_A = 25°C		10	200	nA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC} = 5V, V_{IN} = 0, T_A = 25°C	−200	−10		nA

CONVERTER SECTION V_{CC} = $V_{REF(+)}$ = 5V, $V_{REF(-)}$ = GND, V_{IN} = $V_{COMPARATOR\ IN}$, f_c = 640 kHz

ADC0816CCN −40°C ≤ T_A ≤ +85°C unless otherwise noted

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Resolution		8			Bits
Non-Linearity	(Note 3)		±1/4	±1/2	LSB
Zero Error	(Note 4)		±1/4	±1/2	LSB
Full-Scale Error	(Note 5)		±1/4	±1/2	LSB
Total Unadjusted Error	T_A = 25°C		±1/4	±1/2	LSB
	(Note 6)		±1/4	±3/4	LSB
Quantization Error	(Note 7)			±1/2	LSB
Absolute Accuracy	T_A = 25°C		±3/4	±1	LSB
	(Note 8)		±3/4	±1 1/4	LSB

ADC0817CCN T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Resolution		8			Bits
Non-Linearity	(Note 3)		±1/2	±1	LSB
Zero Error	(Note 4)		±1/4	±1/2	LSB
Full-Scale Error	(Note 5)		±1/4	±1/2	LSB
Total Unadjusted Error	(Note 6)		±1/2	±1	LSB
Quantization Error	(Note 7)			±1/2	LSB
Absolute Accuracy	(Note 8)		±1	±1 1/2	LSB

ADC0816CCN −40°C ≤ T_A ≤ +85°C
ADC0817CCN T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Power Supply Rejection	4.75V ≤ V_{CC} = $V_{REF(+)}$ ≤ 5.25V, (Note 9)		0.05	0.15	%/V
Comparator Input Current	f_c = 640 kHz, (Note 10)	−2	±0.5	2	μA
Ladder Resistance	From Ref(+) to Ref(−)	1	4.5		kΩ

Courtesy National Semiconductor Corp

DC Electrical Characteristics (Continued)

DESIGN GUIDELINES

ADC0816CCN, ADC0817CCN

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
V_{LAD}	Voltage Across Ladder	From Ref(+) to Ref(−)	0.512	5.12	5.25	V
$V_{REF(+)}$	Voltage, Top of Ladder	Measured at Ref(+)		V_{CC}	$V_{CC}+0.1$	V
$\frac{V_{REF(+)}+V_{REF(-)}}{2}$	Voltage, Center of Ladder	Measured at $R_{LADDER}/2$	$\frac{V_{CC}}{2}-0.1$	$\frac{V_{CC}}{2}$	$\frac{V_{CC}}{2}+0.1$	V
$V_{REF(-)}$	Voltage, Bottom of Ladder	Measured at Ref(−)	−0.1	0		V

AC Electrical Characteristics

ADC0816CCN, ADC0817CCN $T_A = 25°C$, $V_{CC} = V_{REF(+)} = 5V$, $V_{REF(-)} = GND$

PARAMETER		CONDITIONS	MIN	TYP	MAX	UNITS
t_{WS}	Start Pulse Width	*(Figure 5)*	200	100		ns
t_{WALE}	Minimum ALE Pulse Width	*(Figure 5)*	200	100		ns
t_s	Address Set-Up Time	*(Figure 5)*	50	25		ns
t_H	Address Hold Time	*(Figure 5)*	50	25		ns
t_D	Analog MUX Delay Time From ALE	Common Tied to Comparator In, $R_S + R_{ON} \leq 5\ k\Omega$, $C_L = 10\ pF$		1	2.5	µs
t_{H1}, t_{H0}	TRI-STATE Control to Q Logic State	$C_L = 50\ pF$		125	250	ns
t_{1H}, t_{0H}	TRI-STATE Control to Hi-Z	$C_L = 10\ pF$, $R_L = 10k$		125	250	ns
t_c	Conversion Time	$f_c = 640\ kHz$, *(Figure 5)*	90	100	114	µs
f_c	Clock Frequency		10	640	1200	kHz
t_{EOC}	EOC Delay Time	*(Figure 5)*	1		8	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs At MUX Inputs		10 5	15 7.5	pF pF
C_{OUT}	TRI-STATE Output Capacitance	At TRI-STATE Outputs, (Note 11)		5	7.5	pF

Note 11: Capacitance guaranteed by periodic testing.

Courtesy National Semiconductor Corp.

Timing Diagram

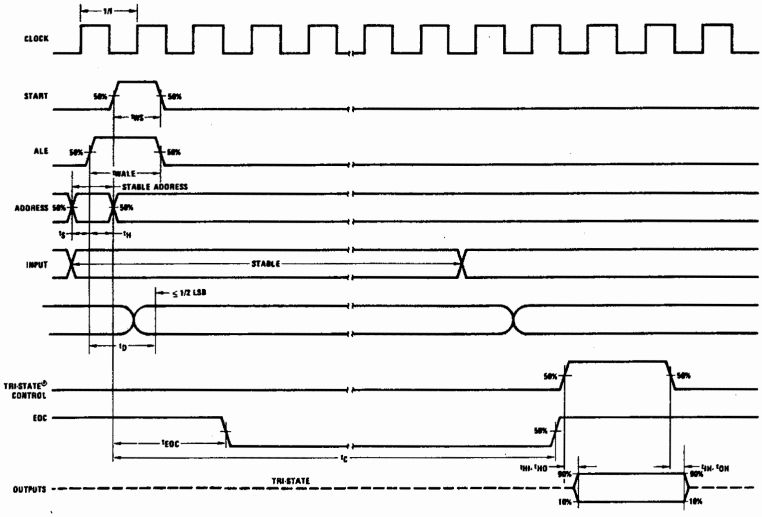


FIGURE 5

Typical Performance Characteristics

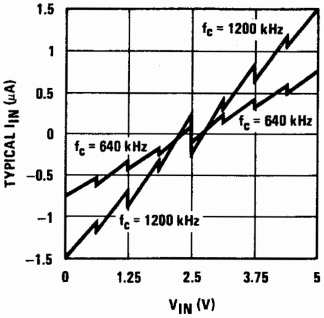


FIGURE 6. Comparator I_{IN} vs V_{IN}
($V_{CC} = V_{REF} = 5V$)

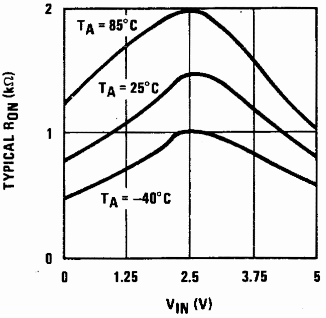


FIGURE 7. Multiplexer R_{ON} vs V_{IN}
($V_{CC} = V_{REF} = 5V$)

Courtesy National Semiconductor Cor

Functional Description

Multiplexer: The device contains a 16-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. Table I shows the input states for the address line and the expansion control line to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

Additional single-ended analog signals can be multiplexed to the A/D converter by disabling all the multiplexer inputs. The additional external signals are connected to the comparator input and the device ground. Additional signal conditioning (i.e., prescaling, sample and hold, instrumentation amplification, etc.) may also be added between the analog input signal and the comparator input.

TABLE I

SELECTED ANALOG CHANNEL	ADDRESS LINE				EXPANSION CONTROL
	D	C	B	A	
IN0	L	L	L	L	H
IN1	L	L	L	H	H
IN2	L	L	H	L	H
IN3	L	L	H	H	H
IN4	L	H	L	L	H
IN5	L	H	L	H	H
IN6	L	H	H	L	H
IN7	L	H	H	H	H
IN8	H	L	L	L	H
IN9	H	L	L	H	H
IN10	H	L	H	L	H
IN11	H	L	H	H	H
IN12	H	H	L	L	H
IN13	H	H	L	H	H
IN14	H	H	H	L	H
IN15	H	H	H	H	H
All Channels OFF	X	X	X	X	L

X = don't care

CONVERTER CHARACTERISTICS

The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach *(Figure 1)* was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +1/2 LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0816, ADC0817, the approximation technique is extended to 8 bits using the 256R network.

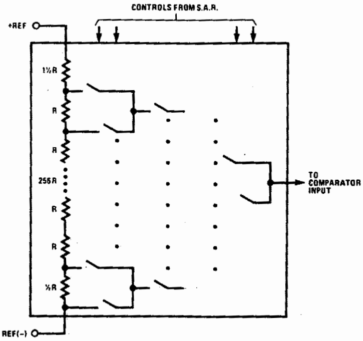


FIGURE 1. Resistor Ladder and Switch Tree

Courtesy National Semiconductor Corp.

Functional Description (Continued)

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 1 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0816 as measured using the procedures outlined in AN-179. The characteristic is generated with the analog input signal applied to the comparator input.

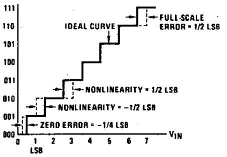


FIGURE 2. 3-Bit A/D Transfer Curve

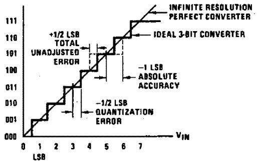


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

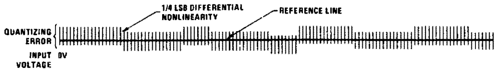
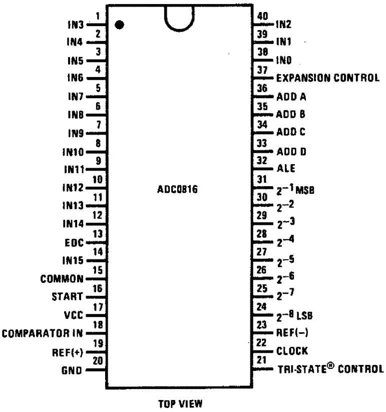


FIGURE 4. Typical Error Curve

Connection Diagram

Dual-In-Line Package



Courtesy National Semiconductor Corp.

Applications Information

OPERATION

Ratiometric Conversion

The ADC0816, ADC0817 is designed as a complete Data Acquisition System (DAS) for ratiometric conversion systems. In ratiometric systems, the physical variable being measured is expressed as a percentage of full-scale which is not necessarily related to an absolute standard. The voltage input to the ADC0816 is expressed by the equation

$$\frac{V_{IN}}{V_{fs} - V_Z} = \frac{D_X}{D_{MAX} - D_{MIN}} \qquad (1)$$

V_{IN} = Input voltage into the ADC0816
V_{fs} = Full-scale voltage
V_Z = Zero voltage
D_X = Data point being measured
D_{MAX} = Maximum data limit
D_{MIN} = Minimum data limit

A good example of a ratiometric transducer is a potentiometer used as a position sensor. The position of the wiper is directly proportional to the output voltage which is a ratio of the full-scale voltage across it. Since the data is represented as a proportion of full-scale, reference requirements are greatly reduced, eliminating a large source of error and cost for many applications. A major advantage of the ADC0816, ADC0817 is that the input voltage range is equal to the supply range so the transducers can be connected directly across the supply and their outputs connected directly into the multiplexer inputs, *(Figure 8)*.

Ratiometric transducers such as potentiometers, strain gauges, thermistor bridges, pressure transducers, etc., are suitable for measuring proportional relationships; however, many types of measurements must be referred to an absolute standard such as voltage or current. This means a system reference must be used which relates the full-scale voltage to the standard volt. For example, if $V_{CC} = V_{REF} = 5.12V$, then the full-scale range is divided into 256 standard steps. The smallest standard step is 1 LSB which is then 20 mV.

Resistor Ladder Limitations

The voltages from the resistor ladder are compared to the selected input 8 times in a conversion. These voltages are coupled to the comparator via an analog switch tree which is referenced to the supply. The voltages at the top, center and bottom of the ladder must be controlled to maintain proper operation.

The top of the ladder, Ref(+), should not be more positive than the supply, and the bottom of the ladder Ref(−) should not be more negative than ground. The center of the ladder voltage must also be near the center of the supply because the analog switch tree changes from N-channel switches to P-channel switches. These limitations are automatically satisfied in ratiometric systems and can be easily met in ground referenced systems.

Figure 9 shows a ground referenced system with a separate supply and reference. In this system, the supply must be trimmed to match the reference voltage. For instance, if a 5.12V reference is used, the supply should be adjusted to the same voltage within 0.1V.

The ADC0816 needs less than a milliamp of supply current so developing the supply from the reference is readily accomplished. In *Figure 10* a ground referenced system is shown which generates the supply from the reference. The buffer shown can be an op amp of sufficient drive to supply the milliamp of supply current and the desired bus drive, or if a capacitive bus is driven by the outputs a large capacitor will supply the transient supply current as seen in *Figure 11*. The LM301 is overcompensated to insure stability when loaded by the 10 µF output capacitor.

The top and bottom ladder voltages cannot exceed V_{CC} and ground, respectively, but they can be symmetrically less than V_{CC} and greater than ground. The center of the ladder voltage should always be near the center of the supply. The sensitivity of the converter can be increased, (i.e., size of the LSB steps decreased) by using a symmetrical reference system. In *Figure 12*, a 2.5V reference is symmetrically centered about $V_{CC}/2$ since the same current flows in identical resistors. This system with a 2.5V reference allows the LSB bit to be half the size of a 5V reference system.

Converter Equations

The transition between adjacent codes N and N + 1 is given by:

$$V_{IN} = V_{REF(+)}\left[\frac{N}{256} + \frac{1}{512}\right] \pm V_{TUE} \qquad (2)$$

The center of an output code N is given by:

$$V_{IN} = V_{REF(+)}\left[\frac{N}{256}\right] \pm V_{TUE} \qquad (3)$$

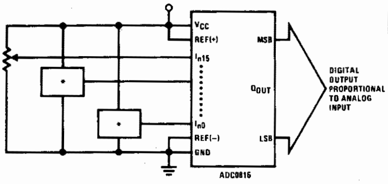
The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN}}{V_{REF(+)}} \times 256 \pm \text{Absolute Accuracy} \qquad (4)$$

where: V_{IN} = Voltage at comparator input
$V_{REF(+)}$ = Voltage at Ref(+)
$V_{REF(-)}$ = GND
V_{TUE} = Total unadjusted error voltage(typically $V_{REF(+)}/512$)

Courtesy National Semiconductor Corp.

Applications Information (Continued)

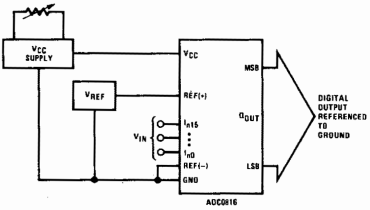


$$Q_{OUT} = \frac{V_{IN}}{V_{REF}} = \frac{V_{IN}}{V_{CC}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

*Ratiometric transducers

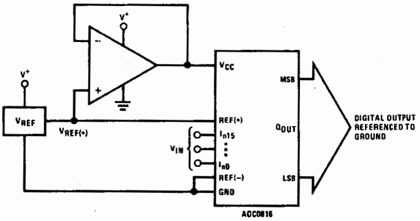
FIGURE 8. Ratiometric Conversion System



$$Q_{OUT} = \frac{V_{IN}}{V_{REF}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

FIGURE 9. Ground Referenced Conversion System Using Trimmed Supply



$$Q_{OUT} = \frac{V_{IN}}{V_{REF}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

FIGURE 10. Ground Referenced Conversion System with Reference Generating V_{CC} Supply

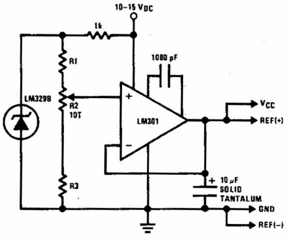
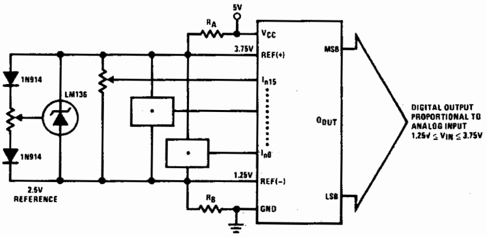


FIGURE 11. Typical Reference and Supply Circuit



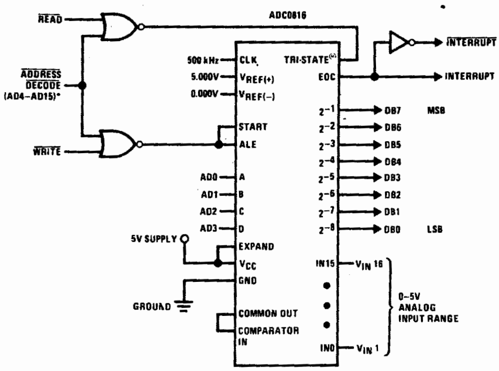
$R_A = R_B$

*Ratiometric transducers

FIGURE 12. Symmetrically Centered Refeience

Courtesy National Semiconductor Corp.

Typical Application

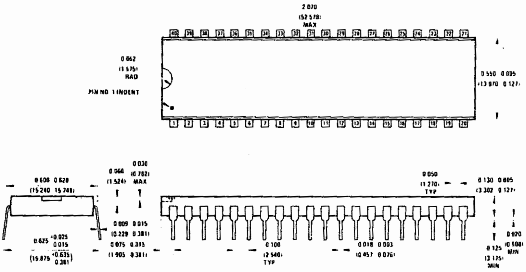


*Address latches needed for 8085 and SC/MP interfacing the ADC0816 to a microprocessor

MICROPROCESSOR INTERFACE TABLE

PROCESSOR	READ	WRITE	INTERRUPT (COMMENT)
8080	MEMR	MEMW	INTR (Thru RST Circuit)
8085	RD	WR	INTR (Thru RST Circuit)
Z–80	RD	WR	INT (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA · φ2 · R/W	VMA · φ2 · R/W	IRQA or IRQB (Thru PIA)

Physical Dimensions inches (millimeters)



Molded Dual-In-Line Package (N)
Order Number ADC0816CCN
or ADC0817CCN
NS Package Number N40A

Manufactured under one or more of the following U.S. patents: 3083262, 3189758, 3231797, 3303356, 3317671, 3323071, 3381071, 3408542, 3421025, 3426423, 3440498, 3518750, 3519897, 3557431, 3560765, 3566218, 3571630, 3575609, 3579059, 3593069, 3597640, 3607469, 3617859, 3631312, 3633052, 3638131, 3648071, 3651565, 3693248.

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 737-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 05-27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied; and National reserves the right, at any time without notice, to change said circuitry.

Courtesy National Semiconductor Corp.

Index

THE BLACKSBURG GROUP

According to Business Week magazine (Technology July 6, 1976) large scale integrated circuits or LSI "chips" are creating a second industrial revolution that will quickly involve us all. The speed of the developments in this area is breathtaking and it becomes more and more difficult to keep up with the rapid advances that are being made. It is also becoming difficult for newcomers to "get on board."

It has been our objective, as The Blacksburg Group, to develop timely and effective educational materials and aids that will permit students, engineers, scientists and others to quickly learn how to apply new technologies to their particular needs. We are doing this through a number of means, textbooks, short courses, monthly computer interfacing columns and through the development of educational "hardware" or training aids.

Our group members make their home in Blacksburg, found in the Appalachian Mountains of southwestern Virginia. While we didn't actively start our group collaboration until the Spring of 1974, members of our group have been involved in digital electronics, minicomputers and microcomputers for some time.

Some of our past experiences and on-going efforts include the following:

—The design and development of the Mark-8 computer, featured in Radio-Electronics magazine in July 1974. This is generally recognized as the first widely available hobby computer. It was based upon the 8008 processor chip. Since then we have designed the Micro-Designer (MD-1) and the Mini-Micro Designer (MMD-1). This last computer was also featured in Radio-Electronics as the Dyna-micro.

—The Blacksburg Continuing Education Series™covers subjects ranging from basic electronics through microcomputers, operational amplifiers, and active filters. Text experiments and examples have been provided in each book. We are strong believers in the use of detailed experiments and examples to reinforce basic concepts. This series originally started as our Bugbook® series and many titles are now being translated into Chinese, Japanese, German and Italian.

—We have pioneered the use of small, self-contained computers in hands-on courses aimed at microcomputer users. Our expanding line of solderless breadboarding modules or OUTBOARDS® make the design and testing of circuits much easier than was possible in the past. Our educational hardware is marketed by E & L Instruments, Inc., Derby, CT 06418, USA.

—Our short course programs have been presented throughout the world. Two series of programs are offered, one through Tychon Incorporated and the other through the Virginia Polytechnic Institute and State University Extension Division. Each provides hands-on experiences with digital electronics and microcomputer hareware and software. Continuing Education Units (CEUs) are provided. Courses are presented to open groups, companies, schools and other groups We are strong believers in hands-on experience in these courses, so much time is spent in laboratory sessions.

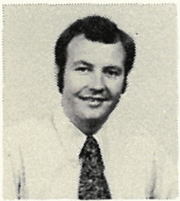
—Our monthly column, "Microcomputers Interfacing" appears in four domestic electronic publications as well as in four overseas periodicals, reaching about three-quarters of a million readers. The columns are currently being translated into German and Italian.

Besides our educational activities, our members have also been involved in microcomputer software development and scientific instrument automation.

Mr. David Larsen and Dr. Peter Rony are on the faculty of the Departments of Chemistry and Chemical Engineering at Virginia Polytechnic Institute and State University. Mr. Jonathan Titus and Dr. Christopher Titus are with Tychon, Inc., all of Blacksburg, Virginia.

MICROCOMPUTER-ANALOG CONVERTER SOFTWARE & HARDWARE INTERFACING

This book has been written to introduce the reader to the concepts and techniques of interfacing digital computers to analog devices. The designs and programs presented in this book are equally applicable to all of the 8080-type microcomputers, including the 8080A, 8085, Z-80, etc.



Jonathan A. Titus is the president of Tychon, Incorporated in Blacksburg, Virginia. Most of his current work involves technical writing and the application of microcomputers for data acquisition and control. He has written and coauthored a number of articles on computers for both professional and popular applications.

Jon's first microcomputer experience was with the 8008 and his Mark-8 computer was featured as the first widely available hobby computer. His interests now center around the 8080 and 16-bit microcomputers.

He has coinstructed courses with the American Chemical Society and now works with the Tychon hardware and software course program.

Dr. Christopher A. Titus is a microcomputer application engineer with Tychon, Incorporated in Blacksburg, Virginia. He received his Ph.D. from Virginia Polytechnic Institute while working on microcomputer automated chemical instruments. He has co-authored a number of instrumentation articles and has had papers presented at major, national engineering and science conferences.

Chris has programmed with the Intel 8008, Intel 8080, and also the MOS Technology 6502 microcomputer. He has written editor, assembler, disassembler, and debug software as well as complete operating systems for microcomputers. He is also a proficient PDP-8 programmer and digital designer.

Dr. Titus has been instructing in computer and related areas since 1973. His current interests include systems software, data-acquisition systems, and evaluation of hardware/software tradeoffs.

David G. Larsen is an instructor in the Department of Chemistry at Virginia Polytechnic Institute & State University, where he teaches undergraduate and graduate courses in analog and digital electronics. He is co-author of other books in the Blacksburg Continuing Education Series™ and the monthly columns on microcomputer interfacing. He is co-instructor, along with Dr. Rony, of a series of one- to five-day workshops on the digital and microcomputer revolution, taught under the auspices of the Extension Division of the University, that attract professionals from all parts of the world.

Dr. Peter R. Rony is professor in the Department of Chemical Engineering at Virginia Polytechnic Institute & State University. Digital electronics and microcomputers will play an increasingly important role in process control, a subject of considerable interest to chemical engineers. He is co-author of many other books in the Blacksburg Continuing Education Series™ and monthly columns on microcomputer interfacing that appear in *American Laboratory, Computer Design, Ham Radio Magazine,* the German magazine *Electroniker,* and other U.S. and foreign magazines.

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

$9.50/21540