A data acquisition system design based on a single-board computer

by

Tou-Chung Hou

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE

Interdepartmental Program: Biomedical Engineering
Major: Biomedical Engineering

Approved:

Iowa State University
Ames, Iowa

1986

TABLE OF CONTENTS

INTRODUCTION

Over the past four decades the computer industry has experienced four generations of development, physically marked by rapid changes of building blocks from relays and vacuum tubes (1940-1950s), to discrete diodes and transistors (1950-1960s), to small- and medium-scale integrated circuits (SSI/MSI) (1960-1970s), and to large- and very-large-scale integrated circuits (LSI/VLSI) (1970s and beyond). Increases in device speed and reliability and reduction in hardware cost and physical size have greatly enhanced computer performance.

Computers are increasingly in demand in the area of structural analysis, weather forecasting, petroleum exploration, fusion energy research, medical diagnosis, aerodynamics simulation, artificial intelligence, expert system, industrial automation, remote sensing, military defense, genetic engineering, socioeconomics, and other scientific and engineering applictions. They are used almost everywhere. They are undeniably becoming more and more important in human lives.

This project focuses on a computer-based system for medical data acquisition. The project goals encouraged the use of the high-level language BASIC for programming. Unless the actual application requires a higher execution speed, the user need not consider using a low-level language such as assembly language or machine language.

LITERATURE REVIEW

New industrial, commercial, and consumer products containing microcomputers are announced almost daily. Many manufacturers are applying microcomputer technology to their products to increase versatility, simplify operation, reduce cost or improve reliability. A similar trend is occurring in the medical device industry. Laboratory automation, data acquisition from analytical instruments, and patient monitoring are but a few of areas benefiting from the microcomputer.

In medical usage, an Intel 8748 microcomputer was used to build a computer-based system to help a handicapped (quadriplegic) computer programmer (Ramey et al., 1982). A Motorola 6800 microprocessor was used to construct a computer-based system for the disabled (Bolton and Taylor, 1981). This system used breath-powered signals to help a tetraplegic select characters for a typewriter. It could also link to another computer directly or to a modem for transmission of data via a telephone line. A commercial (Datel) data acquisition system was modified for electrophysiological application (Demjanenko and Sachs, 1982). The Datel ST series lacked a programmable sampling clock and a multichannel sample and hold. Both of these features are often vital to biophysical experiments where it is necessary to sample at precise intervals and/or change the sampling rate during an experiment. This system added a programmable interval timer and a sample and hold unit to the Datel ST unit, making it more adaptable for biophysical experiments than the original device. A 6802 central processing unit (CPU) was used to construct a microprocessor-electrocardiograph (Dotsinsky et al., 1985). The

procedure time was greatly reduced, taking only a few seconds for data acquisition. There was no need for baseline centering, lead switching, or amplifier setting. The requirements for cable and amplifier shielding and electrode/skin contact were greatly reduced too. The use of a fast microdot printer/plotter eliminated the problem of mechanical fragility of the recording system and greatly reduced the need for servicing. A single-board computer (SBC 80/24) was used to build a computer-based system for the measurement and analysis of an expiratory flow-control curve (El-Dhaher et al., 1983).

A Z8 BASIC system controller was used to build a computer-based communication aids to help the vocally handicapped (Rolander, 1984). A small keyboard was used as an input device to the Z8 BASIC controller. The processed signal was then sent to two speech synthesizers to produce the desired words or phrases. The processed signal could also be sent to a liquid crystal display to show the words or phrases. This device was easy to operate.

Comparing the above examples, it can be seen that there are different purposes, usages, and design schemes. But some common design criteria are evident:

1. The cost of the system is low.
2. The reliability of the system is high.
3. The performance of the system must be high.
4. The system must be easy to maintain.
5. The system should be easy to expand.

6. The system is light in weight and small in size.

7. The system needs less power for operation.

Many times, the computer-based system can't meet all the criteria. Then we must consider which one is the most important, which one is the second most important,...,etc. Later, we use our priorities in designing the computer-based system. For example, a computer-based automatic control system in an airplane must have very high reliability, even though the price is high. The high price compensates the high reliability. Sometimes designing a computer-based system for a special purpose becomes too complex. This project describes a data acquisition system design using a single-board computer. Some general design ideas for a computer-based system will be given.

DISCUSSION OF SINGLE-CHIP BASIC MICROCOMPUTERS

There are three single-chip BASIC microcomputers (MCUs) - Zilog Z8671, National Semiconductor (N.S.C.) INS8073, and Intel 8052AH-BASIC. The addresses of 13 manufacturers which use these single-chip BASIC MCUs in their single-board computers are presented in the Appendix.

All three of the MCUs have these features: auto-startup with user's ROM; on-board universal asychronous receiver/transmitter (UART); can call machine language subroutines to increase execution speed; +5 V supply voltage; input/output (I/O) pins are transistor-transistor-logic (TTL) compatible; direct access to register and memory locations; can use both hexadecimal and decimal numbers; line editor (interactive debug program); can support memory-mapped I/O; on-chip oscillator; and a 40-pin package. A comparison of the three MCUs is given in Table 1.

If the different features of the three MCUs are compared, you will note that the N.S.C. INS8073 appears to be the lowest in performance. National Semiconductor Corporation has ceased production of the INS8073. The Intel 8052AH-BASIC MCU is superior to the Zilog Z8671. There are four advantages of the Z8671 MCU and were the reasons for its choice.

1. Low cost - about $20; the price of the Intel 8052AH-BASIC is approximately $80.
2. It is an established product and is described by a lot of documentation.
3. It is an easy device to use.

TABLE 1. Features of the three single-chip BASIC MCUs

--- FEATURE ---	ZILOG Z8671 MCU	N.S.C. INS8073	INTEL 8052AH-BASIC
BASIC ROM size	2K	1.75K	8K
BASIC commands & statements	16	25	55
Other BASIC functions	0	7	32
Logic functions	1	3	4
Arith./relational operations	4/6	4/6	5/6
BASIC variables	26	26	>26
Error codes	<25	12	15
Baud rates	8	4	auto-srch
Prog. counter/timer	2	N	3
Built-in clock & EPROM prog.	N	N	Y
Max. memory addressing	124K	65K	128K
I/O ports & I/O pins	4;32	3;27	4;32
Register files	144	12	256
Interrupts	6	2	6
Machine lang. instructions	43	33	111
Flags	6	3	5
Clock frequency	8MHz	4MHz	12MHz
Price (dollars)	20	30	80

4. The author is more familiar with it than with the Intel MCU.

Introducing the MC-1Z

The MC-1Z is a single-board computer that can be used in "real-time" control applications. It is manufactured by Basicon Corporation and costs approximately $160. It is compact, easy to use and affords a variety of applications through its 40 I/O lines. It contains the following major components:

1. Z8671 MCU with resident tiny BASIC
2. 8255A programmable peripheral interface (8255A PPI)
3. MM57284 programmable real-time clock/calendar

4. 2K by 8 CMOS RAM or 8K by 8 CMOS RAM
5. Application socket for a 2732 or 2764 EPROM or 2K by 8 or 8K by 8 additional RAM
6. Serial interface buffers and negative voltage converter

Figure 1 details the makeup of the MC-1Z as used in this project. A memory map of the MC-1Z is shown in Figure 2.

The Z8671 MCU

The Z8671 MCU is an 8-bit MCU. It has a BASIC/DEBUG interpreter in ROM. Because the BASIC/DEBUG interpreter is present in the Z8671 MCU, programming is made much easier. The Z8671 MCU allows fast hardware tests, and bit-by-bit examination of any memory location, I/O port, or register. It also allows bit manipulation and logical operations. It has a self-contained line editor to support interactive debugging, further speeding up program development.

Two kinds of memory exist in the Z8671 MCU: on-chip registers and external ROM and RAM. BASIC/DEBUG assigns addresses 0 through 255 to the on-chip registers. Maximum memory expansion is 62K bytes of external program memory and 62K bytes of data memory. This provides up to 124K bytes of useable memory space. In addition, 32 I/O lines, a 144-byte register file, on-board UART and two counter/timers are provided. The functional block diagram of the Z8671 MCU is shown in Figure 3.

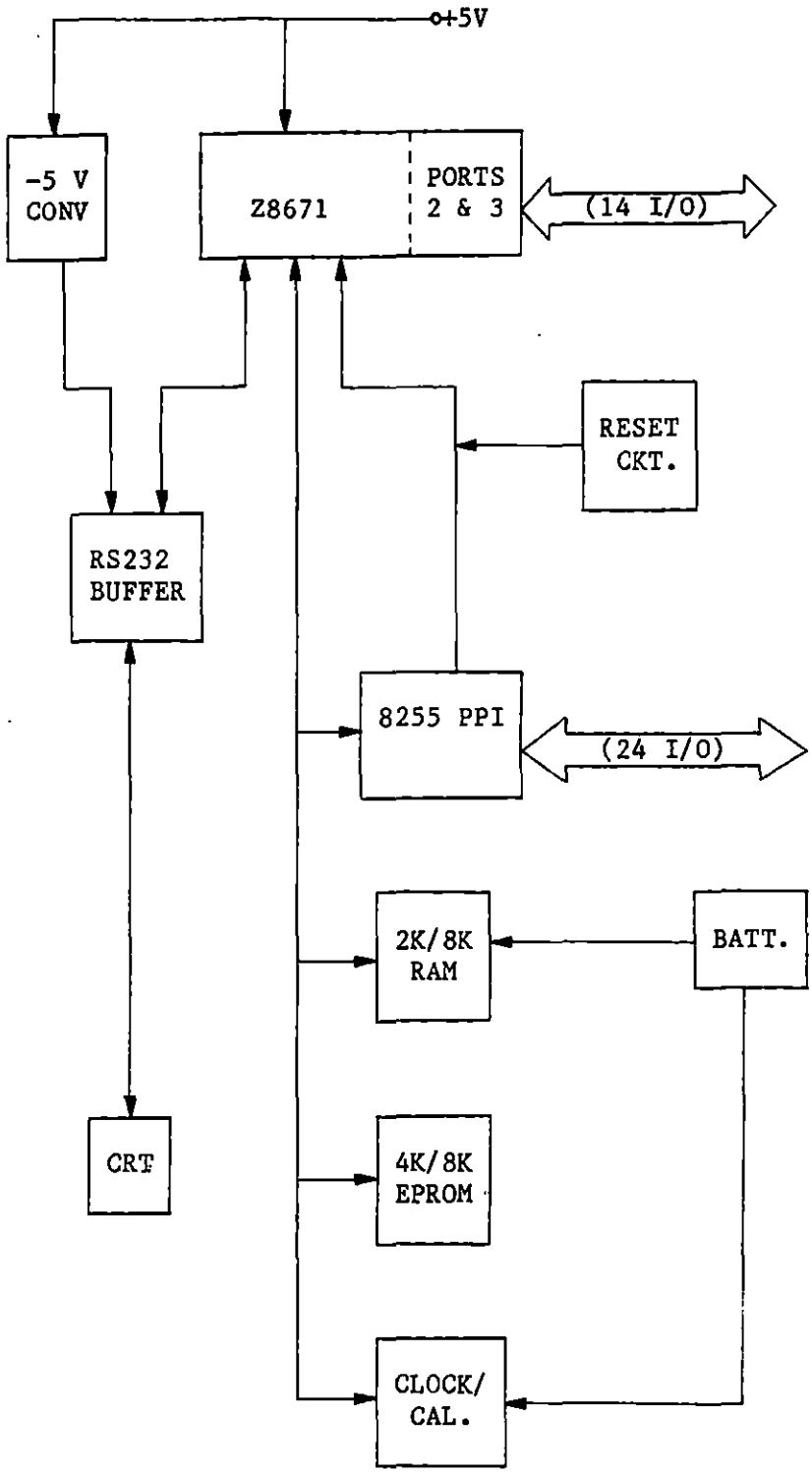


FIGURE 1. Block diagram of the MC-1Z

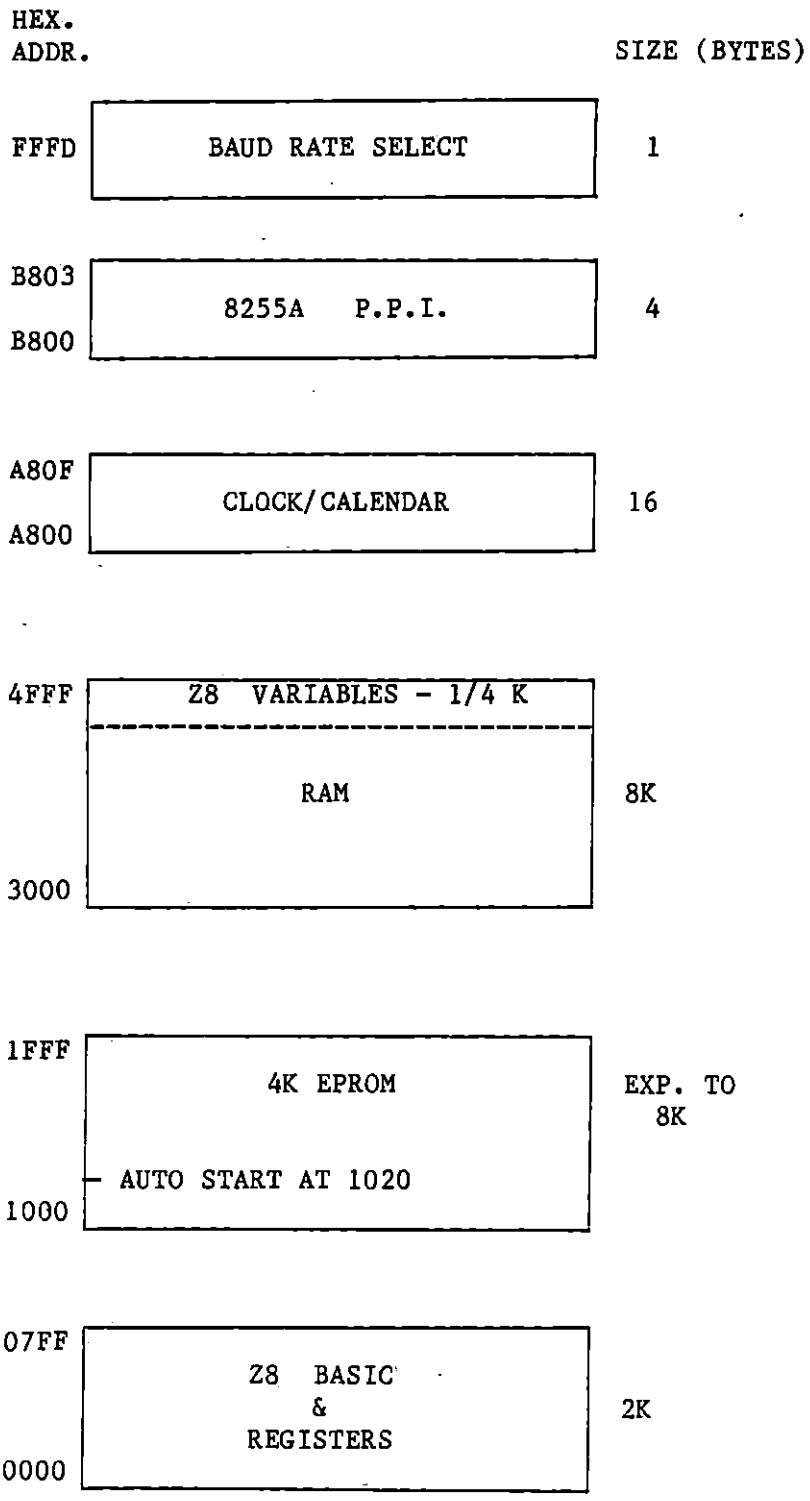


FIGURE 2. The MC-1Z MCU memory map

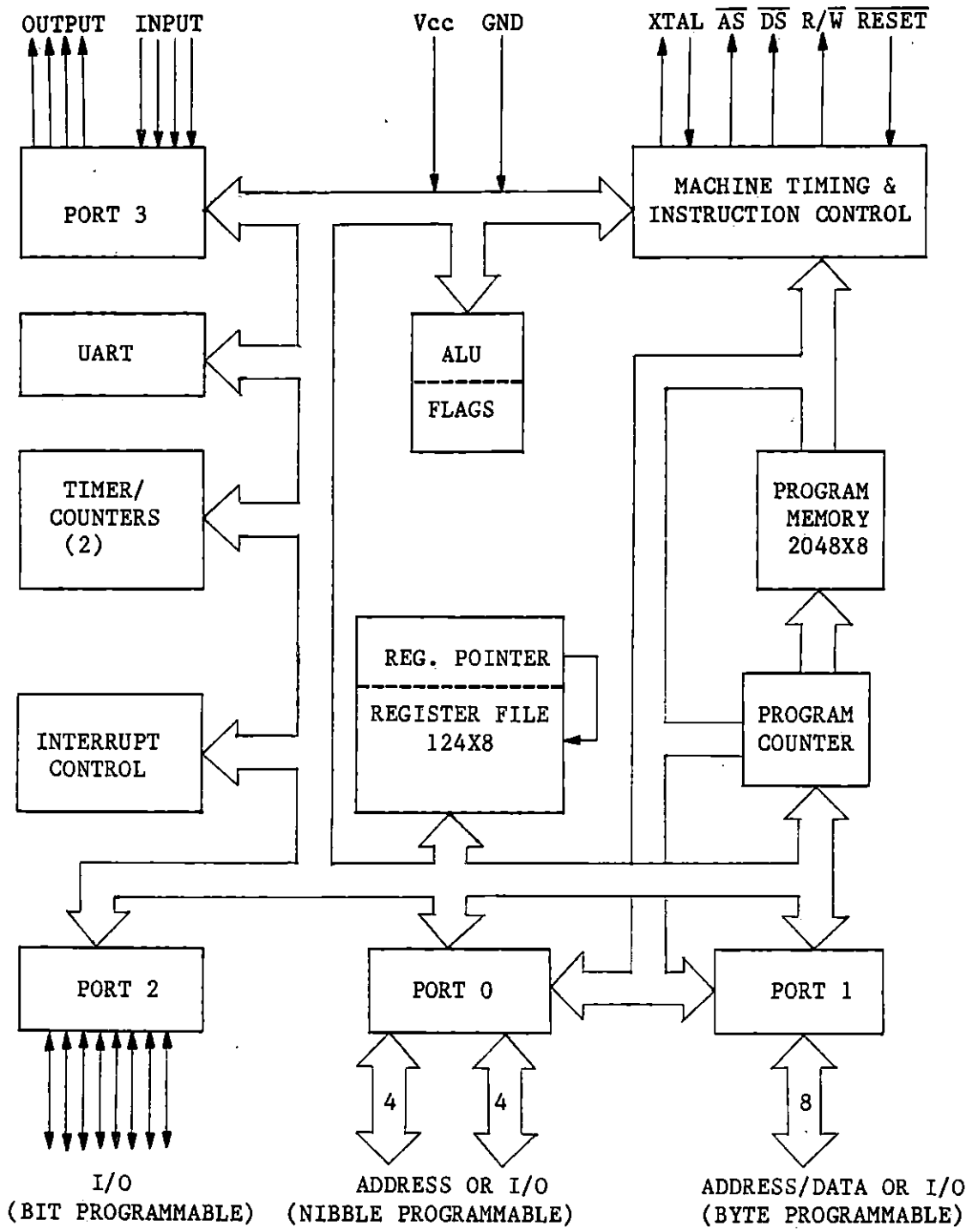


FIGURE 3. The functional diagram of the Z8671 MCU

The 8255A PPI

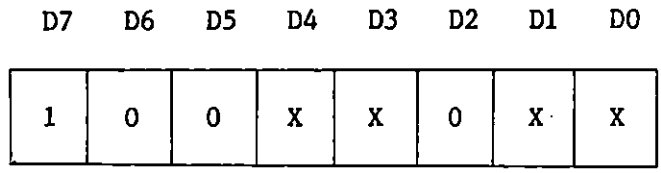
The 8255A PPI is a general-purpose I/O interfacing device. It provides 24 I/O lines organized as three 8-bit I/O ports labeled A, B, and C. All of the bits in ports A or B are programmed as one byte. The four high- and four low-order bits of port C can be programmed as two separate nibbles.

The 8255A PPI is a very versatile device. It can be programmed to look like three simple I/O ports (mode 0), two handshaking I/O ports (mode 1), or a bidirectional I/O port with five handshaking signals (mode 2). The modes can also be intermixed. For example, port A can be programmed to operate in mode 2, while port B operates in mode 0. There is also a bit set/reset mode that allows individual bits of port C to be set or reset for control purposes.

The 8255A PPI occupies four sequential memory locations in the MC-1Z:

%B800	port A
%B801	port B
%B802	port C
%B803	PPI control register

For most applications including this project the 8255A PPI is used in mode 0. A single control word is written to the control register (%B803) to define the inputs and outputs. Figure 4 shows this control register. Table 2 shows the 16 combinations for the inputs and outputs of ports A, B, and C.



X=0 : OUTPUT ; X=1 : INPUT

FIGURE 4. The control register used to program the inputs and outputs of the 8255A PPI in mode 0

TABLE 2. The 16 combinations for the inputs and outputs of ports A, B, and C

ADDRESS DATA @%B803	PORT A	PORT B	PORT C, HIGH NIBBLE	PORT C, LOW NIBBLE
%80	output	output	output	output
%81	output	output	output	input
%82	output	input	output	output
%83	output	input	output	input
%88	output	output	input	output
%89	output	output	input	input
%8A	output	input	input	output
%8B	output	input	input	input
%90	input	output	output	output
%91	input	output	output	input
%92	input	input	output	output
%93	input	input	output	input
%98	input	output	input	output
%99	input	output	input	input
%9A	input	input	input	output
%9B	input	input	input	input

Each time the 8255A PPI is reset, all of the ports are configured as inputs and the lines are floating. Each time the 8255A PPI control register is written to, all selected outputs are set low.

The MM58274 real-time clock/calendar

The MC-1Z has a real-time clock/calendar. The clock/calendar is an N.S.C. MM58274 which provides tenths of seconds through years, including leap year calculation. The clock is crystal controlled and requires low power. In the MC-1Z the real time clock resides in memory locations %A800 thru %A80F as shown in Table 3.

TABLE 3. The address, counter, and mode of the real-time clock

ADDRESS	COUNTER[a]	MODE
%A800	Control register	Split read and write
%A801	Tenth of seconds	Read only
%A802	Unit of seconds	Read or write
%A803	Tens of seconds	Read or write
%A804	Units of minutes	Read or write
%A805	Tens of minutes	Read or write
%A806	Units of hours	Read or write
%A807	Tens of hours	Read or write
%A808	Units of days	Read or write
%A809	Tens of days	Read or write
%A80A	Units of months	Read or write
%A80B	Tens of months	Read or write
%A80C	Units of years	Read or write
%A80D	Tens of years	Read or write
%A80E	Day of week	Read or write
%A80F	Clock setting/interrupt reg.	Read or write

[a]All counters are four bit counters and utilize the lower four bits of the data bus (D0-D3).

The clock must be initialized after power up. During initialization, the clock is stopped and the interrupts are disabled by setting the control register to 5, @%A800=5. The hour setting can be specified for either 12 or 24 hour mode. If 12 hour mode is selected:
If am:

@%A80F=0 If this year is a leap year.

@%A80F=4 If this year is a leap year plus one year.

@%A80F=8 If this year is a leap year plus two years.

@%A80F=12 If this year is a leap year plus three years.

If pm:

@%A80F=2 If this year is a leap year.

@%A80F=6 If this year is a leap year plus one year.

@%A80F=10 If this year is a leap year plus two years.

@%A80F=14 If this year is a leap year plus three years.

If 24 hour mode is selected:

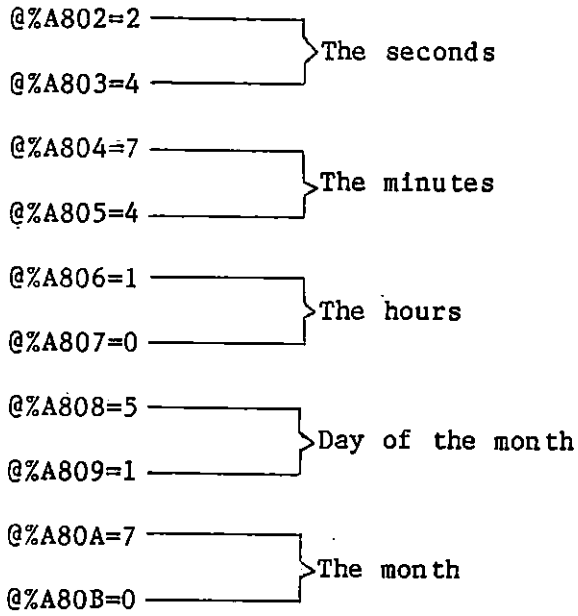
@%A80F=1 If this year is a leap year.

@%A80F=5 If this year is a leap year plus one year.

@%A80F=9 If this year is a leap year plus two years.

@%A80F=13 If this year is a leap year plus three years.

Finally the clock is set by loading memory locations %A802 through %A80F with desired data. For example, to set 1:47:42 pm on July 15, 1986 (12 hour mode):



@%A80C=6 ┐
         ├ The year
@%A80D=8 ┘

@%A80E=3 Tuesday, the third day of the week

@%A80F=10 This is a leap year plus two years.

It is conventional to assign Sunday as the first day of the week. So, Tuesday is the third day of the week.

When everything is set the clock is started with either of two commands: @%A800=0 (interrupt timer is running) or @%A800=1 (interrupt timer is off). The clock is stopped with a @%A800=4 command.

The serial interface buffers

There are two serial interface buffers (RS232/TTL and TTL/RS232 conversion) in the MC-1Z. RS232 was developed in the early 1960s as a standard governing the interconnection of terminals and modems. The RS232 standard defines a single-ended transmission technique limited to 20,000 baud with a 50-ft cable. The conversion buffers and the Z8671's UART use 3 lines (transmitted data, received data, and signal ground) to connect to a terminal.

The most striking feature about RS232 is that the logical levels are not TTL-compatible. TTL is fine for short-distance cables of 5 to 10 ft, provided that the data rate is not too high. However, as the cable length increases, the capacitive and DC loading effects reduce the noise margins to unacceptable levels.

An RS232 receiver will interpret a voltage more negative than -3 V as a logic 1 and a voltage more positive than +3 V as a logic 0. RS232 transmitters are specified to output a voltage more negative than -5 V

for logic 1 and more positive than +5 V for a logic 0. The actual output voltage depends on the supply voltages used. With this scheme 2 V of noise immunity is guaranteed. This should be compared with 0.4 V for standard TTL. The conversion buffers are placed between the RS232 lines and the UART to assure that terminal-computer communication can occur.

Additional I/O lines

The MC-1Z still has 14 I/O lines which come directly from the Z8671 MCU. They are assigned as port 2 and port 3. Eight of the 14 lines belong to port 2, the others belong to port 3. The control registers for ports 2 and 3 are registers 246 and 247, respectively. The control word for port 3 is complex and does not need to be illustrated here. The majority of uses will define P3(1) to P3(3) as simple inputs and P3(4) to P3(6) as simple outputs. The command for this is @247=%41. P3(0) (input) and P3(7) (output) are committed to serial I/O with a terminal.

The control register for port 2 is register 246. A control word is sent to 246 to set what bits of port 2 are inputs and outputs. Figure 5 shows how to set the inputs and outputs of port 2.

For example:

@246=0	All lines are output lines.
@246=%F	The high nibble is output, the low nibble is input.
@246=%F0	The high nibble is input, the low nibble is output.
@246=%FF	All lines are input lines.

D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	X	X	X	X

X=0 : OUTPUT ; X=1 : INPUT

FIGURE 5. The control register for port 2

There is also an EPROM programming module and an accompanying utility PROM (ZUTIL-1.00) to support the MC-1Z. The utility PROM offers fourteen commands for program development on the MC-1Z. These fourteen commands are divided into five groups:

1. Alter memory and check memory: Alter memory (A), Copy memory (C), Display memory (D), and Fill memory (F).
2. Copy a program inside RAM to EPROM: Erase check (E), Program EPROM (P), and Verify EPROM (V).
3. Copy a program inside EPROM to RAM: Display ROM (R) and Copy ROM to memory (G).
4. Mark the top of a program and find the end of a program: Mark top (M) and Locate data (L).
5. Accessory: Help (H), Set time (S), and Time check (T).

Once the program module is attached and the utility PROM is installed in the MC-1Z socket, the above commands are very useful in program development.

A BLOCK DIAGRAM OVERVIEW THE Z8 DATA ACQUISITION SYSTEM

The Basicon MC-1Z single-board computer was used as the main part of the Z8 data acquisition system. Its 24 parallel programmable I/O lines which come from the 8255A PPI were connected to the A/D converter, D/A drivers, relay drivers, sense switches, and a keypad.

In this system, port A was configured as an input port for the A/D converter, port B was configured as an output port for the two D/A converters, relay drivers (and relays). The high nibble of port C was configured as input lines for the sense switches and the keypad and the low nibble of port C was configured as output lines to a 4-to-16 decoder. The decoder output lines are the control lines for the Z8 data acquisition system. A block diagram of the system is shown in Figure 6.

The advantages of this system are the following.

1. It only needs one (+5 V) power supply for operation.
2. It is light in weight and compact to carry.
3. The system can be reset with a pushbutton without losing your application program.
4. A battery will keep the RAM and real-time clock/calendar alive.
5. The utility PROM (ZUTIL-1.00) (a 2732 EPROM installed in the MC-1Z ROM socket) and a program module can be attached. With a few simple instructions from the keyboard, you can copy an application program from RAM to EPROM.

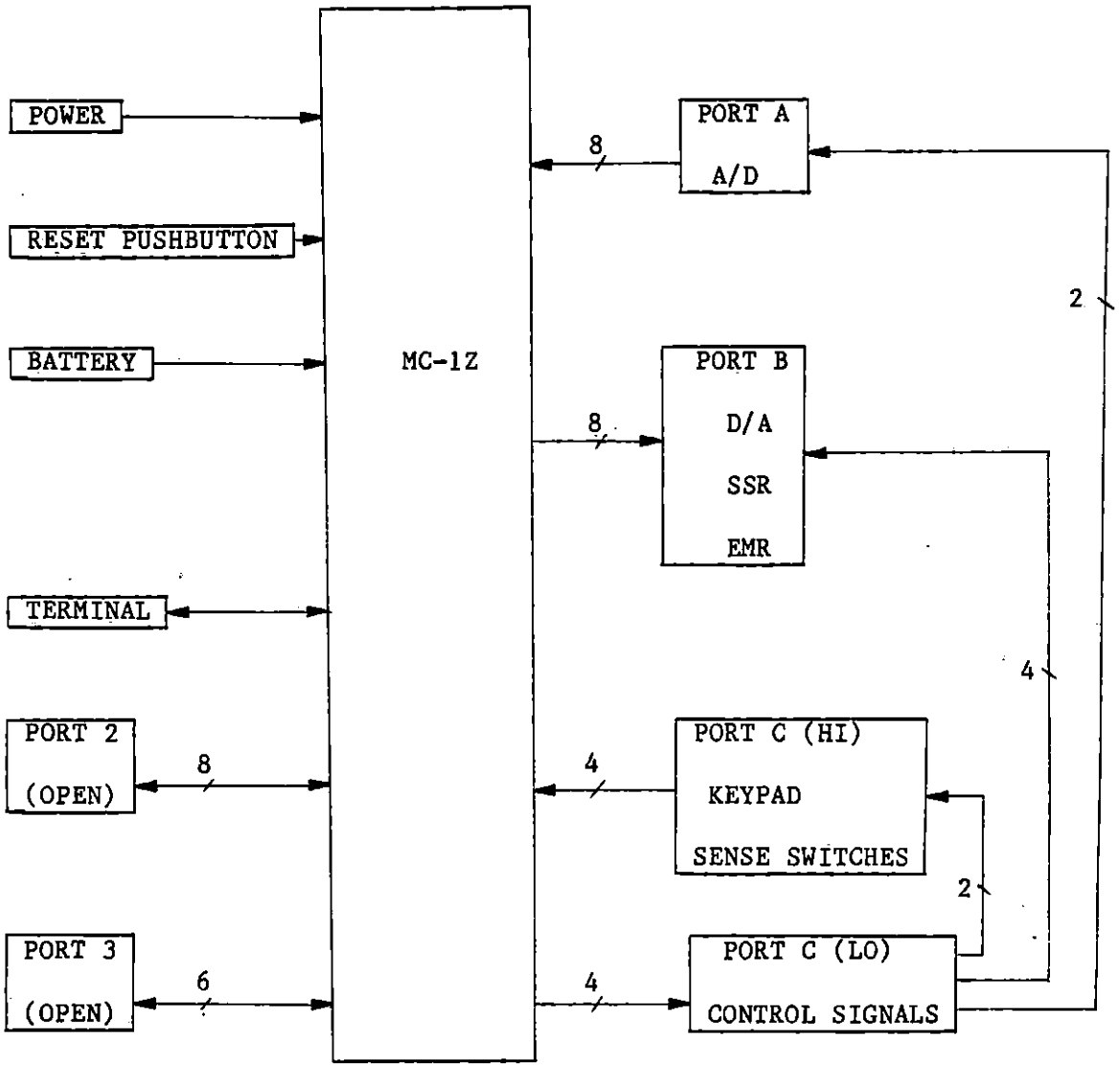


FIGURE 6. A block diagram of the Z8 data acquisition system

Subsystems

A/D

Port A was configured as an input port to accept the output lines (DB0-DB7) of the N.S.C. ADC0844 A/D converter. The A/D circuit is shown in Figure 7. A single circuit phone jack is provided for each A/D input channel. A variable potentiometer is also provided to set any desired voltage from 0 to 5 V. When a phone plug is inserted into the jack, the potentiometer is deactivated and the signal is coming via the phone plug.

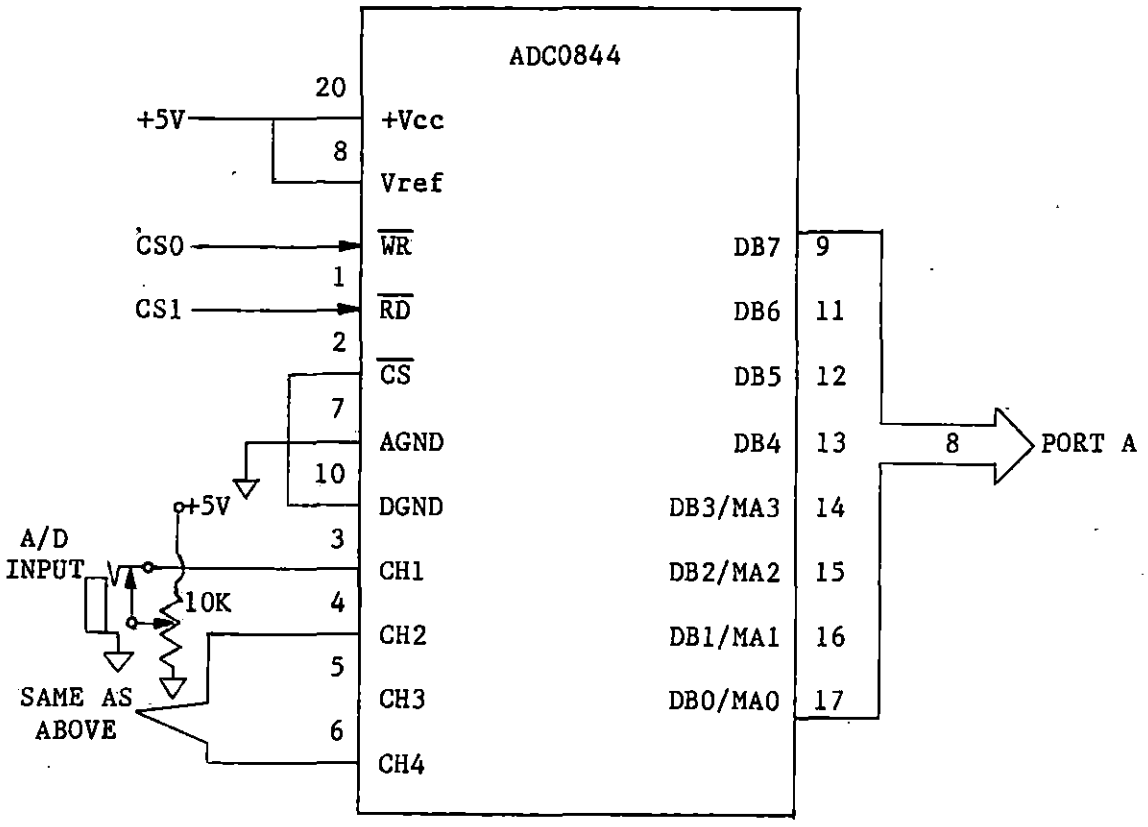


FIGURE 7. The A/D circuit

The ADC0844 is an 8-bit A/D converter with an analog input multiplexer (MUX). The 4 channel MUX can be programmed for 3 modes operation: single-ended, differential, or pseudo-differential. See Table 4.

TABLE 4. ADC MUX addressing

MUX ADDRESS							Channel #					MUX
MA3	MA2	MA1	MA0	$\overline{CS}$	$\overline{WR}$	$\overline{RD}$	CH1	CH2	CH3	CH4	AGND	MODE
X[a]	L	L	L	L[a]	⌐↑	H[a]	+	−				
X	L	L	H	L		H	−	+				
X	L	H	L	L		H		+	−			DIFFERENTIAL
X	L	H	H	L		H			−	+		
L	H	L	L	L	⌐↑	H	+				−	
L	H	L	H	L		H		+			−	
L	H	H	L	L		H			+		−	SINGLE-ENDED
L	H	H	H	L		H				+	−	
H	H	L	L	L	⌐↑	H	+			−		
H	H	L	H	L		H		+		−		PSEUDO-
H	H	H	L	L		H			+	−		DIFFERENTIAL
X	X	X	X	L	⌐↑	L	PREVIOUS CHANNEL CONFIGURATION					

[a]H = High voltage ; L = Low voltage ; X = Don't care.

In this project, only the single-ended mode was used. Since the MUX address latches (MA0-MA3) are common with data bus lines (DB0-DB3), four steps were needed for channel selection:

1. Port A is configured as an output port.
2. The ADC0844 A/D chip's $\overline{WR}$ pin is set low.
3. A byte is sent to port A to choose the channel.
4. The ADC0844 A/D chip's $\overline{WR}$ pin is set high.

As perviously mentioned, port A and the control register of the 8255A PPI are located at memory locations %B800 and %B803, respectively. An example which uses BASIC instructions to select channel 1 and to accept the input signal from channel 1 is shown below.

```
@%B803=%88      Set port A to be an output port
@%B802=0        ADC0844 WR low
@%B800=4        Send the proper value to port A to choose channel # 1
@%B802=15       ADC0844 WR high; start conversion
@%B803=%98      Set port A to be an input port
@%B802=1        ADC0844 RD low
Y=@%B800        Accept input signal from port A
@%B802=15       ADC0844 RD high
```

The conversion time of the ADC0844 is 40 microseconds. If only BASIC instructions are used to do data processing, there will be no problem. If machine-language instructions are used, the time interval for acquiring the data should be longer than 40 microseconds. Otherwise, the ADC0844 will not have enough time to convert the data.

D/A

Port B was configured as an output port to connect to the input lines of an 8-bit buffer (74LS244). The buffer output lines were connected to the input lines of two 8-bit D/A converters (AD558) to form two D/A outputs. Figure 8 shows the D/A circuitry.

The following BASIC commands select D/A #1 and D/A #2 and set their output voltages.

```
@%B802=2        Choose D/A #1
```

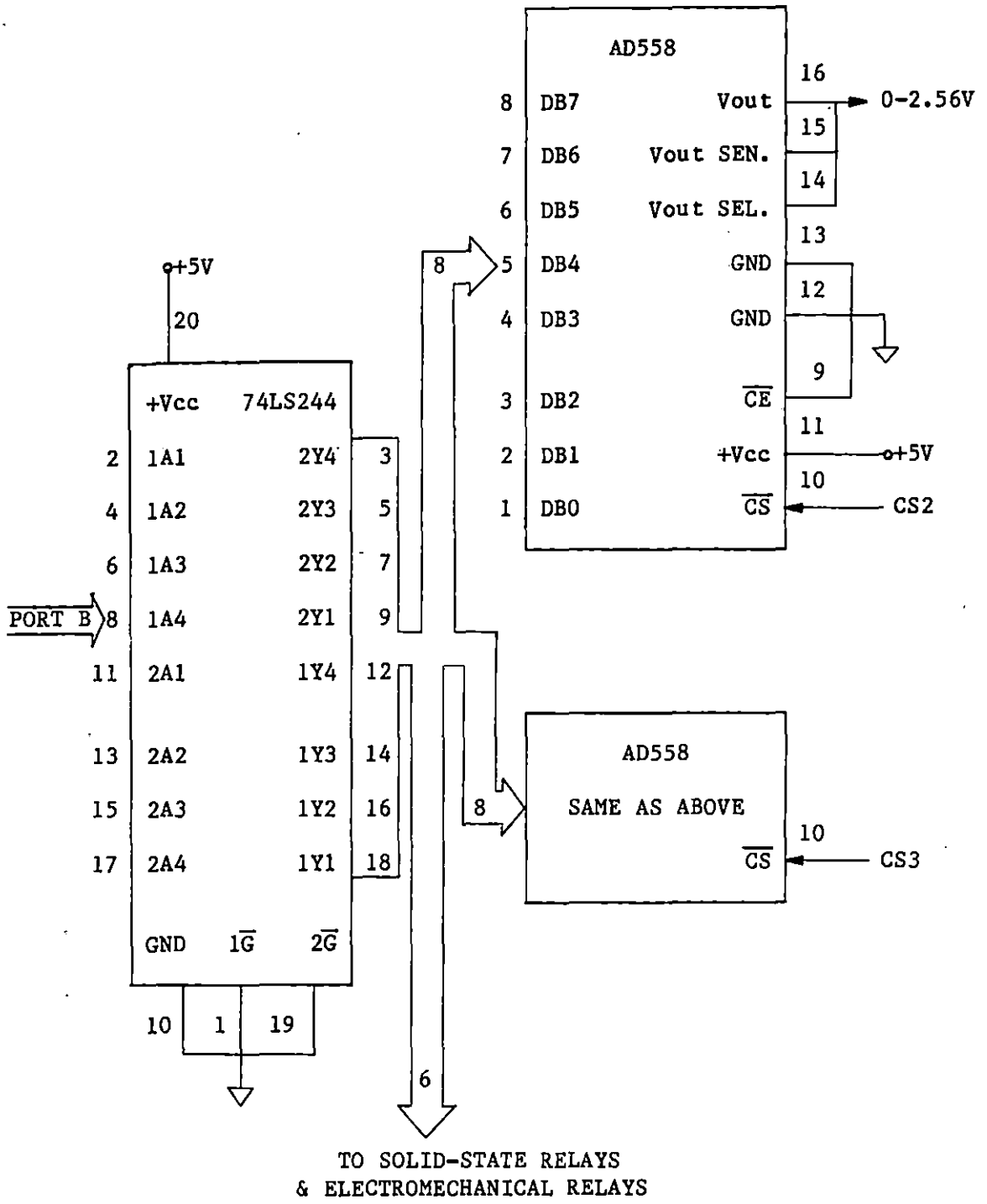


FIGURE 8. The D/A circuit

@%B801=0	D/A #1 outputs zero volts
@%B801=255	D/A #1 outputs 2.56 volts
@%B802=3	Choose D/A #2
@%B801=128	D/A #2 outputs 1.28 volts

Not only the D/A outputs but also the relays were connected to port B. The 8-bit buffer provides the current to drive the D/A outputs and the relays.

The AD558 is a voltage-output 8-bit D/A converter. It will operate with any power supply voltage between +4.5 and +16.5 V. There are two output ranges (0 to +2.56 V and 0 to +10 V). The 0 to +10 V output range requires a power supply of +11.4 to +16.5 V. In this system, the 0 to +2.56 V output range was chosen since the system's power supply voltage was +5 V.

Solid-state relays (SSRs)

The output lines (1Y1-1Y4) of the 74LS244 buffer were wired to a bistable latch (74LS75). The latch's four output lines were wired to an inverting current buffer (SN75492), and the current buffer's output lines were connected to four logic lines of the Basicon electrical control design (ECD) series 6 I/O module mounting system (MS-4-E) which has four data/control lines, four field pairs, and four solid-state relays (SSRs). Each data/control lines has a 3.3 Kohm pull-up resistor and a LED. Socketed LED indicators are included at each module for on/off status. Each field pair includes a plug-in replaceable 5 A fuse to protect the wiring and/or load. The four field pairs are connected to four 110V sockets. Figure 9 shows the SSR circuit.

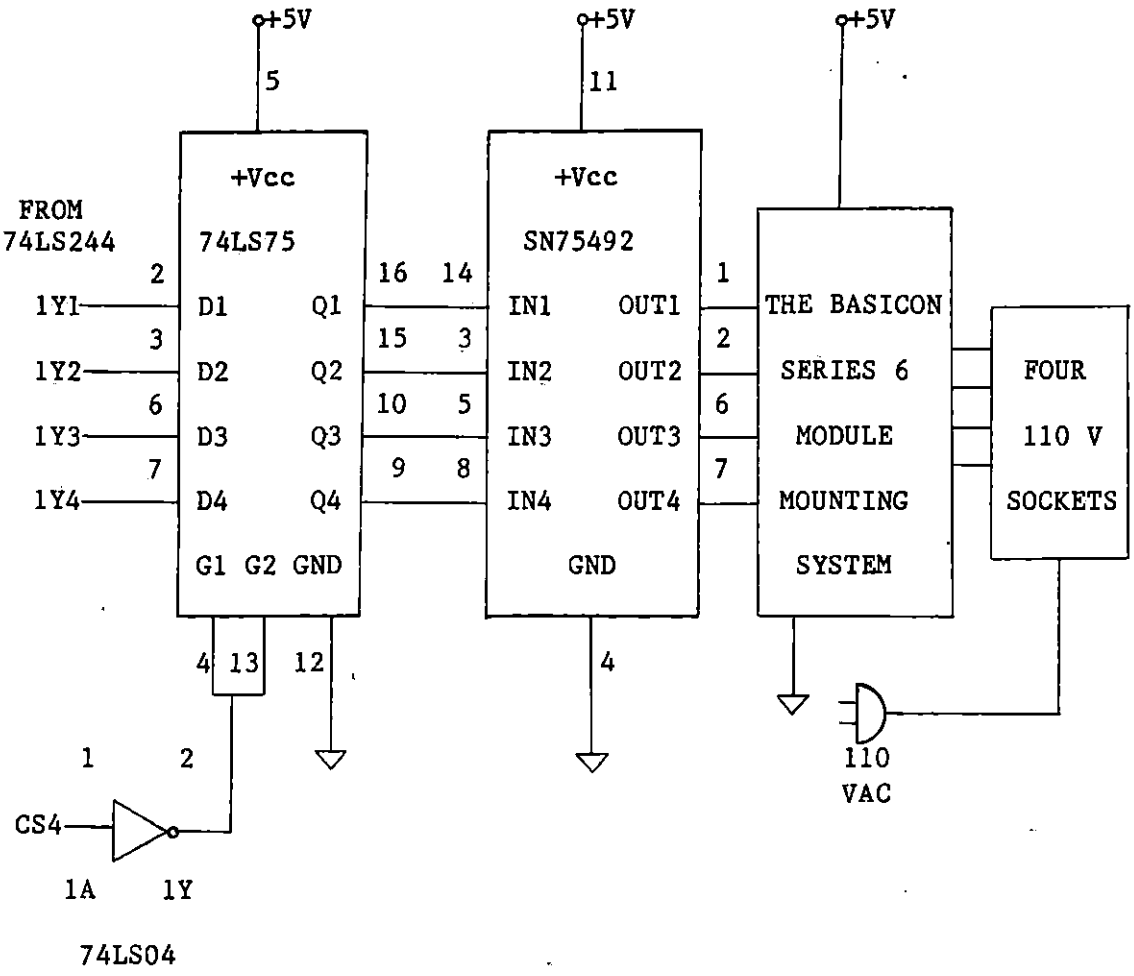


FIGURE 9. The SSR circuit

The following BASIC commands select the SSR circuit and operate the relays.

@%B802=4	Choose SSR circuit
@%B801=1	Let SSR #1 on, others off
@%B801=2	Let SSR #2 on, others off
@%B801=4	Let SSR #3 on, others off
@%B801=8	Let SSR #4 on, others off

```
@%B801=15          Let all SSRs on
```

To turn any or all SSRs off, the command is @%B801=0. The latch is used for temporary storage of binary information. If the D/As and SSRs are used alternately, the latch will hold the SSR commands while the MC-1Z handles the D/A outputs. The D/A converter has its own built-in latch and thus the two processes will not interfere with each other. The SN75492 current buffer supplies the necessary current to drive the SSRs.

The SSRs are useful for controlling high-current loads and for isolation. For example, if a short circuit happens in a light bulb circuit controlled by the SSR there will be no damage to the MC-1Z.

Electromechanical relays (EMRs)

Two of the output lines (2Y1-2Y2) from the 74LS244 buffer were connected to another latch (74LS75). The latch's two output lines were connected to the two remaining input lines of the SN75492 current buffer. The current buffer's two remaining output lines were then connected to the two EMRs. Figure 10 shows the EMR circuit.

The following BASIC commands select the EMR circuit and activate the relays.

```
@%B802=5           Choose EMR circuit
@%B801=16          Let EMR #1 on, the other off
@%B801=32          Let EMR #2 on, the other off
@%B801=48          Let all EMRs on
```

To turn either or both EMRs off, the command is @%B801=0. The latch in EMR circuit allows for independent EMR operation.

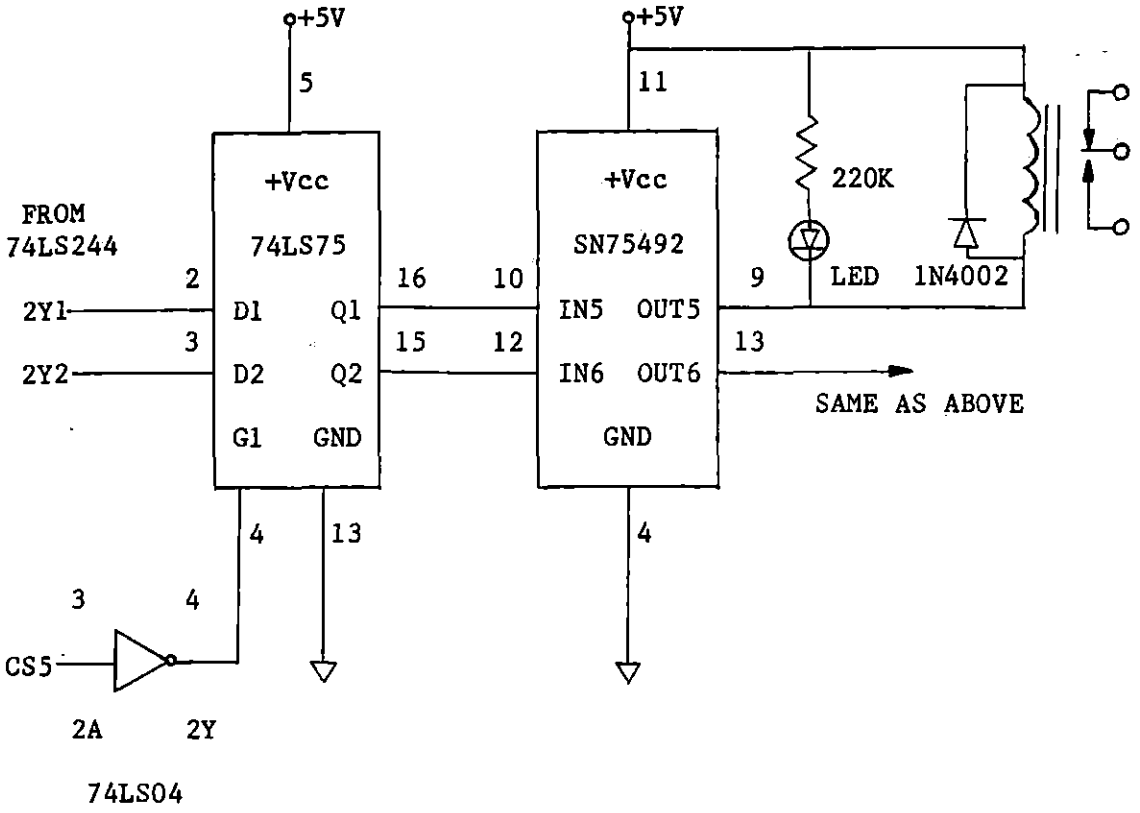


FIGURE 10. The EMR circuit

The sense switches

Four single-pole double-throw (SPDT) toggle switches were connected to a quad inverse SR latch (74LS279). The latch's output lines were connected through another buffer (74LS244) to the high nibble of port C. Figure 11 shows the sense switch circuit.

The following BASIC commands select the sense switch circuit and determine which switches are on or off.

```
@%B802=6                    Choose the sense switch circuit
X=AND(@%B802, 240)/16       Determine which switches are on or off
```

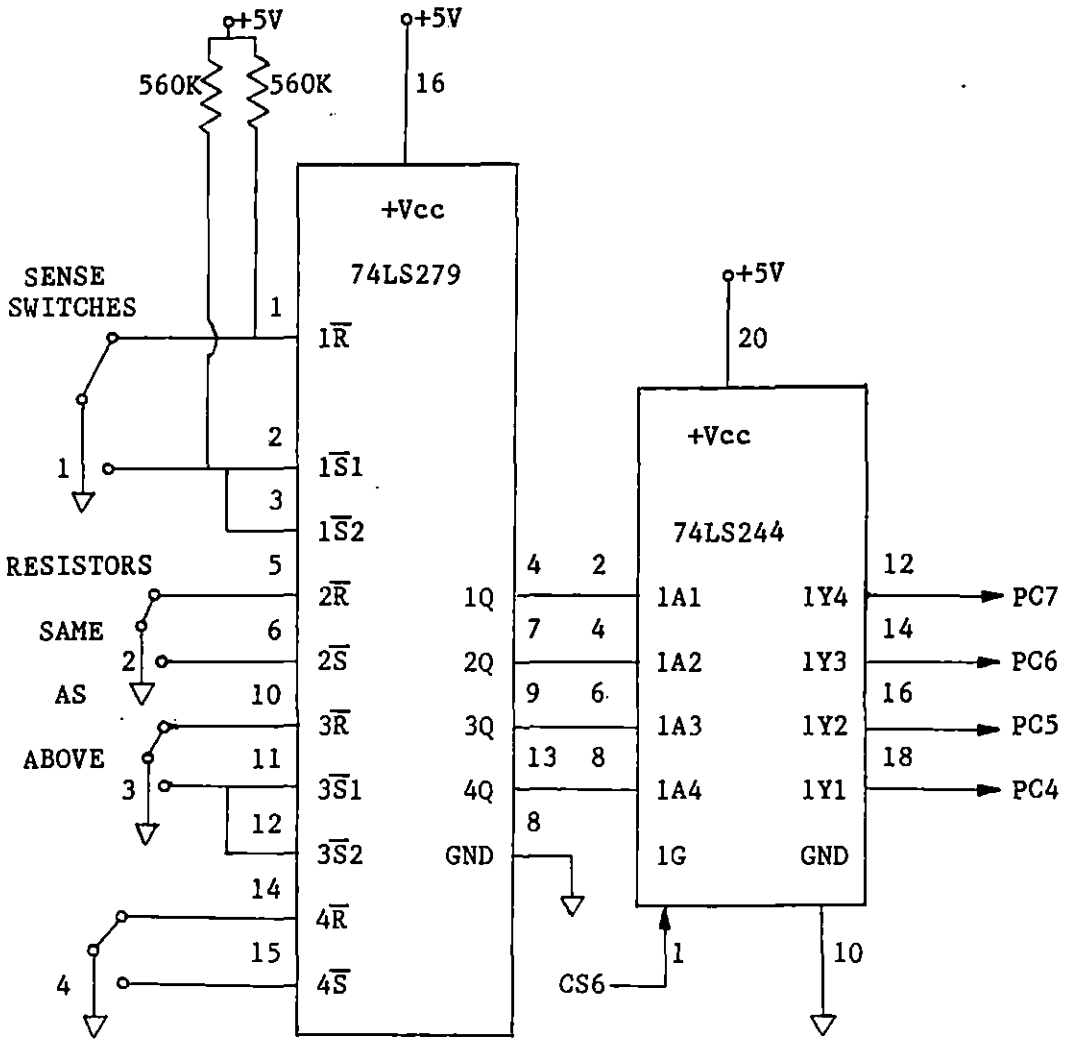


FIGURE 11. The sense switch circuit

If X=1, switch #1 is on and others are off; if X=3, switches #1 and 2 are on and the others are off.

Toggle switches can produce contact bounce which could subsequently cause false data. The inverse SR latch operates as a debouncing circuit. The sense switch circuit shares the high nibble of port C with the keypad

circuit. Thus a buffer was used as a means to separate the signals coming from the sense switches and the keypad.

The keypad

The keypad (16 keys, 4X4 matrix) has 8 output lines, 4 for the rows and 4 for the columns. The keypad's output lines were connected to a 16-key encoder (MM74C922) and the encoder's output lines were connected to the high nibble of port C. Figure 12 shows the keypad circuit.

The following BASIC commands select the keypad circuit and determine which pushbutton is pressed.

```
@%B802=7                        Choose the keypad circuit
Y=AND(@%B801, 240)/16           Determine which pushbutton is pressed
```

If Y=0, the #1 pushbutton is pressed; if Y=2, the #2 pushbutton is pressed; if Y=14, the E pushbutton is pressed; and if Y=15, the D pushbutton is pressed.

The 16-key encoder provides all the necessary logic to fully encode an array of single-pole single-throw (SPST) pushbuttons. It also has an internal debounce circuit which needs only a single external capacitor connected from the keybounce mask line (KBM) to ground. An internal register in the encoder remembers the last key pressed even after the key is released. The TRI-STATE outputs provide for easy expansion and bus operation and are low power transistor-transistor logic (LPTTL) compatible.

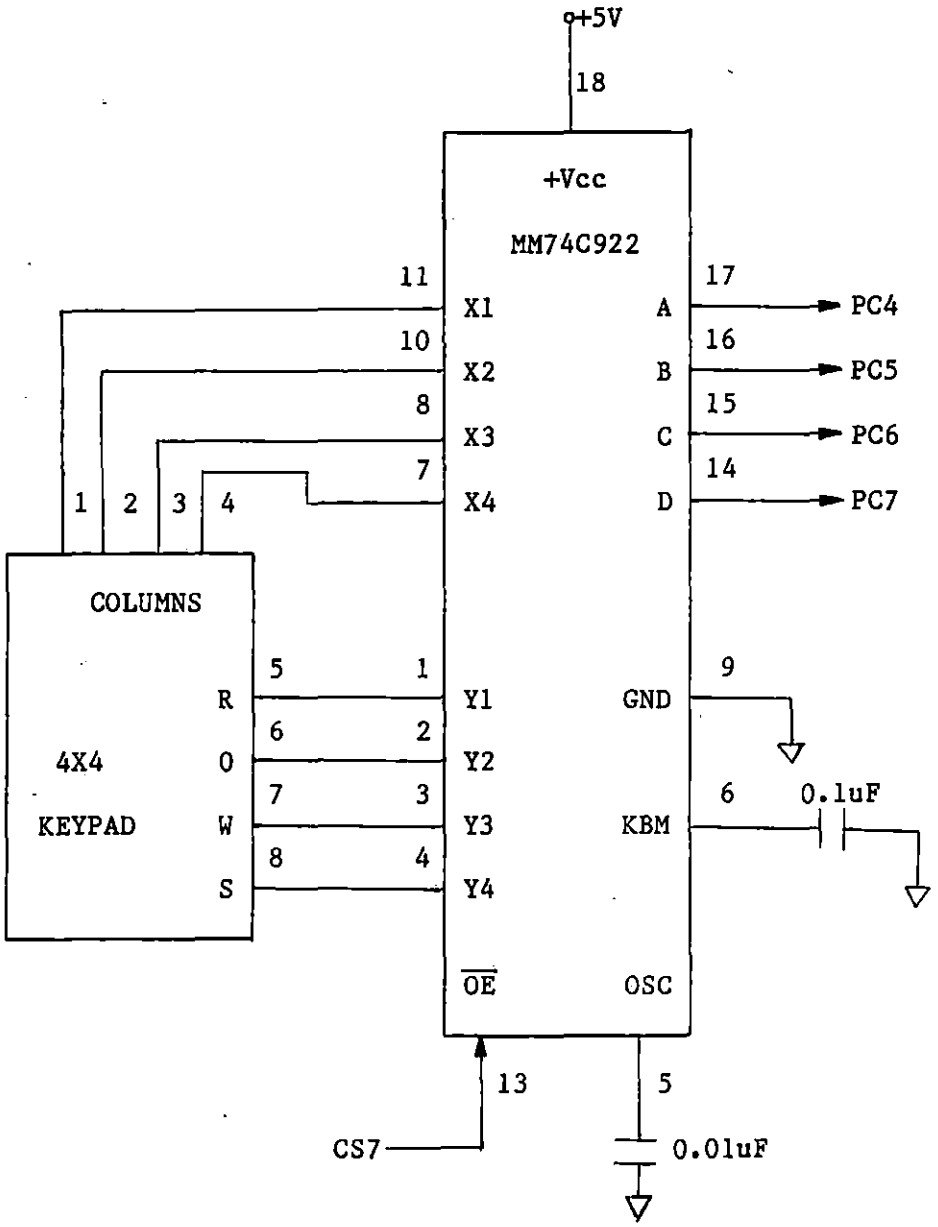


FIGURE 12. The keypad circuit

System device control

The low nibble of port C was configured as output lines and were connected to a 4-line-to-16-line decoder (MM74C154). Only 8 (0-7) of the 16 output lines of the decoder were used as the control signals for the system. The remaining lines are for later expansion. The control lines were assigned as 0 to CS0, 1 to CS1, ..., etc. Figure 13 shows the circuit for the control lines.

CS0 and CS1 which were connected to the ADC0844 A/D chip; CS2 and CS3 were connected to the AD558 D/A chips; CS4 was connected through an inverter (74LS04) to latch the SSR circuit; CS5 was connected through the above mentioned inverter to latch the EMR circuit; CS6 was connected to the buffer of the sense switch circuit; and CS7 was connected to the keypad encoder. The following BASIC commands enable CS0, CS1, CS2, CS3, CS4, CS5, CS6, and CS7.

@%B802=0	Enable CS0 (edge triggered)
@%B802=1	Enable CS1 (edge triggered)
@%B802=2	Enable CS2 (level triggered)
@%B802=3	Enable CS3 (level triggered)
@%B802=4	Enable CS4 (level triggered)
@%B802=5	Enable CS5 (level triggered)
@%B802=6	Enable CS6 (level triggered)
@%B802=7	Enable CS7 (level triggered)

Only one "CS" signal can be enabled at a time. All others are disabled (logic high).

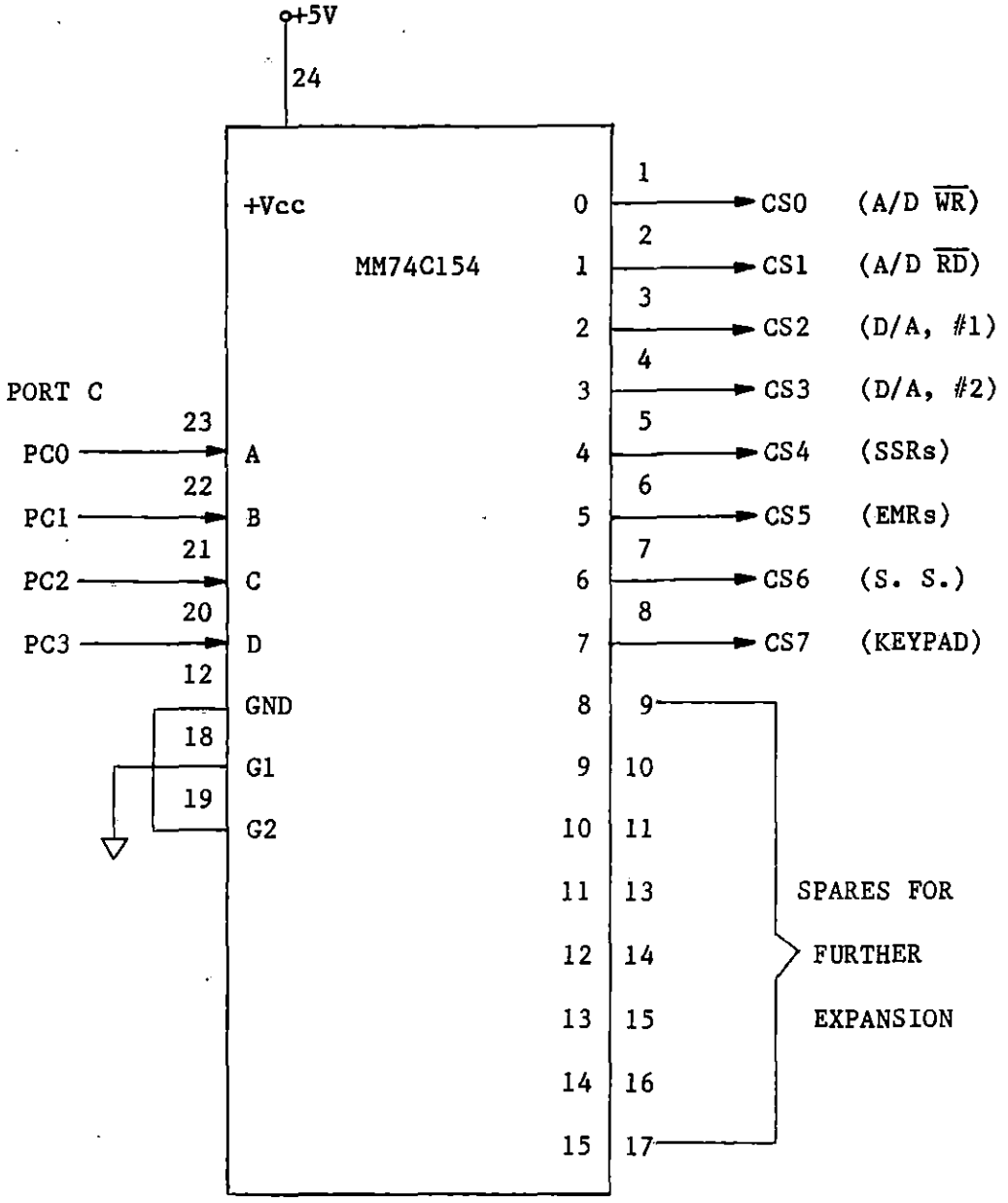


FIGURE 13. The system device control

SOFTWARE DEMONSTRATION OF CAPABILITIES

After the hardware is constructed, software is used for a demonstration which can let the user easily understand the whole system. Two Zilog books that will be invaluable to the user are "Z8671 single-chip BASIC interpreter: BASIC/DEBUG software reference manual" and "Z8 microcomputer: Technical manual".

The following program will demonstrate the system's capabilities. Program description follows the listing.

```
1 REM ----- Z8 DEMO (4-9-86)
10 A=%B800:B=A+1:C=A+2:D=A+3
20 @D=%98
30 @B=0:@C=2:@C=3:@C=4:@C=5
40 PRINT:PRINT
50 "WHAT DO YOU WANT TO DO?"
60 "1)-----A/D"
70 "2)-----D/A"
80 "3)-----SENSE SWITCHES"
90 "4)-----KEYPAD"
100 "5)-----ELECTROMECHANICAL RELAYS"
110 "6)-----SOLID-STATE RELAYS"
120 "7)-----END THIS PROGRAM"
130 "PRESS 1-7 FOR CHOICE"
140 INPUT H
150 IF H<=0 "TRY AGAIN.":GOTO 130
```

```
160 IF H>7 "TRY AGAIN.":GOTO 130
170 IF H=1 GOSUB 300
180 IF H=2 GOSUB 400
190 IF H=3 GOSUB 500
200 IF H=4 GOSUB 600
210 IF H=5 GOSUB 700
220 IF H=6 GOSUB 800
230 IF H=7 STOP
240 GOTO 40

300 REM ----- A/D SUBROUTINE
305 "PRESS 1-4 TO SELECT CHANNEL 1-4 AS INPUT CHANNEL."
310 "IF THE INPUT SIGNAL EQUALS ZERO, THIS SUBROUTINE IS EXITED."
315 INPUT X
320 IF X<=0 "TRY AGAIN.":GOTO 315
330 IF X>4 "TRY AGAIN.":GOTO 315
340 @D=%88:@C=%F
350 @C=0:@A=X+3:@C=%F
360 @D=%98:@C=1:Y=@A:@C=%F:PRINT Y
370 IF Y=0 GOTO 390
380 GOTO 340
390 RETURN

400 REM ----- D/A SUBROUTINE
405 "PRESS 1 OR 2 TO SELECT CHANNEL 1 OR 2 D/A OUTPUT."
410 "PRESS 3 TO LEAVE THIS SUBROUTINE."
```

```
415 INPUT G
420 IF G<=0 "TRY AGAIN.":GOTO 415
430 IF G>3 "TRY AGAIN.":GOTO 415
440 IF G=3 GOTO 480
445 @C=G+1
450 S=0
455 @B=S:S=S+1
460 IF S<255 GOTO 455
465 @B=S:S=S-1
470 IF S>0 GOTO 465
475 GOTO 405
480 RETURN

500 REM ----- SENSE SWITCH SUBROUTINE
510 "IF ALL SENSE SWITCHES ARE UP, THIS SUBROUTINE IS EXITED."
520 @C=6
530 M=AND(@C,240)/16:PRINT M
540 IF M=15 GOTO 560
550 GOTO 530
560 RETURN

600 REM ----- KEYPAD SUBROUTINE
610 "PRESS THE D PUSHBUTTON OF THE KEYPAD TO LEAVE THIS SUBROUTINE."
620 @C=7
630 N=AND(@C,240)/16:PRINT N
640 IF N=15 GOTO 660
```

```
650 GOTO 630
660 RETURN

700 REM ----- ELECTROMECHANICAL RELAYS SUBROUTINE
705 "PRESS 1 TO ACTIVATE RELAY 1; PRESS 2 TO ACTIVATE RELAY 2."
710 "PRESS 3 TO LEAVE THIS SUBROUTINE."
715 @C=5
720 INPUT K
730 IF K<=0 "TRY AGAIN.":GOTO 720
740 IF K>3 "TRY AGAIN.":GOTO 720
745 IF K=3 GOTO 795
750 IF K=2 GOTO 775
755 @B=0
760 @B=16:GOSUB 900
765 @B=0
770 GOTO 705
775 @B=0
780 @B=32:GOSUB 900
785 @B=0
790 GOTO 705
795 RETURN

800 REM ----- SOLID-STATE RELAYS SUBROUTINE
805 "PRESS 1 TO ACTIVATE ALL RELAYS; PRESS 2 TO ACTIVATE RELAYS
    SEQUENTIALLY."
810 "PRESS 3 TO LEAVE THIS SUBROUTINE."
```

```
815 @C=4
820 INPUT Z
830 IF Z<=0 "TRY AGAIN.":GOTO 820
840 IF Z>3 "TRY AGAIN.":GOTO 820
845 IF Z=3 GOTO 895
850 IF Z=2 GOTO 870
855 @B=0:@B=15:GOSUB 900
860 @B=0
865 GOTO 805
870 @B=0:@B=1:GOSUB 900
875 @B=2:GOSUB 900
880 @B=4:GOSUB 900
885 @B=8:GOSUB 900
890 @B=0:GOTO 805
895 RETURN

900 REM ----- DELAY SUBROUTINE
910 V=0
920 V=V+1
930 IF V<25 GOTO 920
940 RETURN
```

The main program consists of lines 1 to 270. It accomplishes the following:

1. Initialization (lines 1 to 30): Set port A & port C high nibble to be input lines and port B & port C low nibble to be output lines; set A/D outputs at zero volts and deactivate all the relays.
2. User instructions (lines 40 to 130).
3. Error check and subroutine selection (lines 140 to 240): Determine whether the input value from the keyboard is correct or not and where to jump.

Lines 300 to 390 compose the A/D subroutine which does the following:

1. User instructions (lines 305 to 310): Use as a directory for what the A/D subroutine is doing.
2. Error check and channel selection (lines 315 to 330): Determine whether the input value from the keyboard is correct or not, decide which input channel is to be chosen.
3. Main body of subroutine (lines 340 to 390): Print the digital value of the input signal on the CRT screen.

Lines 400 to 480 make up the D/A subroutine which does the following:

1. User instructions (lines 400 to 410): Explain what this subroutine is doing.
2. Error check and channel selection (lines 415 to 440): Determine whether the input value from the keyboard is correct or not and decide which output channel is to be chosen.

3. Main body of subroutine (lines 445 to 480): Produce a triangular wave signal.

The subroutine for interpreting the sense switch settings resides at lines 500 to 560. If the switch is on, a high signal is sent to the corresponding line of port C high nibble. Otherwise, a low signal is sent to the same line. Upon receiving a signal from port C high nibble, the subroutine prints a digital value on the CRT screen. If all the switches are off, the program leaves this subroutine.

Lines 600 to 660 compose the subroutine for reading the keypad. It does approximately the same thing as the sense switch subroutine. The only difference between the two subroutines is the source of the signal. In one, the signal comes from the sense switches and in the other, it comes from the keypad.

The subroutine for controlling the electromechanical relays consists of lines 700 to 795. It does the following:

1. User instructions (lines 705 to 710): Explain what this subroutine is doing.
2. Error check and relay selection (lines 715 to 750): Decide whether the input value from the keyboard is correct or not and choose the exact relay to turn on or off.
3. Execution (lines 755 to 795): Activate/deactivate relay 1 or relay 2.

Lines 800 to 895 make up the subroutine for controlling the solid-state relays. It does the following:

1. User instructions (lines 805 to 810): Explain what this subroutine is doing.
2. Error check and relay selection (lines 815 to 850): Check whether the input value from the keyboard is correct or not and make program selection.
3. Execution (lines 855 to 895): Activate/deactivate all relays simultaneously or sequentially.

A three-second delay subroutine resides at lines 900 to 940. It is needed because the program activates and deactivates the relays in a short amount of time. If there is no delay subroutine in this program the user is not sure that the relays have been activated.

A Program to Count Heart Rate

One way to count the heart rate of a human subject correctly is by use of the electrocardiogram (ECG). The pattern of electrical activity associated with the contraction of cardiac muscle during a heartbeat can be recorded by the procedure of electrocardiography. This procedure produces a recording called an ECG.

A normal ECG consists of a regularly spaced series of waves, portions of which are designated P, Q, R, S, and T. Figure 14 shows a normal ECG waveform. The P wave is caused by electrical currents that are produced as the atria depolarize with contraction. The QRS complex (actually three separate waves - a Q wave, an R wave, and an S wave) is caused by the depolarization of the ventricles. The T wave is caused by currents that are generated as the ventricles repolarize. The blood which is pumped by the ventricles supplies the whole body, so the

ventricles need more strength to pump the blood. That is the reason why the amplitude of the QRS complex is much higher than that of the P wave. The QRS complex is a good target for use in counting the heart rate. If the number of the QRS complexes during a period of time are counted, a determination of the heart rate in beats/minute can be made.

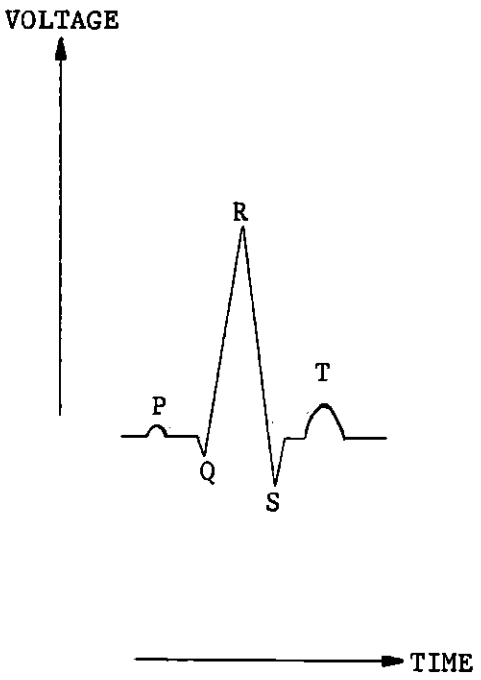


FIGURE 14. A normal ECG waveform

An ECG simulator was used to produce an appropriate signal and a differential amplifier amplified the signal. The amplified signal was then connected to channel 1 of the Z8 data acquisition system. An ECG simulator was used since the frequency and amplitude can be easily changed to test the program. The differential amplifier increased the signal level and rejected any common-mode noise. Counter/timer(C/T) 1 of

the MC-1Z MCU was used as a timer. This required connecting line P3(6) to line P3(1). Because the duration of the QRS complex of the ECG signal is approximately 0.08 seconds (Milnor, 1974), a program using only BASIC instructions would be too slow to detect the QRS complex. Thus a machine language subroutine was written to detect the QRS complex and BASIC instructions handled the remaining parts of the program. The following program displays the heart rate of a human subject on the CRT.

```
10 A=%B800:B=A+1:C=A+2:D=A+3
20 @D=%98
30 @B=0:@C=2:@C=3:@C=4:@C=5
40 @38=130:@39=30
50 @247=%41:@243=0:@242=0:@241=%46
60 GO@%3900
70 PRINT:PRINT
80 IF @242=0 "HEART RATE= 0 BEAT/MINUTE":GOTO 110
90 H=256-@242:Z=150*60/H
100 PRINT "HEART RATE=";Z;"BEATS/MINUTE"
110 "PRESS 1 TO CONTINUE; PRESS ANY OTHER KEY TO STOP"
120 INPUT X
130 IF X=1 GOTO 20
140 STOP
```

LOC.	OP. CODE		HEX. CODE	
%3900	SRP	%30	31	30

```
%3902   LD      r0, 0           0C      00
%3904   LD      r1, 1           1C      01
%3906   LD      r2, %F          2C      0F
%3908   LD      r3, %88         3C      88
%390A   LD      r4, %98         4C      98
%390C   LD      r5, 4           5C      04
%390E   LD      r6, %B8         6C      B8
%3910   LD      r7, 3           7C      03
%3912   LD      r8, %B8         8C      B8
%3914   LD      r9, 2           9C      02
%3916   LD      rA, %B8         AC      B8
%3918   LD      rB, 0           AC      00
%391A   LD      R40, %FF        E6      28      FF
%391D   LD      R41, %FF        E6      29      FF
%3920   LD      R34, 4          E6      22      04
%3923   DEC     R34             00      22
%3925   JP      NZ,%3923        ED      39      23
%3928   DECW    R40,41          80      28
%392A   JP      Z,%398B         6D      39      8B
%392D   LDE     @rr6,r3         92      36
%392F   LDE     @rr8,r2         92      28
%3931   LDE     @rr8,r0         92      08
%3933   LDE     @rrA,r5         92      5A
%3935   LDE     @rr8,r2         92      28
%3937   LDE     @rr6,r4         92      46
%3939   LDE     @rr8,r1         92      18
```

%393B	LDE	@rC,@rrA	82	CA	
%393D	LDE	@rr8,r2	92	28	
%393F	CP	R38,R60	A4	26	3C
%3942	JP	ULT,%3920	7D	39	20
%3945	LD	R241, %4A	E6	F1	4A
%3948	LD	R35, 4	E6	23	04
%394B	DEC	R35	00	23	
%394D	JP	NZ,%394B	ED	39	4B
%3950	LDE	@rr6,r3	92	36	
%3952	LDE	@rr8,r2	92	28	
%3954	LDE	@rr8,r0	92	08	
%3956	LDE	@rrA,r5	92	5A	
%3958	LDE	@rr8,r2	92	28	
%395A	LDE	@rr6,r4	92	46	
%395C	LDE	@rr8,r1	92	18	
%395E	LDE	rD,@rrA	82	DA	
%3960	LDE	@rr8,r2	92	28	
%3962	CP	R39,R61	A4	27	3D
%3965	JP	UGT,%3948	BD	39	48
%3968	LD	R36, 4	E6	24	04
%396B	DEC	R36	00	24	
%396D	JP	NZ,396B	ED	39	6B
%3970	LDE	@rr6,r3	92	36	
%3972	LDE	@rr8,r2	92	28	
%3974	LDE	@rr8,r0	92	08	
%3976	LDE	@rrA,r5	92	5A	

%3978	LDE	@rr8,r2	92	28	
%397A	LDE	@rr6,r4	92	46	
%397C	LDE	@rr8,r1	92	18	
%397E	LDE	rE,@rrA	82	EA	
%3980	LDE	@rr8,r2	92	28	
%3982	CP	R38,R60	A4	26	3E
%3985	JP	ULT,%3968	7D	39	68
%3988	LD	R241, %42	E6	F1	42
%398B	RET		AF		

The BASIC program resides in lines 10 to 140. It does the following.

1. Lines 10 to 30 set port A and port C high nibble to be input lines and port B and port C low nibble to be output lines. This part also sets the A/D outputs at zero volts and deactivates all the relays.
2. Line 40 sets the high- and low-voltage levels for detecting the QRS complex of the ECG signal. If the high-voltage level is set too high or too low, the program will not detect the QRS complex or the P wave and the T wave will be erroneously detected, respectively. The high-voltage level should be set below the height of QRS complex but above those of the P wave and the T wave. If only one voltage level for detecting the QRS complex is used, any noise will cause false counting of the QRS complex. The setting of the low-voltage level assures that the detected QRS complex has dropped completely.

3. Line 50 causes a 9600 Hz signal from timer 0 to be divided by 64 to produce a 150 Hz signal. It also loads C/T 1 with the value of 256 and holds C/T 1 off.
4. Lines 80 to 100 acquire a value from C/T 1, make a calculation (which will be described later), and print the heart rate on the CRT screen. Because the value from C/T 1 is 256 or less (assume the value is X), the actual value would be (256-X). The actual value is produced by a 150 HZ signal, so the real time interval is (256-X)/150. The detecting procedure counts the time interval of two consecutive QRS complexes. The true heart rate thus is 150*60/(256-X).
5. Lines 110 to 140 print instructions to the user to allow a repeat heart rate determination or program termination.

Memory locations %3900 to %398B hold the machine language subroutine. The subroutine does the following.

1. Locations %3900 to %3919 store the numbers that the subroutine uses for port setting and channel selection.
2. Locations %391A to %3944 check for a no-signal condition for 10 seconds. If no signal is detected the subroutine is exited, otherwise it continues.
3. Locations %3945 to %3967 turn C/T 1 on and then wait for the amplitude of the QRS complex to drop below the low-voltage level.
4. Locations %3968 to %398B check for another QRS complex to appear with its amplitude above the high-voltage level. If so, C/T 1 is turned off.

CONCLUSION

This single-board-computer data acquisition system has four A/D input channels, two D/A output channels, four SSRs, two EMRs, four sense switches, and a keypad. The A/D can be programmed as four single-ended channels or three pseudo-differential channels or two differential channels. The D/A output channel can supply 0-2.56 V. It should not be used as a control power supply. The four SSRs and two EMRs can be used to control other devices. The four sense switches and a keypad can be used as inputs to control the system.

This system uses the BASIC language for its programming. It can also be programmed in machine-language to increase execution speed. The system needs a power supply of +5 V to operate. Because a back-up battery is included, if the system is shut off, the RAM-stored data will be maintained. If a programming module (and its utility PROM) are used, you can copy an application program from RAM to EPROM. You can also copy the EPROM back to RAM for debugging purposes.

The techniques in building a computer system will improve with time. The size of the computer will decrease and its performance and numbers of functions will increase. The Z8 data acquisition system is compact and portable. Its capabilities are approximately the same as an older PDP-8-based data acquisition system in use in the Biomedical Engineering Program at Iowa State University.

The Z8 data acquisition system described illustrates the diversity of a single-board computer and its application to biomedical and clinical research. Even more importantly, it indicates the impact that

single-board computers are having on instrumentation development. The flexibility that can result from using a single-board computer is undeniable. It is easy to adapt a single-board computer-based instrument to a new circumstance by simply changing an EPROM. The potential also exists for reducing the time from project inception to project completion as a result of the reduced need for hardware design and fabrication.

BIBLIOGRAPHY

Basicon, Inc. 1984. MC-1Z Microcontroller. Basicon, Inc., 11895 N.W. Cornell Road, Portland, OR.

Bolton, M. P.; Taylor A. C. 1981. A universal computer and interface system for the disabled (unicaid). J. Biomedical Engineering 3(4):281-284.

Brown, A. W. S. 1981. A microprocessor-based ultrasonic limb movement monitoring system. J. Biomedical Engineering 3(4):275-279.

Demjaneko, V.; Sachs, F. 1982. Computer interface for electrophysiology applications: Simple modifications to a commercial (Datel) single-board data-acquisition system. J. Medical & Biological Engineering & Computing 20(1):65-69.

Dostinsky, I. A.; Christov, C. L.; Daskalov, I. K. 1985. A microprocessor-electrocardiograph. J. Medical & Biological Engineering & Computing 23(3):209-211.

El-Dhaher, A. H. G.; Kaouri, H. A.; Mustafa, K. Y. 1983. Microprocessor-based system for the measurement and analysis of an expiratory flow-volume curve. J. Medical & Biological Engineering & Computing 21(3):277-284.

Imperiale, Cosimo. 1983. Microprogrammed data acquisition system for renography. J. Clinical Engineering 8(3):235-241.

Milnor, William R. 1974. The electrocardiogram. Pages 883-891 in Vernon B. Mountcastle, ed. Medical physiology. 13th ed. The C. V. Mosby Company Press, Baltimore, Maryland.

Plexico, Perry S. 1980. Microcomputer applications in biomedical research. J. Medical Instrumentation 14(6):307-310.

Ramey, R. L.; Johnson, B. W.; Aylor, J. H. 1982. Microcomputer-based aid for the handicapped computer programmer. J. Medical & Biological Engineering & Computing 20(6):640-644.

Rolander, Clas I. 1984. Communication aids for the vocally handicapped using voice synthesis technology, an LCD text display and a single-chip microcomputer. M.S. thesis. Iowa State University, Ames.

Westenskow, Dwayne R.; Bowman, Robert J.; Ohlson, Kevin B.; Raemer, Daniel B. 1980. Microprocessors in intensive care medicine. J. Medical Instrumentation 14(6):311-313.

Wuthnow, Mark; Manoli, Samir; Schroder, Darrell. 1984. Microcomputer-based arrhythmia monitor. J. Clinical Engineering 9(4):291-298.

Zilog, Inc. 1980. Z8 PLZ/ASM: Assembly language programming manual. Zilog publication ref. no.: 03-3023-02. Zilog Inc., 1315 Dell Ave., Campbell, CA.

Zilog, Inc. 1981. Z8671 single-chip BASIC interpreter: BASIC/DEBUG software reference manual. Zilog publication ref. no.: 03-3149-02. Zilog, Inc., 1315 Dell Ave., Campbell, CA.

Zilog, Inc. 1983. Z8 single-chip microcomputers technical overview. Zilog Inc., 1315 Dell Ave., Campbell, CA.

Zilog, Inc. 1984. Z8 microcomputer: Technical manual. Zilog publication ref. no.: 03-3047-02. Zilog Inc., 1315 Dell Ave., Campbell, CA.

ACKNOWLEDGEMENTS

I would like to acknowledge the enthusiastic support offered by my major professor, Dr. Curran S. Swift, and I would like to thank him for his openness and willingness to help. I would also like to thank Dr. Richard E. Horton and Dr. David L. Carlson for serving as members of my committee.

In addition, I would like to extend my thanks to Mike A. Anderson and Dexter K. Ishii for the many learning experiences we shared together. I would also like to acknowledge the very important and constant encouragement that has been provided by my parents and wife.

Finally, I would like to acknowledge the much appreciated financial support that I have received from the Biomedical Engineering Program and Iowa State University.

APPENDIX: MANUFACTURERS OF SINGLE-BOARD 'BASIC' COMPUTERS

1. Antona Corporation. 2100 S Sawtelle Blvd., Suite 205, West Los Angeles, CA 90025 (213) 473-8995 (I[1])
2. Basicon, Inc. 11895 NW Cornell Road, Portland, OR 97229 (503) 626-1012 (I, N[2], Z[3])
3. Digi-Key Corporation. 701 Brooks Avenue South, P.O. Box 677, Thief River Falls, MN 56701 (218) 681-6674 (N)
4. Global Automation. 2829 Lewis Lane, Owensboro, KY 42301 (502) 683-9871 (I)
5. H. H. S. Microcontrollers. 5876 Old State Road, Edinboro, PA 16412 (814) 734-4338 (Z)
6. Lehmann & Associates. P.O. Box 566, Maumee, OH 43537 (419) 891-0687 (Z)
7. Messand Microprocessor Engineering. 1509 Francis Street, Albany, CA 94706 (415) 526-5155 (Z)
8. Micro Linear Controls. 4900 Memco Lane, Racine, WI 53404 (414) 639-1105 (I)
9. Micromint, Inc. 561 Willow Avenue, Cedarhurst, NY 11516 (516) 374-6973 (I, Z)

---

[1]I = Intel 8052AH-BASIC product.

[2]N = National Semiconductor INS8073 product.

[3]Z = Zilog Z8671 product.

10. Octagon Systems Corporation. 6501 W 91st Avenue, Westminster, CO 80030 (303) 426-8540 (N)

11. Schulz Enterprises Inc. 1285 Las Tunas Drive, San Gabriel, CA 91776 (818) 287-5067 (Z)

12. Tech Star Laboratory. Suite 709, R&B Corporation Park, 1701 N. Greenville Avenue, Richardson, TX 75081 (I)

13. Transwave Corporation. Cedar Valley Building, Box 489, Vanderbilt, PA 15486 (412) 628-6370 (N)