ANALOG-TO-DIGITAL CONVERSION TECHNIQUES WITH THE M6800 MICROPROCESSOR SYSTEM

This application note describes several analog-to-digital conversion systems implemented with the M6800 microprocessor and external linear and digital IC's. Systems consisting of an 8- and 10-bit successive approximation approach, as well as dual ramp techniques of 3½- and 4½-digit BCD and 12-bit binary, are shown with flow diagrams, source programs and hardware schematics. System tradeoffs of the various schemes and programs for binary-to-BCD and BCD-to-7 segment code are discussed.

Analog-To-Digital Conversion Techniques with the M6800 Microprocessor System

INTRODUCTION

The MPU (microprocessing unit) is rapidly replacing both digital and analog circuitry in the industrial control environment. It provides a convenient and efficient method of handling data; controlling valves, motors and relays; and in general, supervising a complete processing machine. However, much of the information required by the MPU for the various computations necessary in the processing system may be available as analog input signals instead of digitally formatted data. These analog signals may be from a pressure transducer, thermistor or other type of sensor. Therefore, for analog data an A/D (analog-to-digital) converter must be added to the MPU system.

Although there are various methods of A/D conversion, each system can usually be divided into two sections – an analog subsystem containing the various analog functions for the A/D and a digital subsystem containing the digital functions. To add an A/D to the MPU, both of the sections may be added externally to the microprocessor in the form of a PC card, hybrid module or monolithic chip. However, only the analog subsystem of the A/D need be added to the microprocessor, since by adding a few instructions to the software, the MPU can perform the function of the digital section of the A/D converter in addition to its other tasks. Therefore, a system design that already contains an MPU and requires analog information needs only one or two additional inexpensive analog components to provide the A/D. The microprocessor software can control the analog section of the A/D, determine the digital value of the analog input from the analog section, and perform various calculations with the resulting data. In addition, the MPU can control several analog A/D sections in a timeshare mode, thus multiplexing the analog information at a digital level.

3

Using the MPU to perform the tasks of the digital section provides a lower cost approach to the A/D function than adding a complete A/D external to the MPU. The information presented in this note describes this technique as applied to both successive approximation (SA) A/D and dual ramp A/D. With the addition of a DAC (digital-to-analog converter), a couple of operational amplifiers, and the appropriate MPU software, an 8- or 10-bit successive approximation A/D is available. Expansion to greater accuracies is possible by modifying the software and adding the appropriate D/A converter. The technique of successive approximation A/D provides medium speed with accuracies compatible with many systems. The second technique adds an MC1405 dual ramp analog subsystem to the MPU system and, if desired, a digital display to produce a 12-15 bit binary or a 3½- or 4½-digit BCD A/D conversion with 7-segment display readout. This A/D technique has a relatively slow conversion rate but produces a converter of very high accuracy. In addition to the longer conversion time, the MPU must be totally devoted to the A/D function during the conversion period. However, if maximum speed is not required this technique of A/D allows an inexpensive and practical method of handling analog information.

Figure 1 shows the relative merits of each A/D conversion technique. Listed in this table are conversion time, accuracy and whether interrupts to the MPU are allowed during the conversion cycle.

This note describes each method listed in Figure 1 and provides the MPU software and external system hardware schematics along with an explanation of the basic A/D technique and system peculiarities. In addition, the MPU interface connections for the external A/D hardware schemes are shown. These schemes are a complete 8-bit successive approximation and a 3½-digit dual ramp A/D system, both of which externally perform the conversion and transfer the digital data into the MPU system through a PIA.

For additional information on the MC6800 MPU system or A/D systems, the appropriate data sheets or other available literature should be consulted.

MPU

The Motorola microprocessor system devices used are the MC6800 MPU, MCM6810 RAM, MCM6830 ROM and MC6820 PIA (peripheral interface adapter). The following is a brief description of the basic MPU system as it pertains to the A/D systems presented later in this application note.

The Motorola MPU system uses a 16-bit address bus and an 8-bit data bus. The 16-bit address bus provides 65,536 possible memory locations which may be either storage devices (RAM, ROM, etc.) or interface devices (PIA, etc.). The basic MPU contains two 8-bit accumulators, one 16-bit index register, a 16-bit program counter, a 16-bit stack pointer, and an 8-bit condition code register. The condition code register indicates carry, half carry, interrupt, zero, minus, and 2's complement overflow. Figure 2 shows a functional block of the MC6800 MPU.

The MPU uses 72 instructions with six addressing modes which provide 197 different operations in the MPU. A summary of each instruction and function with the appropriate addressing mode is shown in Appendix A of this note.

Characteristic	Successive Approximation			Dual Ramp			
	8-Bit Software	10-Bit Software	8-Bit Hardware	12-Bit Software	3½-Digit Software	4½-Digit Software	3½-Digit Hardware
External Hardware	8-Bit DAC Op Amp Comparator	10-Bit DAC Op Amp Comparator	8-Bit DAC SAR* Op Amp Comparator	MC1405	MC1405	MC1405	MC1405 MC14435 MC14558 (for 7-segment display)
Conversion Rate	700 µs Constant	1.25 ms Constant	60 µs for MPU, plus A/D Conversion Time	165 ms (max) Variable	60 ms (max) Variable	600 ms (max) Variable	183 µs (min) for MPU, plus A/D Conversion Time
Interrupt Capability	Allowed	Allowed	Allowed	Not Allowed	Not Allowed	Not Allowed	Allowed
Number of Memory Locations Required (Including PIA Configuration)	106	145	42	84	296	328	58
Serial Output Available	Yes	Yes	Yes	No	No	No	No

*Successive Approximation Register

FIGURE 1 – Relative Merits of A/D Conversion Techniques

The RAMs used in the system are static and contain 128 8-bit words for scratch pad memory while the ROM is mask programmable and contains 1024 8-bit words. The ROM and RAM, along with the remainder of the MPU system components, operate from a single +5 volt power supply; the address bus, data bus and PIAs are TTL compatible.

The MPU system requires a 2ϕ non-overlapping clock with a lower frequency limit of 100 kHz and an upper limit of 1 MHz.

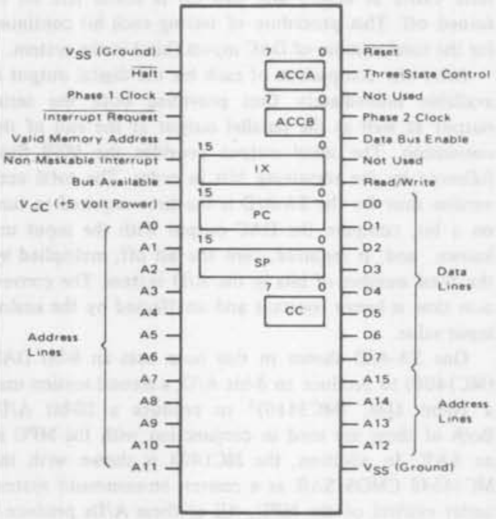


FIGURE 2 – MPU Pin Functions

The PIA is the interface device used between the address and data buses and the analog sections of the A/D. Each PIA contains two essentially identical 8-bit interface ports. These ports (A side, B side) each contain three internal registers that include the data register which is the interface from the data bus to the A/D, the data direction register which programs each of the eight lines of the data register as either an input or an output, and the control register which, in addition to other functions, switches the data bus between the data register and the data direction register. Each port to the PIA contains two addition pins, CA1 and CA2, for interrupt capability and extra I/O lines. The functions of these lines are programmable with the remaining bits in the control register. Figure 3 shows a functional block of the MC6820 PIA.

Each PIA requires four address locations in memory. Two addresses access either of the two (A or B sides) data/data direction registers while the remaining two addresses access either of the two control registers. These addresses are decoded by the chip select and register select lines of the PIA which are connected to the MPU address bus. Selection between the data register and data direction register is made by programming a "1" or "0" in the third least significant bit of each control register . A logic "0" accesses the data direction register while a logic "1" accesses the data register.

By programming "0"s in the data direction register each corresponding line performs as an input, while "1"s in the data direction register make corresponding lines act as outputs. The eight lines may be intermixed between inputs and outputs by programming different combinations of "1"s and "0"s into the data direction register. At the beginning of the program the I/O configuration is programmed into the data direction register, after which the control register is programmed to select the data register for I/O operation.

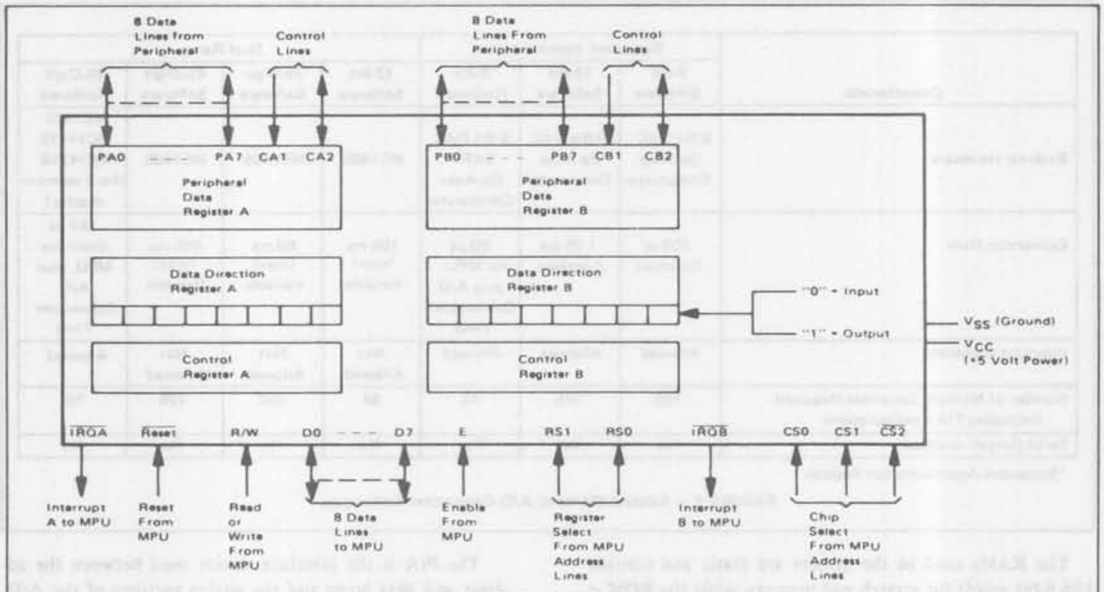


FIGURE 3 – PIA Functions

The printouts shown for each A/D program are the source instructions for the cross assembler from the Motorola timeshare. Since the MPU contains a 16-bit address bus and an 8-bit data bus, the hexadecimal number system provides a convenient representation of these numbers. Although the assembler output is in hexadecimal, the source input may be either binary, octal, decimal or hexadecimal. A dollar sign ($) preceding a number in the source instructions indicates hexadecimal, a percent sign (%) indicates binary and an at sign (@) indicates octal. No prefix indicates the decimal number system.

Only the beginning addresses of the program and labels are shown in the source programs. These beginning addresses may be changed prior to assembling the total system program or the programs may be relocated after assembly with little or no modification.

SUCCESSIVE APPROXIMATION TECHNIQUES

General

One of the more popular methods of A/D conversion is that of successive approximation. This technique uses a DAC (digital-to-analog converter) in a feedback loop to generate a known analog signal to which the unknown analog input is compared. In addition to medium speed conversion rates, it has the advantages of providing not only a parallel digital output after the conversion is completed but also the serial output during the conversion.

Figure 4 shows the block diagram and waveform of the SA-A/D. The DAC inputs are controlled by the successive approximation register (SAR) which is, as presented here, the microprocessor. The DAC output is compared to the analog input (V_{in}) by the analog comparator and its output controls the SAR. At the start of a conversion the MSB of the DAC is turned on by the SAR, producing an output from the DAC equal to half of the full scale value. This output is compared to the analog input and if the DAC output is greater than the input unknown, the SAR turns the MSB off. However, if the DAC output is less than the input unknown, the MSB remains on. Following the trial of the MSB the next most significant bit is turned on and again the comparison is made between the DAC output and the input unknown. The same criteria exists as before and this bit is either left on or turned off. This procedure of testing each bit continues for the total number of DAC inputs (bits) in the system.

After the comparison of each bit the digital output is available immediately thus providing both the serial output as well as the parallel output at the end of the conversion. The serial output provides the MSB first, followed by the remaining bits in order. The total conversion time for the SA-A/D is the time required to turn on a bit, compare the DAC output with the input unknown and, if required, turn the bit off, multiplied by the total number of bits in the A/D system. The conversion time is hence constant and unaffected by the analog input value.

One SA-A/D shown in this note uses an 8-bit DAC (MC1408) to produce an 8-bit A/D; a second version uses a 10-bit DAC (MC3410)* to produce a 10-bit A/D. Both of these are used in conjunction with the MPU as an SAR. In addition, the MC1408 is shown with the MC14549 CMOS SAR as a convert-on-command system under control of the MPU. All of these A/Ds produce a binary output. However, by adding the appropriate software a BCD output or 7-segment-display outputs are available. Also by using a BCD-weighted DAC, the BCD output can be produced directly.

*MC3410 to be announced

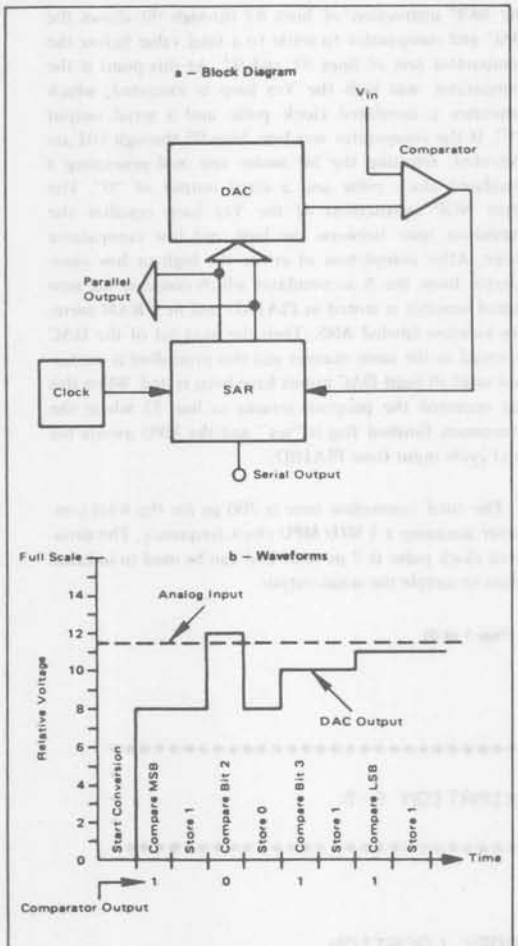


FIGURE 4 – 4-Bit Successive Approximation Converter

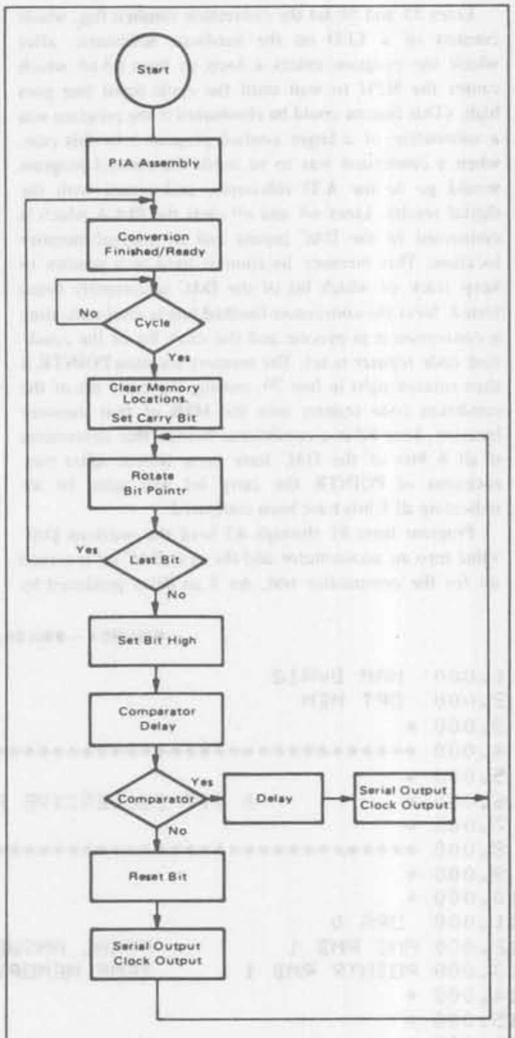


FIGURE 5 – 8-Bit Successive Approximation A/D Flow Diagram

8-Bit SA Program

The flow chart for the 8-bit MPU A/D system is shown in Figure 5; Figures 6 and 7 show the software and the hardware external to the microprocessor. The DAC used is the MC1408L-8 which has active high inputs and a current sink output. An uncompensated MLM301A operational amplifier is used as a comparator while an externally compensated MLM301A or internally compensated MC1741 operational amplifier is used as a buffer amplifier for the input voltage. The output voltage compliance of the DAC is ±0.5 volt; if the current required by the D/A does not match that produced from the output of the buffer amplifier through R1 and R2, then the DAC output will saturate at 0.5 volt above or below ground, thus toggling the comparator. The system is calibrated by adjusting R1 for 1 volt full scale, and zero calibration is set by adjusting R3.

The first MPU instruction for the 8-bit A/D is in line 45 of Figure 6. After assembly, this instruction will be placed in memory location $0A00 as defined in the assembler directive of line 42. The assembled code for this program is relocatable in memory as long as the PIA addresses and storage addresses are unchanged. The program as shown requires 106 memory bytes. Source program lines 45 through 53 configure the PIAs for the proper input/output configuration. PIA1BD is used for various control functions between the MPU system and the external hardware. The exact configuration of this PIA is shown in lines 28 through 33 of Figure 6. PIA1AD provides the 8-bit output needed for the DAC. Lines 51 through 53 set bit 3 of the PIA control register to access the data register for the actual A/D program.

Lines 55 and 56 set the conversion finished flag, which consists of a LED on the hardware schematic, after which the program enters a loop in lines 63-65 which causes the MPU to wait until the cycle input line goes high. (This feature could be eliminated if the program was a subroutine of a larger control program.) In this case, when a conversion was to be made the control program would go to the A/D subroutine and return with the digital results. Lines 68 and 69 clear the PIA-A which is connected to the DAC inputs and an internal memory location. This memory location is used as a pointer to keep track of which bit of the DAC is currently being tested. Next the conversion finished line is reset indicating a conversion is in process and the carry bit of the condition code register is set. The memory location POINTR is then rotated right in line 79, moving the carry bit of the condition code register into the MSB of that memory location. Line 80 is a conditional branch that determines if all 8 bits of the DAC have been tested. After nine rotations of POINTR the carry bit will again be set indicating all 8 bits have been compared.

Program lines 81 through 83 load the previous DAC value into an accumulator and the next DAC bit is turned on for the comparator test. An 8 μs delay produced by the NOP instruction of lines 87 through 90 allows the DAC and comparator to settle to a final value before the comparator test of lines 91 and 92. At this point if the comparator was high the Yes loop is executed, which generates a simulated clock pulse and a serial output "1". If the comparator was low, lines 95 through 101 are executed, resetting the bit under test and generating a simulated clock pulse and a serial output of "0". The three NOP instructions of the Yes loop equalize the execution time between the high and low comparator loops. After completion of either the high or low comparator loop, the A accumulator which contains the new digital number is stored in PIA1AD and in a RAM memory location labeled ANS. Then the next bit of the DAC is tested in the same manner and this procedure is continued until all eight DAC inputs have been tested. When this has occurred the program returns to line 55 where the conversion finished flag is "set" and the MPU awaits the next cycle input from PIA1BD.

The total conversion time is 700 μs for the 8-bit converter assuming a 1 MHz MPU clock frequency. The simulated clock pulse is 7 μs wide and can be used to indicate when to sample the serial output.

FIGURE 6 — 8-Bit SA Software (Page 1 of 3)

```
 1.000  NAM DWA12
 2.000  OPT MEM
 3.000 *
 4.000 ***********************************************************
 5.000 *                                                         *
 6.000 *       8 BIT SUCCESSIVE APPROXIMATION A/D                *
 7.000 *                                                         *
 8.000 ***********************************************************
 9.000 *
10.000 *
11.000  ORG 0
12.000 ANS RMB 1          FINAL ANSWER MEMORY LOCATION
13.000 POINTR RMB 1       TEMP MEMORY LOCATION
14.000 *
15.000 *
16.000 *
17.000  ORG $4004         PIA MEMORY ADDRESSES
18.000 PIA1AD RMB 1       A SIDE, DATA REGISTER
19.000 PIA1AC RMB 1       A SIDE, CONTROL REGISTER
20.000 PIA1BD RMB 1       B SIDE, DATA REGISTER
21.000 PIA1BC RMB 1       B SIDE, CONTROL REGISTER
22.000 *
23.000 *                  PIA1AD USED FOR DIGITAL OUTPUT TO DAC
24.000 *                  PIA1BD USED FOR A/D CONTROL
25.000 *
26.000 *
27.000 *
28.000 ****************PIA1BD PIN CONNECTIONS****************
29.000 *******************************************************
30.000 * PB7   * PB6 *  PB5 * PB4 *  PB3 * PB2 *  PB1 *  PB0 *
31.000 *******************************************************
32.000 * COMP  *  NC *  SC  * CF  *  SD  * NC  * CYCLE * NC  *
33.000 *******************************************************
34.000 *                             *
35.000 *
```

FIGURE 6 – 8-Bit SA Software (Page 2 of 3)

```
*
* COMP-COMPARATOR,SC-SIMULATED CLOCK,SD-SERIAL OUTPUT
* CF-CONVERSION FINISHED, NC-NO CONNECTION
*
*
*
 ORG $0A00          BEGINNING ADDRESS
*
*                            **PIA ASSEMBLY**
 CLR PIA1AC
 CLR PIA1BC
 LDA A #$7C
 STA A PIA1BD
 LDA A #$0FF
 STA A PIA1AD       A SIDE ALL OUTPUTS
 LDA A #$04
 STA A PIA1AC
 STA A PIA1BC
*
RSTART LDA A #$10
 STA A PIA1BD       SET CONVERSION FINISHED
*
*
*
*                            **CYCLE TEST**
*
*
CYCLE LDA A PIA1BD
 AND A #$02
 BEQ CYCLE
*
*
 CLR PIA1AD
 CLR POINTR
*
*
*
 CLR PIA1BD         RESET CONVERSION FINISHED
 SEC
*
*
*
*
CONVRT ROR POINTR
 BCS RSTART
 LDA A PIA1AD       RECALL PREVIOUS DIGITAL OUTPUT
 ADD A POINTR
 STA A PIA1AD       SET NEW DIGITAL OUTPUT
*
*                            **DELAY FOR COMPARATOR**
*
 NOP
 NOP
 NOP
 NOP
 LDA A PIA1BD       COMPARATOR TEST
 BMI YES
*
*                            **LOW COMPARATOR LOOP**
 LDA A PIA1AD
 SUB A POINTR
 LDA B #$20      SERIAL OUT OF "0", CLOCK SET
 STA B PIA1BD
 CLR B           CLOCK RESET
 STA B PIA1BD
 BRA END
*
*                                   **HIGH COMPARATOR LOOP**
YES LDA A PIA1AD
 NOP
 NOP             DELAY
 NOP
 LDA B #$28      SERIAL OUTPUT OF "1", CLOCK SET
 STA B PIA1BD
 LDA B #$08      CLOCK RESET
 STA B PIA1BD
*
END STA A PIA1AD
 STA A ANS
 BRA CONVRT
*
*
*
*
*
*
 MON
```

FIGURE 6 – 8-Bit SA Software (Page 3 of 3)

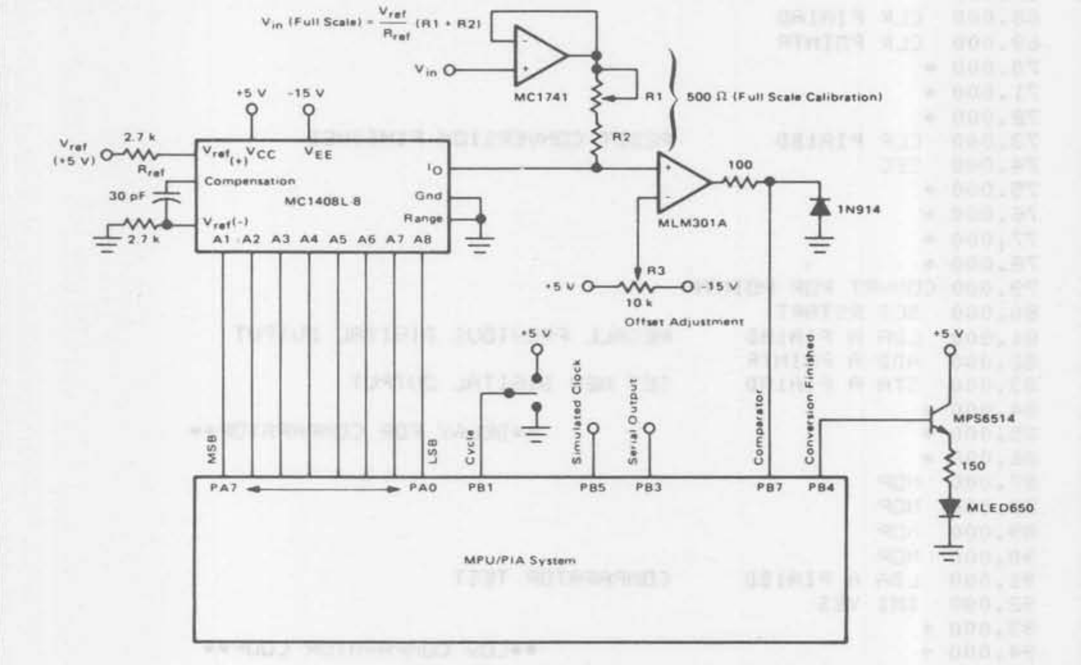


FIGURE 7 – 8-Bit SA Hardware

10-Bit SA Program

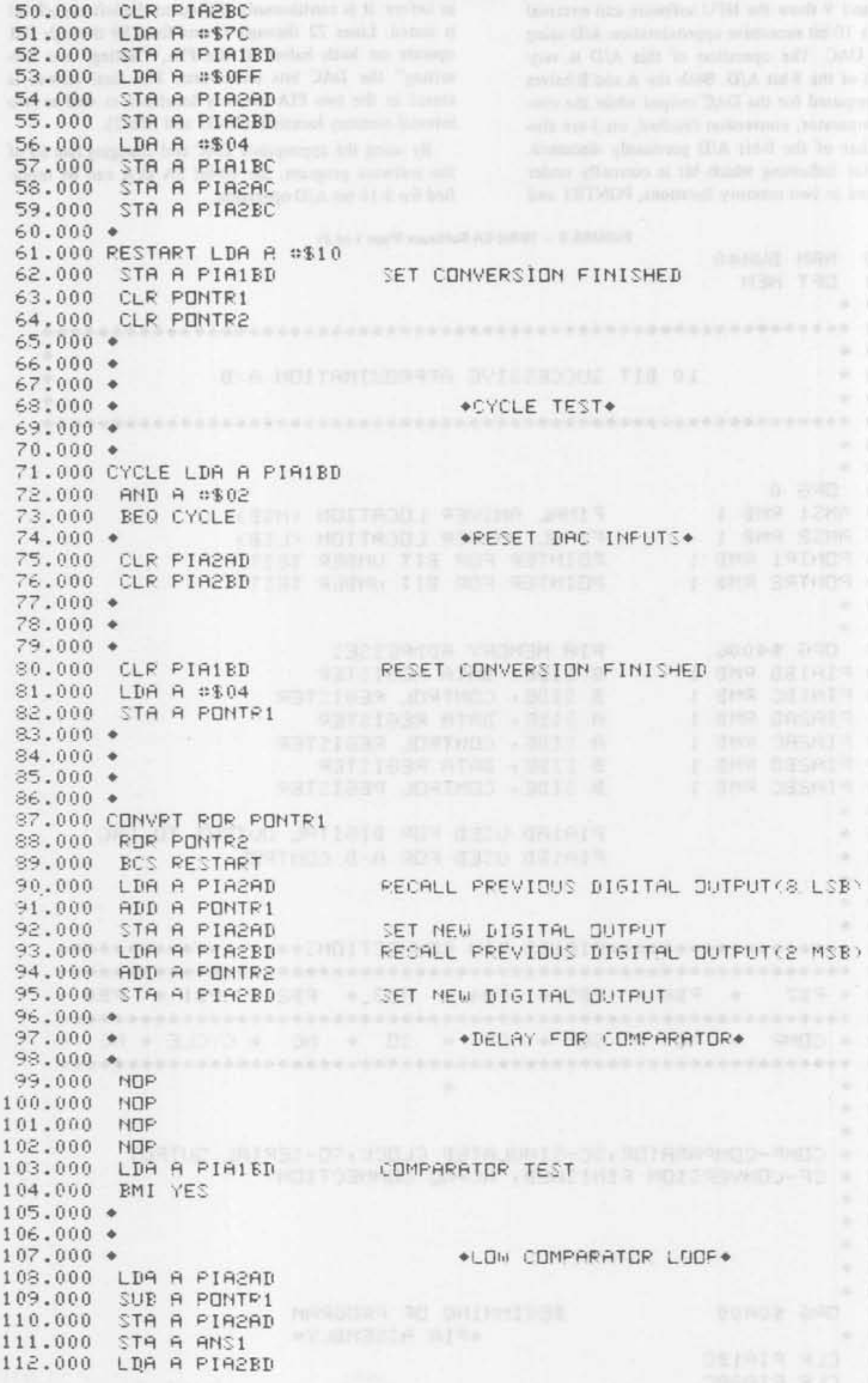
Figures 8 and 9 show the MPU software and external hardware for a 10-bit successive approximation A/D using the MC3410 DAC. The operation of this A/D is very similar to that of the 8-bit A/D. Both the A and B halves of a PIA are required for the DAC output while the control lines (comparator, conversion finished, etc.) are also identical to that of the 8-bit A/D previously discussed. The pointer for indicating which bit is currently under test is contained in two memory locations, PONTR1 and PONTR2. The pointer is initialized in lines 63 and 64 and as before, it is continuously shifted to the left as each bit is tested. Lines 72 through 77 and lines 89 through 101 operate on both halves of the PIA, "setting" and "resetting" the DAC bits under test. The final answer is stored in the two PIA memory locations as well as two internal memory locations (ANS1 and ANS2).

By using the appropriate DAC and changing line 63 of the software program, the 10-bit SA D/A can be modified for 9-16 bit A/D operation.

FIGURE 8 – 10-Bit SA Software (Page 1 of 3)

```
 1.000  NAM DWA40
 2.000  OPT MEM
 3.000 *
 4.000 ***********************************************************
 5.000 *                                                         *
 6.000 *           10 BIT SUCCESSIVE APPROXIMATION A/D           *
 7.000 *                                                         *
 8.000 ***********************************************************
 9.000 *
10.000 *
11.000  ORG 0
12.000 ANS1 RMB 1         FINAL ANSWER LOCATION (MSB)
13.000 ANS2 RMB 1         FINAL ANSWER LOCATION (LSB)
14.000 PONTR1 RMB 1       POINTER FOR BIT UNDER TEST
15.000 PONTR2 RMB 1       POINTER FOR BIT UNDER TEST
16.000 *
17.000 *
18.000  ORG $4006         PIA MEMORY ADDRESSES
19.000 PIA1BD RMB 1       B SIDE, DATA REGISTER
20.000 PIA1BC RMB 1       B SIDE, CONTROL REGISTER
21.000 PIA2AD RMB 1       A SIDE, DATA REGISTER
21.500 PIA2AC RMB 1       A SIDE, CONTROL REGISTER
22.000 PIA2BD RMB 1       B SIDE, DATA REGISTER
23.000 PIA2BC RMB 1       B SIDE, CONTROL REGISTER
24.000 *
25.000 *                  PIA1AD USED FOR DIGITAL OUTPUT TO DAC
26.000 *                  PIA1BD USED FOR A/D CONTROL
27.000 *
28.000 *
29.000 *
30.000 *****************PIA1BD PIN CONNECTIONS*****************
31.000 *********************************************************
32.000 * PB7   * PB6 * PB5 * PB4 * PB3.* PB2 * PB1 * PB0 *
33.000 *********************************************************
34.000 * COMP  * NC  * SC  * CF  *  SD  *  NC  * CYCLE * NC *
35.000 *********************************************************
36.000 *                                   *
37.000 *
38.000 *
39.000 * COMP-COMPARATOR,SC-SIMULATED CLOCK,SD-SERIAL OUTPUT
40.000 * CF-CONVERSION FINISHED, NC-NO CONNECTION
41.000 *
42.000 *
43.000 *
44.000 *
45.000 *
46.000  ORG $0A00            BEGINNING OF PROGRAM
47.000 *                             *PIA ASSEMBLY*
48.000  CLR PIA1BC
49.000  CLR PIA2AC
```

FIGURE 8 – 10-Bit SA Software (Page 2 of 3)



```
 CLR PIA2BC
 LDA A #$7C
 STA A PIA1BD
 LDA A #$0FF
 STA A PIA2AD
 STA A PIA2BD
 LDA A #$04
 STA A PIA1BC
 STA A PIA2AC
 STA A PIA2BC
◆
RESTART LDA A #$10
 STA A PIA1BD          SET CONVERSION FINISHED
 CLR PONTR1
 CLR PONTR2
◆
◆
◆
◆                            ◆CYCLE TEST◆
◆
◆
CYCLE LDA A PIA1BD
 AND A #$02
 BEQ CYCLE
◆                            ◆RESET DAC INPUTS◆
 CLR PIA2AD
 CLR PIA2BD
◆
◆
◆
 CLR PIA1BD            RESET CONVERSION FINISHED
 LDA A #$04
 STA A PONTR1
◆
◆
◆
◆
CONVRT ROR PONTR1
 ROR PONTR2
 BCS RESTART
 LDA A PIA2AD          RECALL PREVIOUS DIGITAL OUTPUT(8 LSB)
 ADD A PONTR1
 STA A PIA2AD          SET NEW DIGITAL OUTPUT
 LDA A PIA2BD          RECALL PREVIOUS DIGITAL OUTPUT(2 MSB)
 ADD A PONTR2
 STA A PIA2BD          SET NEW DIGITAL OUTPUT
◆
◆                            ◆DELAY FOR COMPARATOR◆
◆
 NOP
 NOP
 NOP
 NOP
 LDA A PIA1BD          COMPARATOR TEST
 BMI YES
◆
◆
◆                            ◆LOW COMPARATOR LOOP◆
 LDA A PIA2AD
 SUB A PONTR1
 STA A PIA2AD
 STA A ANS1
 LDA A PIA2BD
```

```
113.000  SUB A PONTR2
114.000  STA A PIA2BD
115.000  STA A ANS2
116.000  LDA B #$20              SERIAL OUTPUT (CLOCK ONLY)
117.000  STA B PIA1BD
118.000  CLR B                   CLOCK RESET
119.000  STA B PIA1BD
120.000  BRA END
121.000 *
122.000 *
123.000 *                            *HIGH COMPARATOR LOOP*
124.000 YES LDA A #$05           TIME EQUALIZATION
125.000 DELAY DEC A
126.000  BNE DELAY
127.000  LDA B #$28              SERIAL OUTPUT
128.000  STA B PIA1BD
129.000  LDA B #$08              CLOCK RESET
130.000  STA B PIA1BD
131.000  NOP
132.000  NOP
133.000 *
134.000 END BRA CONVRT
135.000 *
136.000 *
137.000 *
138.000  MON
```

FIGURE 8 – 10-Bit SA Software (Page 3 of 3)

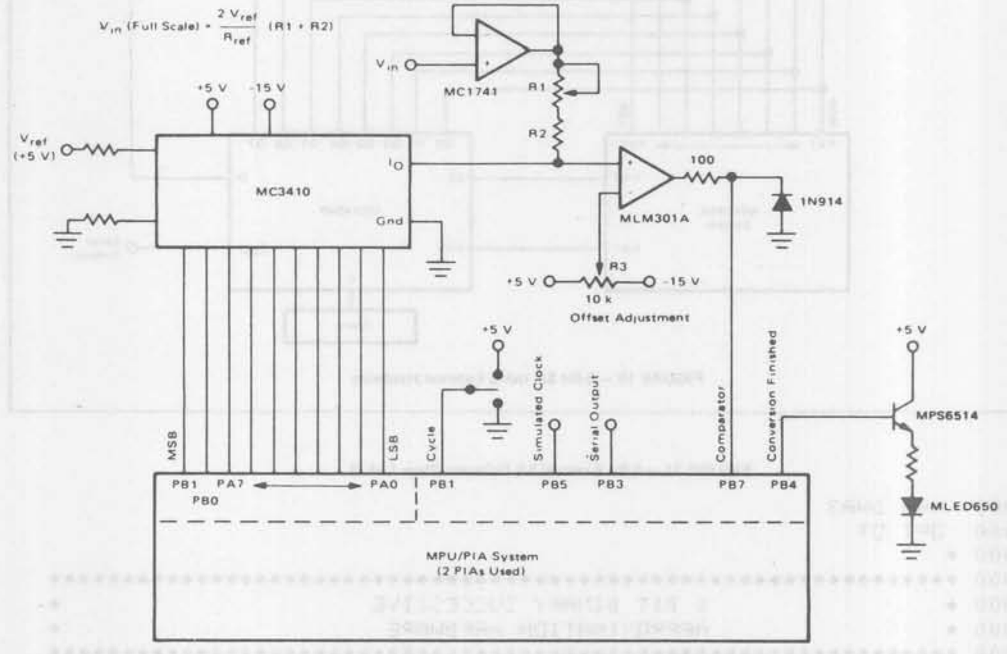


FIGURE 9 – 10-Bit SA Hardware

External SA System

The third successive approximation program, shown in Figures 10 and 11, uses an MC1408 DAC with the MC14549 CMOS SAR for a convert-on-command A/D system. This system is controlled by the MPU through the CA1 and CA2 PIA pins to start a conversion and store the results of this conversion in memory when the conversion is finished. The 8-bit data word from the A/D is brought in to the MPU system through PIA1AD. The advantages of this A/D system are that a minimum number of software instructions are required, a higher speed conversion is possible, and the MPU may be performing other tasks during the conversion. The disadvantage is a higher parts count and increased cost.

The program for this A/D, shown in Figure 11, is written as a subroutine of a larger program. This larger program is simulated with the instructions of lines 28 through 31. The subroutine starts in line 34, unmasking the interrupt input on CA1 and setting CA2 high. (For additional information on use of the CA1 and CA2 lines, see the MC6820 data sheet.) CA2 initiates the conversion. Line 35 is a dummy read statement necessary to clear the data register of the interrupt bit associated with the CA1 input line. Then a wait for interrupt instruction stores the stack in anticipation of the A/D conversion being completed. When the conversion is finished the CA1 line is toggled by the EOC output of the MC14549 and the program goes to line 43 where CA1 is masked and CA2 is set low, thus stopping any further conversion sequences by the A/D. The digital results are loaded into the A accumulator through PIA-A and stored in memory location TEMP. Then the MPU returns from the interrupt and finally returns from the subroutine.

The entire sequence requires 60 μs plus the conversion time of the A/D.

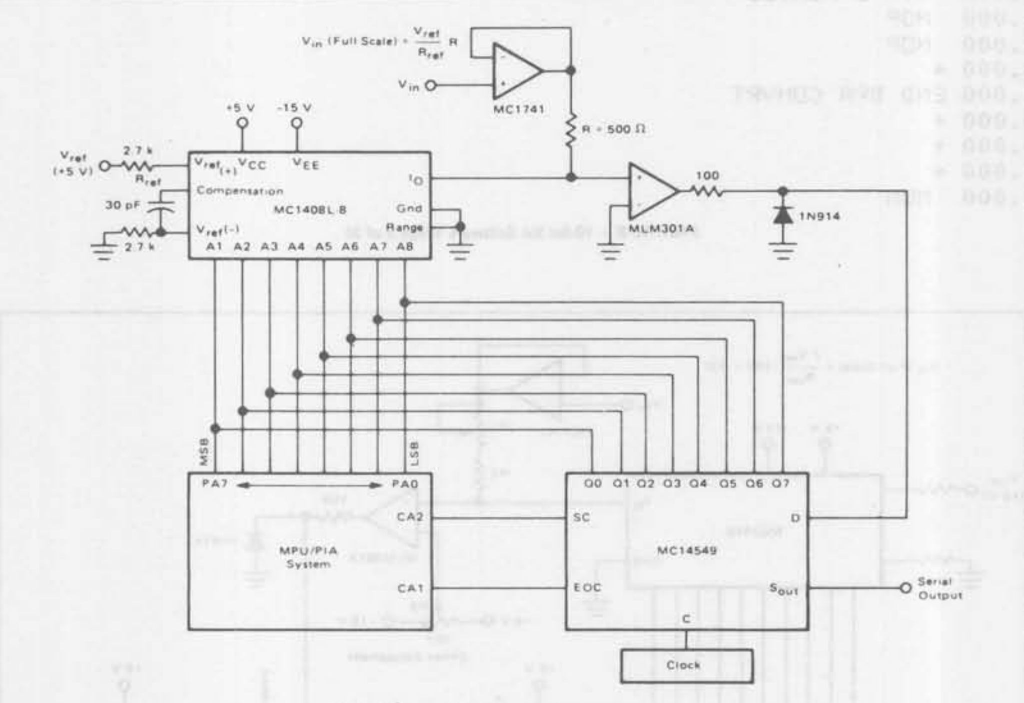


FIGURE 10 – 8-Bit SA Using External Hardware

FIGURE 11 – 8-Bit External SA Software (Page 1 of 2)

```
1.000  NAM DWA3
2.000  OPT OT
3.000 *
4.000 ************************************************************
5.000 *                  8 BIT BINARY SUCCESSIVE                   *
6.000 *                  APPROXIMATION HARDWARE                    *
7.000 ************************************************************
8.000 *
9.000  ORG $0100
```

```
TEMP RMB 1                    8 BIT BINARY DATA
*
*
 ORG $4004
PIA1AD RMB 1                  DATA REGISTER
PIA1AC RMB 1                  CONTROL REGISTER
*
*
*
*
 ORG $0300
 CLR PIA1AC                   PIA ASSEMBLY
 CLR PIA1AD
 LDA A #$3C
 STA A PIA1AC
 LDS #$0020
*
*
 NOP
 JSR CONVRT
END NOP
 BRA END
*
*                          CONVERSION SUBROUTINE
CONVRT LDA A #$3F              CA1 UNMASKED,POS EDGE--CA2 HIGH
 LDA B PIA1AD
 STA A PIA1AC
 WAI
 RTS
*
*
*
*                             INTERRUPT PROGRAM
INTRPT LDA A #$36               CA1 MASKED-CA2 LOW
 STA A PIA1AC
 LDA A PIA1AD
 STA A TEMP
 RTI
*
*
*
 MON
```

FIGURE 11 – 8-Bit External SA Software (Page 2 of 2)

DUAL RAMP TECHNIQUES

General

Another commonly used method for A/D conversion is the dual ramp or dual slope technique. This approach has a longer conversion time than that of the successive approximation method. The conversion time period is also variable and input voltage dependent. However, this method yields an A/D converter of high accuracy and low cost.

As the name implies the dual ramp method consists of two ramp periods for each conversion cycle. Figure 12 shows the basic waveforms for the dual ramp A/D. The ratio in time of the ramp lengths provides a value representing the difference between a reference and an unknown voltage. During time period T1, the input unknown is integrated for a fixed time period (fixed number of clock cycles). The integrator voltage increases from the reference level to a voltage which is proportional to the input voltage. At the end of this time period a reference voltage is applied to the input of the integrator causing the integrator output voltage to decrease until the reference level is again reached. The number of clock cycles that are required to bring the integrator output voltage back to the reference level is proportional to the input unknown voltage.

The dual ramp converters discussed here use the MC1405 analog subsystem in conjunction with the M6800 MPU system. The MC1405 provides the integrator, comparator and reference voltage required for the analog functions of the dual ramp A/D. The analog device also adds an offset current to the integrator input during the ramp up time period to stabilize small voltage readings. The digital section of the A/D must subtract an equivalent number of counts to produce a zero reading display output for a zero input. The interface between the analog and digital subsystems consists of two control lines. These are the comparator output from the analog part, which indicates whether the ramp is above or below the reference level, and a ramp control output from the digital part to switch the integrator input between the input unknown voltage and the reference voltage. The control of these lines, offset subtraction, and calculations with the resulting data must be handled by the digital subsystem, which in this case is the MPU.

For additional information on the dual ramp technique for A/D, consult the data sheet for the MC1405.

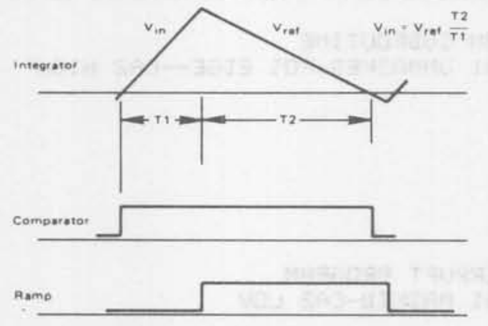


FIGURE 12 – Dual Ramp Waveforms

12-Bit Dual Ramp Program

This version of the dual ramp A/D generates a 12-bit binary output from a 1 volt full scale analog input. Figures 13, 14 and 15 show the flow chart, MPU software and external hardware. The interface of the PIAs used for this A/D is shown both on the schematic and in lines 16 through 22 of the source program. Lines 25 and 26 indicate the two memory locations where the final 12-bit binary result is stored. These locations are $0000 and $0001. The four most significant bits are in location $0000 while the remaining eight bits are in $0001.

Referring to the software of Figure 14, the first instructions (lines 37 through 42) initialize the PIA for its input/output configuration. Source program lines 46 through 49 set the ramp control line of the MC1405 and check the comparator output from the MC1405 to insure that the integrator output is below the reference level at the start of a conversion. Next the "conversion finished" flag is set indicating a conversion ready status. Then the MPU enters a loop (lines 55 through 57) waiting for a cycle input (PB1) from the PIA. When this condition occurs the conversion finished flag is reset while the ramp control line (PB2) goes low, thus starting a conversion cycle. In addition, the index register has been loaded with $2000 which will be decremented to provide the ramp up timing period. When the ramp crosses the threshold level the comparator (PB7) change from low to high causes the MPU to enter the timing cycle of lines 67 through 69. The index register is continuously decremented until reaching zero, at which point the ramp control line (PB2) to the MC1405 is set high (line 74) and the index register is incremented (line 75). This loop continues until the integrator output again reaches the threshold level. Line 76 of the ramp down cycle is a dummy statement included to equalize the timing between the ramp up and ramp down time periods. The proper timing ratio (2:1 in this example) must be maintained for correct A/D operation.

After the termination of the ramp down time period the content of the index register is stored in memory locations $0000 and $0001 (line 82). Next the offset counts are subtracted (512_{10}) from this result by subtracting $01 from memory location $0000. The result is

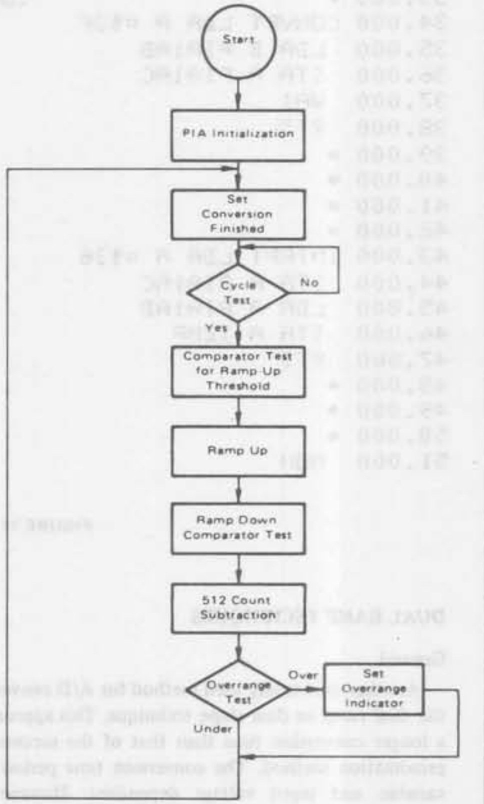


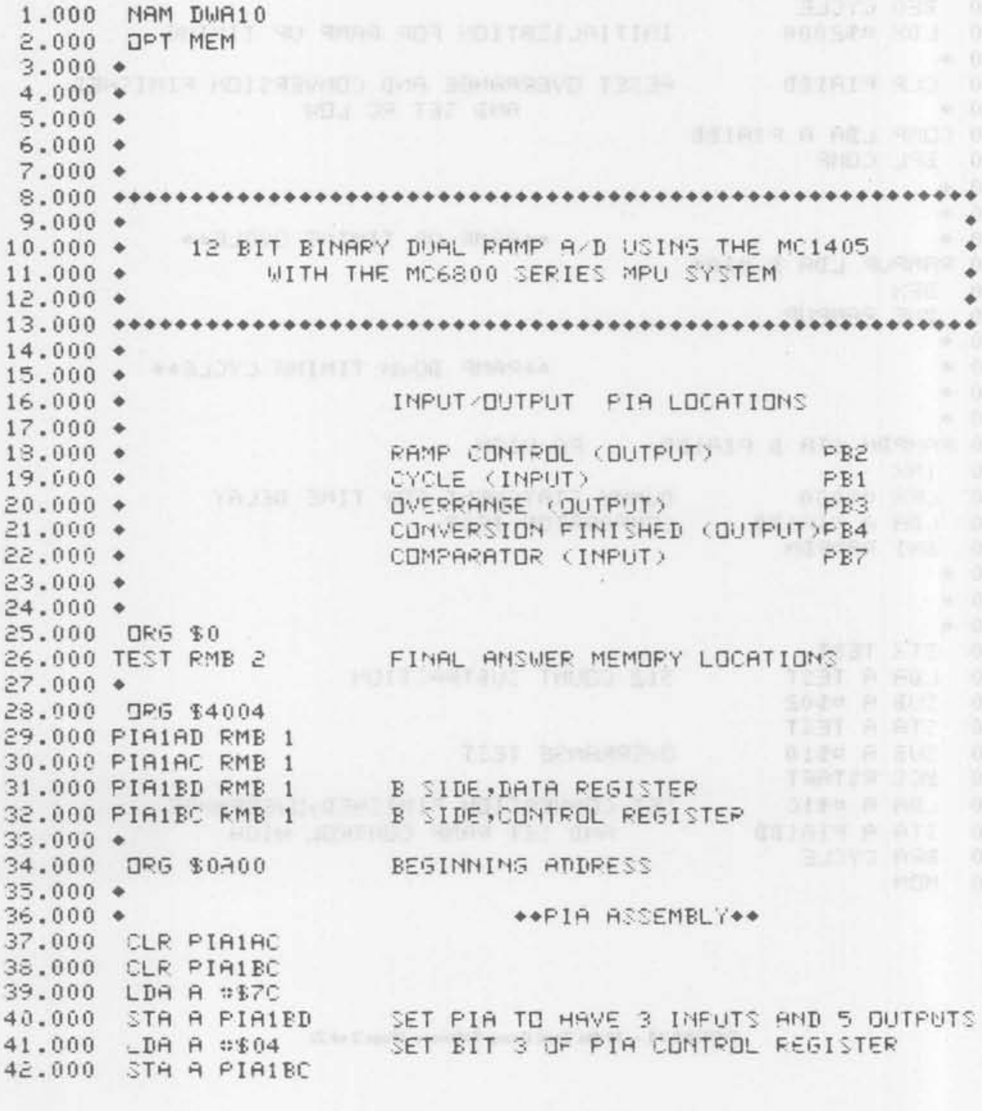
FIGURE 13 – 12-Bit Binary Dual Ramp A/D Flow Diagram

then stored back into the same memory location. Lines 86 and 87 check the contents of memory location TEST for a number greater than 4095_{10}. If this condition occurs, the overrange, conversion finished, and ramp control bits are set high. Otherwise the MPU branches back to line 50 where only the conversion finished and ramp control bits are set high. The program then checks the status of the cycle input waiting for the next conversion.

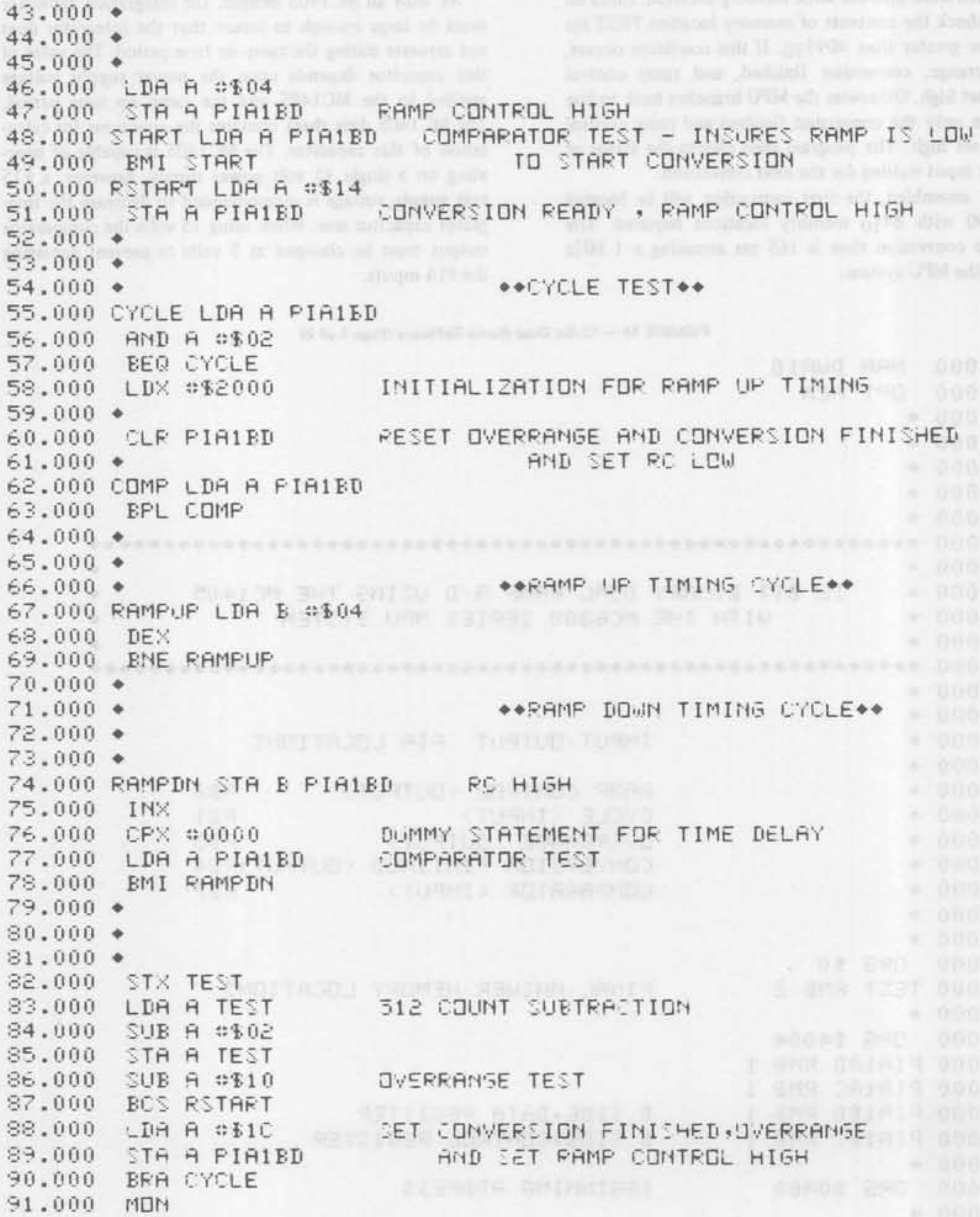
When assembled, the first instruction will be located at $0A00 with 84_{10} memory locations required. The full scale conversion time is 165 ms assuming a 1 MHz clock in the MPU system.

As with all MC1405 designs, the integration capacitor must be large enough to insure that the integrator does not saturate during the ramp up time period. The value of this capacitor depends upon the power supply voltage applied to the MC1405 and the ramp up time period. The MC1405 data sheet contains the equations for calculation of this capacitor. The MC1405 is capable of operating on a single +5 volt power supply; however, a +15 volt supply voltage is recommended to decrease the integrator capacitor size. When using 15 volts the comparator output must be clamped at 5 volts to prevent damaging the PIA inputs.

FIGURE 14 – 12-Bit Dual Ramp Software (Page 1 of 2)



```
 1.000  NAM DWA10
 2.000  OPT MEM
 3.000 *
 4.000 *
 5.000 *
 6.000 *
 7.000 *
 8.000 ***********************************************************
 9.000 *                                                         *
10.000 *    12 BIT BINARY DUAL RAMP A/D USING THE MC1405         *
11.000 *         WITH THE MC6800 SERIES MPU SYSTEM               *
12.000 *                                                         *
13.000 ***********************************************************
14.000 *
15.000 *
16.000 *                    INPUT/OUTPUT  PIA LOCATIONS
17.000 *
18.000 *                    RAMP CONTROL (OUTPUT)             PB2
19.000 *                    CYCLE (INPUT)                     PB1
20.000 *                    OVERRANGE (OUTPUT)                PB3
21.000 *                    CONVERSION FINISHED (OUTPUT)PB4
22.000 *                    COMPARATOR (INPUT)                PB7
23.000 *
24.000 *
25.000  ORG $0
26.000 TEST RMB 2           FINAL ANSWER MEMORY LOCATIONS
27.000 *
28.000  ORG $4004
29.000 PIA1AD RMB 1
30.000 PIA1AC RMB 1
31.000 PIA1BD RMB 1         B SIDE,DATA REGISTER
32.000 PIA1BC RMB 1         B SIDE,CONTROL REGISTER
33.000 *
34.000  ORG $0A00           BEGINNING ADDRESS
35.000 *
36.000 *                              **PIA ASSEMBLY**
37.000  CLR PIA1AC
38.000  CLR PIA1BC
39.000  LDA A #$7C
40.000  STA A PIA1BD        SET PIA TO HAVE 3 INPUTS AND 5 OUTPUTS
41.000  LDA A #$04          SET BIT 3 OF PIA CONTROL REGISTER
42.000  STA A PIA1BC
```



```
*
*
*
 LDA A #$04
 STA A PIA1BD     RAMP CONTROL HIGH
START LDA A PIA1BD   COMPARATOR TEST - INSURES RAMP IS LOW
 BMI START                 TO START CONVERSION
RSTART LDA A #$14
 STA A PIA1BD     CONVERSION READY , RAMP CONTROL HIGH
*
*
*                         **CYCLE TEST**
CYCLE LDA A PIA1BD
 AND A #$02
 BEQ CYCLE
 LDX #$2000       INITIALIZATION FOR RAMP UP TIMING
*
 CLR PIA1BD       RESET OVERRANGE AND CONVERSION FINISHED
*                           AND SET RC LOW
COMP LDA A PIA1BD
 BPL COMP
*
*
*                         **RAMP UP TIMING CYCLE**
RAMPUP LDA B #$04
 DEX
 BNE RAMPUP
*
*                         **RAMP DOWN TIMING CYCLE**
*
*
RAMPDN STA B PIA1BD     RC HIGH
 INX
 CPX #0000        DUMMY STATEMENT FOR TIME DELAY
 LDA A PIA1BD     COMPARATOR TEST
 BMI RAMPDN
*
*
*
 STX TEST
 LDA A TEST       512 COUNT SUBTRACTION
 SUB A #$02
 STA A TEST
 SUB A #$10       OVERRANGE TEST
 BCS RSTART
 LDA A #$1C       SET CONVERSION FINISHED,OVERRANGE
 STA A PIA1BD         AND SET RAMP CONTROL HIGH
 BRA CYCLE
 MON
```

FIGURE 14 – 12-Bit Dual Ramp Software (Page 2 of 2)

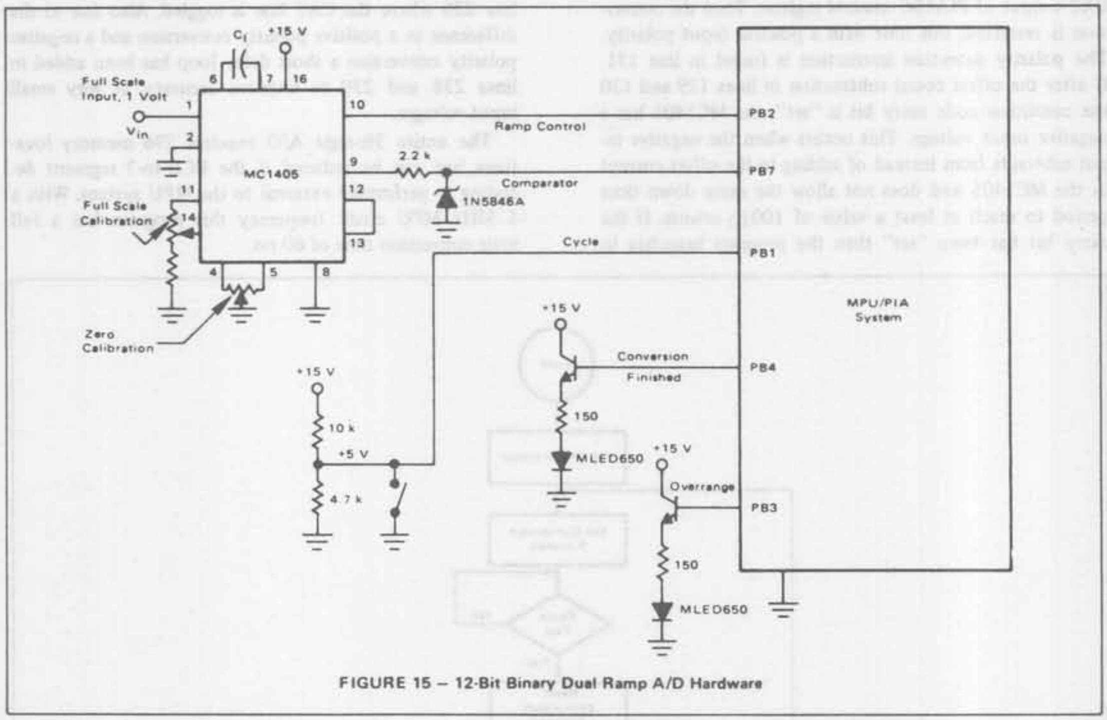


FIGURE 15 – 12-Bit Binary Dual Ramp A/D Hardware

3½-Digit Dual Ramp Program

The flow chart, source program and hardware for a 3½-digit system are shown in Figures 16, 17, and 18 respectively. Referring to Figure 17, the basic conversion routine of lines 96 through 135 in this program is similar to that of the previously discussed 12-bit binary system. The initialization of the index register in line 108 has been changed to increase the ramp up time period. The basic conversion results in a binary number as did the 12-bit version previously discussed. This binary result is converted by the software routine in lines 144 through 180 to produce 3½-digit BCD output. This routine converts up to a 16-bit binary number to the equivalent BCD value. Also the BCD result is converted to a 7-segment display code for use in a LED or LCD readout system. Another feature of the 3½-digit A/D program shown here is a polarity detection scheme. This allows the A/D to handle both positive and negative input voltages.

The external hardware for the 3½-digit A/D requires two full PIAs; one of the four ports is used for interface to the MC1405, cycle input, overrange flag, etc. An I/O configuration similar to that of the 12-bit binary A/D is used. The remaining three ports of the PIAs are used for the 3½-digit display, as shown in Figure 18b.

The conversion initially produces a binary result which is stored in memory locations MSB and MSB+1. This result has 100_{10} offset counts subtracted, and then a polarity check is made. If the polarity that is currently being applied to the input of the MC1405 is positive, the binary number is converted to a BCD number. The technique used for binary-to-BCD conversion is described in Appendix B. The BCD results are stored in memory locations UNTTEN and HNDTHD. Each of these memory locations contains two BCD words. Following the conversion, an overrange test is made in lines 183 through 186 which checks for a maximum of a BCD "1" in the upper four bits of memory location HNDTHD. If an overrange condition occurs, the program branches to lines 227 through 234 where a 1999_{10} is placed in the display and the overrange flag in PIA1BD is "set".

After the overrange test the BCD code is converted to a 7-segment code and stored in the memory location for each PIA port. Segments A through G use PIA outputs 0 through 6 while the half digit output uses PIA2BD output PB7. The conversion technique for BCD-to-7 segment utilizes a look-up table in line 251 with the indexed mode of addressing to access the table. Each of the three full BCD digits is converted to the 7-segment code by first separating the lower BCD and upper BCD word and using the BCD code as the least significant byte of a two byte address for the look-up table. This address is then loaded into the index register and used to locate the corresponding 7-segment code. In the case of the upper BCD digit of each BCD, the memory must be shifted left four times for correct addressing of the look-up table. Finally, the half digit output is added to PIA2BD in lines 197 through 226.

Should the MC1405 have the incorrect polarity on its input, a polarity reversing relay is operated by toggling the

CA2 output of PIA1BC control register. Then the conversion is restarted, this time with a positive input polarity. The polarity detection instruction is found in line 131. If after the offset count subtraction in lines 129 and 130 the condition code carry bit is "set", the MC1405 has a negative input voltage. This occurs when the negative input subtracts from instead of adding to the offset current in the MC1405 and does not allow the ramp down time period to reach at least a value of 100_{10} counts. If the carry bit has been "set" then the program branches to line 236 where the CA2 line is toggled. Also due to the difference in a positive polarity conversion and a negative polarity conversion a short delay loop has been added in lines 238 and 239 to improve accuracy at very small input voltages.

The entire 3½-digit A/D requires 296 memory locations but can be reduced if the BCD-to-7 segment decoding is performed external to the MPU system. With a 1 MHz MPU clock frequency this program has a full scale conversion time of 60 ms.

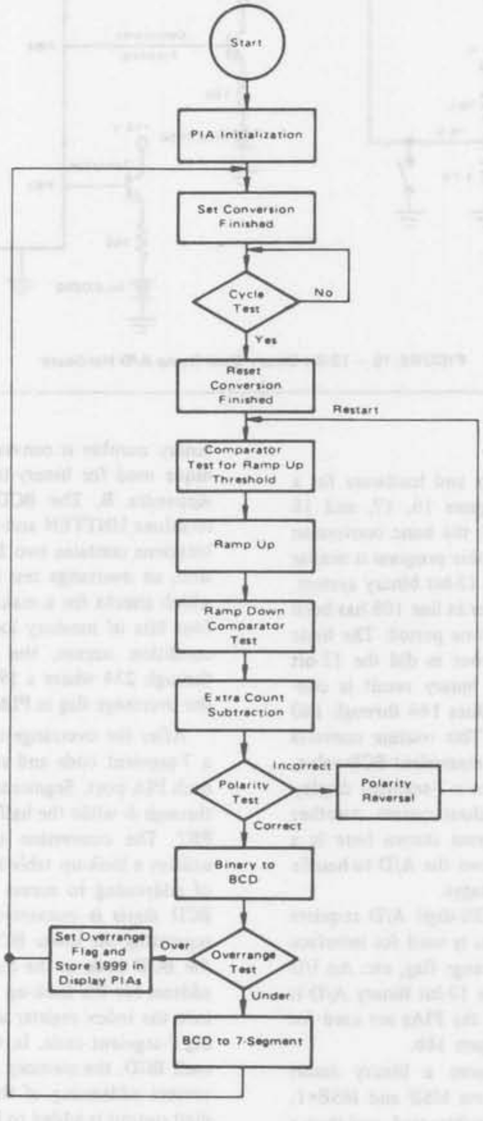


FIGURE 16 — 3½-Digit Dual Ramp A/D Flow Diagram

3

FIGURE 17 — 3½-Digit Dual Ramp Software (Page 1 of 5)

```
 NAM DWA25
 OPT MEM
*
*
*          ****************************************
*          *                                      *
*          *          3 1/2 DIGIT A/D             *
*          *                                      *
*          ****************************************
*
*
* THIS CONVERTER USES A MC1405 IN CONJUNCTION WITH THE
* MC6800 MPU TO PRODUCE A 3 1/2 DIGIT A/D.  THE
* DUAL RAMP METHOD OF A/D CONVERSION IS USED.
*
*   THE INPUTS TO THE MPU CONSIST OF
*
*        CYCLE SWITCH -LOCATED AT PIA1BD (PB1)
*        COMPARATOR   = LOCATED AT PIA1BD (PB7)
*
*   THE OUTPUTS FROM THE MPU CONSIST OF
*
*        RAMP CONTROL- LOCATED AT PIA1BD (PB2)
*        CONVERSION FINISHED = LOCATED AT PIA1BD (PB4)
*        OVERRANGE    - LOCATED AT PIA1BD (PB3)
*        POLARITY     - LOCATED AT PIA1BD (CA2)
*
*        7 SEGMENT OUTPUT
*          TENS   - PIA1AD
*          HUNDREDS - PIA2AD
*          THOUSANDS - PIA2BD
*          TENS OF THOUSANDS OR HALF DIGIT - PIA2BD (PB7)
*
*    THE BINARY ANSWER IS STORED AT MSB AND LSB
*
*    THE BCD ANSWER IS STORED AT UNTTEN,HNDTHD,TENTSD
*
*    THE ANALOG INPUT FOR THE MC1405 MUST HAVE A 2 VOLT
*    MAXIMUM WHILE THE AUTOPOLARITY OUTPUT FROM THE MPU
*    MAY BE USED TO TOGGLE A RELAY TO PROVIDE NEGATIVE
*    INPUT CAPABILITY FOR THE A/D
*
*
*
 ORG $0000
MSB RMB 1
LSB RMB 1
INDEX RMB 2
MSBTEM RMB 1          TEMP STORAGE OF BINARY ANSWER
LSBTEM RMB 1
*
*
*
 ORG $0010
UNTTEN RMB 1
HNDTHD RMB 1
*
*
 ORG $4004
PIA1AD RMB 1          A SIDE DATA REGISTER
```

FIGURE 17 — 3½-Digit Dual Ramp Software (Page 2 of 5)

```
PIA1AC RMB 1         A SIDE CONTROL REGISTER
PIA1BD RMB 1         B SIDE DATA REGISTER
PIA1BC RMB 1         B SIDE CONTROL REGISTER
PIA2AD RMB 1         A SIDE DATA REGISTER
PIA2AC RMB 1         A SIDE CONTROL REGISTER
PIA2BD RMB 1         B SIDE DATA REGISTER
PIA2BC RMB 1         B SIDE CONTROL REGISTER
*
*
 ORG $0A00
*
*                                **PIA ASSEMBLY**
 CLR PIA1AC
 CLR PIA1BC
 CLR PIA2AC
 CLR PIA2BC
 LDA A #$7C
 STA A PIA1BD
 LDA A #$0FF
 STA A PIA1AD
 STA A PIA2AD
 STA A PIA2BD
 LDA A #$34 SETS PIA CONTROL REGISTER BIT 3 HIGH
 STA A PIA1AC
 STA A PIA1BC
 STA A PIA2AC
 STA A PIA2BC
*
 LDA A #$0C          FIRST TWO HEX DIGITS OF LOOK-UP
 STA A INDEX                TABLE ADDRESSES
*               ****************
*               *   BASIC A/D  *
*               ****************
*
*                                   INITIALIZATION
 LDA A #$04
 STA A PIA1BD   RC HIGH
START LDA A PIA1BD COMPARATOR TEST
 BMI START
CYCLE1 LDA A #$14
 STA A PIA1BD   CONVERSION READY AND RC HIGH
*
*
*                                **CYCLE TEST**
CYCLE LDA A PIA1BD
 AND A #$02
 BEQ CYCLE
RESTAR LDX #$07D0
 CLR PIA1BD RESET OVERRANGE, CONVERSION FINISHED AND SET RC LOW
COMP LDA A PIA1BD
 BPL COMP
*                                **RAMP UP TIMING CYCLE**
RAMPUP LDA B #$04
 DEX
 BNE RAMPUP
*
*                                **RAMP DOWN TIMING CYCLE**
*
*
RAMPDN STA B PIA1BD     RC HIGH
```

```
 INX
 CPX #0000           DUMMY STATEMENT FOR TIME DELAY
 LDA A PIA1BD  COMPARATOR TEST
 BMI RAMPDN
*
 STX MSB
 LDA A MSB+1
 LDA B MSB
 SUB A #$64
 SBC B #$00
 BCS POLRY1
 STA A MSB+1
 STA B MSB
 STA A MSBTEM+1
 STA B MSBTEM
*
*
*
*                   ****************
*                   * BINARY TO BCD *
*                   *   CONVERTER   *
*                   ****************
*
 CLR UNTTEN
 CLR HNDTHD
 LDX #$0010
BEGIN LDA A UNTTEN
 TAB
 AND A #$0F
 SUB A #$05
 BMI AT
 ADD B #$03
AT TBA
 AND A #$0F0
 SUB A #$50
 BMI BT
 ADD B #$30
BT STA B UNTTEN
*
 LDA A HNDTHD
 TAB
 AND A #$0F
 SUB A #$05
 BMI CT
 ADD B #$03
CT TBA
 AND A #$0F0
 SUB A #$50
 BMI DT
 ADD B #$30
DT STA B HNDTHD
*
*
*
 ASL LSBTEM
 ROL MSBTEM
 ROL UNTTEN
 ROL HNDTHD
 DEX
 BNE BEGIN
```

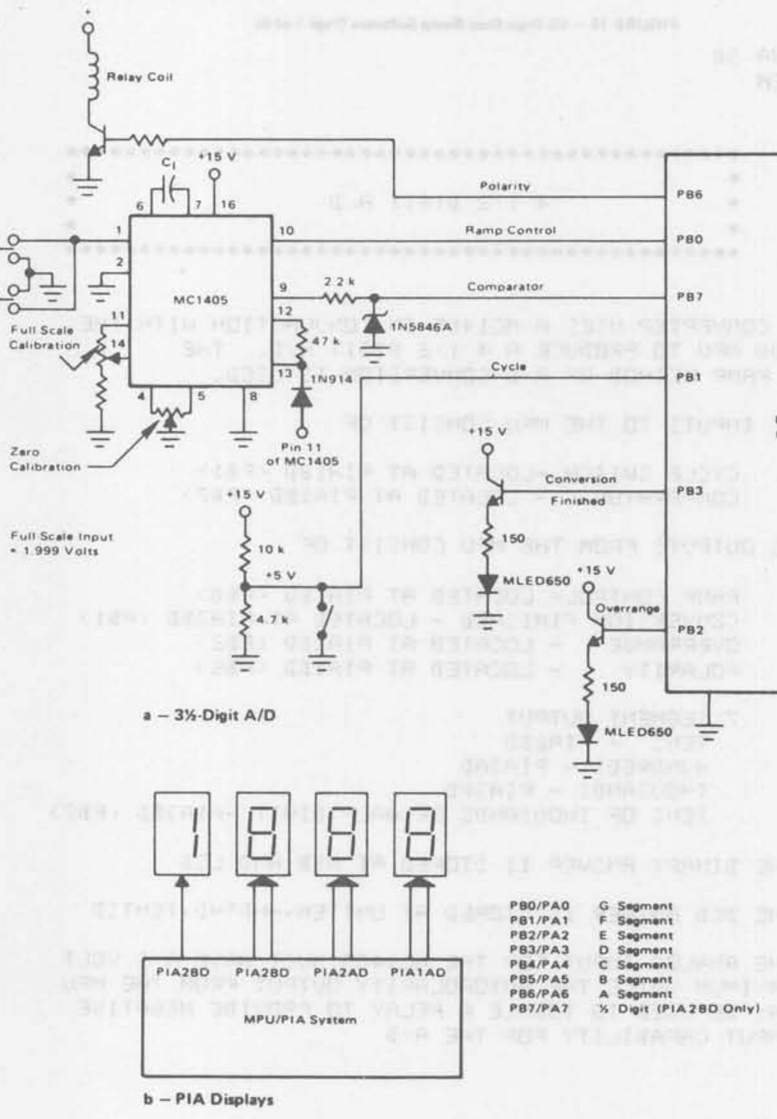
FIGURE 17 – 3½-Digit Dual Ramp Software (Page 3 of 5)

FIGURE 17 — 3½-Digit Dual Ramp Software (Page 4 of 5)

```
*
*                              OVERRANGE TEST
 LDA A HNDTHD
 AND A #$20
 SUB A #$10
 BHI OVRNGE
*
 BRA BCD
POLRY1 BRA POLARY   PATCH TO EXTEND RANGE OF BRANCHES
OVRNG1 BRA OVRNGE
*
*
*
*                  ********************
*                  * BCD TO 7 SEGMENT *
*                  *   CONVERTER      *
*                  ********************
BCD LDA A UNTTEN
 AND A #$0F
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA1AD
 LDA A UNTTEN
 LSR A
 LSR A
 LSR A
 LSR A
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA2AD
 LDA A HNDTHD
 AND A #$0F
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA2BD
 LDA A HNDTHD
 AND A #$10
 SUB A #$10
 BLT END1
 LDA A #$80
 ADD A PIA2BD
 STA A PIA2BD
END1 JMP CYCLE1
*
OVRNGE LDA A #$1C
 STA A PIA1BD
 LDA A #$F3
 STA A PIA1AD
 STA A PIA2AD
 STA A PIA2BD
 JMP CYCLE
*
POLARY LDX #$0100
BR DEX
 BNE BR
 LDA A PIA1BC
 COM A
```

```
241.000  AND A #$08
242.000  ADD A #$34
243.000  STA A PIA1BC
244.000  JMP RESTAR
245.000 *
246.000 *
247.000 *          LOOK-UP TABLE FOR BCD TO 7 SEGMENT
248.000 *                CONVERSION
249.000  ORG $0C00
250.000  FCB $7E,$30,$6D,$79,$33,$5B,$5F,$70,$7F,$73
251.000  END
252.000  MON
```

FIGURE 17 – 3½-Digit Dual Ramp Software (Page 5 of 5)



a – 3½-Digit A/D

b – PIA Displays

FIGURE 18 – 3½-Digit Dual Ramp A/D Hardware

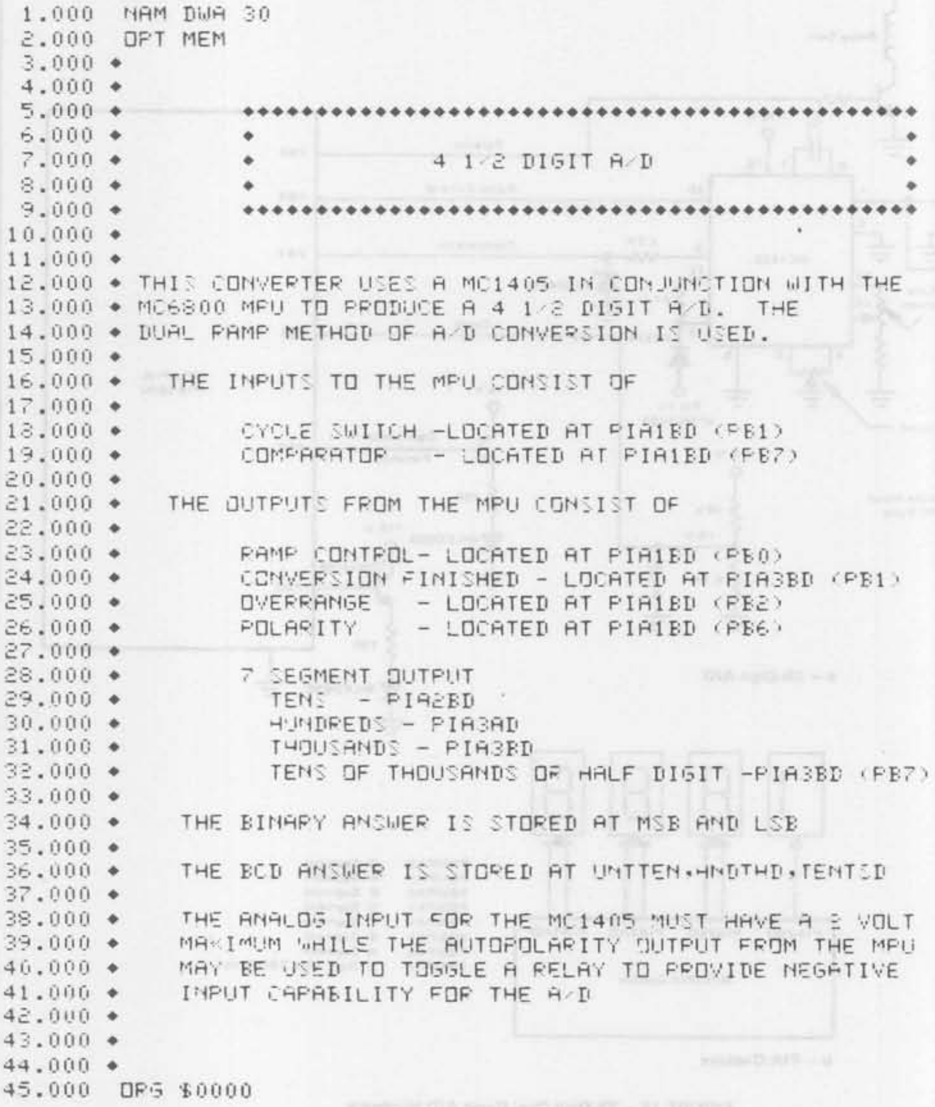
4½-Digit Dual Ramp Program

The microprocessor software for a 4½-digit dual ramp A/D is shown in Figure 19. This program in an extension of the 3½-digit A/D just discussed and has a full scale input voltage of 1.9999 volts. Due to the addition of the extra digit, a fourth PIA port for the 7-segment display is required. The PIA port configuration used for ramp control, comparator, etc. is identical to that used in the 3½-digit A/D.

The addition of the extra digit also implies a longer ramp up time period which is produced by increasing the initialization of the index register in line 115. This longer ramp up time period also requires the change of the extra count subtraction statements of lines 137 and 138 to maintain the extra count subtraction of 10% ramp up time. Also, the longer ramp up time period will require a larger integration capacitor to prevent saturation of the MC1405 integrator. This is of course, assuming the same MPU clock frequency. The remainder of the A/D external hardware is unchanged except for the addition of the fourth full digital display. Figure 18a can be used for the 4½-digit A/D without modification, and Figure 18b can be used with only the addition of another digit.

The software for the binary-to-BCD converter remains the same for the 4½-digit A/D since it is capable of handling up to 16 bits. The conversion routine for BCD-to-7 segment code must be modified to handle the extra digit although the same basic technique is retained.

FIGURE 19 – 4½-Digit Dual Ramp Software (Page 1 of 5)



```
 1.000  NAM DWA 30
 2.000  OPT MEM
 3.000 *
 4.000 *
 5.000 *          *******************************************
 6.000 *          *                                         *
 7.000 *          *            4 1/2 DIGIT A/D              *
 8.000 *          *                                         *
 9.000 *          *******************************************
10.000 *
11.000 *
12.000 * THIS CONVERTER USES A MC1405 IN CONJUNCTION WITH THE
13.000 * MC6800 MPU TO PRODUCE A 4 1/2 DIGIT A/D.  THE
14.000 * DUAL RAMP METHOD OF A/D CONVERSION IS USED.
15.000 *
16.000 *   THE INPUTS TO THE MPU CONSIST OF
17.000 *
18.000 *          CYCLE SWITCH -LOCATED AT PIA1BD (PB1)
19.000 *          COMPARATOR    - LOCATED AT PIA1BD (PB7)
20.000 *
21.000 *   THE OUTPUTS FROM THE MPU CONSIST OF
22.000 *
23.000 *          RAMP CONTROL- LOCATED AT PIA1BD (PB0)
24.000 *          CONVERSION FINISHED - LOCATED AT PIA3BD (PB1)
25.000 *          OVERRANGE    - LOCATED AT PIA1BD (PB2)
26.000 *          POLARITY     - LOCATED AT PIA1BD (PB6)
27.000 *
28.000 *          7 SEGMENT OUTPUT
29.000 *            TENS  - PIA2BD
30.000 *            HUNDREDS - PIA3AD
31.000 *            THOUSANDS - PIA3BD
32.000 *            TENS OF THOUSANDS OR HALF DIGIT -PIA3BD (PB7)
33.000 *
34.000 *    THE BINARY ANSWER IS STORED AT MSB AND LSB
35.000 *
36.000 *    THE BCD ANSWER IS STORED AT UNTTEN,HNDTHD,TENTSD
37.000 *
38.000 *    THE ANALOG INPUT FOR THE MC1405 MUST HAVE A 2 VOLT
39.000 *    MAXIMUM WHILE THE AUTOPOLARITY OUTPUT FROM THE MPU
40.000 *    MAY BE USED TO TOGGLE A RELAY TO PROVIDE NEGATIVE
41.000 *    INPUT CAPABILITY FOR THE A/D
42.000 *
43.000 *
44.000 *
45.000  ORG $0000
```

```
 46.000 MSB RMB 1
 47.000 LSB RMB 1
 48.000 INDEX RMB 2
 49.000 MSBTEM RMB 1                TEMP STORAGE OF BINARY ANSWER
 50.000 LSBTEM RMB 1
 51.000 *
 52.000 *
 53.000 *
 54.000  ORG $0010
 55.000 UNTTEN RMB 1
 56.000 HNDTHD RMB 1
 57.000 TENTSD RMB 1
 58.000 *
 59.000 *
 60.000  ORG $4006
 61.000 PIA1BD RMB 1          B SIDE, DATA REGISTER
 62.000 PIA1BC RMB 1          B SIDE, CONTROL REGISTER
 63.000 PIA2AD RMB 1          A SIDE, DATA REGISTER
 64.000 PIA2AC RMB 1          A SIDE, CONTROL REGISTER
 65.000 PIA2BD RMB 1          B SIDE, DATA REGISTER
 66.000 PIA2BC RMB 1          B SIDE, CONTROL REGISTER
 67.000  ORG $4010
 68.000 PIA3AD RMB 1          A SIDE, DATA REGISTER
 69.000 PIA3AC RMB 1          A SIDE, CONTROL REGISTER
 70.000 PIA3BD RMB 1          B SIDE,DATA REGISTER
 71.000 PIA3BC RMB 1          B SIDE, CONTROL REGISTER
 72.000 *
 73.000 *
 74.000 *
 75.000 *                                 PIA ASSEMBLY
 76.000  ORG $0A00
 77.000  CLR PIA1BC
 78.000  CLR PIA2AC
 79.000  CLR PIA2BC
 80.000  CLR FIA3AC
 81.000  CLR PIA3BC
 82.000  LDA A #$4D
 83.000  STA A PIA1BD
 84.000  LDA A #$0FF    REMAINING PIA'S ALL OUTPUTS
 85.000  STA A PIA2AD
 86.000  STA A PIA2BD
 87.000  STA A PIA3AD
 88.000  STA A PIA3BD
 89.000  LDA A #$34      SETS PIA CONTROL REGISTER BIT 3 HIGH
 90.000  STA A PIA1BC
 91.000  STA A PIA2AC
 92.000  STA A PIA2BC
 93.000  STA A PIA3AC
 94.000  STA A PIA3BC
 95.000 *
 96.000  LDA A #$0C          FIRST TWO HEX DIGITS OF LOOK-UP
 97.000  STA A INDEX                  TABLE ADDRESSES
 98.000 *                ****************
 99.000 *                *   BASIC A/D  *
100.000 *                ****************
101.000 *
102.000 *                                  INITIALIZATION
103.000  LDA A #$04
104.000  STA A PIA1BD   RC HIGH
105.000 START LDA A PIA1BD  COMPARATOR TEST
```

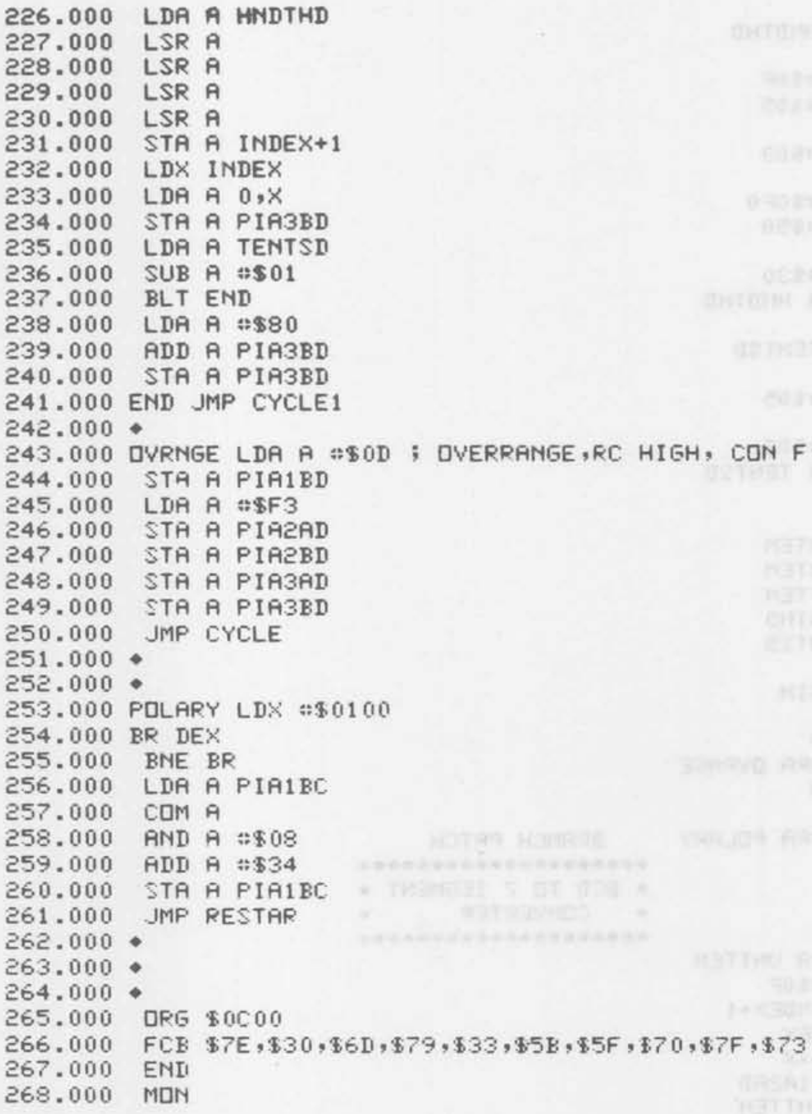
FIGURE 19 – 4½-Digit Dual Ramp Software (Page 2 of 5)

FIGURE 19 – 4½-Digit Dual Ramp Software (Page 3 of 5)

```
 BMI START
CYCLE1 LDA A #14
 STA A PIA1BD   CONVERSION READY AND RC HIGH
*
*
*                                   CYCLE TEST
CYCLE LDA A PIA1BD
 AND A #$02
 BEQ CYCLE
RESTART LDX #$4E20         INITIALIZATION FOR RAMP UP
*                              TIMING
 CLR PIA1BD RESET OVERRANGE, CONVERSION FINISHED AND SET RC LOW
COMP LDA A PIA1BD         COMPARATOR TEST
 BPL COMP
*                                   RAMP UP TIMING CYCLE
RAMPUP LDA B #$04
 DEX
 BNE RAMPUP
*
*                              RAMP DOWN TIMING CYCLE
*
*
RAMPDN STA B PIA1BD     RC HIGH
 INX
 CPX #0000   DUMMY STATEMENT
 LDA A PIA1BD  COMPARATOR TEST
 BMI RAMPDN
*
*                              EXTRA COUNT SUBTRACTION
 STX MSB
 STX MSBTEM
 LDA A MSB
 SUB A #$04        EXTRA COUNT SUBTRACTION
 BMI POLRY1        POLARITY TEST
 STA A MSB
 STA A MSBTEM
*
*
*
*                  *****************
*                   * BINARY TO BCD *
*                   *  CONVERTER    *
*                  *****************
*
 CLR UNTTEN
 CLR HNDTHD
 CLR TENTSD
 LDX #$0010
BEGIN LDA A UNTTEN
 TAB
 AND A #$0F
 SUB A #$05
 BMI AT
 ADD B #$03
AT TBA
 AND A #$0F0
 SUB A #$50
 BMI BT
 ADD B #$30
BT STA B UNTTEN
```

```
*
 LDA A HNDTHD
 TAB
 AND A #$0F
 SUB A #$05
 BMI CT
 ADD B #$03
CT TBA
 AND A #$0F0
 SUB A #$50
 BMI DT
 ADD B #$30
DT STA B HNDTHD
*
 LDA A TENTSD
 TAB
 SUB A #$05
 BMI ET
 ADD B #$03
ET STA B TENTSD
*
*
 ASL LSBTEM
 ROL MSBTEM
 ROL UNTTEN
 ROL HNDTHD
 ROL TENTSD
 DEX
 BNE BEGIN
*
 BRA BCD
DVRNG1 BRA DVRNGE
 BRA BCD
*
POLRY1 BRA POLARY     BRANCH PATCH
*                  *********************
*                  * BCD TO 7 SEGMENT *
*                  *    CONVERTER      *
*                  *********************
BCD LDA A UNTTEN
 AND A #$0F
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA2AD
 LDA A UNTTEN
 LSR A
 LSR A
 LSR A
 LSR A
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA2BD
 LDA A HNDTHD
 AND A #$0F
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA3AD
```

FIGURE 19 – 4½-Digit Dual Ramp Software (Page 4 of 5)



```
226.000  LDA A HNDTHD
227.000  LSR A
228.000  LSR A
229.000  LSR A
230.000  LSR A
231.000  STA A INDEX+1
232.000  LDX INDEX
233.000  LDA A 0,X
234.000  STA A PIA3BD
235.000  LDA A TENTSD
236.000  SUB A #$01
237.000  BLT END
238.000  LDA A #$80
239.000  ADD A PIA3BD
240.000  STA A PIA3BD
241.000 END JMP CYCLE1
242.000 *
243.000 OVRNGE LDA A #$0D ; OVERRANGE,RC HIGH, CON F
244.000  STA A PIA1BD
245.000  LDA A #$F3
246.000  STA A PIA2AD
247.000  STA A PIA2BD
248.000  STA A PIA3AD
249.000  STA A PIA3BD
250.000  JMP CYCLE
251.000 *
252.000 *
253.000 POLARY LDX #$0100
254.000 BR DEX
255.000  BNE BR
256.000  LDA A PIA1BC
257.000  COM A
258.000  AND A #$08
259.000  ADD A #$34
260.000  STA A PIA1BC
261.000  JMP RESTAR
262.000 *
263.000 *
264.000 *
265.000  ORG $0C00
266.000  FCB $7E,$30,$6D,$79,$33,$5B,$5F,$70,$7F,$73
267.000  END
268.000  MON
```

FIGURE 19 – 4½-Digit Dual Ramp Software (Page 5 of 5)

SUMMARY

Many MPU systems require analog information, which necessitates the use of an A/D converter in the microprocessor design. This note has presented two popular A/D techniques used in conjunction with the M6800 microprocessor system. These techniques, successive approximation and dual ramp, were shown using the MPU as the digital control element for the A/D system. This required dedication of the MPU to the A/D function during the conversion. Also shown were systems using the MPU to control the flow of data from an external A/D allowing the MPU to perform other tasks during the conversion.

The variety of programs presented allow the designer to make a selection based upon hardware cost, conversion speed, memory locations and interrupt capability. Although the A/D programs shown here are complete designs, they are general designs and may be tailored to fit each individual application. Also a variety of digital outputs are available including binary, BCD, and 7-segment. In conjunction with the BCD output a 16-bit binary to BCD conversion routine is presented in Appendix B.

REFERENCES

Aldridge, Don: "Autopolarity Circuits for the MC1405 Dual-Slope A-D Converter System", EB-35, Motorola Semiconductor Products Inc.

Aldridge, Don: "Input Buffer Circuits for the MC1505 Dual Ramp A-to-D Converter Subsystem", EB-24, Motorola Semiconductor Products Inc.

Kelley, Steve: "4½-Digit DVM System Using the MC1505 Dual-Slope Converter", EB-36, Motorola Semiconductor Products Inc.

M6800 Microprocessor Applications Manual, Motorola Semiconductor Products Inc.

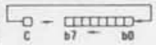
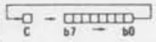
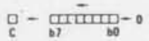
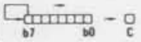
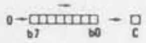
M6800 Microprocessor Programming Manual, Motorola Semiconductor Products Inc.

MC1505/1405 Data Sheet, Motorola Semiconductor Products Inc.

MC6800, MC6820 Data Sheets, *M6800 Systems Reference and Data Sheets*, Motorola Semiconductor Products Inc.

APPENDIX A
MPU INSTRUCTIONS

Accumulator and Memory Instructions

		ADDRESSING MODES															BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG.					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED			(All register labels refer to contents)	5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's	NEG							60	7	2	7ᵕ	6	3				00 − M → M	•	•	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	○	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A 	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A 	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A 	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A 	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A 	•	•	R	↕	⑥	↕
#	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

LEGEND:

OP	Operation Code (Hexadecimal);	+	Boolean Inclusive OR;
~	Number of MPU Cycles;	⊙	Boolean Exclusive OR;
#	Number of Program Bytes;	M̄	Complement of M;
+	Arithmetic Plus;	→	Transfer Into;
−	Arithmetic Minus;	0	Bit = Zero;
·	Boolean AND;	00	Byte = Zero;
M_{SP}	Contents of memory location pointed to be Stack Pointer;		

Note – Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

H	Half-carry from bit 3;
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

Index Register and Stack Manipulation Instructions

		IMMED			DIRECT			INDEX			EXTND			IMPLIED				COND. CODE REG. 5	4	3	2	1	0
POINTER OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BOOLEAN/ARITHMETIC OPERATION	H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	⑦	↕	⑦	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

Jump and Branch Instructions

		RELATIVE			INDEX			EXTND			IMPLIED				COND. CODE REG. 5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BRANCH TEST	H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2										See Special Operations	•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										02	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1	See Special Operations	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	•	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	⑪	•	•	•	•

Condition Code Register Manipulation Instructions

		IMPLIED				COND. CODE REG. 5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	BOOLEAN OPERATION	H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	⑫					
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

1 (Bit V) Test: Result = 10000000?
2 (Bit C) Test: Result = 00000000?
3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4 (Bit V) Test: Operand = 10000000 prior to execution?
5 (Bit V) Test: Operand = 01111111 prior to execution?
6 (Bit V) Test: Set equal to result of N⊕C after shift has occurred.
7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?
8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?
9 (Bit N) Test: Result less than zero? (Bit 15 = 1)
10 (All) Load Condition Code Register from Stack. (See Special Operations)
11 (Bit I) Set when interrupt occurs. If previously set, a Non Maskable Interrupt is required to exit the wait state.
12 (All) Set according to the contents of Accumulator A.

APPENDIX B

BINARY-TO-BCD CONVERSION

A standard technique for binary-to-BCD conversion is that of the Add 3 algorithm. Figures B1 and B2 show a flow diagram and example of this algorithm. The technique requires a register containing the N-bit binary number and enough 4-bit BCD registers to contain the maximum equivalent BCD number for the initial binary number. The conversion starts by checking each BCD register for a value of 5 or greater. If this condition exists in one or all of these registers (initially this condition cannot exist), then a 3 is added to those registers where this condition exists. Next the registers are shifted left with the carry out of the previous register being the carry in to the next register. Again each BCD register is checked for values of 5 or greater. This sequence continues until the registers have been shifted N times, where N is the number of bits in the initial binary word. The BCD registers then contain the resulting BCD equivalent to the initial binary word. The example in Figure B2 starts with an 8-bit binary word consisting of all "1's." This word is converted to the BCD equivalent of 255 by this technique. After 8 shifts the last binary bit has been shifted out of the binary register and the hundreds, tens, and units registers contain a 255.

Figure B3 shows an MC6800 software routine for performing this technique of binary to BCD conversion. The initial binary number is a 16-bit number and occupies memory locations MSB and LSB; this binary number is converted to the equivalent BCD number in memory locations TENTSD, HNDTHD and UNTTEN. Each of these memory locations contains two BCD digits. Eighty-three memory locations are required for program storage with a maximum conversion taking 1.8 ms.

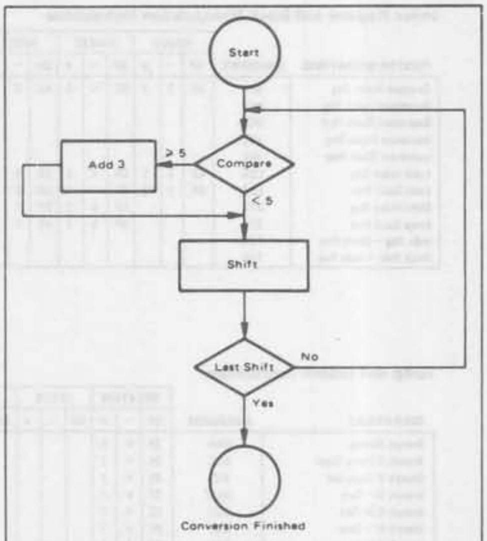


FIGURE B1 – Binary to BCD Conversion Flow Diagram

Hundreds	Tens	Units	8-Bit Binary	
			1 1 1 1 1 1 1 1	
		1	1 1 1 1 1 1 1	Shift
		1 1	1 1 1 1 1 1	Shift
		1 1 1	1 1 1 1 1	Shift
		1 0 1 0	1 1 1 1 1	Add 3 to Units
	1	0 1 0 1	1 1 1 1	Shift
	1	1 0 0 0	1 1 1 1	Add 3 to Units
	1 1	0 0 0 1	1 1 1	Shift
	1 1 0	0 0 1 1	1 1	Shift
	1 0 0 1	0 0 1 1	1 1	Add 3 to Tens
1	0 0 1 0	0 1 1 1	1	Shift
1	0 0 1 0	1 0 1 0	1	Add 3 to Units
1 0	0 1 0 1	0 1 0 1		Shift
2	5	5		Total Shifts 8

FIGURE B2 – Binary to BCD Conversion

FIGURE B3 – Binary-to-BCD Conversion Software (Page 1 of 2)

```
 NAM DWA21
 OPT MEM
*
**************************************************
*                                                *
*          BINARY TO BCD CONVERSION              *
*               ADD 3 ALGORITHM                  *
*                    16 BIT                      *
**************************************************
*
 ORG 0                 INITIAL BINARY NUMBER
MSB RMB 1                MOST SIGNIFICANT 8 BITS
LSB RMB 1                LEAST SIGNIFICANT 8 BITS
*
*
*
 ORG $0010             BCD RESULTS
UNTTEN RMB 1            UNITS AND TENS DIGITS
HNDTHD RMB 1            HUNDREDS AND THOUSANDS
TENTSD RMB 1            TENS OF THOUSANDS DIGIT
*
*
*
```

```
24.000  ORG $0F00       **BEGINNING OF PROGRAM**
25.000  CLR UNTTEN
26.000  CLR HNDTHD
27.000  CLR TENTSD
28.000  LDX #$0010
29.000 BEGIN LDA A UNTTEN     UNITS COMPARISON
30.000  TAB
31.000  AND A #$0F
32.000  SUB A #$05
33.000  BMI AT
34.000  ADD B #$03
35.000 AT TBA                 TENS COMPARISON
36.000  AND A #$0F0
37.000  SUB A #$50
38.000  BMI BT
39.000  ADD B #$30
40.000 BT STA B UNTTEN
41.000 *
42.000  LDA A HNDTHD          HUNDREDS COMPARISON
43.000  TAB
44.000  AND A #$0F
45.000  SUB A #$05
46.000  BMI CT
47.000  ADD B #$03
48.000 CT TBA
49.000  AND A #$0F0
50.000  SUB A #$50
51.000  BMI DT
52.000  ADD B #$30
53.000 DT STA B HNDTHD
54.000 *
55.000  LDA A TENTSD          TENS OF THOUSANDS COMPARISON
56.000  TAB
57.000  SUB A #$05
58.000  BMI ET
59.000  ADD B #$03
60.000 ET STA B TENTSD
61.000 *
62.000 *
63.000  ASL LSB
64.000  ROL MSB
65.000  ROL UNTTEN
66.000  ROL HNDTHD
67.000  ROL TENTSD
68.000  DEX
69.000  BNE BEGIN             END OF CONVERSION CHECK
70.000 *
71.000 *
72.000 *
73.000 *
74.000  END
75.000  MON
```

FIGURE B3 – Binary-to-BCD Conversion Software (Page 2 of 2)

AN-769
Application Note

AUTORANGING DIGITAL MULTIMETER USING THE MC14433 CMOS A/D CONVERTER

This application note describes an autorange digital multimeter using the MC14433. The multimeter includes ac and dc voltage ranges from 200 mV to 200 V, ac and dc current from 2 mA to 2 A full scale, and resistance ranges from 2 kΩ to 2 MΩ full scale.

AUTORANGING DIGITAL MULTIMETER USING THE MC14433 CMOS A/D CONVERTER

This article describes an autorange digital multimeter using the MC14433. The multimeter includes ac and dc voltage ranges from 200 mV to 200 V, ac and dc current from 2 mA to 2 A full scale, and resistance ranges from 2 kΩ to 2 MΩ full scale. The MC14433 DVM chip used provides a 3-1/2-digit A/D converter with autopolarity, autozero and a high input impedance. The chip has overrange and underrange information available to simplify the design of the autoranging meter. Only two input jacks are required for all ranges and functions, eliminating the need for changing leads on the instrument when changing ranges or functions. Although only four ranges are provided for each function, the technique used may be expanded to more ranges if desired.

Range switching is done with the use of mechanical relays. The relays may be replaced with solid-state analog switches; however it was felt that the mechanical relays would provide a higher degree of reliability due to the high voltage and currents being measured with the multimeter.

MC14433 A/D CONVERTER

The MC14433 is a single-chip 3-1/2-digit A/D converter using a modified dual ramp technique of A/D conversion. Housed in a 24-pin package, it features autopolarity, autozero and a high input impedance. Figure 1 shows the pin diagram of the MC14433.

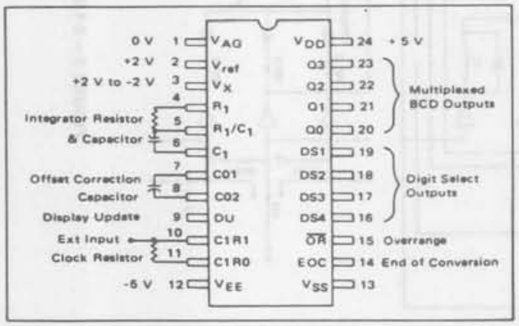


FIGURE 1 – MC14433 Pin Assignment

The output of the MC14433 is 3-1/2-digit multiplexed BCD with the MSD containing not only the half digit but also the polarity of the input, overrange and underrange information. Figure 2 shows the decoding for the MSD. The digit selects for the multiplexed BCD have interdigit blanking to ensure correct BCD data during the time that the digit select is true.

The converter is ratiometric and requires an external reference voltage. This voltage is 2.000 volts for the 1.999 volt range and 200 mV for the 199.9 mV full scale input. Both the unknown and reference inputs and analog ground are high-impedance inputs. External components required are two resistors and two capacitors.

TRUTH TABLE

Coded Condition of MSD	Q3	Q2	Q1	Q0	BCD to 7 Segment Decoding	
+0	1	1	1	0	Blank	
–0	1	0	1	0	Blank	
+0 UR	1	1	1	1	Blank	
–0 UR	1	0	1	1	Blank	
+1	0	1	0	0	4 → 1	Hook up only seg b and c to MSD
–1	0	0	0	0	0 → 1	
+1 OR	0	1	1	1	7 → 1	
–1 OR	0	0	1	1	3 → 1	

Notes for Truth Table

Q3 – ½ digit, low for "1", high for "0"

Q2 – Polarity: "1" = positive, "0" = negative

Q0 – Out of range condition exists if Q0 = 1. When used in conjunction with Q3 the type of out of range condition is indicated, i.e., Q3 = 0 → OR or Q3 = 1 → UR.

When only segment b and c of the decoder are connected to the ½ digit of the display, 4, 0, 7 and 3 appear as 1.

FIGURE 2 – MSD Coding

The MC14433 has an End of Conversion (EOC) pin for indicating the end of one conversion and the start of the next conversion by a positive pulse 1/2 clock period long. The device also contains a display update pin which allows the data to be strobed into the output latches. If at least one positive edge is received prior to the ramp down cycle, new data is strobed to the display. Normally this pin is tied to EOC to allow a data update each conversion cycle.

The MC14433 requires two power supplies. The total voltage must not exceed 18 volts. Pin 13 is the reference level for the output of the MC14433. If this pin is tied to 0 volts, the BCD output, digit selects and EOC will swing from 0 volts to V_{DD}. If, however, pin 13 is tied to V_{EE}, the output swing will be from V_{EE} to V_{DD}.

The clock for the MC14433 is internal to the chip, requiring only a single external resistor to set the frequency. An external clock may be used by driving pin 10. The total conversion time for the MC14433 is approximately 16400 clock periods. This conversion time includes the autozero cycle and the unknown input measurement cycle.

AUTORANGING CIRCUITRY

Figure 3 shows the autoranging DMM. The heart of the autoranging circuitry is an MC14035B CMOS shift

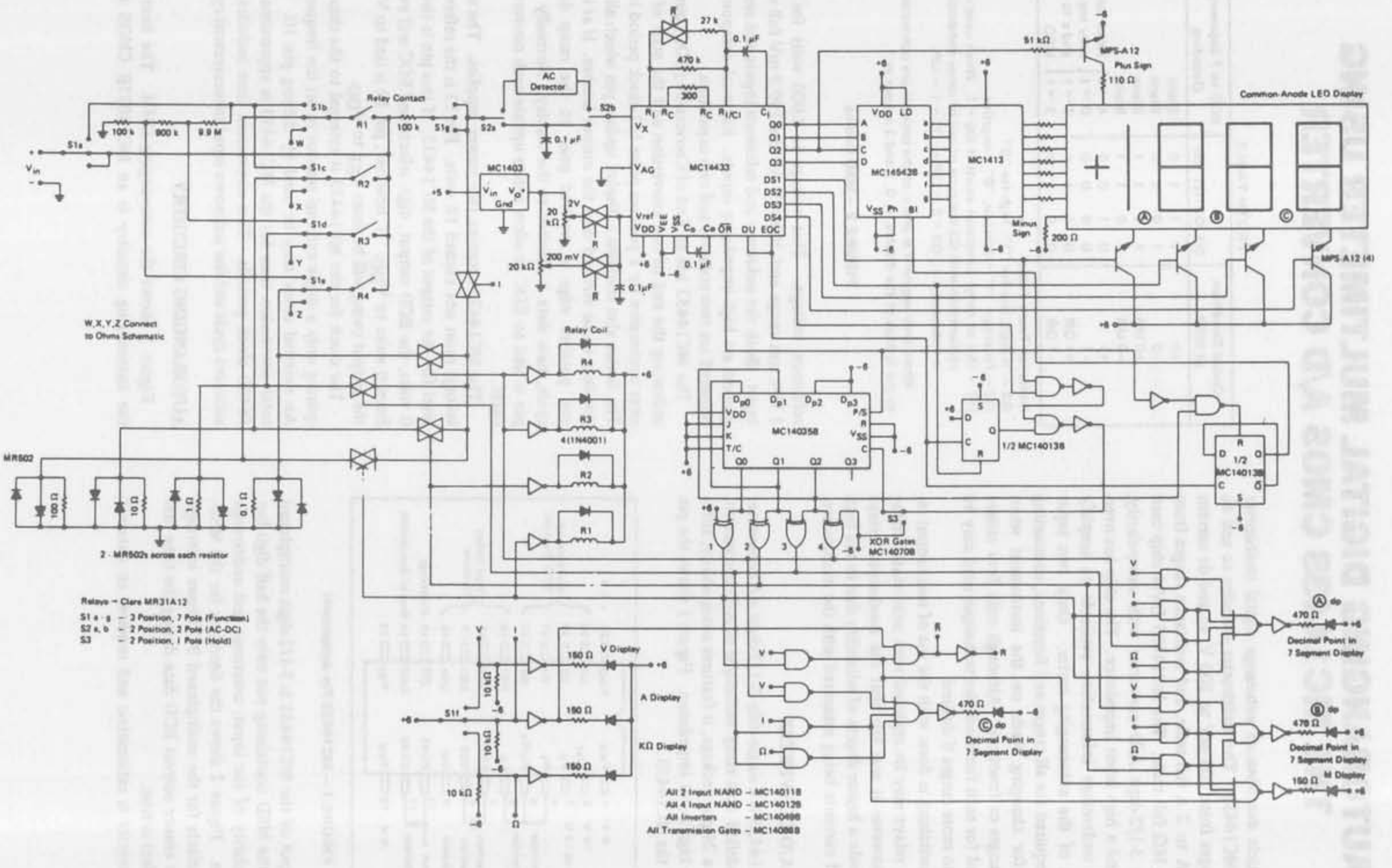
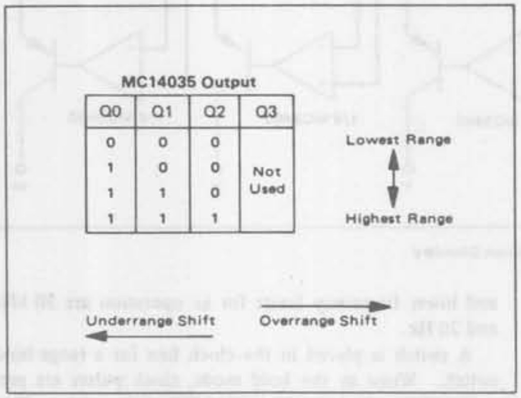


FIGURE 3 — 3-1/2-Digit Autoranging Multimeter

register which can be configured to shift either right or left. The direction of the shift is dependent upon whether an overrange or underrange signal is received at the end of each conversion. If the meter is in range, no shift signal is received. For an overrange condition, a high level is clocked to the right, and for an underrange condition the high level is clocked to the left (see Figure 4). The Exclusive OR gates decode the shift register output to produce only one output high. This output is used to turn on the corresponding range relays.


MC14035 Output

Q0	Q1	Q2	Q3
0	0	0	Not Used
1	0	0	
1	1	0	
1	1	1	

FIGURE 4 – Shift Register Operation for Autoranging DMM

If at the end of the next conversion the MC14433 is still either overrange or underrange, the shift register receives another clock pulse and thus the next range is selected. When an extreme overrange or underrange condition occurs the register is filled with all "ones" or all "zeros" which selects continuously either the highest or lowest range. Input voltages that exceed 200 volts as well as complete overrange conditions for the other functions cause the display to blink on and off. This feature is provided by the second half of the MC14013 flip-flop. The blinking rate is at half the conversion rate.

Figure 5 describes the functional operation for each range and function for the multimeter. The 2-volt reference is used for the ohms function, which means that 2 volts are developed across the unknown resistors at full scale. All current ranges use the 200-mV reference, while for voltage both the 200-mV and the 2-volt reference are used.

MC14066B transmission gates are used to switch between the 2-volt reference and the 200-mV reference. A transmission gate is also used to reduce the integrator resistor for the 200-mV range. In the current mode, transmission gates are used to switch the input of the MC14433 to the appropriate current-measuring resistor. This is necessary to eliminate the problem of measuring the voltage across the contact resistance of the function switch and relays in addition to the voltage across the current resistor. MR501 rectifiers are placed across the current resistors to limit the power dissipation during overrange conditions.

A ±6-volt power supply is used for the multimeter, with the logic sections referenced to the -6-volt level. This power supply is shown in Figure 6 and uses the MC7806 and MC7906 three-terminal regulators.

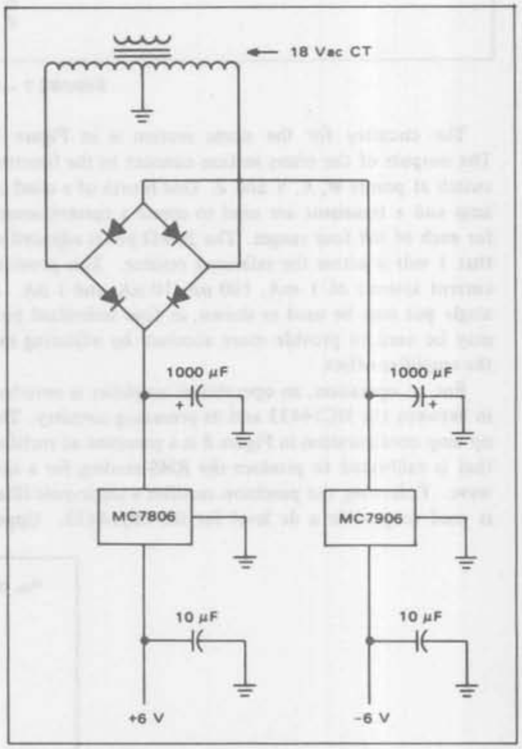


FIGURE 6 – ±6-Volt Power Supply

Relay	Voltage					Current					Resistance				
	Range	dp	Ref Used	Function Display	Resistor Divider	Range	dp	Ref Used	Function Display	Measurement Resistor	Range	dp	Ref Used	Function Display	Current Source
R1	200 mV	199.9	200 mV	mV	1:1	2 mA	1.999	200 mV	mA	100 Ω	2 kΩ	1.999	2 V	kΩ	1 mA
R2	2 V	1.999	2 V	V	1:1	20 mA	19.99	200 mV	mA	10 Ω	20 kΩ	19.99	2 V	kΩ	100 µA
R3	20 V	19.99	2 V	V	10:1	200 mA	199.9	200 mV	mA	1 Ω	200 kΩ	199.9	2 V	kΩ	10 µA
R4	200 V	199.9	2 V	V	100:1	2 A	1.999	200 mV	A	0.1 Ω	2000 kΩ	1999	2 V	kΩ	1 µA

FIGURE 5 – Functional Operation

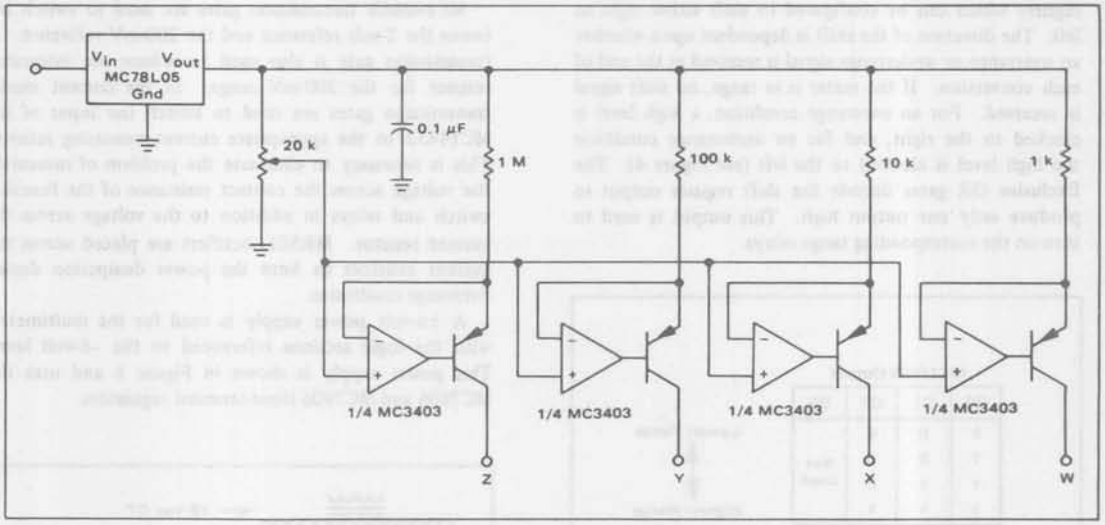


FIGURE 7 – Ohms Section Circuitry

The circuitry for the ohms section is in Figure 7. The outputs of the ohms section connect to the function switch at points W, X, Y and Z. One-fourth of a quad op amp and a transistor are used to create a current source for each of the four ranges. The 20-kΩ pot is adjusted so that 1 volt is across the reference resistor. This provides current sources of 1 mA, 100 µA, 10 µA, and 1 µA. A single pot may be used as shown, or four individual pots may be used to provide more accuracy by adjusting out the amplifier offset.

For ac operation, an operational amplifier is switched in between the MC14433 and its preceding circuitry. The op amp configuration in Figure 8 is a precision ac rectifier that is calibrated to produce the RMS reading for a sine wave. Following the precision rectifier a single-pole filter is used to provide a dc level for the MC14433. Upper and lower frequency limits for ac operation are 30 kHz and 20 Hz.

A switch is placed in the clock line for a range hold switch. When in the hold mode, clock pulses are prevented from clocking the MC14035B shift register. This feature allows several measurements to be made on a high range without the multimeter switching back to the low range between measurements.

The meter must not only be protected from destroying itself during overrange conditions but must also continue to make proper overrange measurements so that the next range may be selected. The analog input to the MC14433 is internally diode protected. The multimeter has a 100-kΩ resistor in series with this input to limit the current during overvoltage measurements.

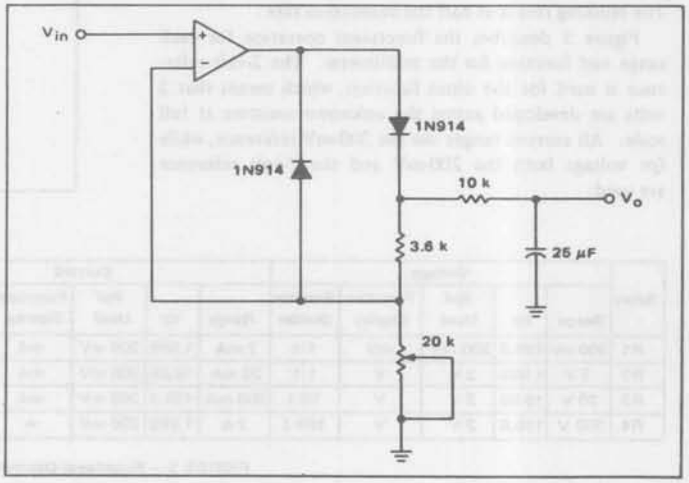


FIGURE 8 – AC Circuitry

AN-770

Application Note

DATA ACQUISITION NETWORKS WITH NMOS AND CMOS

This article describes an eight-channel data acquisition network (DAN) using the Motorola MC14433 CMOS A/D converter and the M6800 microprocessor family. The A/D conversion technique used with the MC14433 is a modified dual ramp featuring auto-zero, auto-polarity, and high input impedance. Both hardware and M6800 software are shown for the DAN.

DATA ACQUISITION NETWORKS WITH NMOS AND CMOS

LSI technology is making it easier and less expensive to design and build complex electronic systems. This fact holds true for Data Acquisition Networks (DANs) due to the new single chip A/Ds and microprocessor systems. Thus, it is now feasible to build your own data acquisition network instead of buying a completed system, and thereby save money.

This article discusses an eight-channel DAN using the Motorola MC14433 CMOS A/D converter and the M6800 microprocessor. The number of channels can be expanded or reduced very simply. In addition to the eight channel DAN the program for a single channel system is shown. The inputs to the system, positive or negative polarity, are multiplexed with a CMOS analog multiplexer.

MC14433 A/D CONVERTER

The MC14433 is a single chip 3½ digit A/D converter using a modified dual ramp technique of A/D conversion. Housed in a 24 pin package it features auto-polarity, autozero and a high input impedance. Figure 1 shows the pin diagram of the MC14433.

The output of the MC14433 is 3½ digit multiplexed BCD with the MSD containing not only the half digit but also polarity of the input, overrange and underrange information. Figure 2 describes the decoding for the MSD. The digit selects for the multiplexed BCD have interdigit blanking to ensure correct BCD data during the time that the digit select is true.

The A/D converter is ratiometric and requires an external reference voltage. This reference voltage is 2.000 volts for the 1.999 volt range and 200 mV for a 199.9 mV full scale input. Both the unknown and reference inputs and analog ground are high impedance inputs. Other external components required are clock resistor, integrator resistor and capacitor, and offset capacitor. Precision components are not required.

Of particular interest for the data acquisition systems are the display update (DU) and the end of conversion (EOC) pins. The EOC pin indicates the end of one conversion cycle and the start of the next conversion by a positive pulse one-half clock period long. The display update pin is an input to the chip which allows the data to be strobed into the output latches. If at least one positive edge is received prior to the ramp down cycle, new data is strobed to the display. In a stand alone A/D system, EOC is connected to DU.

Also of significance to the data acquisition network is the input polarity detection sequence for the MC14433. Polarity for the current conversion cycle is determined in the previous conversion cycle. Thus if the polarity is reversed, a second conversion cycle must be made in order to obtain a correct measurement.

The MC14433 requires two power supplies. The total voltage across the device must not exceed 18 volts. Pin 13 is the reference level for the output circuitry of the MC14433. If this pin is tied to 0 volts, the BCD output, digit select and EOC will swing from 0 volts to V_{DD}. If however, pin 13 is tied to V_{EE}, the output swing will be from V_{EE} to V_{DD}.

The clock for the MC14433 is internal to the chip, requiring only a single external resistor to set the frequency. An external clock may be used by driving pin 10.

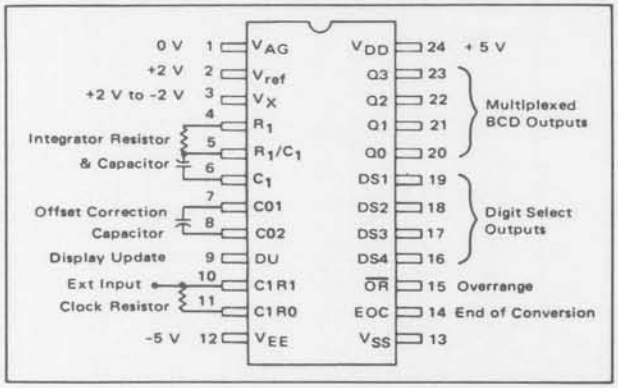


FIGURE 1 – MC14433 Pin Assignment

TRUTH TABLE

Coded Condition of MSD	Q3	Q2	Q1	Q0	BCD to 7 Segment Decoding
+0	1	1	1	0	Blank
−0	1	0	1	0	Blank
+0 UR	1	1	1	1	Blank
−0 UR	1	0	1	1	Blank
+1	0	1	0	0	4 → 1 } Hook up
−1	0	0	0	0	0 → 1 } only seg b
+1 OR	0	1	1	1	7 → 1 } and c to
−1 OR	0	0	1	1	3 → 1 } MSD

Notes for Truth Table

Q3 – ½ digit, low for "1", high for "0"

Q2 – Polarity: "1" = positive, "0" = negative

Q0 – Out of range condition exists if Q0 = 1. When used in conjunction with Q3 the type of out of range condition is indicated, i.e., Q3 = 0 → OR or Q3 = 1 → UR.

When only segment b and c of the decoder are connected to the ½ digit of the display, 4, 0, 7 and 3 appear as 1.

FIGURE 2 – MSD Coding

The total conversion time for the MC14433 is approximately 16400 clock periods. This conversion time includes the auto-zero cycle and the unknown input measurement cycle. The clock frequency may be operated up to about 400 kHz producing a conversion time of 40 ms.

MPU

The Motorola microprocessor system devices used are the MC6800 MPU, MCM6810 RAM, MCM6830 ROM and MC6820 PIA (peripheral interface adapter). The following is a brief description of the basic MPU system as it pertains to the A/D systems presented later in this application note.

The Motorola MPU system uses a 16-bit address bus and an 8-bit data bus. The 16-bit address bus provides 65,536 possible memory locations which may be either storage devices (RAM, ROM, etc.) or interface devices (PIA, etc.). The basic MPU contains two 8-bit accumulators, one 16-bit index register, a 16-bit program counter, a 16-bit stack pointer, and an 8-bit condition code register. The condition code register indicates carry, half carry, interrupt, zero, minus, and 2's complement overflow. Figure 3 shows a functional block of the MC6800 MPU.

The MPU uses 72 instructions with six addressing modes which provide 197 different operations in the MPU. A summary of each instruction and function with the appropriate addressing mode is shown in the MC6800 data sheet.

The RAMs used in the system are static and contain 128 8-bit words for scratch pad memory while the ROM is mask programmable and contains 1024 8-bit words. The ROM and RAM, along with the remainder of the MPU system components, operate from a single +5 volt power supply; the address bus, data bus and PIAs are TTL compatible.

The MPU system requires a 2ϕ non-overlapping clock such as the MC6875* with a lower frequency limit of 100 kHz and an upper limit of 1 MHz.

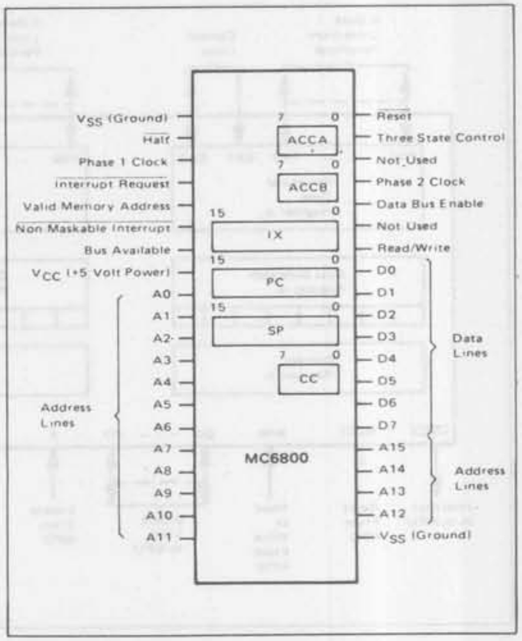


FIGURE 3 – MPU Pin Functions

The PIA is the interface device used between the address and data buses and the analog sections of the A/D. Each PIA contains two essentially identical 8-bit interface ports. These ports (A side, B side) each contain three internal registers that include the data register which is the interface from the data bus to the A/D, the data direction register which programs each of the eight lines of the data register as either an input or an output, and the control register which, in addition to other functions, switches the data bus between the data register and the data direction register. Each port to the PIA contains two addition pins, CA1 and CA2, for interrupt capability and extra I/O lines. The functions of these lines are programmable with the remaining bits of the control register. Figure 4 shows a functional block of the MC6820 PIA.

Each PIA requires four address locations in memory. Two addresses access either of the two (A or B sides) data/data direction registers while the remaining two addresses access either of the two control registers. These addresses are decoded by the chip select and register select lines of the PIA which are connected to the MPU address bus. Selection between the data register and data direction register is made by programming a "1" or "0" in the third least significant bit of each control register. A logic "0" accesses the data direction register while a logic "1" accesses the data register.

By programming "0"s in the data direction register each corresponding line performs as an input, while "1"s in the data direction register make corresponding lines act as outputs. The eight lines may be intermixed between inputs and outputs by programming different combinations of

*MC6875 to be introduced second quarter 1977

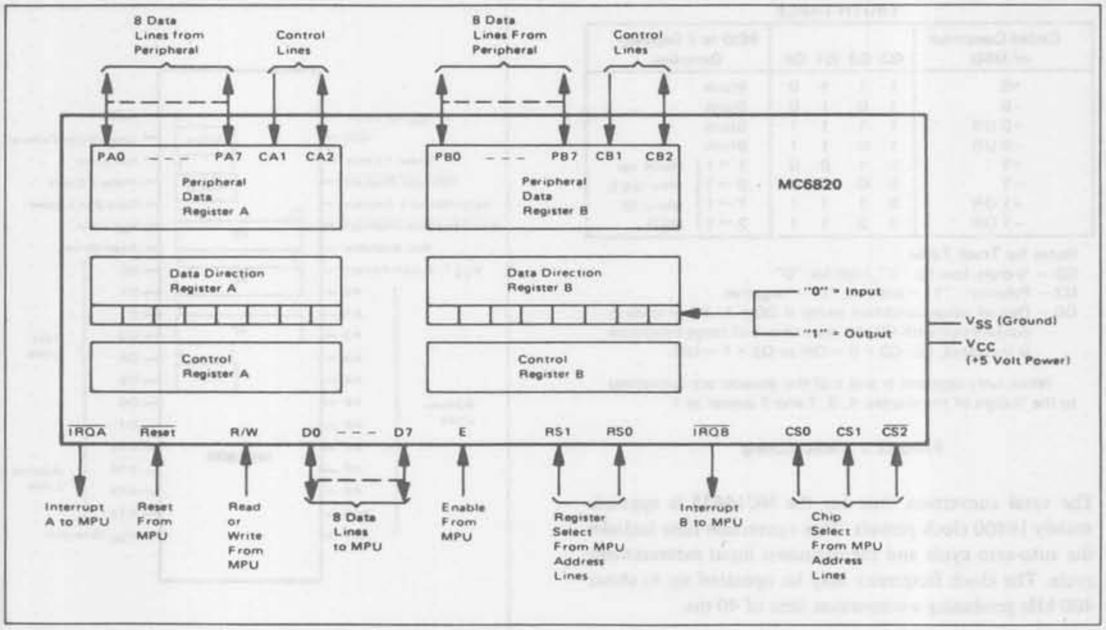


FIGURE 4 – PIA Functions

"1"s and "0"s into the data direction register. At the beginning of the program the I/O configuration is programmed into the data direction register, after which the control register is programmed to select the data register for I/O operation.

8-CHANNEL DATA ACQUISITION NETWORK

Figures 5 and 6 are the flow diagram for the 8-channel data acquisition network. Figure 5 shows the basic operation of the program while Figure 6 provides more detail on the A/D conversion routine. These flow diagrams relate to the actual software shown in Figure 8. The hardware required for the data acquisition is shown in Figure 9; as can be seen, it is fairly simple, consisting of the MC14433, MC1403* reference, MC14051B analog multiplexer, and an MC6820 PIA. The PIA is used as the interface between the microprocessor address and the data bus to the A/D. The microprocessor and associated memory are not shown due to a wide variety of forms possible depending upon the task that the total system is performing.

The reference for the MC14433 is an MC1403 bandgap reference which provides an output voltage of 2.5 volts. This voltage is divided down by the 20 kΩ pot to the 2.000 volt reference required by the MC14433. If a 200 mV reference is used, full scale for the DAN will be 199.9 mV.

The analog multiplexing required to handle the eight input channels is provided by a MC14051B CMOS multiplexer. This device selects one of eight inputs with a 3-bit binary code. The device is capable of switching dual polarity (plus or minus inputs) with a single polarity control voltage.

*MC1403 to be introduced first quarter 1977.

The MC14433 BCD output and digit select outputs are connected to the B side of the PIA as shown in lines 21-28 of the software routine. These lines of the software are comment lines only and do not result in code for the microprocessor. The B side data register of the PIA is labeled throughout the program as PIA1BD while the control register is labeled PIA1BC. The control I/O lines (CB1 and CB2) of the B side PIA are connected to EOC and DU of the MC1433.

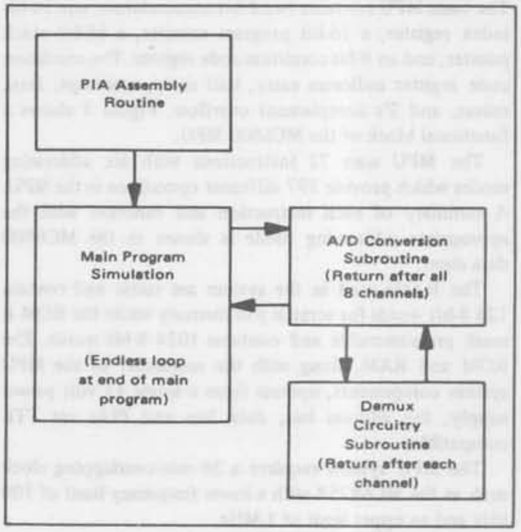

PIA Assembly Routine
Main Program Simulation
(Endless loop at end of main program)
A/D Conversion Subroutine (Return after all 8 channels)
Demux Circuitry Subroutine (Return after each channel)

FIGURE 5 – Basic Operation of 8 Channel DAN

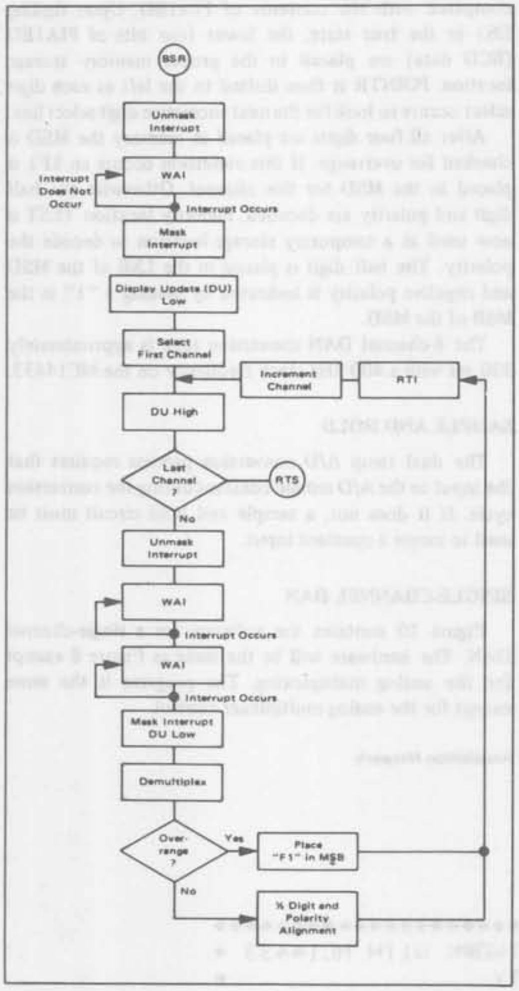


FIGURE 6 – A/D Conversion Subroutine Flow Chart

The first executable instruction for the program is in line 55 and starts a section called PIA assembly. The PIA sets the A side data register as all outputs and the B side data register as all inputs. From there the program goes to the main program simulation which, as its name implies, is a simulation of the user's main program. At such time in the user's program that some analog information is required, the A/D conversion subroutine starting in line 75 is executed. This routine synchronizes the program with the A/D conversion cycle and selects the first channel to be measured.

After the A/D conversion cycle for the first channel is completed the microprocessor is interrupted by the EOC of the MC14433. The interrupt program of line 88 is then executed; this demultiplexes the BCD output of the MC14433 and stores the data in memory. After completing he interrupt program the microprocessor returns to the A/D conversion subroutine and the next channel is selected. When the measurement of channel 2 is completed, the interrupt program is then executed and the resulting data stored away in memory. This procedure is repeated until all eight channels are read, after which the MPU returns to the main program. At this point the data obtained in the A/D conversion subroutine may be processed as required.

Looking at the software for the 8-channel data acquisition network in more detail, program storage of the final results begins in memory location $0010. Each BCD character is stored in the four LSBs of these memory locations. See Figure 7 for explanation of data storage. Each of the eight channel readings requires four memory

Channel Number	Memory Address	Digit	Data Example	Input Voltage
1	0010	MSD	01	1.729 V
	0011		07	
	0012		02	
	0013	LSD	09	
2	0014	MSD	F1	Overrange
	0015		09	
	0016		09	
	0017	LSD	09	
3	0018	MSD	08	-0.130 V
	0019		01	
	001A		03	
	001B	LSD	00	
4	001C	MSD	09	-1.130 V
	001D		01	
	001E		03	
	001F	LSD	00	
5	0020	MSD	00	0.000 V
	0021		00	
	0022		00	
	0023	LSD	00	
6	0024	MSD	01	1.000 V
	0025		00	
	0026		00	
	0027	LSD	00	
7	0028	MSD	F1	Overrange
	0029		09	
	002A		09	
	002B	LSD	09	
8	002C	MSD	09	-1.000 V
	002D		00	
	002E		00	
	002F	LSD	00	

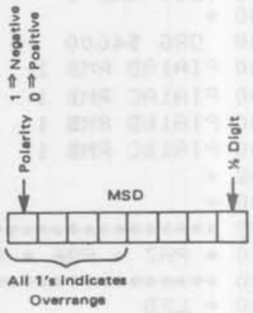


FIGURE 7 – Data Storage Definition

cycle the index register points to the MSD of that channel. This address is also stored at memory location called STORL.

Memory location TEST has two purposes; the first is for keeping track of which WAI was executed when the MPU was in the interrupt routine. This is required since more than one A/D conversion cycle is required for each channel. For the first channel three EOC pulses are required, while the remaining channels require only two A/D conversion cycles. The extra A/D conversion cycle in the first channel is used to synchronize the A/D converter to the MPU system. The second A/D conversion cycle in the first channel and the first conversion cycle of the remaining channels ensure that the polarity is correct for the current input. This is required since the MC14433 determines polarity in the previous conversion cycle.

Since the display update pin is edge triggered it must be taken high and low again in each conversion cycle when the data is to read by the MPU. The DU pin is taken high prior to the WAI for the measurement and low in the interrupt routine after the EOC occurs.

As mentioned previously, the multiplexed BCD data from the MC14433 is demultiplexed in the interrupt routine. A "1" is placed in bit 4 of POINTR which is select occurs to look for the next successive digit select line.

After all four digits are placed in memory the MSD is checked for overrange. If this condition occurs an $F1 is placed in the MSD for this channel. Otherwise the half digit and polarity are decoded. Memory location TEST is now used as a temporary storage location to decode the polarity. The half digit is placed in the LSB of the MSD and negative polarity is indicated by placing a "1" in the MSB of the MSD.

The 8-channel DAN conversion time is approximately 320 ms with a 400 kHz clock frequency on the MC14433.

SAMPLE AND HOLD

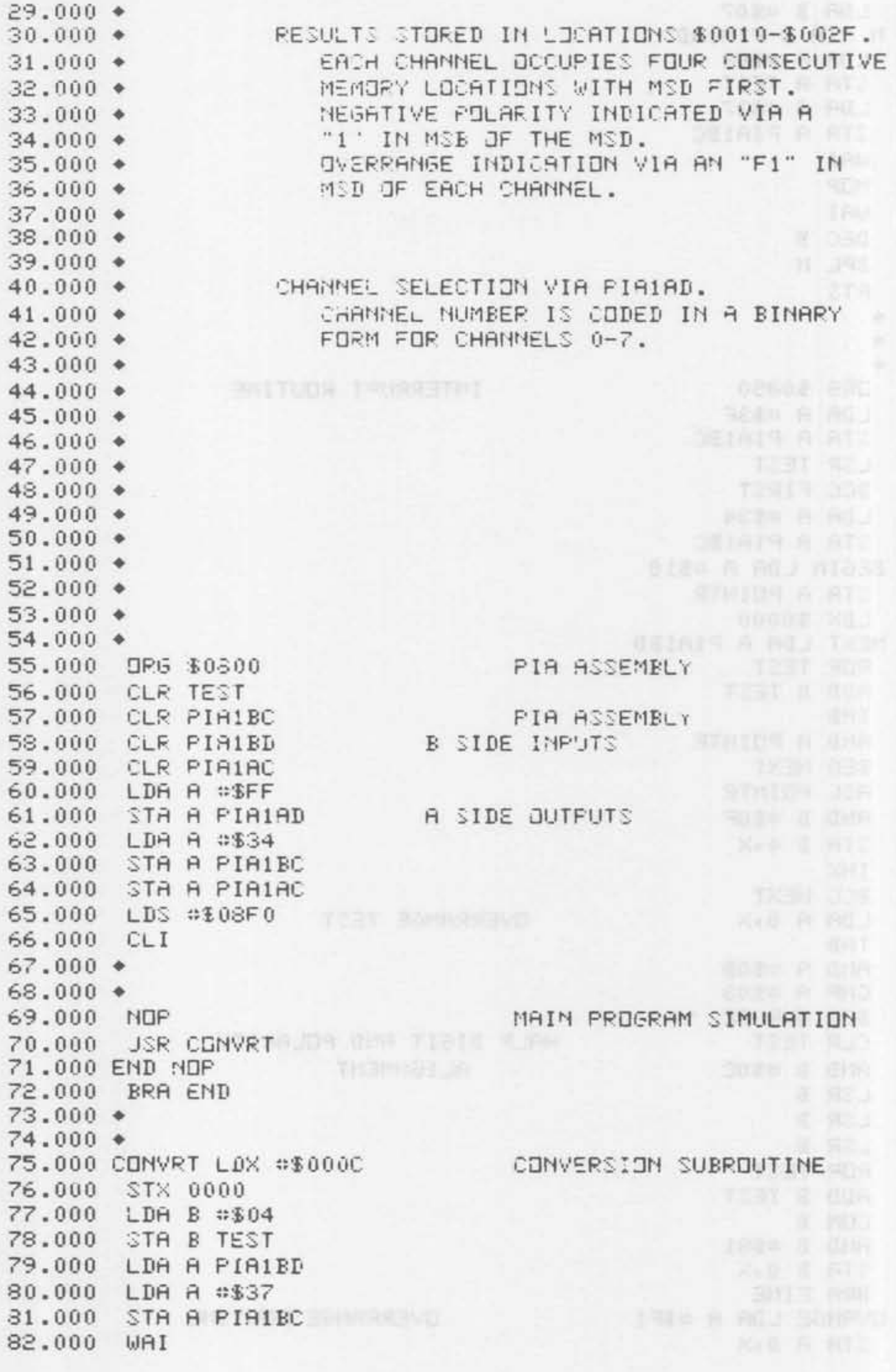
The dual ramp A/D conversion process requires that the input to the A/D remain constant during the conversion cycle. If it does not, a sample and hold circuit must be used to insure a constant input.

SINGLE-CHANNEL DAN

Figure 10 contains the software for a single-channel DAN. The hardware will be the same as Figure 8 except for the analog multiplexing. The program is the same except for the analog multiplexer control.

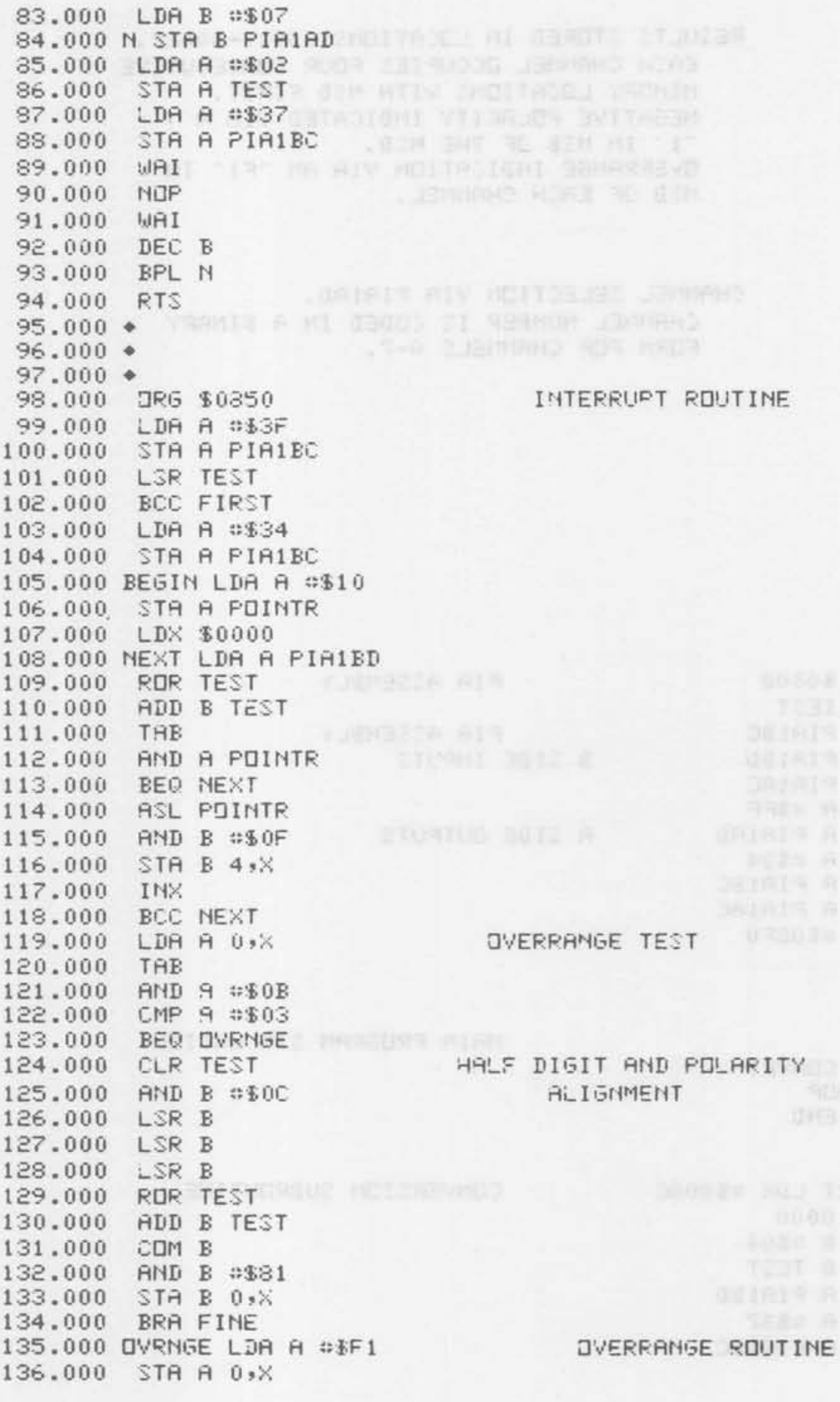
FIGURE 8 – 8-Channel Data Acquisition Network

```
 NAM DWA36
 OPT MEM,OTAPE
◆
◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆ 8 CHANNEL DATA AQUISTION NETWORK WITH MC14433 ◆
◆             WITH AUTOPOLARITY                 ◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆
 ORG $0000
STORL RMB 2         POINTER FOR DATA STORAGE LOCATION
POINTR RMB 1                POINTER FOR DIGIT SELECT
TEST RMB 1
◆
 ORG $4000
PIAIAD RMB 1                   A SIDE,DATA REGISTER
PIAIAC RMB 1                   A SIDE,CONTROL REGISTER
PIAIBD RMB 1                   B SIDE,DATA REGISTER
PIAIBC RMB 1                   B SIDE,CONTROL REGISTER
◆
◆                   ◆◆PIA CONFIGURATION◆◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆ PA7 ◆ PA6 ◆ PA5 ◆ PA4 ◆ PA3 ◆ PA2 ◆ PA1 ◆ PA0 ◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆ LSD               MSD ◆ MSB               LSB ◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
◆     DIGIT SELECT      ◆         BCD           ◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```



```
29.000 *
30.000 *          RESULTS STORED IN LOCATIONS $0010-$002F.
31.000 *             EACH CHANNEL OCCUPIES FOUR CONSECUTIVE
32.000 *             MEMORY LOCATIONS WITH MSD FIRST.
33.000 *             NEGATIVE POLARITY INDICATED VIA A
34.000 *             "1" IN MSB OF THE MSD.
35.000 *             OVERRANGE INDICATION VIA AN "F1" IN
36.000 *             MSD OF EACH CHANNEL.
37.000 *
38.000 *
39.000 *
40.000 *          CHANNEL SELECTION VIA PIA1AD.
41.000 *             CHANNEL NUMBER IS CODED IN A BINARY
42.000 *             FORM FOR CHANNELS 0-7.
43.000 *
44.000 *
45.000 *
46.000 *
47.000 *
48.000 *
49.000 *
50.000 *
51.000 *
52.000 *
53.000 *
54.000 *
55.000  ORG $0800                  PIA ASSEMBLY
56.000  CLR TEST
57.000  CLR PIA1BC                 PIA ASSEMBLY
58.000  CLR PIA1BD           B SIDE INPUTS
59.000  CLR PIA1AC
60.000  LDA A #$FF
61.000  STA A PIA1AD         A SIDE OUTPUTS
62.000  LDA A #$34
63.000  STA A PIA1BC
64.000  STA A PIA1AC
65.000  LDS #$08F0
66.000  CLI
67.000 *
68.000 *
69.000  NOP                        MAIN PROGRAM SIMULATION
70.000  JSR CONVRT
71.000 END NOP
72.000  BRA END
73.000 *
74.000 *
75.000 CONVRT LDX #$000C           CONVERSION SUBROUTINE
76.000  STX 0000
77.000  LDA B #$04
78.000  STA B TEST
79.000  LDA A PIA1BD
80.000  LDA A #$37
81.000  STA A PIA1BC
82.000  WAI
```

FIGURE 8 – 8-Channel Data Acquisition Network



```
 LDA B #$07
N STA B PIA1AD
 LDA A #$02
 STA A TEST
 LDA A #$37
 STA A PIA1BC
 WAI
 NOP
 WAI
 DEC B
 BPL N
 RTS
*
*
*
 ORG $0850                          INTERRUPT ROUTINE
 LDA A #$3F
 STA A PIA1BC
 LSR TEST
 BCC FIRST
 LDA A #$34
 STA A PIA1BC
BEGIN LDA A #$10
 STA A POINTR
 LDX $0000
NEXT LDA A PIA1BD
 ROR TEST
 ADD B TEST
 TAB
 AND A POINTR
 BEQ NEXT
 ASL POINTR
 AND B #$0F
 STA B 4,X
 INX
 BCC NEXT
 LDA A 0,X                       OVERRANGE TEST
 TAB
 AND A #$0B
 CMP A #$03
 BEQ OVRNGE
 CLR TEST                     HALF DIGIT AND POLARITY
 AND B #$0C                        ALIGNMENT
 LSR B
 LSR B
 LSR B
 ROR TEST
 ADD B TEST
 COM B
 AND B #$81
 STA B 0,X
 BRA FINE
OVRNGE LDA A #$F1                    OVERRANGE ROUTINE
 STA A 0,X
```

FIGURE 8 — 8-Channel Data Acquisition Network

```
137.000 FINE STX STORL
138.000  RTI
139.000 FIRST LDA A PIA1BD    DUMMY LOAD TO CLR INTERRUPT
140.000  RTI
141.000  ORG $0350              HARDWARE INTERRUPT VECTOR
142.000  FDB $03F0
143.000  MON
```

FIGURE 8 – 8-Channel Data Acquisition Network

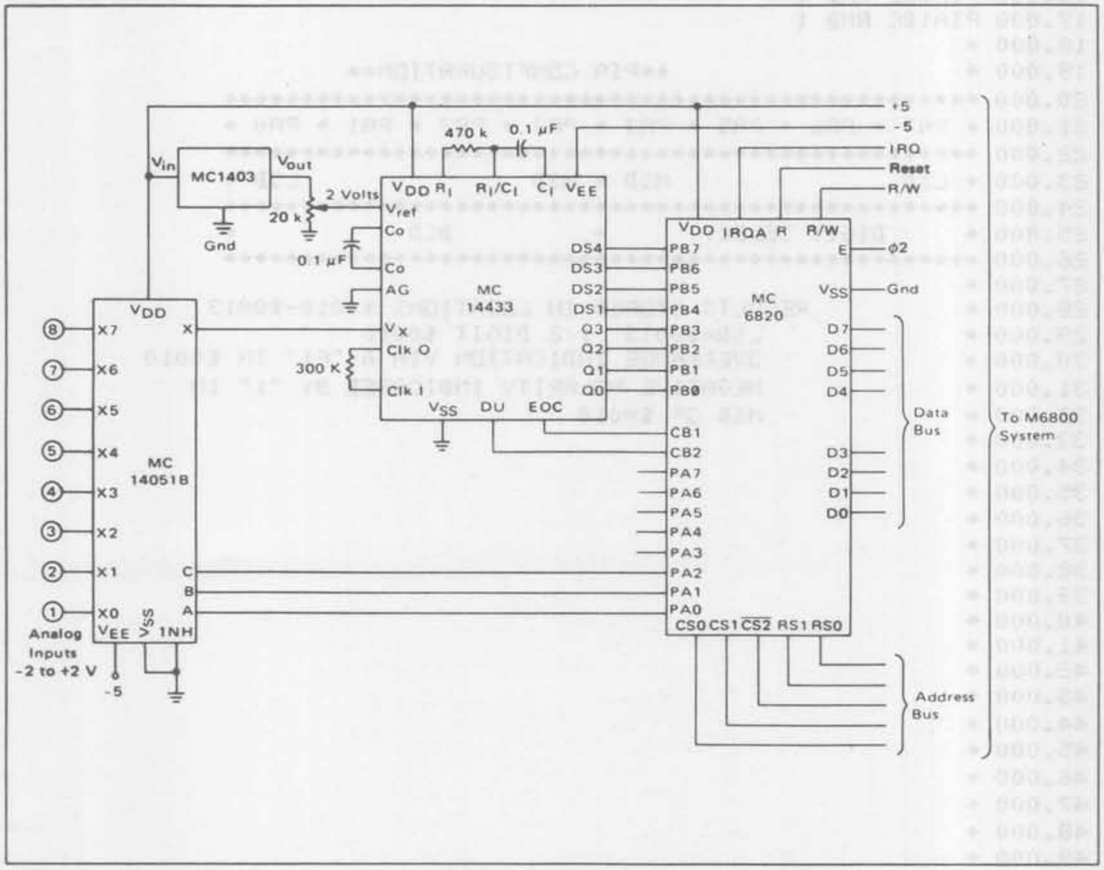
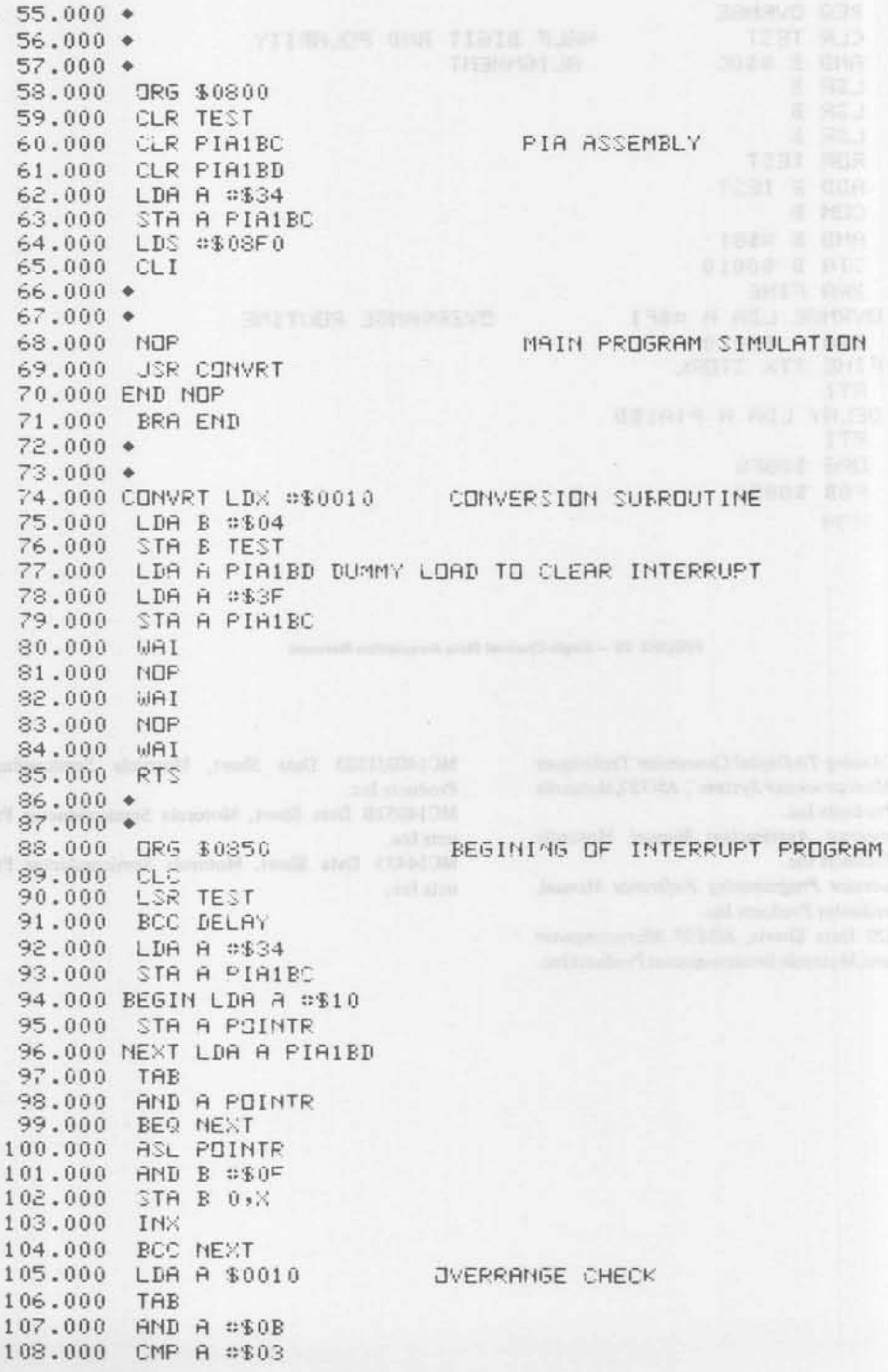


FIGURE 9 – 8-Channel Data Acquisition Hardware

```
 NAM DWA35
 OPT MEM,OTAPE
*
*                  *********
*                  *MC14433*
************************************************
* SINGLE CHANNEL DATA ACQUISITION NETWORK WITH *
*                 AUTOPOLARITY                 *
************************************************
*
 ORG $0002
POINTR RMB 1               POINTER FOR DIGIT SELECT
TEST RMB 1
*
 ORG $4002
PIA1BD RMB 1
PIA1BC RMB 1
*
*                   **PIA CONFIGURATION**
*************************************************
* PA7 * PA6 * PA5 * PA4 * PA3 * PA2 * PA1 * PA0 *
*************************************************
* LSD               MSD * MSB               LSB *
*************************************************
*     DIGIT SELECT      *         BCD           *
*************************************************
*
*          RESULTS STORED IN LOCATIONS $0010-$0013
*             LSD=$0013  1/2 DIGIT $0010
*             OVERRANGE INDICATION VIA A "F1" IN $0010
*             NEGATIVE POLARITY INDICATED BY "1" IN
*             MSB OF $0010
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
```

FIGURE 10 – Single-Channel Data Acquisition Network



```
◆
◆
◆
 ORG $0800
 CLR TEST
 CLR PIA1BC                      PIA ASSEMBLY
 CLR PIA1BD
 LDA A #$34
 STA A PIA1BC
 LDS #$08F0
 CLI
◆
◆
 NOP                             MAIN PROGRAM SIMULATION
 JSR CONVRT
END NOP
 BRA END
◆
◆
CONVRT LDX #$0010     CONVERSION SUBROUTINE
 LDA B #$04
 STA B TEST
 LDA A PIA1BD DUMMY LOAD TO CLEAR INTERRUPT
 LDA A #$3F
 STA A PIA1BC
 WAI
 NOP
 WAI
 NOP
 WAI
 RTS
◆
◆
 ORG $0850           BEGINING OF INTERRUPT PROGRAM
 CLC
 LSR TEST
 BCC DELAY
 LDA A #$34
 STA A PIA1BC
BEGIN LDA A #$10
 STA A POINTR
NEXT LDA A PIA1BD
 TAB
 AND A POINTR
 BEQ NEXT
 ASL POINTR
 AND B #$0F
 STA B 0,X
 INX
 BCC NEXT
 LDA A $0010          OVERRANGE CHECK
 TAB
 AND A #$0B
 CMP A #$03
```

FIGURE 10 – Single-Channel Data Acquisition Network

```
109.000  BEQ OVRNGE
110.000  CLR TEST             HALF DIGIT AND POLARITY
111.000  AND B #$0C            ALIGNMENT
112.000  LSR B
113.000  LSR B
114.000  LSR B
115.000  ROR TEST
116.000  ADD B TEST
117.000  COM B
118.000  AND B #$81
119.000  STA B $0010
120.000  BRA FINE
121.000 OVRNGE LDA A #$F1            OVERRANGE ROUTINE
122.000  STA A $0010
123.000 FINE STX STORL
124.000  RTI
125.000 DELAY LDA A PIA1BD
126.000  RTI
127.000  ORG $08F8
128.000  FDB $0850
129.000  MON
```

FIGURE 10 – Single-Channel Data Acquisition Network

REFERENCES

Aldridge, Don: *"Analog-To-Digital Conversion Techniques with the M6800 Microprocessor System"*, AN757, Motorola Semiconductor Products Inc.

M6800 Microprocessor Applications Manual, Motorola Semiconductor Products Inc.

M6800 Microprocessor Programming Reference Manual, Motorola Semiconductor Products Inc.

MC6800, MC6820 Data Sheets, *M6800 Microcomputer System Design Data*, Motorola Semiconductor Products Inc.

MC1403/1503 Data Sheet, Motorola Semiconductor Products Inc.

MC14051B Data Sheet, Motorola Semiconductor Products Inc.

MC14433 Data Sheet, Motorola Semiconductor Products Inc.

MOTOROLA

AN963

Interfacing The MC6108 A/D To a Microprocessor — It's Easier Than You Think!

Prepared by
Dennis R. Morgan
Motorola, Inc. Analog IC Division

INTRODUCTION

This application note will supplement information in the MC6108 data sheet by describing the detailed requirements for interfacing the Analog-to-Digital converter to a microprocessor. The hardware requirements, and the programming necessary to execute a conversion and read the data, in several different configurations, will be discussed. The microprocessor used in developing this application note is the MC6802 (operating off a 3.58 MHz crystal), a representative sample of the MC6800 family.

Because of the short conversion time of the MC6108, "Wait" states and "Wait for Interrupt" instructions are generally not needed with most microprocessors. The microprocessor can issue a CONVERT instruction, and immediately thereafter, issue a READ instruction, regardless of whether the MC6108 is read through a port (MC6821), or read off the bus directly.

MC6108 OPERATION

The MC6108 is a high-speed, 8-bit, A/D converter using the familiar Successive Approximation technique. Referring to the block diagram in Figure 1, the device includes the SAR, 8-bit DAC, comparator, 2.5 volt precision reference, matched resistors for +10, +5 and ±5 volt inputs, and control logic.

By connecting the internal temperature stable 2.5 volt reference to the Gain R pin, a reference current of ≈1 mA is supplied to the DAC. That current is gained up by x4 by the DAC, and then attenuated by the digital code from the SAR. The analog input signal is applied to R_{in} (for 0 to +10 volts), or R_{off} for 0 to +5 volts, and the current from that signal is compared with the DAC's output current by the comparator during the successive approximation process. The converter will accept a ±5 volt input by connecting the 2.5 volt reference to R_{off},

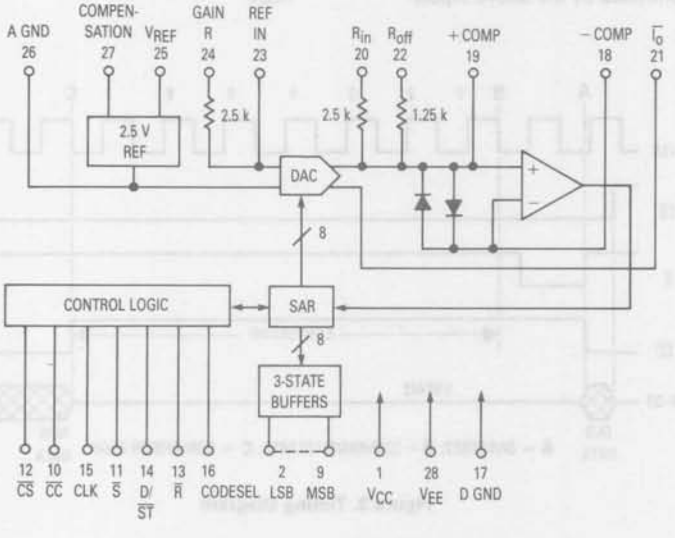


Figure 1. Block Diagram

and the input to R_{in}. Other input voltages can be accommodated by using external resistors at Ref In and +Comp, instead of the internal resistors, and grounding Gain R, R_{in}, and R_{off}. In the circuits tested for this application note, the analog side of the MC6108 was configured as shown in Figure 2.

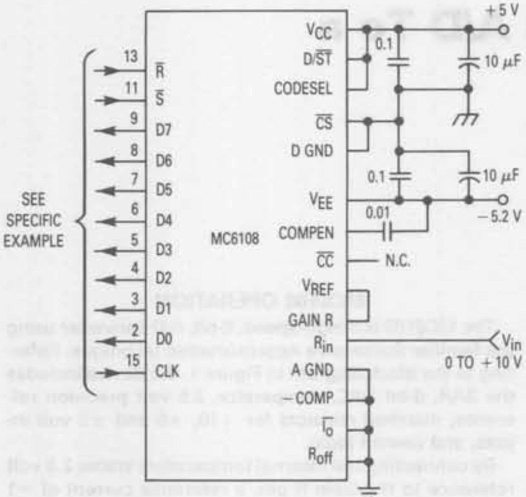


Figure 2. Analog Connections

Proper operation with a 5 MHz clock is guaranteed, although with careful attention to detail, clock rates as high as 10 MHz can be used. The control signals include Chip Select ($\overline{\text{CS}}$), Clock, Read ($\overline{\text{R}}$), Start ($\overline{\text{S}}$), Conversion Complete ($\overline{\text{CC}}$), Data/$\overline{\text{Status}}$ (D/$\overline{\text{ST}}$), and Code Select (CodeSel). The digital inputs and outputs are TTL compatible, with 3-state capability controlled by the above inputs.

Figure 3 shows the timing of the MC6108 during a conversion. The clock need not be synchronous with the other signals, and can run continuously. $\overline{\text{CS}}$ enables the device, and $\overline{\text{S}}$ must receive an active low pulse to initiate the conversion. The timing requirements are two: 1) $\overline{\text{CS}}$, $\overline{\text{S}}$, and Clock must be simultaneously low for a minimum of 50 ns (they may go low in any sequence); and 2) at least one low-to-high clock transition must occur during the $\overline{\text{S}}$ low time. After $\overline{\text{S}}$ switches high, the conversion starts on the next clock rising edge. The conversion requires 7 clock cycles thereafter.

$\overline{\text{CC}}$ (Conversion Complete) switches high at the beginning of the conversion process (when $\overline{\text{S}}$, $\overline{\text{CS}}$, and Clk are low) to indicate "busy," and switches low at the clock's rising edge corresponding to the end of the conversion. The data outputs (D7-D0) are in a high impedance (3-state) mode during the conversion, and go active (within 40 ns) after $\overline{\text{CC}}$ switches low. The outputs are also in the 3-state mode whenever $\overline{\text{CS}}$ is high.

Not shown in Figure 3 is the effects of the $\overline{\text{R}}$ ($\overline{\text{READ}}$) input. $\overline{\text{R}}$ affects the state of the outputs in that when $\overline{\text{R}}$ is low, the outputs are active (if $\overline{\text{CS}}$ is low and the MC6108 is not converting), and taking $\overline{\text{R}}$ high puts the outputs into the 3-state mode. The difference between $\overline{\text{R}}$ and $\overline{\text{CS}}$ is that $\overline{\text{CS}}$, when high, inhibits a conversion, whereas $\overline{\text{R}}$ does not affect the conversion process. Therefore, in the following examples where the MC6108 is connected directly to the microprocessor bus, $\overline{\text{CS}}$ is hard-wired low, $\overline{\text{S}}$ will initiate the conversions, and $\overline{\text{R}}$ will be controlled by the address decoder when data is to be read. Where the MC6108 is read through a port, $\overline{\text{R}}$ is hard-wired low.

ADDRESS DECODING

In order for the MC6802 microprocessor to read data from memory or memory-like devices (the MC6108 is a "read only" device), the timing requirements of Figure 5 must be satisfied. The requirements are that the data from the MC6108 must be valid while R/$\overline{\text{W}}$ is high, while VMA (Valid Memory Address) is high (which ensures the address to the decoder is valid), and while the E clock is high.

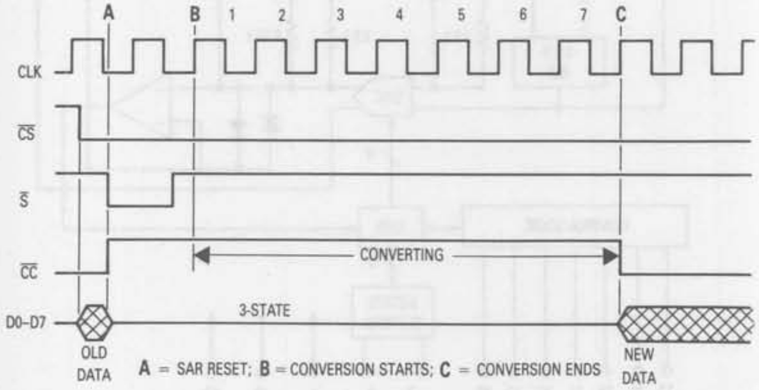


Figure 3. Timing Diagram

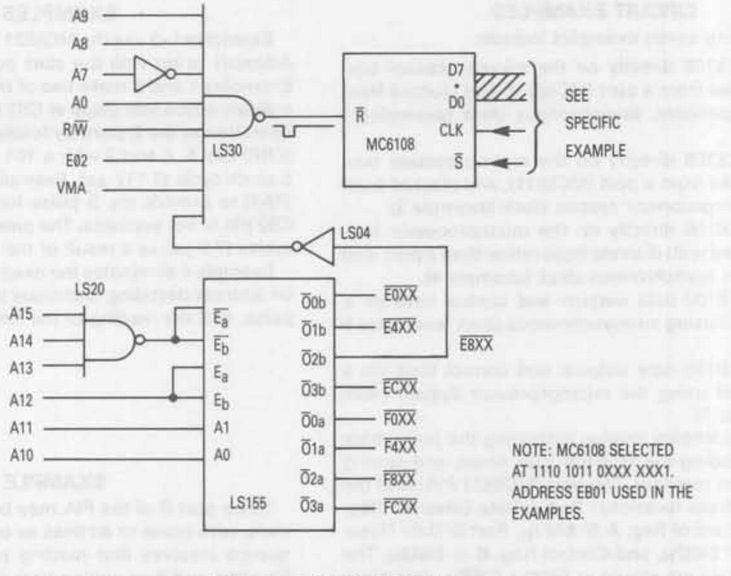


Figure 4. Address Decoder

In the microprocessor circuit used to develop the following examples, the address block $E800_H$–$EFFF_H$ was unused, and so the address $EB01_H$ was chosen for the MC6108. Since the entire block of 1K bytes was free, an incomplete decoding was possible, simplifying the decoder. Figure 4 is the schematic of the decoder, which uses three LS ICs, plus two inverters. Address lines A15 through A7, and A0, are used to activate the MC6108 at address $EB01_H$, although any address satisfying the code 1110 1011 0xxx xxx1 will work. If other addresses satisfying that code interfere with other devices in the system, then a more complete decoding involving A6–A1 is necessary.

R/W̄, E (phase 2 clock), and VMA are also included in the decoder to satisfy the requirements of the MC6802 microprocessor. The LS155 (along with the LS20) provides a decode (active low) for each of the 1K blocks from $E000_H$ through $FFFF_H$. The LS30 provides the rest of the decoding, and provides an active low output which is connected directly to the R̄ input on the MC6108.

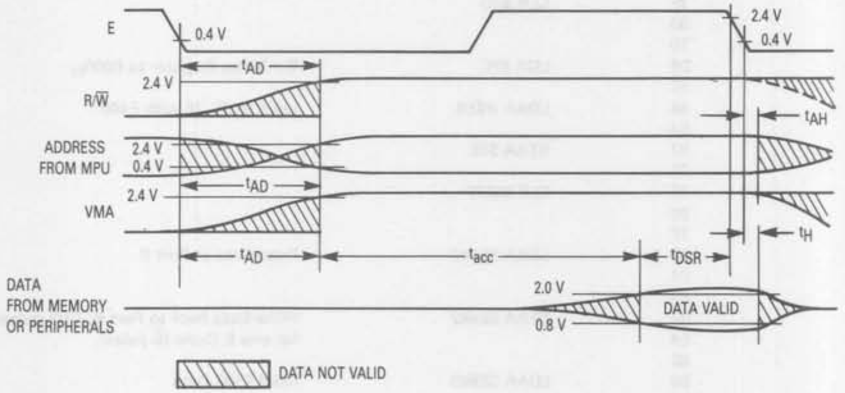


Figure 5. MC6802 Read Data From Memory

CIRCUIT EXAMPLES

The following seven examples include:

- The MC6108 directly on the microprocessor bus, controlled from a port (MC6821), and clocked from an independent, asynchronous clock (examples 1 and 2).
- The MC6108 directly on the microprocessor bus, controlled from a port (MC6821), and clocked from the microprocessor system clock (example 3).
- The MC6108 directly on the microprocessor bus, controlled with discrete logic rather than a port, and using an asynchronous clock (example 4).
- The MC6108 data outputs **and** control lines on a port, and using an asynchronous clock (examples 5 and 6).
- The MC6108 data outputs **and** control lines on a port, and using the microprocessor system clock (example 7).

All of the examples involve initializing the port where necessary, reading the MC6108 1024 times, and storing the 1K bytes in memory. The port (MC6821 PIA) is at the following address locations: Port A/Data Direction Reg. A @ E480$_H$, Control Reg. A @ E481$_H$, Port B/ Data Direction Reg. B @ E482$_H$, and Control Reg. B @ E483$_H$. The 1K bytes of data are stored at E000$_H$–E3FF$_H$, and those beginning and ending addresses are stored at 007C$_H$–007F$_H$ (for use with the Index Register). The addresses used for the instructions in the listings are for reference only.

EXAMPLES 1–4

Examples 1–3 use the MC6821 PIA (Peripheral Interface Adapter) to provide the start pulse ($\overline{S}$) to the MC6108. Examples 1 and 3 make use of the PIA's ability to output a single active low pulse at CB2 in response to a "write" operation to the B port, by loading the B control register (CRB) bits 5, 4 and 3 with a 101. The pulse width is one E clock cycle (1.117 μs). Example 2 uses bit 7 of port A (PA7) to provide the $\overline{S}$ pulse for those cases where the CB2 pin is not available. The pulse width is seven E clock cycles (7.8 μs) as a result of the instructions used.

Example 4 eliminates the need for a port, instead using an address decoding technique to provide both the Start pulse, and the reading of the data.

EXAMPLE #1

Since port B of the PIA may be used for other peripherals, with some or all lines as outputs, this program sequence involves first reading port B's Peripheral Data Register, and then writing back the same information, so as to not disturb the outputs. The write operation creates the pulse at the CB2. The data from the MC6108 is read immediately thereafter. See Figure 6 for the schematic, and Figure 7 for the timing involved.

Address	OpCode	Mnemonic	Notes
			Start Initialization-
01	86	LDAA #$2C	
02	2C		
03	B7	STAA $E483	Set CB2 to Output
04	E4		
05	83		
			-Start Read/Store Program-
06	86	LDAA #$E0	Load 007C, 7D with E000
07	E0		
08	97	STAA $7C	
09	7C		
0A	7F	CLR $7D	
0B	00		
0C	7D		
0D	DE	LDX $7C	Set Index Register to E000$_H$
0E	7C		
0F	86	LDAA #$E4	Load 007E, 7F with E400
10	E4		
11	97	STAA $7E	
12	7E		
13	7F	CLR $007F	
14	00		
15	7F		
16	B6	LDAA $E482	Read Data at Port B
17	E4		
18	82		
19	B7	STAA $E482	Write Data back to Port B; CB2 pulses low for one E Cycle ($\overline{S}$ pulse).
1A	E4		
1B	82		
1C	B6	LDAA $EB01	Read 6108 Data
1D	EB		
1E	01		
1F	A7	STAA $00,X	Store 6108 Data
20	00		

Address	OpCode	Mnemonic	Notes
21	08	INX	Increment Index Register
22	9C	CPX $7E	Compare Index Reg. with 7E/7F$_H$ (E400)
23	7E		
24	2C	BGE $03	Branch if ≥ 0 to $0029
25	03		
26	7E	JMP $0016	Jump to $0016 — Read Next Byte
27	00		
28	16		
29	??	??	1K Bytes stored — Next Instruction

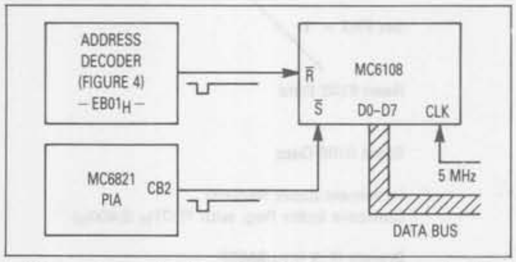


Figure 6. Reading Data Off The Bus — Example #1

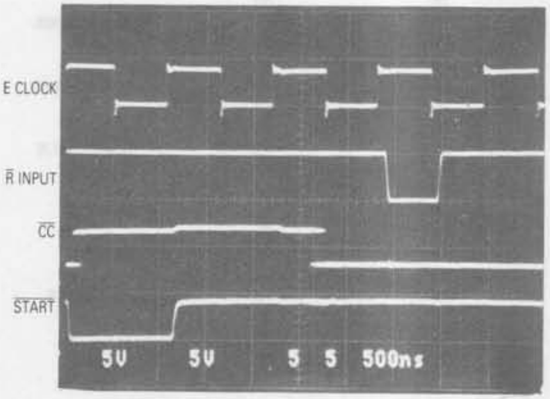


Figure 7. Example #1 Timing

EXAMPLE #2

This example involves using bit 7 of the PIA's port A (PA7) to provide the $\overline{S}$ pulse. The program sequence involves writing to the A port to bring PA7 low, then high, and then reading the data from the MC6108. See Figure 8 for the schematic, and Figure 9 for the timing involved.

Address	OpCode	Mnemonic	Notes
00	7F	CLR $E481	-Start Initialization- Access PIA's DDRA
01	E4		
02	81		
03	86	LDAA #$80	
04	80		
05	B7	STAA $E480	Set PA7 = Output (PA7 will provide $\overline{S}$ pulse)
06	E4		
07	80		
08	86	LDAA #$04	
09	04		
0A	B7	STAA $E481	Access Per. Data Reg. A
0B	E4		
0C	81		
0D	86	LDAA #$80	
0E	80		
0F	B7	STAA $E480	Set PA7 = 1
10	E4		
11	80		-End Initialization-
12	86	LDAA #$E0	-Start Read/Store Program- Load 007C, 7D with E000$_H$
13	E0		
14	97	STAA $7C	
15	7C		
16	7F	CLR $7D	
17	00		
18	7D		
19	DE	LDX $7C	Set Index Register to E000$_H$
1A	7C		

```
Address   OpCode   Mnemonic       Notes
1B        86       LDAA #$E4      Load 007E, 7F with E400H
1C        E4
1D        97       STAA $7E
1E        7E
1F        7F       CLR $007F
20        00
21        7F
22        7F       CLR $E480      Set PA7 = 0
23        E4
24        80
25        86       LDAA #$80                      - S̄ pulse
26        80
27        B7       STAA $E480     Set PA7 = 1
28        E4
29        80
2A        B6       LDAA $EB01     Read 6108 Data
2B        EB
2C        01
2D        A7       STAA $00,X     Store 6108 Data
2E        00
2F        08       INX            Increment Index Register
30        9C       CPX $7E        Compare Index Reg. with 7E/7FH (E400H)
31        7E
32        2C       BGE $03        Branch IF ≥ 0 to $0037
33        03
34        7E       JMP $0022      Jump to $0022 — Read Next Byte
35        00
36        22
37        ??       ?              1K Bytes stored — Next Instruction
```

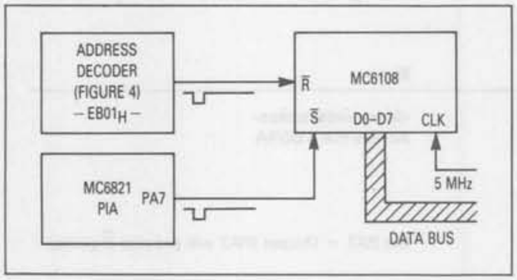


Figure 8. Reading Data Off The Bus — Example #2

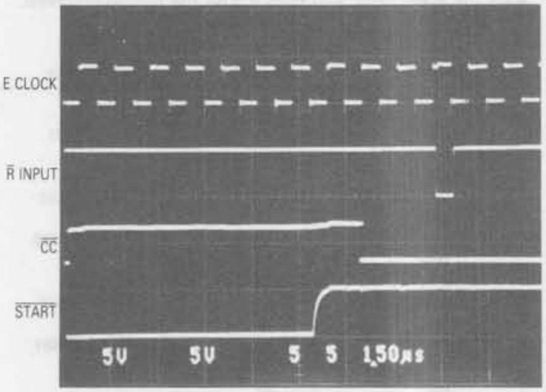


Figure 9. Example #2 Timing

EXAMPLE #3

This example is similar to example #1, except that the microprocessor's system clock (E) is used by the MC6108, rather than the faster 5 MHz clock. Because of the clock cycles required by the MC6108 to complete its conversion, 3 No Op instructions (6 clock cycles) are inserted between the start command (S̄ pulse), and the reading of the data. The program sequence involves reading port B, and then writing back the same information so as to not affect any outputs. The write operation creates the S̄ pulse at the CB2 output. The data is then read (after the 3 No Ops). See Figure 10 for the schematic, and Figure 11 for the timing involved.

```
Address        OpCode      Mnemonic          Notes

                                             -Start Initialization-
  01             86        LDAA #$2C
  02             2C
  03             B7        STAA $E483        Set CB2 to Output
  04             E4
  05             83
                                             -Start Read/Store Program-
  06             86        LDAA #$E0         Load 007C, 7D with E000
  07             E0
  08             97        STAA $7C
  09             7C
  0A             7F        CLR $7D
  0B             00
  0C             7D
  0D             DE        LDX $7C           Set Index Register to E000H
  0E             7C
  0F             86        LDAA #$E4         Load 007E, 7F with E400
  10             E4
  11             97        STAA $7E
  12             7E
  13             7F        CLR $007F
  14             00
  15             7F
  16             B6        LDAA $E482        Read Data at Port B
  17             E4
  18             82
  19             B7        STAA $E482        Write Data back to Port B; CB2 pulses low
  1A             E4                          for one E Cycle (S pulse)
  1B             82
  1C–1E          3x01      3 No Ops          MC6108 is converting
  1F             B6        LDAA $EB01        Read 6108 Data
  20             EB
  21             01
  22             A7        STAA $00,X        Store 6108 Data
  23             00
  24             08        INX               Increment Index Register
  25             9C        CPX $7E           Compare Index Reg. with 7E/7FH (E400)
  26             7E
  27             2C        BGE $03           Branch IF ≥ 0 to $002C
  28             03
  29             7E        JMP $0016         Jump to $0016 — Read Next Byte
  2A             00
  2B             16
  2C             ??        ??                1K Bytes stored — Next Instruction
```

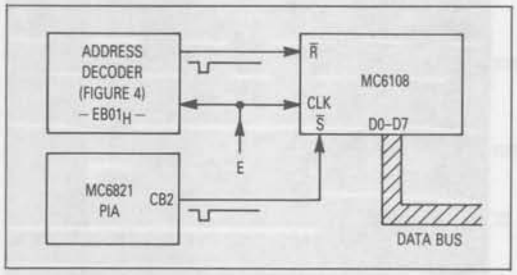


Figure 10. Reading Data Off The Bus — Example #3

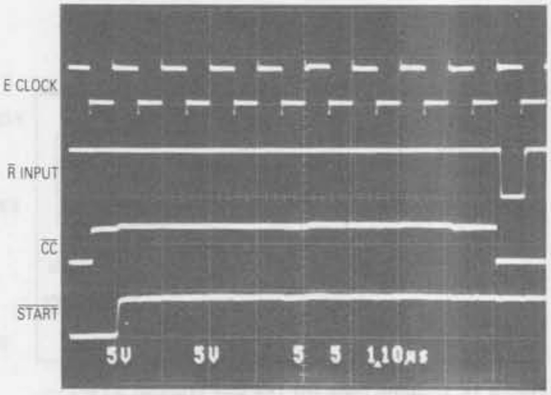


Figure 11. Example #3 Timing

EXAMPLE #4

This example does not use a PIA port for either the start ($\overline{S}$) or the Read ($\overline{R}$) operation, but instead uses some gates in addition to the address decoder of Figure 4. Address line A0 is removed from its place in Figure 4, and instead implemented as shown in Figure 12 so as to provide decoding at two addresses ($EB00_H$ and $EB01_H$). The microprocessor is made to read address $EB00_H$, creating an active low pulse at $\overline{S}$. The width of the pulse is 0.56 µs (1/2 E clock cycle), which is wide enough when using a 5 MHz clock for the MC6108. Due to the requirements of the MC6108, the minimum clock frequency usable in this configuration is 2x E clock frequency. Reading the data, by means of the $\overline{R}$ input is the same as in the previous examples. The two LS04 inverters in series with the lower OR gate provide a propagation delay to allow for the propagation delay of the address decoder block, thus preventing glitches at the $\overline{S}$ input. Figure 13 shows the timing involved.

Address	OpCode	Mnemonic	Notes
			-Start Read/Store Program-
00	86	LDAA #$E0	Load 007C, 7D with $E000_H$
01	E0		
02	97	STAA $7C	
03	7C		
04	7F	CLR $7D	
05	00		
06	7D		
07	DE	LDX $7C	Set Index Register to $E000_H$
08	7C		
09	86	LDAA #$E4	Load 007E, 7F with $E400_H$
0A	E4		
0B	97	STAA $7E	
0C	7E		
0D	7F	CLR $007F	
0E	00		
0F	7F		
10	B6	LDAA $EB00	Read $EB00 — create $\overline{S}$ pulse through Address Decode Logic
11	EB		
12	00		
13	B6	LDAA $EB01	Read 6108 Data
14	EB		
15	01		
16	A7	STAA $00,X	Store 6108 Data
17	00		
18	08	INX	Increment Index Register
19	9C	CPX $7E	Compare Index Reg. with $7E/7F_H$ ($E400_H$)
1A	7E		
1B	2C	BGE $03	Branch IF ≥ 0 to $0020
1C	03		
1D	7E	JMP $0010	Jump to $0010 — Read Next Byte
1E	00		
1F	10		
20	??	?	1K Bytes stored — Next Instruction

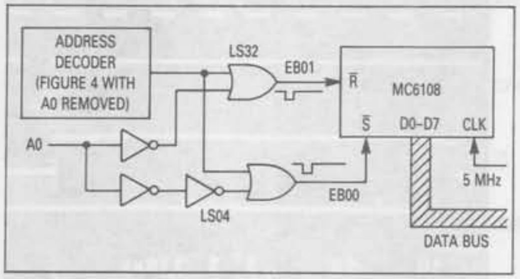


Figure 12. Reading Data Off The Bus Without a Port — Example #4

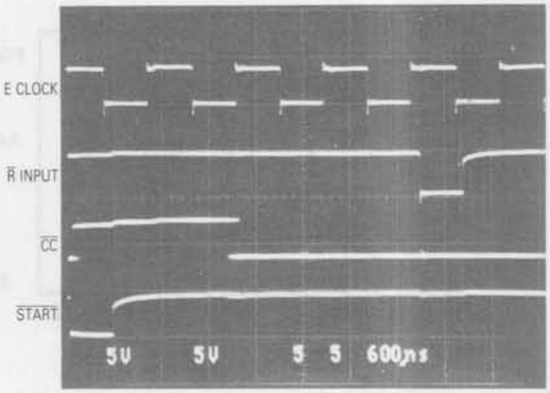


Figure 13. Example #4 Timing

EXAMPLES 5–7

Examples 5–7 use the MC6821 PIA for reading the data (through its B port) from the MC6108, as well as for the control. Examples 5 and 7 make use of the PIA's ability to output a single active low pulse at CB2 in response to a "write" operation to the B port, by loading the B control register (CRB) bits 5, 4, and 3 with a 101. The pulse width is one E clock cycle (1.117 μs). Example 6 uses bit 7 of port A (PA7) to provide the $\overline{S}$ pulse for those cases where the CB2 pin is not available. The pulse width is seven E clock cycles (7.8 μs) as a result of the instructions used.

EXAMPLE #5

The program sequence for this example is to write a 00_H to the B port to create the pulse at CB2 (writing to inputs does not affect them), and then reading the same port to obtain the MC6108's data. The conversion sequence requires one No Op before the read instruction due to the setup time required by the PIA. See Figure 14 for the schematic, and Figure 15 for the timing involved.

```
Address   OpCode   Mnemonic        Notes
---------------------------------------------------------------------------------
                                   -Start Initialization-
  01        7F     CLR $E483       Access PIA's DDRB
  02        E4
  03        83
  04        7F     CLR $E482       Set PB7-0 = Inputs            Initialize
  05        E4                                                    PIA
  06        82
  07        86     LDAA #$2C
  08        2C
  09        B7     STAA $E483      Access Per. Data Reg. B,
  0A        E4                     Set CB2 to Output
  0B        83                     -End Initialization-
                                   -Start Read/Store Program-
  10        86     LDAA #$E0       Load 007C, 7D with E000H
  11        E0
  12        97     STAA $7C
  13        7C
  14        7F     CLR $7D
  15        00
  16        7D
  17        DE     LDX $7C         Set Index Register to E000H
  18        7C
  19        86     LDAA #$E4       Load 007E, 7F with E400H
  1A        E4
  1B        97     STAA $7E
  1C        7E
  1D        7F     CLR $007F
  1E        00
  1F        7F
  20        7F     CLR $E482       Write 00H to Port B; CB2 pulses low for
  21        E4                     one E Cycle (S̄ pulse)
  22        82
  23        01     No Op           Required for PIA's setup time
  24        B6     LDAA $E482      Read port B (Read MC6108)
  25        E4
  26        82
  27        A7     STAA $00,X      Store Port B Data
  28        00
  29        08     INX             Increment Index Reg.
  2A        9C     CPX $7E         Compare Index Reg. with 7E/7FH (E400H)
  2B        7E
  2C        2C     BGE $03         Branch IF ≥ 0 to $0031H
  2D        03
  2E        7E     JMP $0020       Jump to $0020H — Read Next Byte
  2F        00
  30        20
  31        ??     ?               1K Bytes stored — Next Instruction
```

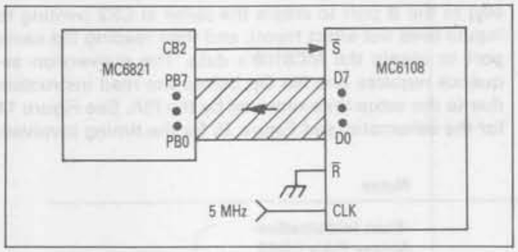


Figure 14. Reading Data Through a Port — Example #5

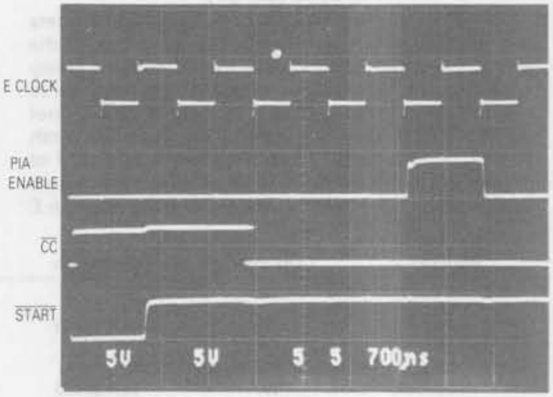


Figure 15. Example #5 Timing

EXAMPLE #6

The program sequence for the conversion is to write to the A port to bring PA7 low, and then high, and then read the B port to obtain the MC6108's data. The conversion occurs within the cycle time between PA7 switching high and reading the data, requiring no WAIT or NO-OP instructions in between. See Figure 16 for the schematic, and Figure 17 for the timing involved.

Address	OpCode	Mnemonic	Notes
			-Start Initialization-
01	7F	CLR $E481	Access PIA's DDRA
02	E4		
03	81		
04	86	LDAA #$80	
05	80		
06	B7	STAA $E480	Set PA7 = Output (PA7 will provide $\overline{S}$ pulse)
07	E4		
08	80		
09	86	LDAA #$04	
0A	04		
0B	B7	STAA $E481	Access Per. Data Reg. A
0C	E4		
0D	81		
0E	86	LDAA #$80	
0F	80		
10	B7	STAA $E480	Set PA7 = 1
11	E4		
12	80		
13	7F	CLR $E483	Access PIA's DDRB
14	E4		
15	83		
16	7F	CLR $E482	Set PB7–0 = Inputs
17	E4		
18	82		
19	86	LDAA #$04	
1A	04		
1B	B7	STAA $E483	Access Per. Data Reg. B
1C	E4		
1D	83		**-End Initialization-**
			-Start Read/Store Program-
20	86	LDAA #$E0	Load 007C, 7D with E000$_H$
21	E0		|
22	97	STAA $7C	|
23	7C		|

Address	OpCode	Mnemonic	Notes
24	7F	CLR $7D	
25	00		
26	7D		
27	DE	LDX $7C	Set Index Register to E000$_H$
28	7C		
29	86	LDAA #$E4	Load 007E, 7F with E400$_H$
2A	E4		
2B	97	STAA $7E	
2C	7E		
2D	7F	CLR $007F	
2E	00		
2F	7F		
30	7F	CLR $E480	SET PA7 = 0
31	E4		
32	80		
33	86	LDAA #$80	− S̄ pulse
34	80		
35	B7	STAA $E480	Set PA7 = 1
36	E4		
37	80		
38	B6	LDAA $E482	Read Port B (Read MC6108)
39	E4		
3A	82		
3B	A7	STAA $00,X	Store Port B Data
3C	00		
3D	08	INX	Increment Index Reg.
3E	9C	CPX $7E	Compare Index Reg. with 7E/7F$_H$ (E400$_H$)
3F	7E		
40	2C	BGE $03	Branch IF ≥ 0 to \$0045$_H$
41	03		
42	7E	JMP $0030	Jump to \$0030$_H$ — Read Next Byte
43	00		
44	30		
45	?	?	1K Bytes stored ← Next Instruction

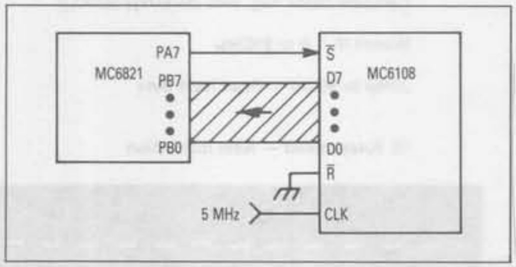


Figure 16. Reading Data Through a Port — Example #6

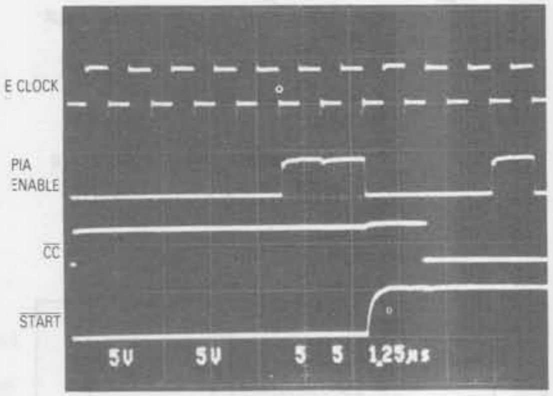


Figure 17. Example #6 Timing

EXAMPLE #7

The program sequence for this example (using the system E clock rather than a 5 MHz clock for the MC6108) is to write a 00$_H$ to the B port to create the S̄ pulse at CB2 (writing to inputs does not affect them), and then reading the same port to obtain the MC6108's data. Between those instruction 4 No Ops are required to give the MC6108 the necessary clock cycles to finish the conversion. See Figure 18 for the schematic, and Figure 19 for the timing involved.

Address	OpCode	Mnemonic	Notes	
			-Start Initialization-	
01	7F	CLR $E483	Access PIA's DDRB	
02	E4			
03	83			
04	7F	CLR $E482	Set PB7–0 = Inputs	
05	E4			Initialize PIA
06	82			
07	86	LDAA #$2C		
08	2C			
09	B7	STAA $E483	Access Per. Data Reg. B,	
0A	E4		Set CB2 to Output	
0B	83		**-End Initialization-**	
			-Start Read/Store Program-	
10	86	LDAA #$E0	Load 007C, 7D with $E000_H$	
11	E0			
12	97	STAA $7C		
13	7C			
14	7F	CLR $7D		
15	00			
16	7D			
17	DE	LDX $7C	Set Index Register to $E000_H$	
18	7C			
19	86	LDAA #$E4	Load 007E, 7F with $E400_H$	
1A	E4			
1B	97	STAA $7E		
1C	7E			
1D	7F	CLR $007F		
1E	00			
1F	7F			
20	7F	CLR $E482	Write 00_H to Port B; CB2 pulses low for one E Cycle ($\overline{S}$ pulse)	
21	E4			
22	82			
23–26	4x01	4 No Ops	MC6108 is converting	
27	B6	LDAA $E482	Read Port B (Read MC6108)	
28	E4			
29	82			
2A	A7	STAA $00,X	Store Port B Data	
2B	00			
2C	08	INX	Increment Index Reg.	
2D	9C	CPX $7E	Compare Index Reg. with $007E/7F_H$ ($E400_H$)	
2E	7E			
2F	2C	BGE $03	Branch IF ≥ 0 to $\$0034_H$	
30	03			
31	7E	JMP $004B	Jump to $0020 — Read Next Byte	
32	00			
33	20			
34	?	?	1K Bytes stored — Next Instruction	

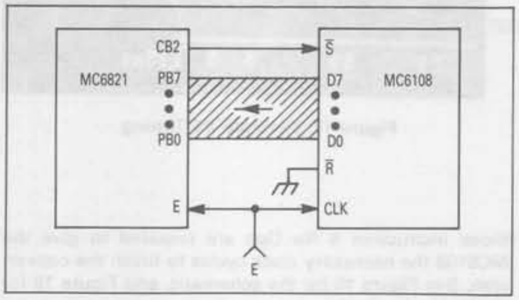


Figure 18. Reading Data Through a Port — Example #7

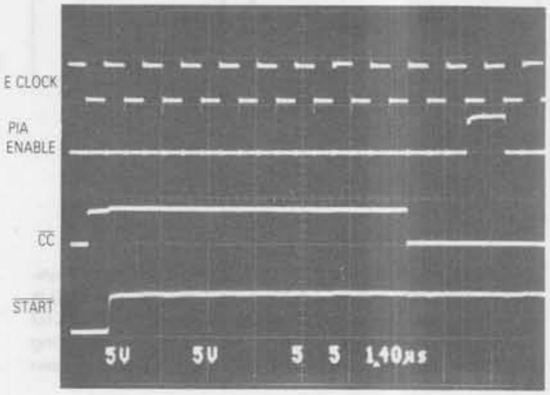


Figure 19. Example #7 Timing

OTHER EXAMPLES

Figure 20 illustrates a method for controlling several MC6108 A/D converters. The four devices are permanently enabled ($\overline{CS}$ = low) as shown in Figure 2. The PIA is set up to output a single active low pulse at the CB2 pin, as described prior to Example #1 in this application note, to initiate the conversion. The convert command ($\overline{S}$ pulse) is provided to all four converters simultaneously. The address decoder, composed of the 74LS30, LS04s, LS11, and LS155, results in one of the four converters being read by the microprocessor, depending on which address (EB00 through EB03) is selected. It should be noted that the decoder shown in this example is incomplete, as address lines A7-A2 are not included. Each individual application will determine the need for more complete decoding.

Some variations of the circuit shown are:

1) Extend the decoder to include address line A2, and use one line of the other half of the LS155 (only the "A" half is shown) to provide the $\overline{S}$ pulse to the converters, eliminating the need for the PIA.
2) Use the other half of the LS155 to provide individual $\overline{S}$ pulses to each converter.

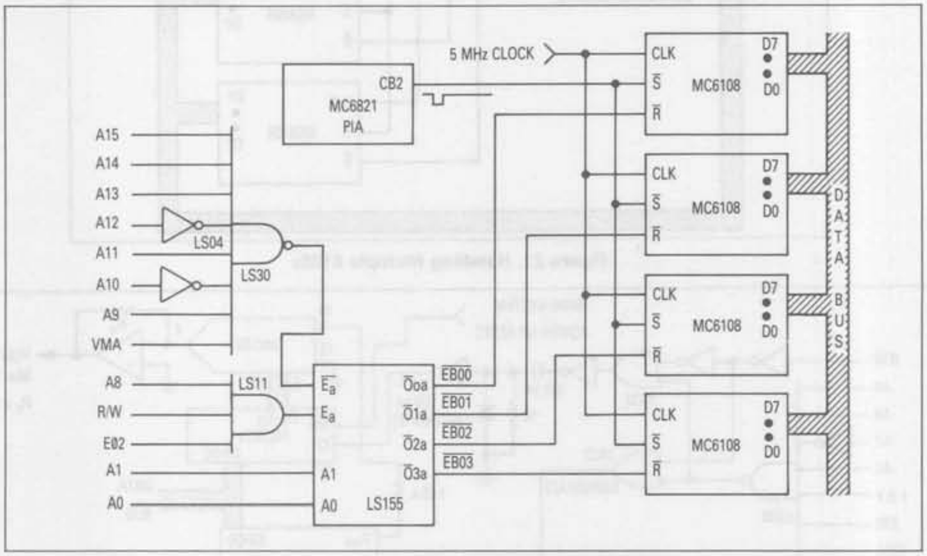


Figure 20. Handling Multiple MC6108s

Figure 21 illustrates an additional configuration for controlling several MC6108 A/D converters. In this case, the MC6821 PIA is used to both initiate the conversion, and read the data. The active low pulse at CB2 is provided to all the converters simultaneously. The selected MC6108 is then activated by bringing its $\overline{READ}$ pin low, by means of the appropriate line at the A port. The data is then read through the B port, and then the $\overline{READ}$ input is taken high. The remaining pins of each MC6108 are connected as shown in Figure 2.

Figure 22 illustrates the circuitry for reading in an analog signal, processing it according to the system requirements, and then producing an analog signal out by means of the DAC-08 D/A converter. The digital data to be converted to analog is stored in the 74LS273 octal latch when its CP input receives an active low pulse from the address decoder. In Figure 22, the latch is considered a "write only" location at address EB01. On the rising edge of the CP pulse, the data is transferred to the DAC-08 by means of the Q outputs. The output of the DAC is a current proportional to the reference current and the digital data presented to it. The op amp converts that current to an output voltage by means of the feedback resistor R_X (Max $V_{out} = R_X \times 2$ mA).

The reference current for the DAC-08 is supplied from the MC6108's reference supply (V_{ref}). Settling time of the output voltage is approximately 1 μs with any of the currently available fast op amps, such as the MC34001 family, MC33070 family, MC34074 family, or the MC34080 family. The DAC-08 can be powered from the same +5 and −5.2 volt supplies used for the MC6108.

The MC6108 receives its Start command ($\overline{S}$ pulse) from an MC6821 PIA's CB2 output, as described in Example #1. Since the MC6108 can be considered a "Read Only" memory location, and the 74LS273 latch a "Write Only" location, they are placed at the same address (EB01 in Figure 22), but set to respond to Read and Write commands (LDAA and STAA for the MC6802 MPU). In Figure 22, the address decoder is the same as in Figure 4, except that the R/$\overline{W}$ line has been relocated. Reading the MC6108 is the same as in previous examples. When writing to the 74LS273 latch, the output pulse from the decoder (at the upper LS32 gate) is shortened to 300 ns to ensure that its rising edge occurs while data on the data bus is still valid.

CONCLUSION

The examples have shown that interfacing the MC6108 A/D converter to a microprocessor is a relatively simple process. The flexibility associated with the various control

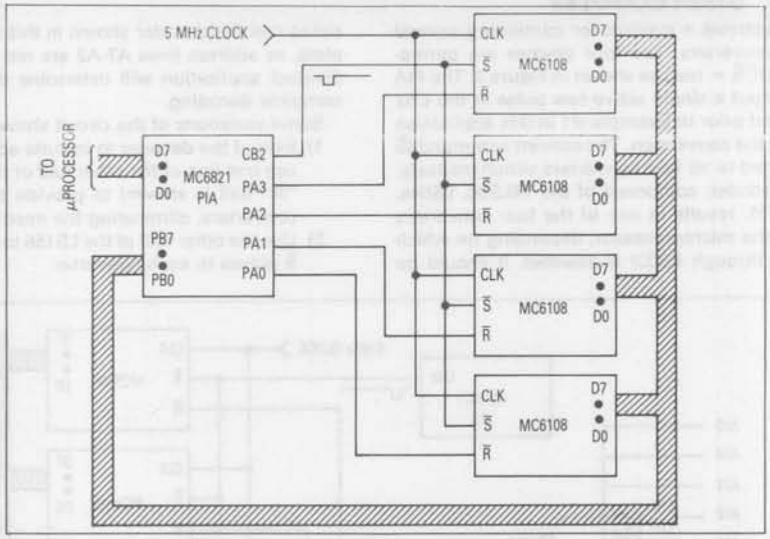
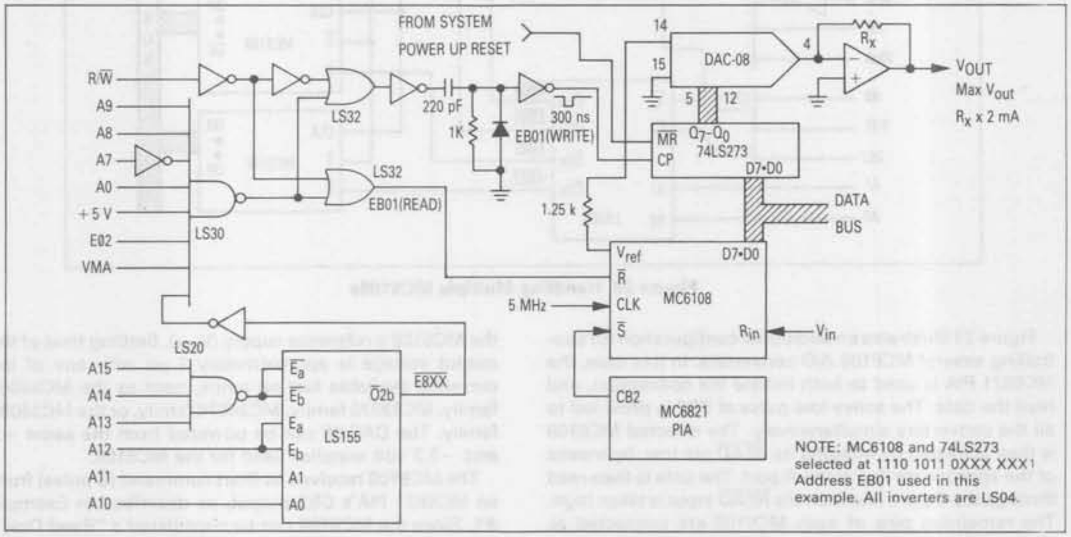


Figure 21. Handling Multiple 6108s



NOTE: MC6108 and 74LS273 selected at 1110 1011 0XXX XXX1 Address EB01 used in this example. All inverters are LS04.

Figure 22. The MC6108 A/D and DAC-08 D/A

lines allow several combinations of a port and address decoder to be used for controlling the converter, and for reading the data.

The examples indicate there is an inverse relationship between the amount of hardware and the length of programming required. Each individual application will determine the right combination of the two.

The MC6108's high speed (1.8 μs) facilitates programming the system since interrupts and long wait states are generally not required. In most cases, the READ instruction can immediately follow the CONVERT instructions.

REFERENCES

— MC6108 Data Sheet, 1986
— MC6802 Data Sheet, 1979
— MC6821 Data Sheet, 1978
— DAC-08 Data Sheet, 1986