1975

AN-757
Application Note

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

Prepared by:
Don Aldridge
Applications Engineering

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.

MOTOROLA *Semiconductor Products Inc.*

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.

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.

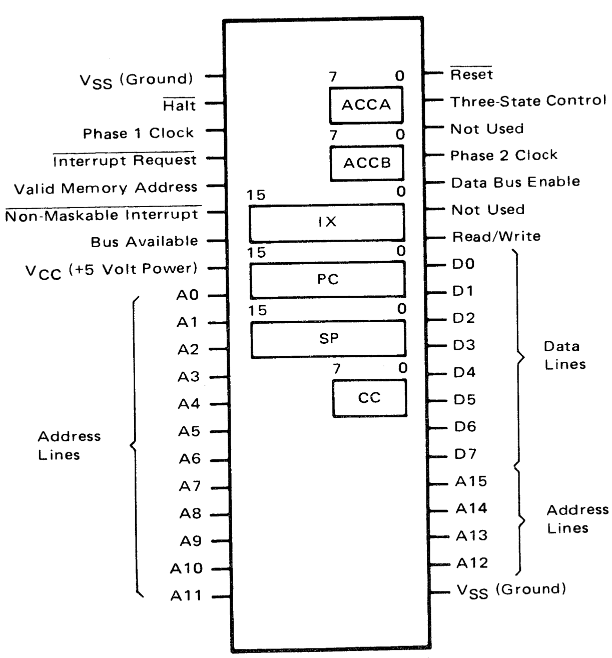


FIGURE 2 – MPU Pin Functions

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.

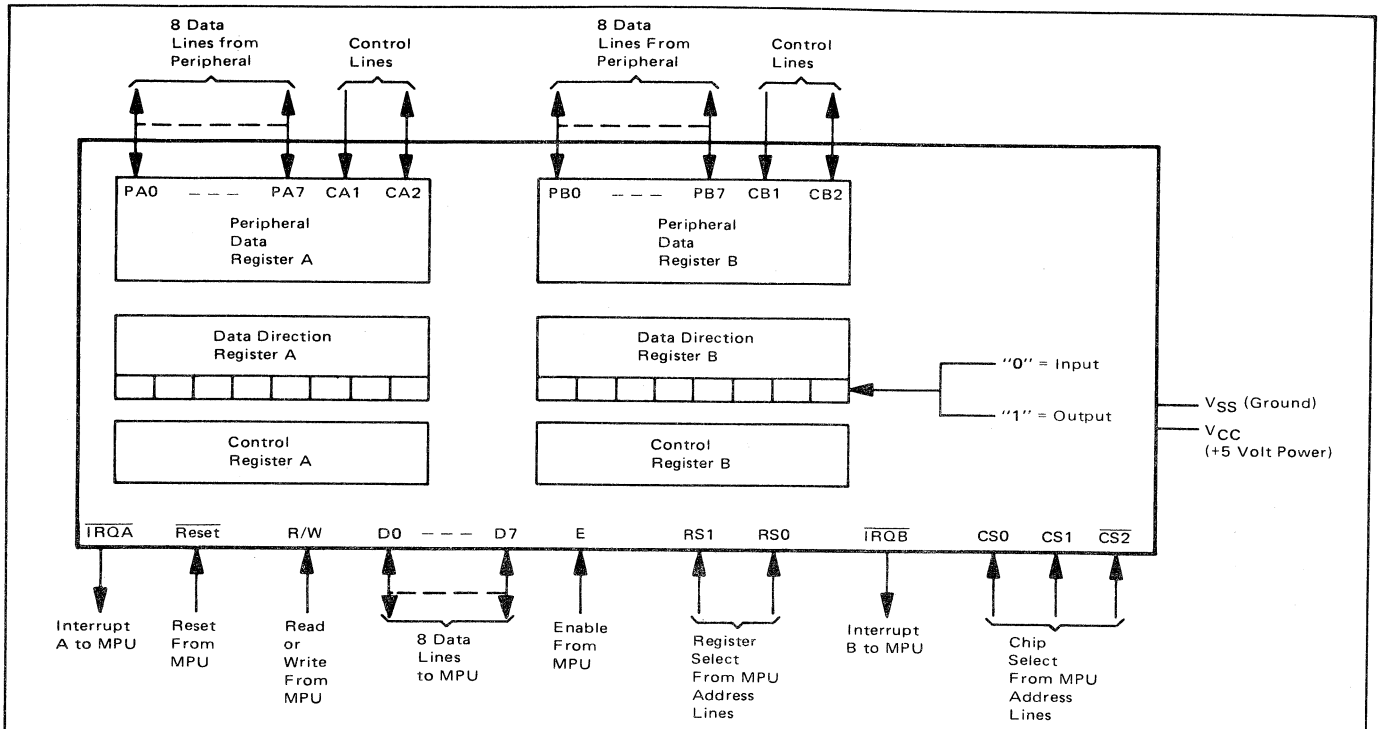


FIGURE 3 – PIA Functions

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

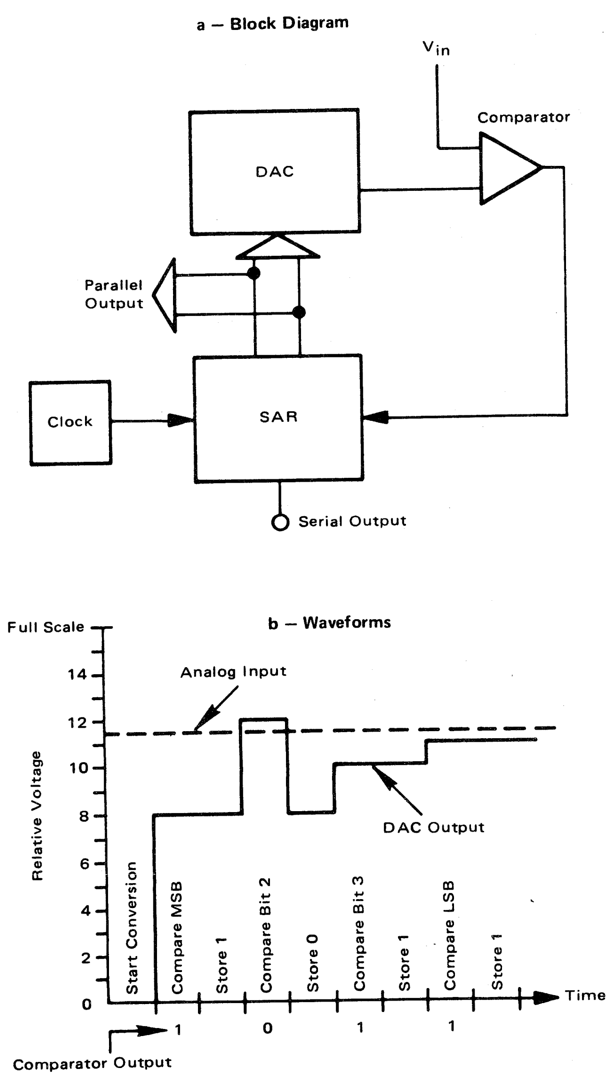


FIGURE 4 — 4-Bit Successive Approximation Converter

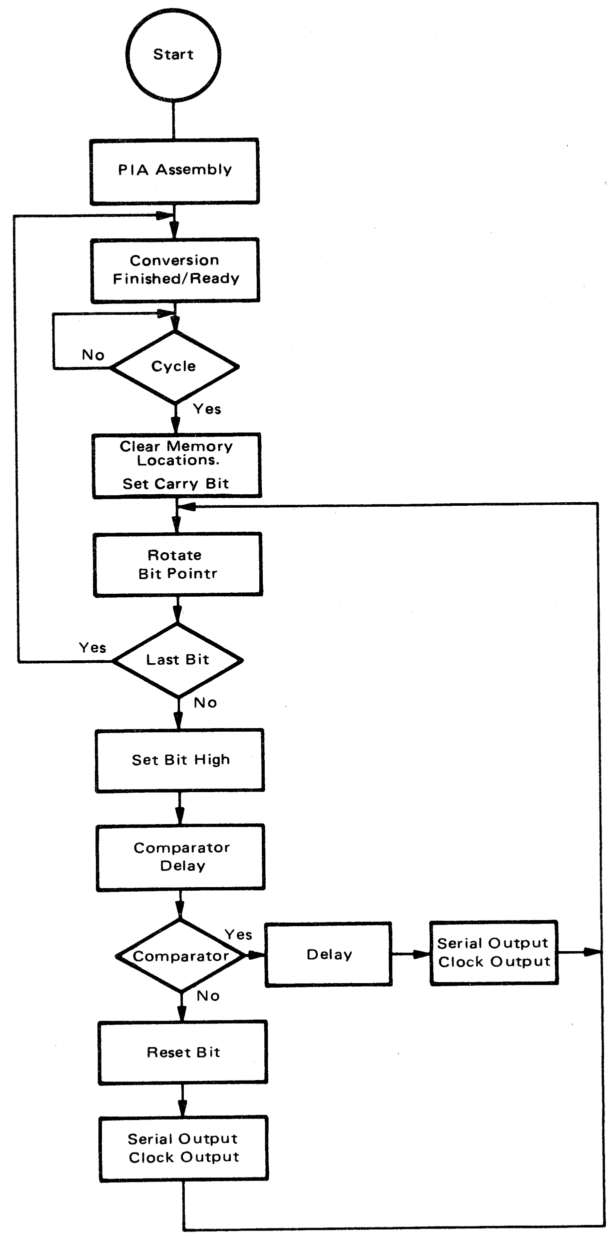


FIGURE 5 — 8-Bit Successive Approximation A/D Flow Diagram

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  * SO  * NC  * CYCLE * NC  *
33.000 *********************************************************
34.000 *                     *
35.000 *
```

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

```
*
* COMP-COMPARATOR,SC-SIMULATED CLOCK,SO-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)

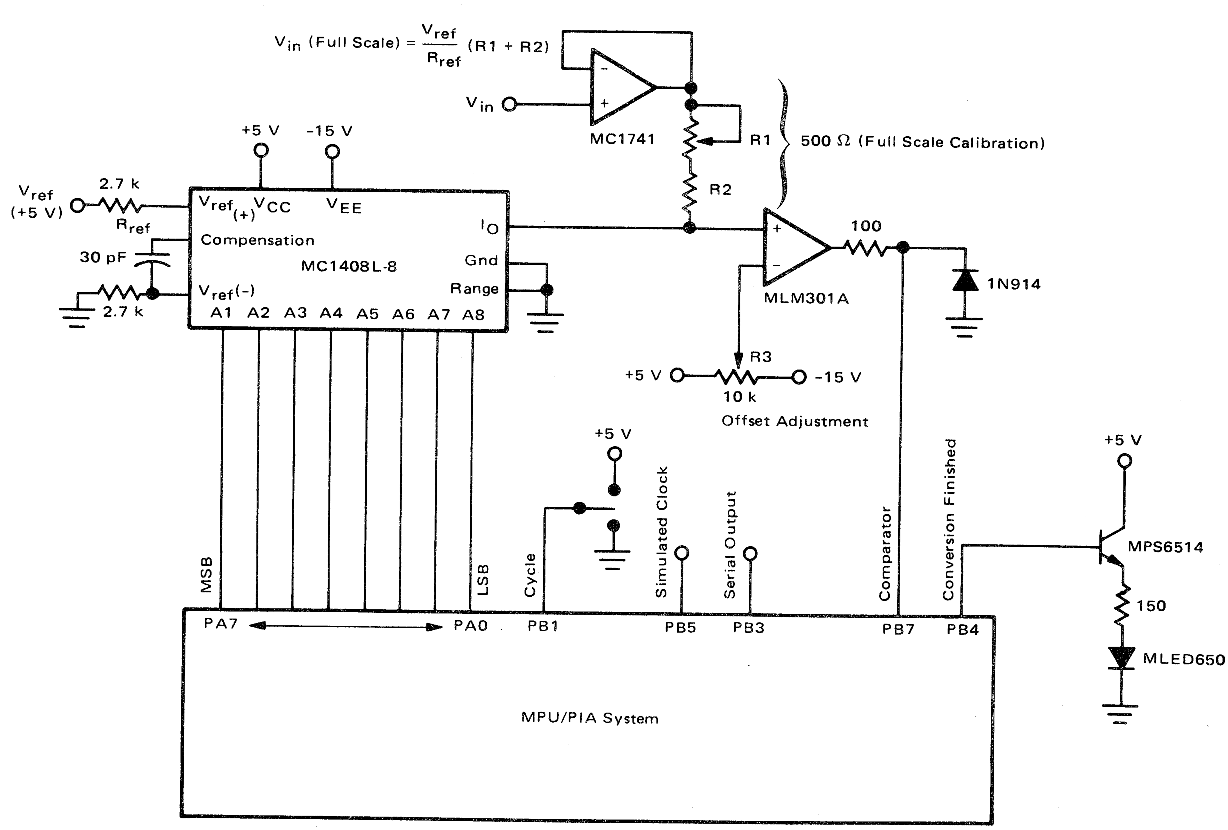


FIGURE 7 — 8-Bit SA Hardware

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  * SO  * NC  * CYCLE * NC  *
35.000 ***************************************************************
36.000 *                                 *
37.000 *
38.000 *
39.000 * COMP-COMPARATOR,SC-SIMULATED CLOCK,SO-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
```

```
 SUB A PONTR2
 STA A PIA2BD
 STA A ANS2
 LDA B #$20              SERIAL OUTPUT (CLOCK ONLY)
 STA B PIA1BD
 CLR B                   CLOCK RESET
 STA B PIA1BD
 BRA END
*
*
*                              *HIGH COMPARATOR LOOP*
YES LDA A #$05           TIME EQUALIZATION
DELAY DEC A
 BNE DELAY
 LDA B #$28              SERIAL OUTPUT
 STA B PIA1BD
 LDA B #$08              CLOCK RESET
 STA B PIA1BD
 NOP
 NOP
*
END BRA CONVRT
*
*
*
 MON
```

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

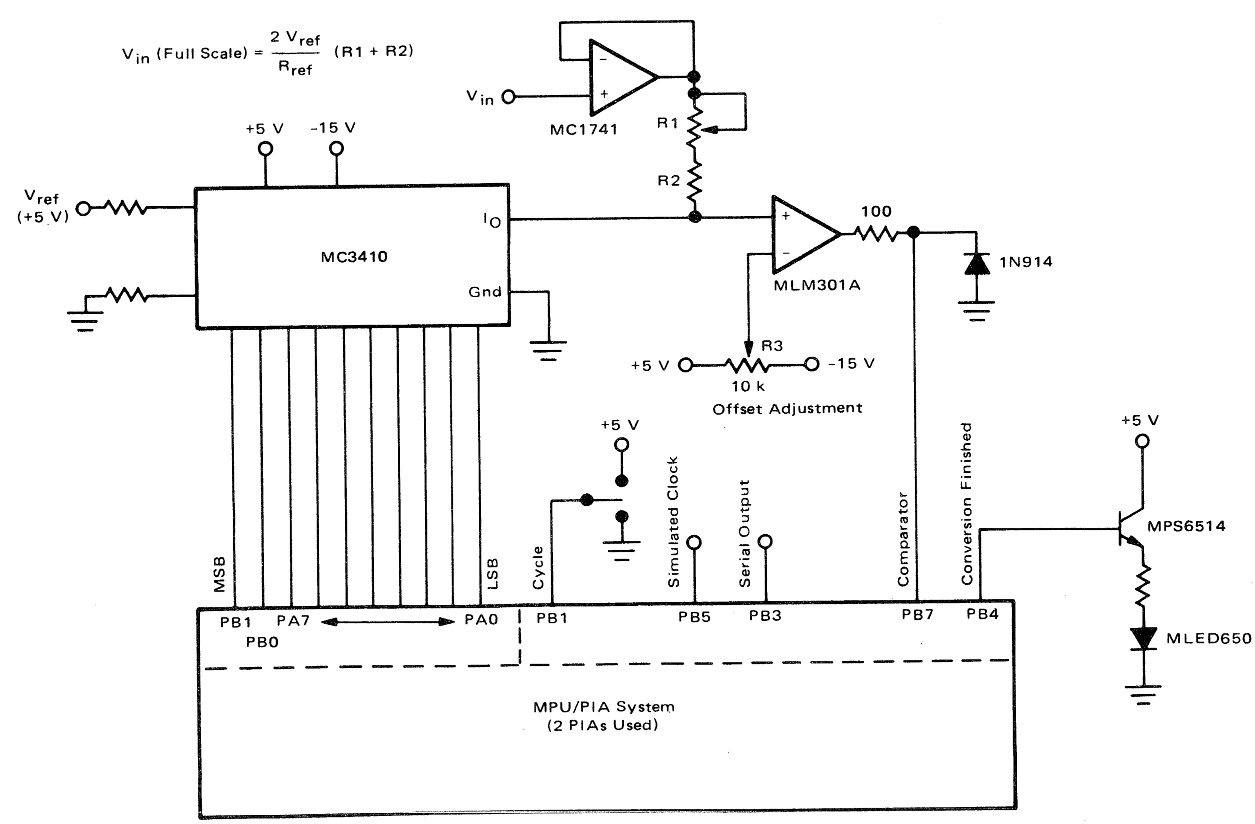


FIGURE 9 — 10-Bit SA Hardware

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.

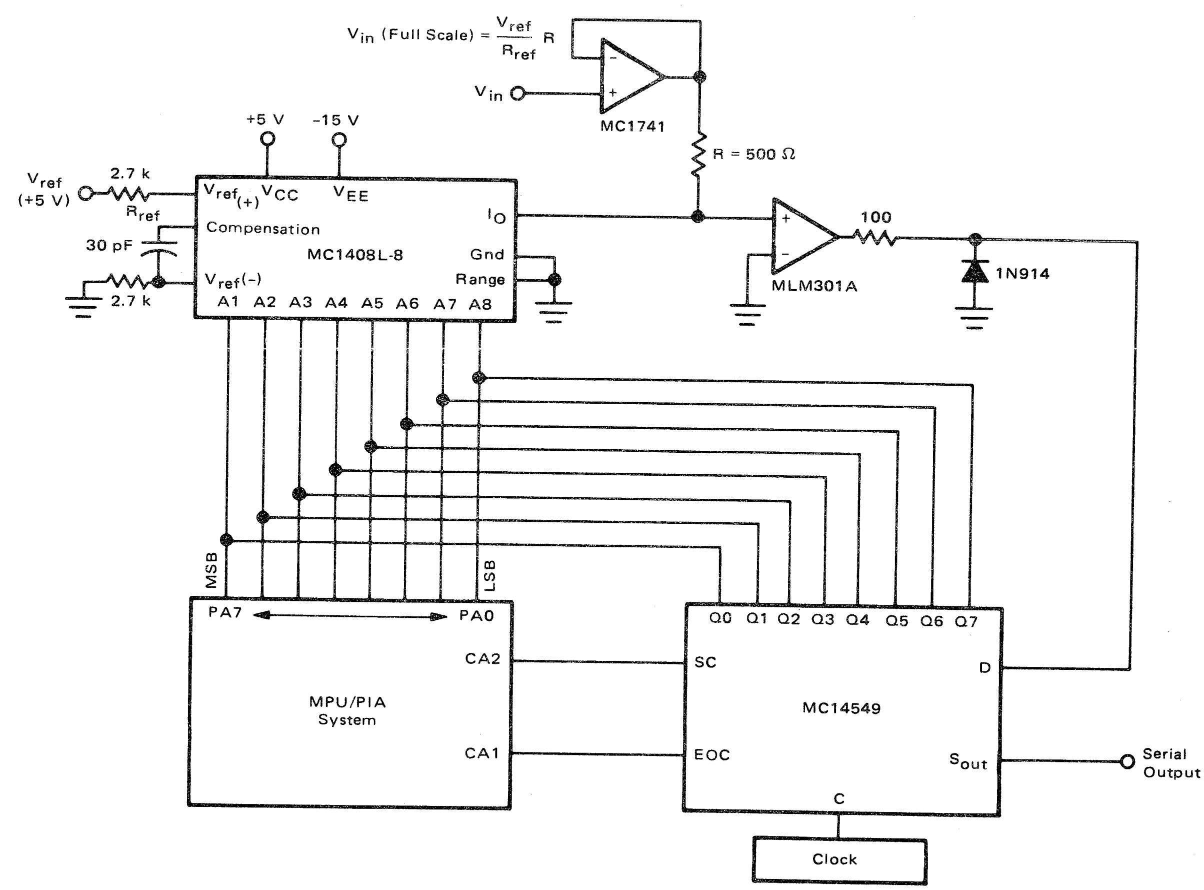


FIGURE 10 — 8-Bit SA Using External Hardware

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.

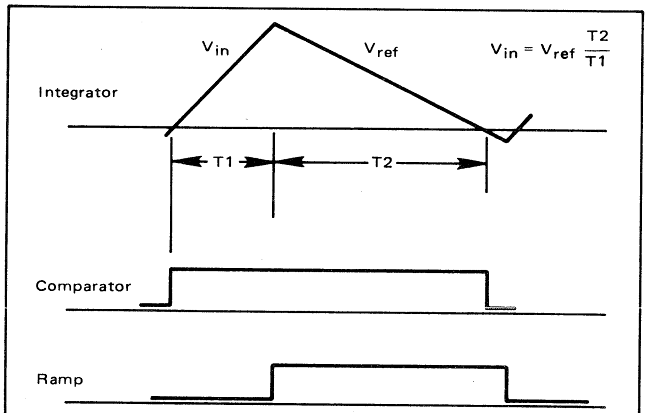


FIGURE 12 — Dual Ramp Waveforms

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

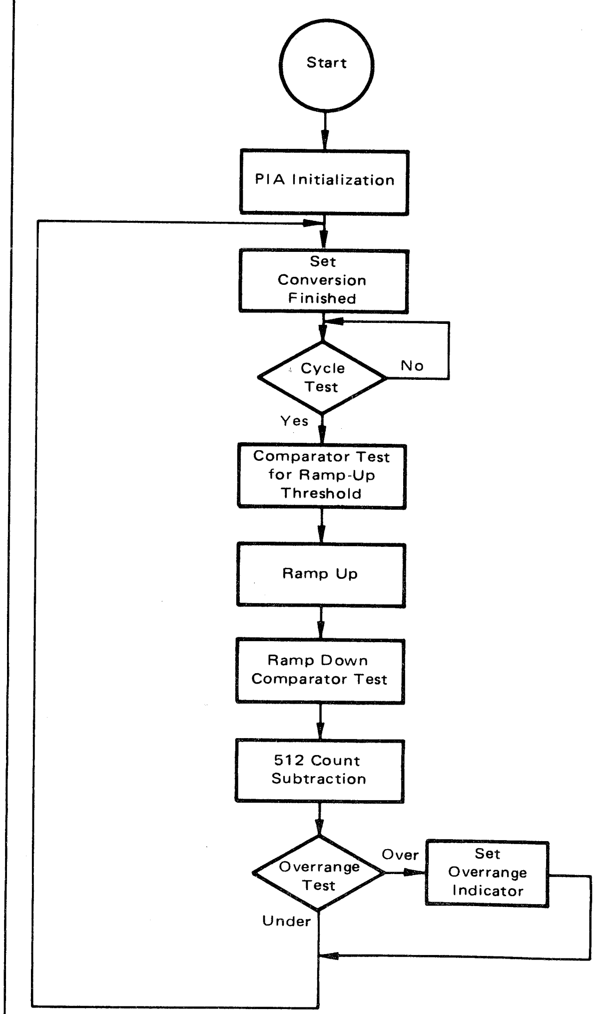


FIGURE 13 — 12-Bit Binary Dual Ramp A/D Flow Diagram

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)

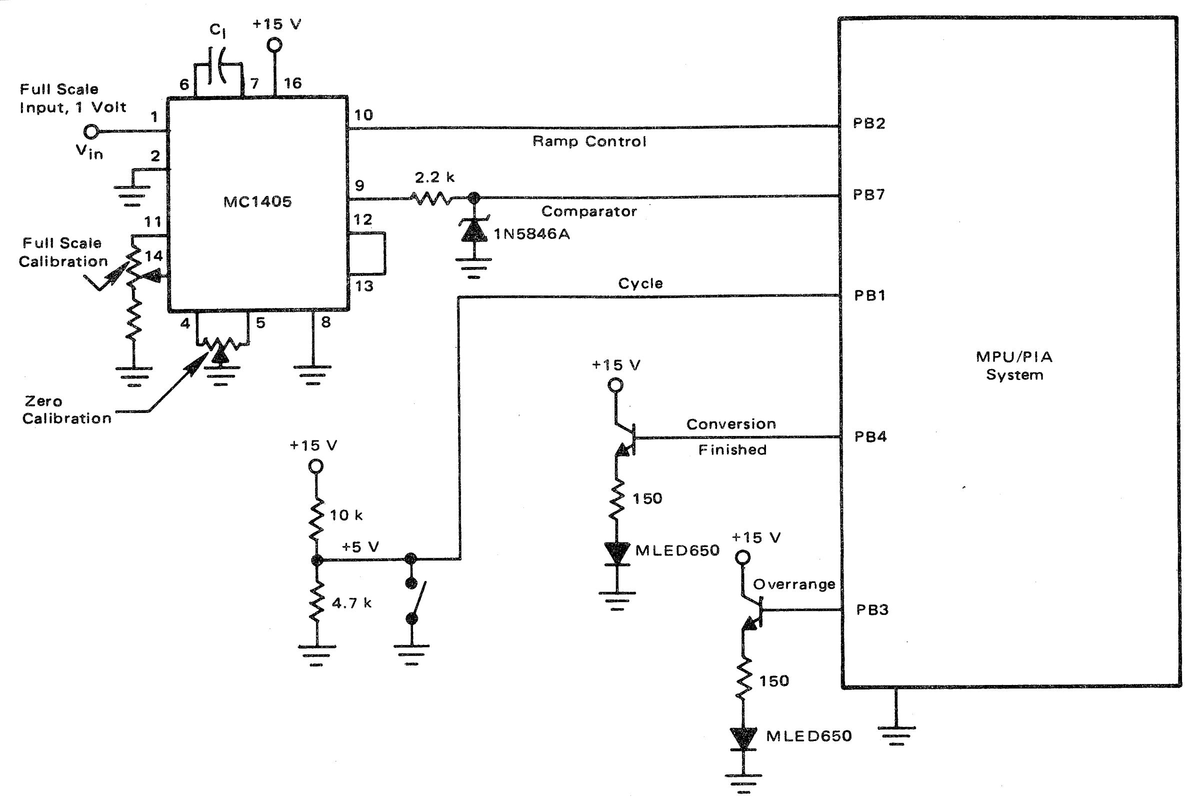


FIGURE 15 – 12-Bit Binary Dual Ramp A/D Hardware

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.

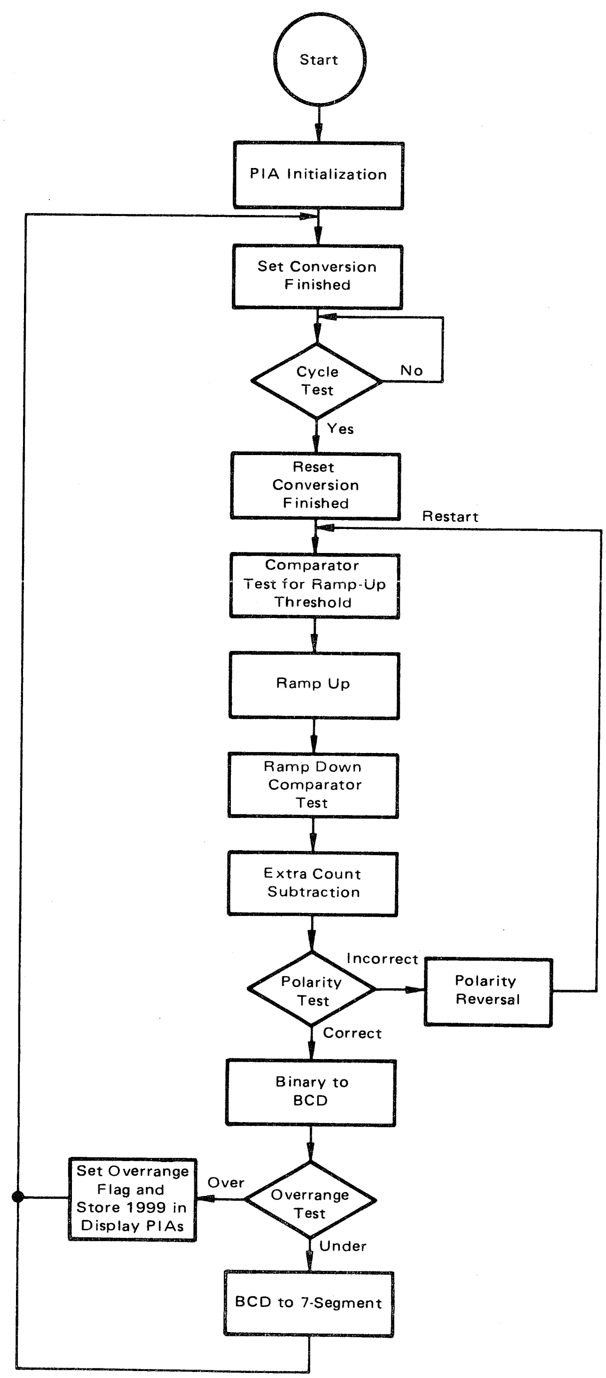


FIGURE 16 — 3½-Digit Dual Ramp A/D Flow Diagram

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
UNITTEN 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)

```
*
*                                          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
```

```
AND A #$08
ADD A #$34
STA A PIA1BC
JMP RESTAR
*
*
*                          LOOK-UP TABLE FOR BCD TO 7 SEGMENT
*                                  CONVERSION
ORG $0C00
FCB $7E,$30,$6D,$79,$33,$5B,$5F,$70,$7F,$73
END
MON
```

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

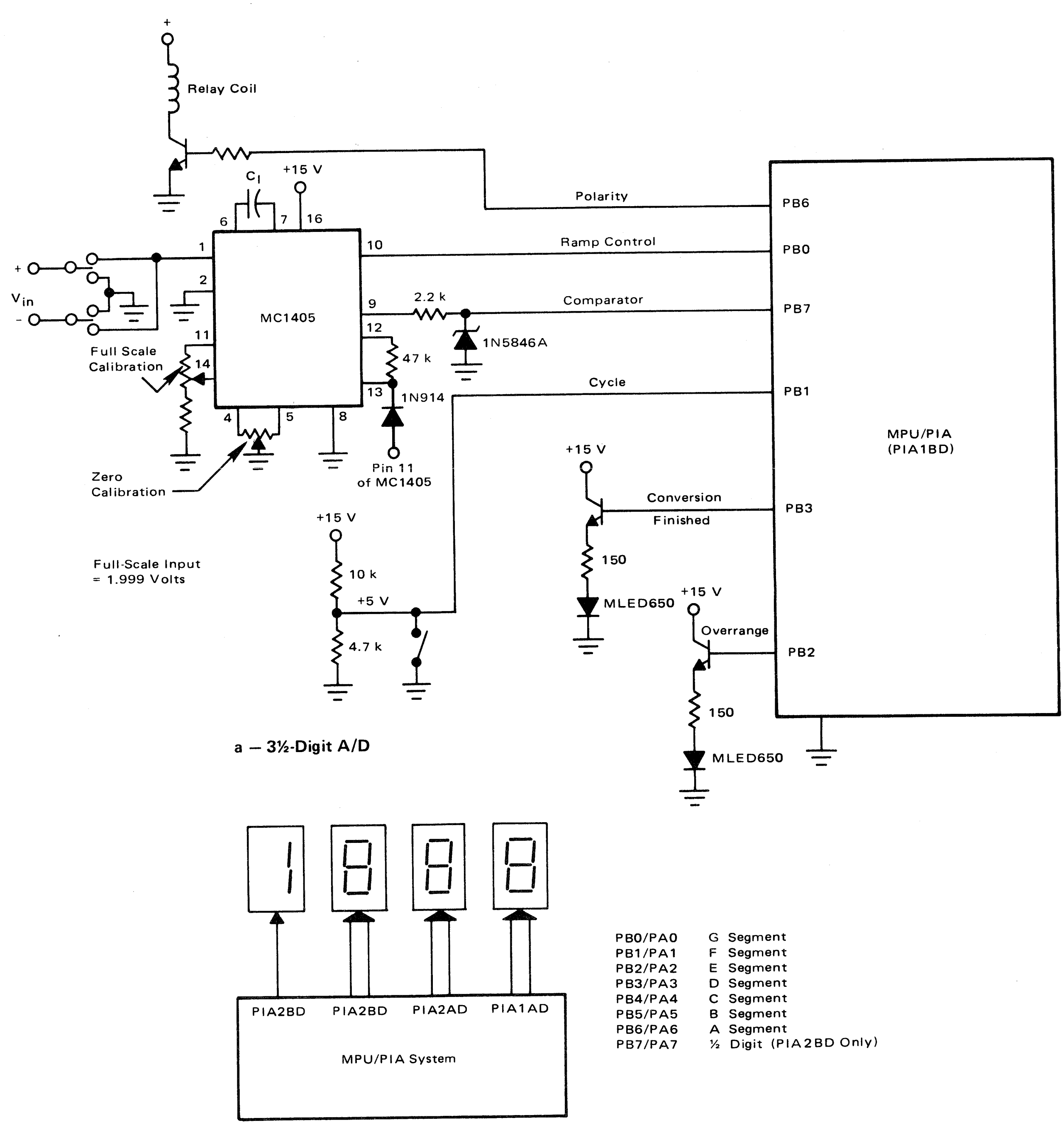


FIGURE 18 – 3½-Digit Dual Ramp A/D Hardware

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
```

```
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
TENTSD RMB 1
*
*
 ORG $4006
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 $4010
PIA3AD RMB 1            A SIDE, DATA REGISTER
PIA3AC RMB 1            A SIDE, CONTROL REGISTER
PIA3BD RMB 1            B SIDE, DATA REGISTER
PIA3BC RMB 1            B SIDE, CONTROL REGISTER
*
*
*
*                                           PIA ASSEMBLY
 ORG $0A00
 CLR PIA1BC
 CLR PIA2AC
 CLR PIA2BC
 CLR PIA3AC
 CLR PIA3BC
 LDA A #$4D
 STA A PIA1BD
 LDA A #$0FF      REMAINING PIA'S ALL OUTPUTS
 STA A PIA2AD
 STA A PIA2BD
 STA A PIA3AD
 STA A PIA3BD
 LDA A #$34        SETS PIA CONTROL REGISTER BIT 3 HIGH
 STA A PIA1BC
 STA A PIA2AC
 STA A PIA2BC
 STA A PIA3AC
 STA A PIA3BC
*
 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
```

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
OVRNG1 BRA OVRNGE
 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)

```
 LDA A HNDTHD
 LSR A
 LSR A
 LSR A
 LSR A
 STA A INDEX+1
 LDX INDEX
 LDA A 0,X
 STA A PIA3BD
 LDA A TENTSD
 SUB A #$01
 BLT END
 LDA A #$80
 ADD A PIA3BD
 STA A PIA3BD
END JMP CYCLE1
*
OVRNGE LDA A #$0D ; OVERRANGE,RC HIGH, CON F
 STA A PIA1BD
 LDA A #$F3
 STA A PIA2AD
 STA A PIA2BD
 STA A PIA3AD
 STA A PIA3BD
 JMP CYCLE
*
*
POLARY LDX #$0100
BR DEX
 BNE BR
 LDA A PIA1BC
 COM A
 AND A #$08
 ADD A #$34
 STA A PIA1BC
 JMP RESTAR
*
*
*
 ORG $0C00
 FCB $7E,$30,$6D,$79,$33,$5B,$5F,$70,$7F,$73
 END
 MON
```

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

External Dual Ramp System

The final dual ramp A/D system to be discussed uses the MC1405 with an MC14435 CMOS dual ramp digital subsystem to provide a complete A/D converter external to the MPU system. This system provides an inexpensive A/D that is easily interfaced to an MPU system through a PIA and requires a minimum number of additional software instructions for control. Also, the microprocessor is available for performing other tasks during the A/D conversion.

When the MPU requires analog information, the data is brought into the MPU system through a PIA and placed in memory for further use. The flow of this information is under control of the MPU system via an interrupt program. Figures 20 and 21 show the external devices with the MPU and the software instructions required to start the conversion and transfer the data from the A/D. Like the external successive approximation method described previously, this dual ramp technique reduces the number of MPU instructions required and increases the throughput of the overall MPU system. However, the increase in exterrnal hardware may offset these advantages. Also, additional external hardware is required for autopolarity and a 7-segment display.

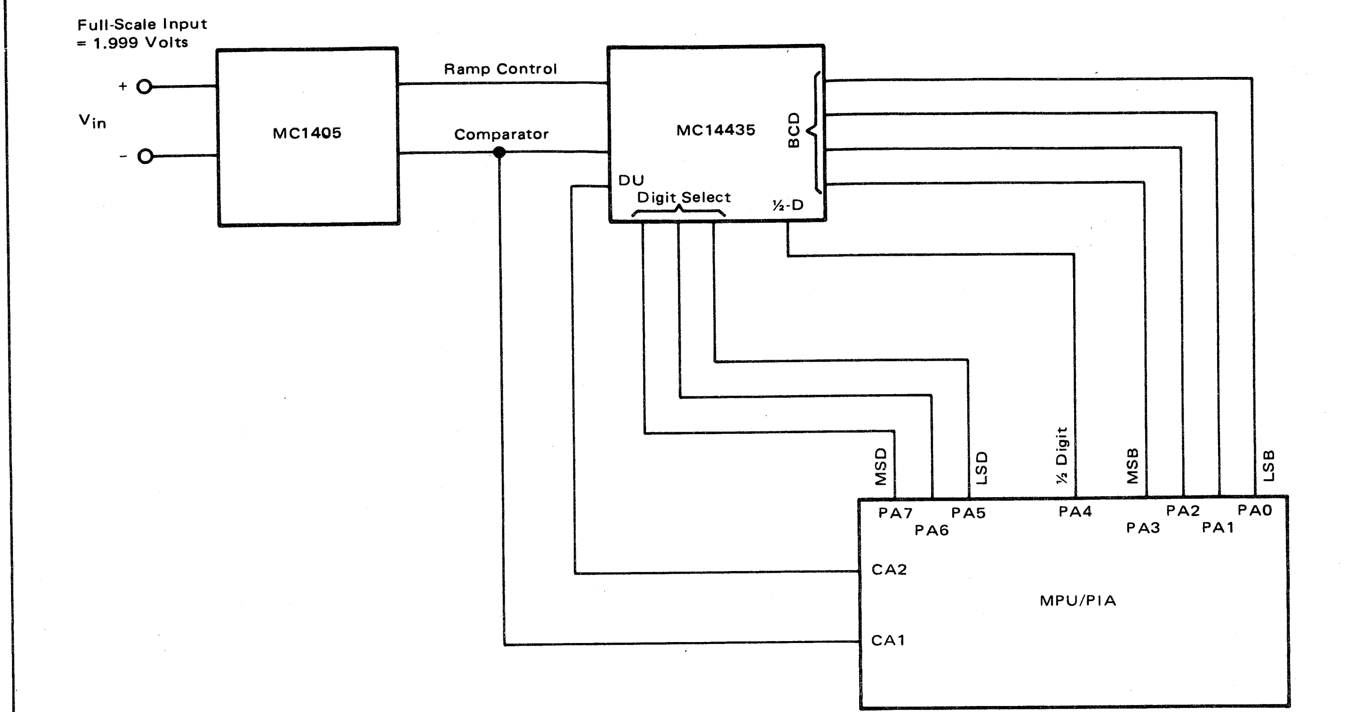


FIGURE 20 – 3½-Digit Dual Ramp Using External Hardware

One port of a PIA is required for the interface to the MPU. The I/O configuration of this PIA is shown in lines 18 through 25 of the source program (Figure 21). The output of the MC14435 digital subsystem consists of three multiplexed BCD digits with the half digit output provided on a separate pin. The three most significant bits of the PIA port are connected to the digit select lines of the MC14435 while the four LSBs are connected to the BCD lines of the MC14435. The remaining PIA bit is connected to the half digit output. Lines 36 through 39 simulate the main MPU program which branches to the A/D subroutine starting in line 42. When this occurs the display update pin of the MC14435 (CA2) is set low which allows only the next data update to enter the MC14435 output latches. The wait for interrupt (WAI) instruction (line 44) stores the MPU stack and waits until the comparator output causes an interrupt on CA1.

At this point the processor is interrupted and vectored to the program beginning at line 50 causing it to demultiplex the BCD data on the output of the MC14435. The least significant digit (LSD) is first selected by the pointer of lines 50 and 51. When a low condition on this LSD line occurs, the BCD data is stored via the indexed mode of addressing in memory location \$0100. The pointer is then shifted to the next position (line 57) and when the digit select line goes low the BCD data is stored in the next sequential memory location (\$0101). Then the MSD BCD value is placed in memory location \$0102 when the MSD digit select goes low. After the multiplexed BCD data has been placed in memory, the half digit is placed in memory location \$0103. At this point the display update line to the MC14435 is returned to a high position and the MPU returns from the interrupt and then from the subroutine back to the main program which requested the data.

A minimum of 183 μs is required to transfer the A/D data to the MPU. This time period is dependent upon the A/D clock frequency which controls the digit select lines.

FIGURE 21 – 3½-Digit External Dual Ramp Software (Page 1 of 3)

```
 1.000   NAM DWA4
 2.000   OPT OT
 3.000 *
 4.000 **************************************************************
 5.000 *             3 1/2 DIGIT DUAL RAMP USING EXTERNAL           *
 6.000 *                         HARDWARE                           *
 7.000 **************************************************************
 8.000 *
 9.000   ORG $0010
10.000 POINTR RMB 1                    POINTER FOR DIGIT SELECT
11.000 *
12.000 *
13.000   ORG $400A
14.000 PIA1AD RMB 1
```

FIGURE 21 — 3½-Digit External Dual Ramp Software (Page 2 of 3)

```
PIA1AC RMB 1
*
*
*               **PIA1AD CONFIGURATION**
*************************************************
* PA7 * PA6 * PA5 * PA4 * PA3 * PA2 * PA1 * PA0 *
*************************************************
* MSD         LSD  *1/2 D* MSB             LSB *
*************************************************
*   DIGIT SELECT   *     *          BCD         *
*************************************************
*
*                         RESULTS STORED IN LOCATIONS 0100-0103
*                            LSD=0100  1/2 DIGIT=0103
*
 ORG $0900
 CLI
 CLR PIA1AC                  PIA ASSEMBLY
 CLR PIA1AD
 LDA A #$3C
 STA A PIA1AC
 LDS #$0020
*
*
 NOP                       MAIN PROGRAM SIMULATION
 JSR CONVRT
END NOP
 BRA END
*
*
CONVRT LDA A #$35          A/D CONVERSION SUBROUTINE
 LDA B PIA1AD              DUMMY READ OF PIA DATA REGISTER
 STA A PIA1AC
 WAI
 RTS
*
*
*
*
BEGIN LDA A #$20           BEGINING OF INTERRUPT PROGRAM
 STA A POINTR
 LDX #$0100
NEXT LDA A PIA1AD
 TAB
 AND A POINTR
 BNE NEXT
 ROL POINTR
 AND B #$0F
 STA B 0,X
 INX
 BCC NEXT
 LDA A PIA1AD
 AND A #$10
 LSR A
 LSR A
 LSR A
 LSR A
 STA A 0,X
 LDA A #$3C
 STA A PIA1AC
```

```
75.000  RTI
76.000 *
77.000 *
78.000 *
79.000  MON
```

FIGURE 21 — 3½-Digit External Dual Ramp Software (Page 3 of 3)

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.

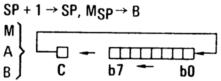
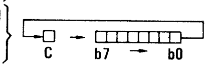
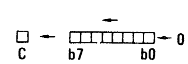
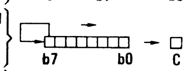
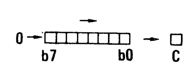
MC6800, MC6820 Data Sheets, *M6800 Systems Reference and Data Sheets,* Motorola Semiconductor Products Inc.

MC14435 Data Sheet, Motorola Semiconductor Products Inc.

APPENDIX A
MPU INSTRUCTIONS

Accumulator and Memory Instructions

ADDRESSING MODES

		IMMED			DIRECT			INDEX			EXTND			IMPLIED			BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG.					
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	(All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 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				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's	NEG							60	7	2	70	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);
~ Number of MPU Cycles;
# Number of Program Bytes;
+ Arithmetic Plus;
− Arithmetic Minus;
· Boolean AND;
M_{SP} Contents of memory location pointed to be Stack Pointer;

+ Boolean Inclusive OR;
⊕ Boolean Exclusive OR;
$\overline{M}$ Complement of M;
→ Transfer Into;
0 Bit = Zero;
00 Byte = Zero;

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			BOOLEAN/ARITHMETIC OPERATION	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										$N \oplus V = 0$	●	●	●	●	●	●
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	●	●	●	●	●	●
Branch If Higher	BHI	22	4	2										C + Z = 0	●	●	●	●	●	●
Branch If ⩽ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	●	●	●	●	●	●
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	$0 \rightarrow C$	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	●	R	●	●	●	●
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	●	●	●	●	R	●
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	●	S	●	●	●	●
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	●	●	●	●	S	●
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	12					
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	●	●	●	●	●	●

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 $N \oplus C$ 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.

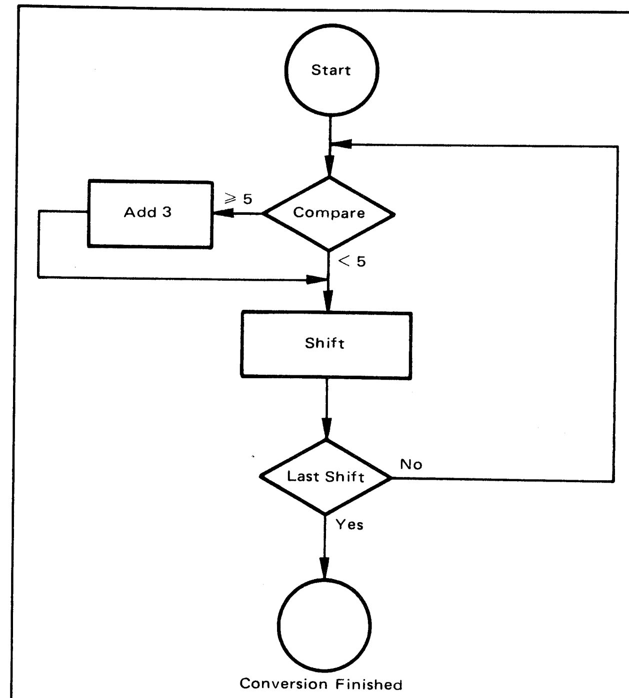


FIGURE B1 – Binary to BCD Conversion Flow Diagram

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
*
*
*
```

```
        ORG $0F00          **BEGINNING OF PROGRAM**
        CLR UNTTEN
        CLR HNDTHD
        CLR TENTSD
        LDX #$0010
BEGIN   LDA A UNTTEN       UNITS COMPARISON
        TAB
        AND A #$0F
        SUB A #$05
        BMI AT
        ADD B #$03
AT      TBA                TENS COMPARISON
        AND A #$0F0
        SUB A #$50
        BMI BT
        ADD B #$30
BT      STA B UNTTEN
*
        LDA A HNDTHD       HUNDREDS COMPARISON
        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       TENS OF THOUSANDS COMPARISON
        TAB
        SUB A #$05
        BMI ET
        ADD B #$03
ET      STA B TENTSD
*
*
        ASL LSB
        ROL MSB
        ROL UNTTEN
        ROL HNDTHD
        ROL TENTSD
        DEX
        BNE BEGIN          END OF CONVERSION CHECK
*
*
*
*
        END
        MON
```

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

MOTOROLA Semiconductor Products Inc.

Printed in Switzerland