ANALOG/DIGITAL PROCESSING WITH MICROCONTROLLERS

John Katausky, Ira Horden, Lionel Smith
Application Engineers
Intel Corp.
5000 W. Williams Field Road
Chandler, AZ 85224

Microcontrollers are rapidly becoming the backbone of silicon computing systems. From a technical standpoint, the most significant attribute, aside from the inclusion of RAM and ROM, that segregates a microcontroller from a microprocessor is I/O manipulation. In general, I/O manipulation is an intimate part of a microcontroller's architecture. The instruction set and architecture of a microcontroller allows the CPU to directly control the I/O facilities on the device. This is in direct contrast to a microprocessor where the I/O is essentially a "sea" of addresses and it is up to the hardware designer to place some type of I/O hardware in this I/O "sea". It should be obvious that simply adding ROM and RAM to a microprocessor WILL NOT create a microcontroller.

This intimate contact with I/O gives the microcontroller a distinct advantage over the microprocessor in applications that are I/O intensive. Microcontrollers can test, set, complement, or clear I/O port pins much faster than a microprocessor and they can also make decisions, based on the state of other hardware features, such as timer/counters with equal speed. This integration of I/O, in both hardware and software makes the microcontroller "ideal" for many types of intelligent instrumentation.

4K ROM/EPROM - 8K ROM ON 8052
128 BYTES OF RAM - 256 ON THE 8052
2-16 BIT TIMER/COUNTERS - 3 ON THE 8052
FULL DUPLEX UART
5 VECTORED INTERRUPTS - 6 ON THE 8052
4 REGISTER BANKS
BIT MANIPULATION (BOOLEAN PROCESSOR)
32 DIRECTLY ADDRESSABLE I/O PINS
MULTIPLY AND DIVIDE INSTRUCTIONS
SUPPORTS 64K OR RAM AND ROM-128K TOTAL

TABLE 1. A BREIF LISTING OF THE MCS-51'S FEATURE SET.

Intel's MCS-51 series of microcontrollers contain many features that can be integrated directly into many types of instruments. TABLE 1 is a brief listing of these features. To illustrate the power of the 8051 this paper will elaborate on two techniques for performing analog to digital (A to D) conversion. Both of these examples assume that some additional hardware is attached to the I/O pins of the 8051.

S/A CONVERSION TECHNIQUES

Successive approximation analog to digital conversion involves a "binary search" of an unknown voltage relative to a "fixed" known reference. The reference is selectively divided by multiples of two until the desired accuracy is reached. Figure 1 is a flowchart of a successive approximation converter. This technique usually requires a digital to analog converter to divide the reference voltage and a voltage comparator to compare the unknown voltage to the "divided" reference. Digital to analog converters and voltage comparators are readily available and relatively inexpensive. A block diagram of an 8051 based A to D converter is shown in Figure 2.

Many industrial A to D converters require 12 bits of accuracy. A 12 bit converter provides good "dynamic range" and is capable of resolving 1 part in 4096. If the applied input voltage ranges from 0 to 10 Volts, a 12 bit converter can resolve 2.4 millivolts within this range. The theoretical accuracy of a 12 bit converter is .024% +/- 1/2 least significant bit.

The power of the 8051 in this type of application is best revealed by examining the software required to implement the successive approximation algorithm. The routine for the 8051 is shown in Table 2.

The execution times given assume a 12 Mhz crystal. Compare this to the following routine which is a 4 Mhz Z-80

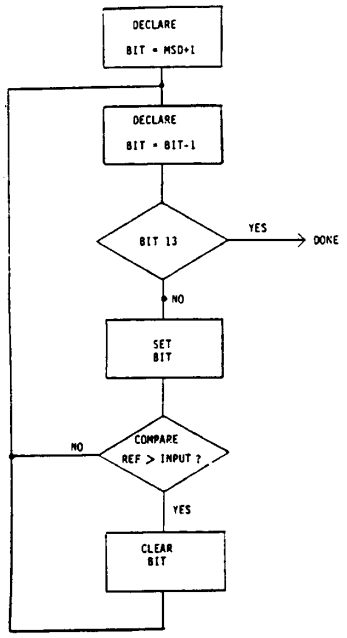

FIGURE 1. SUCCESSIVE APPROXIMATION CONVERSION ALGORITHM

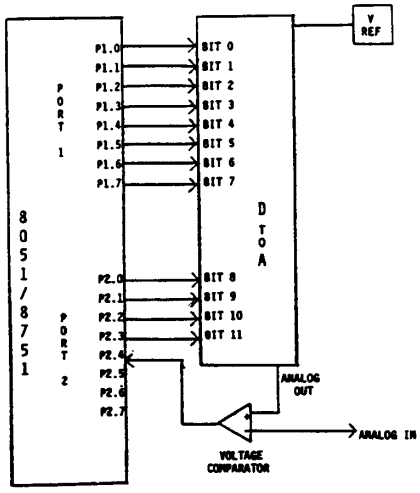

FIGURE 2. BLOCK DIAGRAM OR SUCCESSIVE APPROXIMATION A TO D CONVERTER

TABLE 2. SUCCESSIVE APPROXIMATION ROUTINE FOR THE 8051.

```
      INSTRUCTION          BYTES  TIME
      ;
      ;CLEAR PORT PINS
      ;
      MOV     P1,#0          3      2
      ANL     P2,#0F0H       3      2
      ;
      ;START CONVERSION
      ;
      SETB    P2.3           2      1
      JNB     P2.4,L1        3      2
      CLR     P2.3           2      1
L1:   SETB    P2.2           2      1
      JNB     P2.4,L2        3      2
      CLR     P2.2           2      1
L2:   SETB    P2.1           2      1
      JNB     P2.4,L3        3      2
      CLR     P2.1           2      1
L3:   SETB    P2.0           2      1
      JNB     P2.4,L4        3      2
      CLR     P2.0           2      1
L4:   SETB    P1.7           2      1
      JNB     P2.4,L5        3      2
      CLR     P1.7           2      1
L5:   SETB    P1.6           2      1
      JNB     P2.4,L6        3      2
      CLR     P1.6           2      1
L6:   SETB    P1.5           2      1
      JNB     P2.4,L7        3      2
      CLR     P1.5           2      1
L7:   SETB    P1.4           2      1
      JNB     P2.4,L8        3      2
      CLR     P1.4           2      1
L8:   SETB    P1.3           2      1
      JNB     P2.4,L9        3      2
      CLR     P1.3           2      1
L9:   SETB    P1.2           2      1
      JNB     P2.4,L10       3      2
      CLR     P1.2           2      1
L10:  SETB    P1.1           2      1
      JNB     P2.4,L11       3      2
      CLR     P1.1           2      1
L11:  SETB    P1.0           2      1
      JNB     P2.4,L12       3      2
      CLR     P1.0           2      1
      ;
L12:  CONVERSION COMPLETE
```

TOTAL	**90**	**46 US**

NOTE: TIMING IS TYPICAL
WORST CASE = 52 US
BEST CASE = 40 US

executing the same algorithm with the D to A hardware attached to an I/O port is shown in Table 3 (assume that all bits on PORT3 are grounded, except the comparator ihput).

TABLE 3. SUCCESSIVE APPROXIMATION ROUTINE FOR THE Z-80.

```
      INSTRUCTION          BYTES    TIME

      ;
      ;CLEAR PORT PINS
      ;
       LD      A,0           2      1.75
      OUT      (PORT1),A     2      2.75
      OUT      (PORT2),A     2      2.75
      ;
      ;START CONVERSION
      ;
       LD      A,08H         2      1.75
      OUT      (PORT2),A     2      2.75
       IN      A,(PORT3)     2      2.75
       OR      A             1      1.00
       IN      A,(PORT2)     2      2.75
       JP      Z,L1          3      2.50
      AND      0F7H          2      1.75
L1:    OR      04H           2      1.75
      OUT      (PORT2),A     2      2.75
       IN      A,(PORT3)     2      2.75
       OR      A             1      1.00
       IN      A,(PORT2)     2      2.75
       JP      Z,L2          3      2.50
      AND      0FBH          2      1.75
L2:    OR      02H           2      1.75

      REPEAT BETWEEN L1 AND L2 10
      MORE TIMES AND SET/RESET THE
      APPROPRIATE I/O BITS

-------------------------------------------

TOTAL                       179     180 US

AGAIN TIMING IS TYPICAL
      WORST CAST = 190.25 US
       BEST CASE = 169.25 US
```

One may argue that by "memory mapping" the Z-80's I/O ports the execution time could be enhanced because the user could take advantage of the Z-80's SET and RESET memory BIT instructions. In reality, a few bytes of memory are saved, but very little time!. This is because the Z-80's memory oriented BIT instructions are VERY slow, requiring between 3 and 5 microseconds with a 4 Mhz clock!

This is not to say that the Z-80 isn't a credible 8-bit processor. The weakness is that decisions (i.e. JUMPS) cannot be made directly on the state of a given I/O pin. JUMP instructions, on most processors, are made on the state of the flags - after some type of logical or arithmetic operation! This means that information must be moved to an internal CPU register before a decision can be made. This "moving" of information back and forth between internal registers and I/O makes the microprocessor quite inefficient, relative to the microcontroller when I/O manipulation is involved. Note that with the 8051 algorithm never "moves" data from one location to another - it directly sets, tests, and clears bits. This characteristic gives the 8051 its distinct execution advantage.

Another strength of the 8051 in this type of application, relates to the fact that I/O port pins can be set, cleared, complemented, and tested with the same speed that a microprocessor can act on it's internal registers. Note that the 8051 takes only 1 microsecond to fetch an opcode and set or clear a port pin. A microprocessor must first fetch and decode the opcode, then place the appropriate I/O or memory address on the bus, then perform the necessary operation. All of this "communication" over the microprocessor bus significantly slows down the microprocessor.

DUAL SLOPE INTEGRATING CONVERTER

Integrating A to D converters operate by an indirect method of converting a voltage to a time period, then measuring the time period with a counter. Integrating techniques are quite slow, relative to successive approximation, but they are capable of providing very accurate measurements - 5 1/2 or more decimal digits - if proper analog techniques are employed. They also have the added advantage of allowing the integration period to be a multiple of 60 Hz (16.67 ms) which can eliminate inaccuracies caused by the ever present "power line". Virtually all digital voltmeters use some type of integrating technique. Figure 3 is a block diagram of a typical integrating

A to D converter.

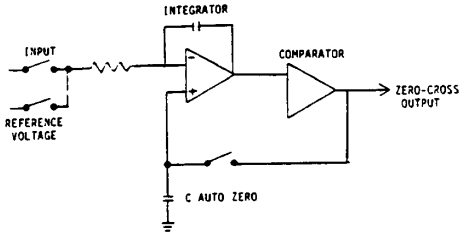

FIGURE 3. INTEGRATING A TO D CONVERTER

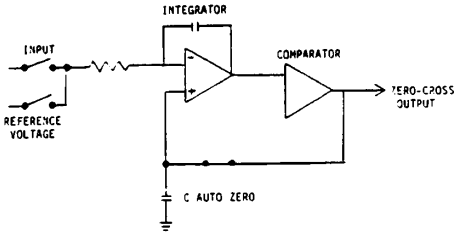

FIGURE 4A. AUTO ZERO PHASE

Figures 4A, 4B, and 4C show the three typical phases involved in the dual slope technique. Figure 4A illustrates the auto-zero phase. In this phase the integrating "loop" is closed and the offset of the analog integrator is accumlated in C auto zero. In Figure 4B, the input switch is closed and the integrator integrates the input voltage for a fixed time period T1. In figure 4C, the reference switch is closed and the integrator integrates the reference voltage until the comparator senses a zero crossing condition. The time it takes for this phase to occur is directly proportional to the amplitute of the input voltage. Additional circuitry can be added to determine the polarity of the input voltage, then switch in a reference of oppsite polarity, but the basic technique remains the same.

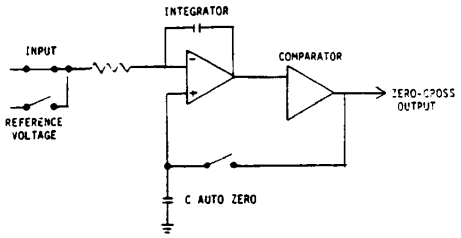

FIGURE 4B. INPUT INTEGRATION PHASE

The 8051 is an ideal controller for an intelligent integrating A to D system. The 16 bit timer/counters can provide better than 4 1/2 decimal digits of accuracy, the serial port can be used to transmit the analog reading to a printer or another processor, the CPU can be interrupted by the 60 Hz line so conversions can start at percise intervals, and software can be used to calculate and save average, peak, or RMS readings.

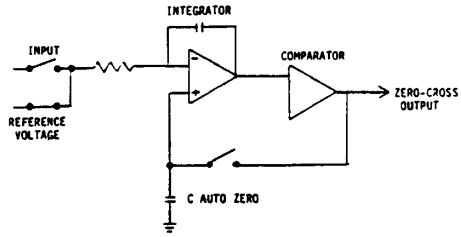

FIGURE 4C. REFERENCE INTEGRATION PHASE

Another "nice" benefit of this type of converter is that very few I/O port pins are required to control the A to D hardware, so opto-isolators can be used to completely isolate the 8051

"digital system" from the analog hardware. Opto-isolators provide an additional "bonus" in that they may provide logical level shifting if needed by the analog circuitry. Figure 5 shows how an 8051 might be connected to the analog sub-system. In practice, the analog switches can be almost anything ranging from CMOS to VFETs. The code needed to generate the "basic" integrating A to D function is shown in Table 4.

Timer interrupts could be used so that the CPU could be doing other things while the conversion was in process. Note that very little CPU time is needed to perform the actual A to D function.

CONCLUSION

This paper illustrated possible methods of using the 8051 in A to D "instrumentation" types of applications. The power of the 8051's microcontroller architecture relates to the fact that logical "decisions" can be made directly on the state of the resident I/O hardware. This fact alone gives the 8051 a distinct advantage in "bit intensive" applications. Software and hardware support tools include in-circuit emulators, an assembler, and a high level language, PLM-51. Presently, the 8051 is available in 3 technology "flavors"- HMOS II, HMOS-EPROM, and CHMOS, so depending on your individual application, you can have it your way.

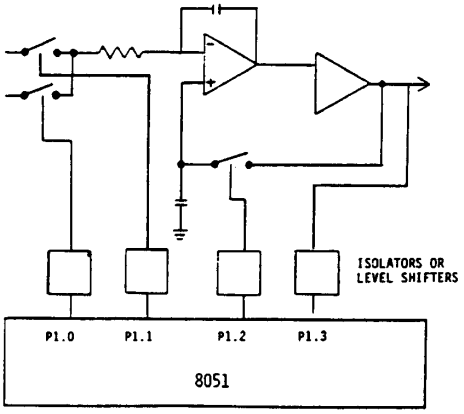

FIGURE 5. TYPICAL 8051 CONTROLLED ANALOG SUB-SYSTEM

TABLE 4. SOFTWARE FOR INTEGRATING A TO D CONVERTER

```
;
;START PROGRAM
;
CLR   TR0             ;TURN TIMER OFF
;
MOV   TH0,#HIGH TAZ   ;LOAD AUTO ZERO
MOV   TL0,#LOW TAZ    ;TIME
;
ANL   P1,#0F0H        ;MAKE A/D INACTIVE
SETB  P1.2            ;AUTO ZERO PHASE
SETB  TR0             ;TURN TIMER ON
JNB   TF0,$           ;LOOP TIL OVERFLOW
;
CLR   TR0             ;TURN TIMER OFF
CLR   TF0             ;RESET TOV FLAG
;
MOV   TH0,#HIGH INTT  ;LOAD INTEGRATION
MOV   TL0,#LOW INTT   ;TIME
;
CLR   P1.2            ;END AUTO ZERO
SETB  P1.1            ;START INTEGRATION
SETB  TR0             ;START TIMER
JNB   TF0,$           ;WAIT FOR OVERFLOW
;
CLR   P1.1            ;END INTEGRATION
;
; NOW, INTEGRATE THE REFERENCE
;
SETB  P1.0
;
;AT THIS POINT TIMER 0 HAS A VALUE OF
;TWO, THE TIMER IS EQUAL TO ZERO, WHEN
;IT OVERFLOWS AND IT WAS INCREMENTED
;TWICE DURING THE LAST TWO INSTRUCTIONS
;
;NOW, WAIT FOR ZERO CROSS
;
JNB   P1.3,$
;
;TURN THE TIMER OFF
;
CLR   TR0
;
;NOW, TIMER 0 ~ Vin + 3 COUNTS
;
```