ANALOG DEVICES

AN-276
APPLICATION NOTE

ONE TECHNOLOGY WAY • P.O. BOX 9106 • NORWOOD, MASSACHUSETTS 02062-9106 • 617/329-4700

Analog-to-Digital Conversion Using Voltage-to-Frequency Converters

by Paul Klonowski

INTRODUCTION

A voltage-to-frequency converter (VFC) is a device which accepts at its input an analog voltage or current signal and provides at its output a train of pulses or square waves at a frequency which is proportional to the input value. A voltage-to-frequency converter can thus be used as a building block in an analog-to-digital (A/D) conversion system, by using the VFC to clock a counter for a certain period of time (the "count time" or "gate time" or "conversion time") and reading the output digital word. This digital word will be proportional to the analog input.

There are a number of advantages to using a voltage-to-frequency converter in an analog-to-digital conversion scheme. First, unlike converters based on binary-weighted networks, monotonicity is inherent under all supply and temperature conditions. Second, the fact that the signal is converted to an easily-transmitted serial bit stream allows the analog circuitry (the VFC and analog signal-conditioning circuits) to be located close to the signal source and the digital circuitry (the counter, timing gate and display circuitry) to be located elsewhere. This is especially advantageous when a large number of channels are required; the remote VFCs can be used to provide "converter-per-channel" data acquisition. Finally, since the digital number is accumulated over a large number of cycles, integration of and therefore reduction of unwanted signals is inherent.

The time required to convert an analog signal into a digital word is related to the maximum full-scale frequency of the VFC and the required resolution of the measurement. For example, the Analog Devices AD650 VFC has a full-scale frequency of 1MHz. If this device is used in an application where a resolution of 16 bits, or 1 part in 65,536 is desired, then the time required to convert the analog signal into a 16-bit digital word will be 65.536 ms. Resolution of 18 bits, or 1 part in 262,144 will require a count time slightly greater than 0.262 seconds. In general, the required count time for an analog-to-digital conversion using a VFC is:

$$T_{COUNT} = \frac{N}{FS_{out}}$$

where N is the number of codes for a given resolution and FS_{out} is the VFC full-scale output frequency.

Although VFC-based A/D converters are slower than successive-approximation and flash converters, they are comparable in speed to integrating A/D converters. VFC-based A/D converters are thus well suited for low frequency applications such as temperature and strain-gauge measurements. The resolution that the VFC can provide in these types of applications offsets the relatively long count time required to acquire the digital word corresponding to the analog input value.

The intent of this application note is to demonstrate different methods of using voltage-to-frequency converters as analog-to-digital building blocks. The focus, then, is on interfacing the output of a VFC. For detailed information on handling a variety of types of inputs (temperature, strain-gauge and photodiode signals), it is suggested that the reader consult the individual AD650, AD651 and AD654 data sheets and the AD654 application note, all available from the Analog Devices Literature Center (Phone # (617) 329-4700). Signal multiplexing schemes, isolation circuits, and individual device details are also included in the material listed above.

PULSE COUNTING

One way to perform an analog-to-digital conversion using a voltage-to-frequency converter is to have a single-chip microcomputer count the number of pulses that occur in a fixed time period. The total number of pulses counted during this period is then proportional to the input voltage of the VFC. For example, if a 1V full-scale input produces a 100kHz signal from the VFC and the count period is 100ms, then the total full-scale count will be 10,000. Scaling from this maximum is then used to determine the input voltage, i.e., a count of 5,000 corresponds to an input voltage of 0.5V.

Figure 1 shows the Analog Devices AD654 VFC output connected to the counter input, T1, of the Intel 8051 microcomputer. The AD654 is a low-cost, single-supply monolithic VFC with a full-scale frequency of up to 500kHz. The 8051 is a member of the Intel MCS-51 family of 8-bit microcomputers whose family members differ mainly in their internal memory capabilities. In the following text, the term "8051" is used to generically refer to all members of the MCS-51 family.

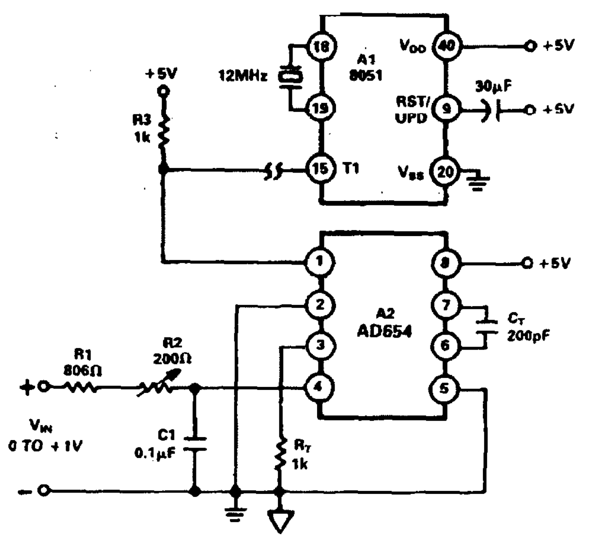

Figure 1. AD654 Pulse Counter

The analog input of the AD654 in Figure 1 is a 0 to +1V signal. The timing resistor and capacitor, R_T and C_T, are selected such that this 0 to +1V signal seen at Pin 4 results in a 0 to 500kHz output frequency. The pull-up resistor, R3, ensures that the AD654 output meets the logic levels required at T1 (Pin 15) of the 8051.

The 8051 has two 16-bit timer/event counters on-chip (the 8052 and the 8032 have three). These counters, Timer 0 and Timer 1, can be programmed independently to operate as 16-bit time-interval or event counters. The use of Timer 0 and Timer 1 is determined by two 8-bit registers, TMOD (timer mode) and TCON (timer control). The TMOD register is shown in Figure 2 below:

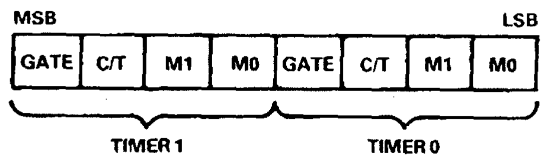

Figure 2. 8051 TMOD Register

"M1" and "M0" are used to select the mode of each timer. Mode 01 configures the timer as a 16-bit time-interval or event counter. "C/T" is the timer or counter selector and is cleared for timer operation. In this application, Timer 0 was configured as the timer (to provide the fixed time interval) and Timer 1 was configured as the counter (to count the pulses). Hereafter, the two timers will be referred to as Timer 0 and Counter 1. When running, Timer 0 increments at a rate equal to the external clock divided by twelve. Using a 12MHz crystal, this corresponds to once every microsecond. "GATE" is the gating control. When this bit is clear, timer/counter "X" is enabled whenever the "TRx" control bit found in the TCON register is set. The "TRx" bit is controlled via software. When the "GATE" bit is set, timer/counter "X" is enabled whenever the "TRx" bit is set and the signal level appearing at the $\overline{\text{INTx}}$ pin (Pin 12 or 13 for Timer 0 or 1, respectively) is high. Thus, when a GATE bit is clear that timer is controlled by software only, and when it is set that timer is controlled by a combination of software and hardware. In this application the GATE bits are clear; however in the next application they will be set.

Table I lists the software routine PLSECNT, which counts the number of falling edges that appear at T1 (the Counter 1 input) in a 50ms window. After Counter 1 is cleared, the value 15539 is loaded into Timer 0. Since Timer 0 is a 16-bit timer, the maximum possible count is 65535. With the Timer 0 interrupt enabled, a count of 65536 will cause a jump to the starting address (0BH) of the Timer 0 interrupt service program. With Timer 0 starting at 15539 and incrementing once every µs (based on a 12MHz clock), there will be 49,997 counts or 49.997ms before jumping to the service program. The 3µs difference from 50ms is made up in the speed of the interrupt response. The interrupt response latency ranges from 3µs to 7µs when using a 12MHz crystal. During this 50ms count period, control resides with the main program. Thus the 8051 is not tied up while Counter 1 is counting for 50ms. After the interrupt service is reached, Counter 1 and Timer 0 are stopped and the contents of Counter 1 are moved into RAM, where it may be accessed at the user's convenience. Control is then returned to the main program for which the subroutine was written. With a maximum frequency of 500kHz and a count window of 50ms, the maximum value of Counter 1 will be 25,000. This provides resolution greater than 14 bits. Appropriate scaling from this 1V full-scale reference point may then be performed in software.

```
         ORG    00H
         AJMP   MAIN
PLSECNT  ORG    60H          ;PULSE COUNT SUBROUTINE
         MOV    TMOD, #51H   ;Put Time 0 and Count 1 in Mode 01
         MOV    TL1, #00H    ;Initialize Counter 1 Register
         MOV    TH1, #00H
         MOV    TL0, #0B3H   ;Load Time 0 With 15536 + 3.Will
         MOV    TH0, #3CH    ;Ovflw After 50ms + 3µs Delay
         SETB   PT0          ;Prioritize Time 0 Interrupt
         SETB   ET0          ;Enable Time 0 Interrupt
         SETB   EA           ;Enable Global Interrupt
         SETB   TR0          ;Start Timer
         SETB   TR1          ;Start Counter
         RET                 ;Return to Main Program
         ORG    0BH          ;TIME 0 INTERRUPT SUBROUTINE
         CLR    TR1          ;Stop Counter
         CLR    TR0          ;Stop Timer
         AJMP   COUNT
         ORG    40H
COUNT    MOV    50H,TL1      ;Move Counter Contents Into RAM
         MOV    51H,TH1
         RETI                ;Return from Interrupt
         ORG    100H
MAIN     –      –            ;Main Program for Which PLSECNT
                             ;Subroutine Was Written
```

Table I. 8051 Pulse Count Routine

PERIOD TIMING

Another method of performing analog-to-digital conversion using a voltage-to-frequency converter (VFC) and a microcomputer is to have the microcomputer time the period of the VFC output frequency. For example, an output frequency of 25kHz has a period of 40µs. If a timer that is incremented once per microsecond is gated to this signal, a total count of 40 will result. An output frequency of 250Hz has a period of 4ms. The same timer gated to this period signal will then produce a total count of 4000.

One of the advantages of period timing over pulse counting is that the count window is dependent upon the output frequency of the VFC; in many cases the count window will be shorter for period timing than for pulse counting. This will be especially important in systems where a number of channels are being converted. Recall that in the Pulse Counter application the count window was 50ms—whether the output frequency was 50kHz or 50Hz. With Period Timing, the count window is the inverse of the output frequency. Thus, a 50kHz signal will have a count window of 20µs, while a 50Hz signal will have a count window of 20ms. In fact, it's not until the output frequency reaches 20Hz that the Period Timing count window is equal to the 50ms Pulse Counter count window.

Figure 3 shows the circuitry necessary to perform an analog-to-digital conversion via period timing using the Analog Devices AD650 voltage-to-frequency converter. The AD650 has a maximum full-scale frequency of 1MHz with a nonlinearity error of 0.1% max at this frequency. The AD650 in Figure 3 is configured such that a 0 to +10V input will result in a 0 to 50kHz output. Because the AD650's output is made up of pulses, the SN7474 D-type flip-flop is used to convert these pulses into a square wave. The SN7474 Pin 3 and Pin 5 waveforms sketched in Figure 3 show that the width of either the high-level or low-level output appearing at Pin 5 is the same as one period of the AD650 output frequency. Note also that when Pin 1 on the SN7474 is held low, Pin 5 is also held low.

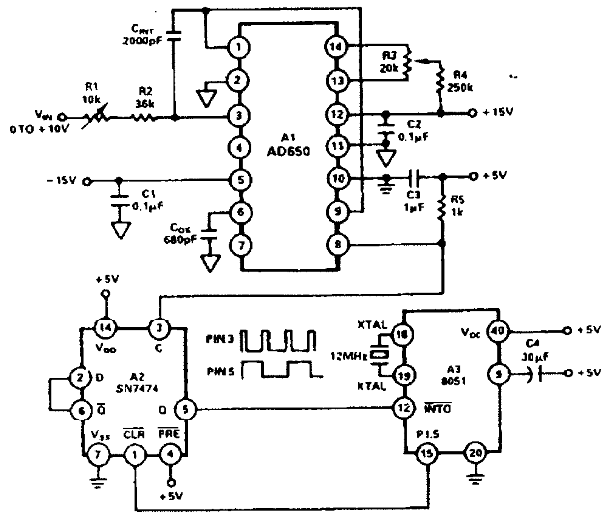

Figure 3. AD650 Period Timing

Recall that the $\overline{\text{INTO}}$ pin (pin 12) on the 8051 is the Timer 0 gate pin. (See Pulse Counter discussion). When the "GATE" bit is set in the TMOD register, Timer 0 will run only when $\overline{\text{INTO}}$ at Pin 12 is high and the TR0 bit in the TCON register has been set via software. Thus, connecting the "Q" output of the SN7474 to the $\overline{\text{INTO}}$ pin on the 8051 will ensure the timer runs for one period of the AD650 frequency.

One problem that could arise is the TR0 bit in software being set in the middle of a high-level edge at the $\overline{\text{INTO}}$ pin. In this case, Timer 0 would run for a fraction of a period rather than one full period. This is countered by tying the 8051 Port 1 Bit 5 (P1.5) pin to the SN7474 $\overline{\text{CLR}}$ pin. When $\overline{\text{CLR}}$ is low and $\overline{\text{PRE}}$ is high, Q is low. When $\overline{\text{CLR}}$ and $\overline{\text{PRE}}$ are both high, Q changes state on every positive edge appearing at the clock (C) pin. Setting P1.5 low, setting TR0 (in software) and then setting P1.5 high will thus ensure that Timer 0 runs for one full period.

Table II shows the software subroutine PCNT, which increments Timer 0 once per microsecond for one AD650 output frequency period. Note that there are two interrupt service programs, one for $\overline{\text{INTO}}$ and one for Timer 0. The $\overline{\text{INTO}}$ service program is accessed after a negative edge appears at the $\overline{\text{INTO}}$ pin (Pin 12), signifying the end of one period. The timer is then stopped and its contents are loaded into RAM. The user may then access the contents at his convenience.

```
        ORG     00H
        AJMP    MAIN
PCNT    ORG     90H           ;PERIOD COUNTER SUBROUTINE
        MOV     TMOD, #05H    ;Put Time0 in Mode 1, Enable INTO Pin
        CLR     P1.5          ;Initially Set INTO Pin Low
        SETB    IT0           ;Specify Edge Triggered Interrupt
        MOV     TL0, #00H     ;Initialize Timer
        MOV     TH0, #00H
        SETB    EX0           ;Enable INTO Interrupt
        SETB    ET0           ;Enable Timer 0 Interrupt
        SETB    EA            ;Enable All Interrupts
        SETB    TR0           ;Start Timer
        SETB    P1.5          ;Enable Gate INTO Pin
        RET                   ;Return to Main Program

        ORG     03H           ;INTO Subroutine Service Program
        CLR     TR0           ;Stop Timer
        CLR     EA            ;Disable Interrupts
        AJMP    COUNT         ;Jump to Count

        ORG     0BH           ;TIMER 0 SUBROUTINE
                               SERVICE PROGRAM
        CLR     TR0           ;Stop Timer
        CLR     EA            ;Disable Interrupts
        AJMP    OFLW          ;Jump to OFLW

        ORG     40H
OFLW    MOV     60H, #FF      ;Load RAM With Overflow
        MOV     61H, #FF      ;Value
        CLR     P1.5          ;Set INTO Pin Low
        RETI                  ;Return From Subroutine

        ORG     50H
COUNT   MOV     60H,TH0       ;Load RAM with Counter Contents
        MOV     61H,TL0
        CLR     P1.5          ;Set INTO Pin Low
        RETI                  ;Return From Subroutine

        ORG     100H
MAIN    -       -             ;Main Program for Which
                              ;Subroutine Was Written
```

Table II. Period Timing Routine

The Timer 0 service program serves to limit the count window to approximately 65.5ms. It is accessed when the contents of Timer 0 reach 65536. This will occur when the AD650 input voltage is approximately 3.05mV, or the output frequency is approximately 15.26Hz. This service program then loads the overflow value "65535" into RAM. After both interrupt subroutines, and after initialization of the PCNT subroutine, control returns to the main program for which the subroutine was written. Thus the 8051 is not tied up while the period timing is occurring.

One possible source of error in this application is jitter, which is the range of variation in the period of the output frequency. This variation in period would result in a variation in the number of pulses counted from one period to the next. The magnitude of this error can be greatly reduced in software by taking the average of a number of period counts and using this average value for calculations.

COMPLETE 16-BIT RESOLUTION ANALOG-TO-DIGITAL SYSTEM

A complete 16-bit resolution analog-to-digital conversion system is shown in Figure 4. This system uses the Analog Devices AD651 as its VFC. The AD651 is a 2MHz full-scale output synchronous voltage-to-frequency converter, requiring an external clock to generate the full-scale frequency rather than relying on the stability of passive components. This allows the AD651 to achieve linearity and stability far superior to any other monolithic VFC available. Other key elements in this system include an Intersil 7208 single chip counter-decoder-LED driver, a 4MHz TTL oscillator, and two 4020B binary counters.

The AD651 is configured in the 0 to +10V input, 2MHz output mode. Pull-up resistor, R1, is used to feed the AD651 output frequency directly into the counter input pin of the 7208 counter-decoder-driver. The 4020Bs are 14-stage binary ripple counters which have clock and master reset inputs (Pins 10 and 11), and buffered outputs from the first stage and the last eleven stages. The 4020Bs are used to generate the count window for the 7208. By dividing the 4MHz TTL clock by 2^{18} (two divide by 2^9 stages cascaded) a 15.2588Hz signal results. The 7208 will count the negative edges of the AD651 output pulses when pin 13 is low. With a 15.2588Hz signal, the count window will be 32.768ms.

Besides being tied to the count enable pin, the 15.26Hz signal is also tied to the 74LS221 dual monostable multivibrator.

Figure 5 shows the purpose of the 74LS221 one-shot. After the count window on the 7208 has been closed (that is, the count enable input has gone high), the data must be latched and then decoded in order to drive the LEDs. After this data has been latched, the counter must then be cleared in order to get an accurate count during the next count window. The purpose of the 74LS221 is to provide the required $\overline{\text{STORE}}$ input and $\overline{\text{RESET}}$ input pulses. Although the 7208 only requires pulse widths greater than 50µs, $R_3 \cdot C_2$ and $R_4 \cdot C_3$ have been selected to provide a pulse width of approximately 500µs.

The 7208 automatically handles the decoding and driving of the LEDs. The only other components required are resistors R_5 and R_6 and capacitor C_4. These non-critical passive components control the display multiplex rate of the

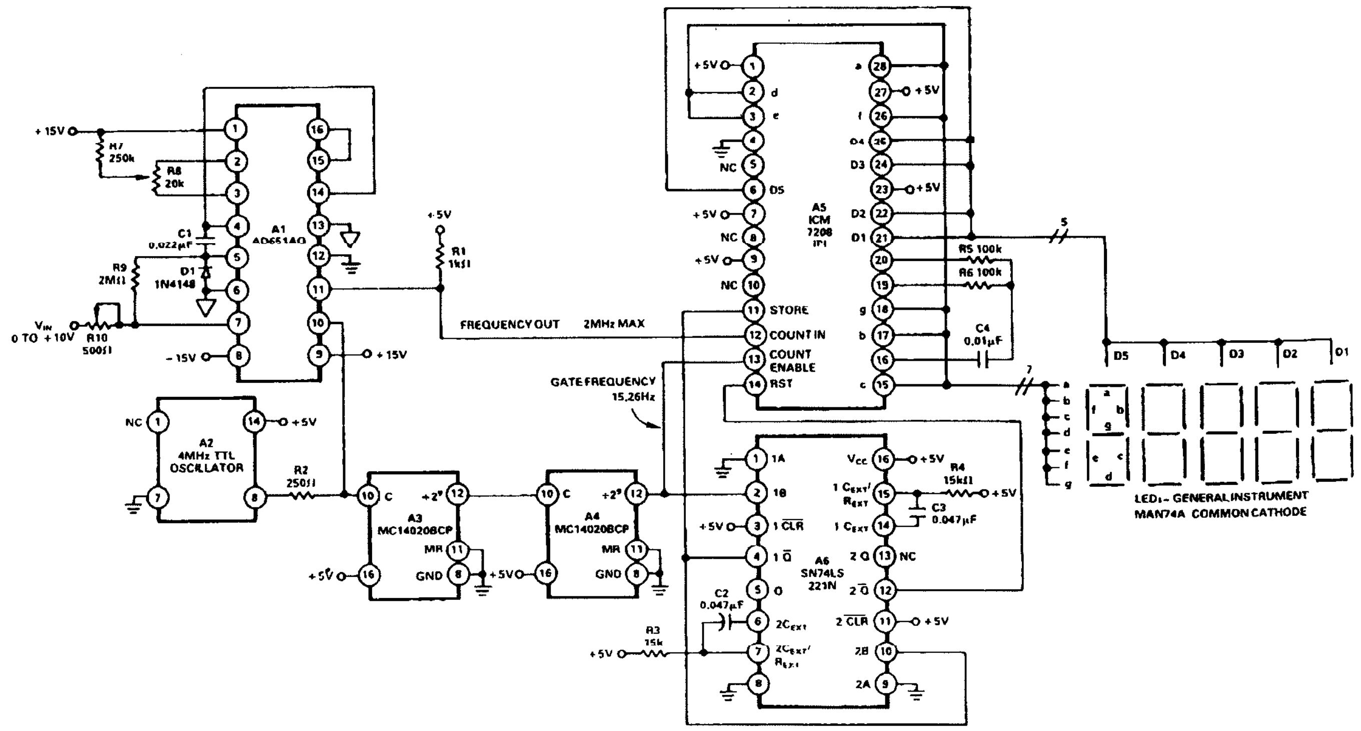

Figure 4. Complete 16-Bit Analog-to-Digital System

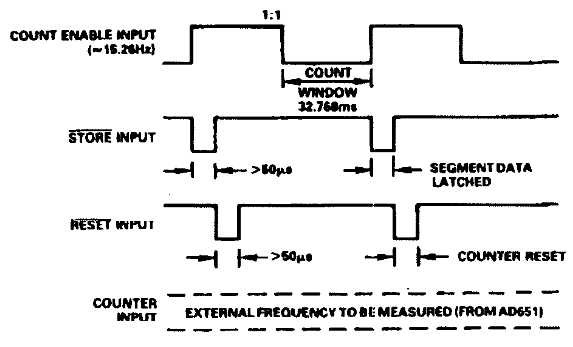

Figure 5. ICM7208 Timing Diagrams

LEDs and were selected following the manufacturer's guidelines to provide a multiplex rate between 50 and 200Hz.

To achieve maximum performance with the AD651, the fixed gate interval (or the "count window") should be generated using a multiple of the AD651 SVFC clock input, as was done in this application. Counting in this manner eliminates any errors due to the clock (whether it be jitter, drift with time or temperature, etc.) since it is the ratio of the clock and output frequencies that is being measured.

The resolution of the A/D conversion, of course, is determined by the clock frequency and the gate time. If, for instance, a resolution of 12 bits is desired and the clock frequency is 1MHz (resulting in an AD651 FS frequency of 500kHz) the gate time will be:

$$\left(\frac{\text{FS Freq}}{N}\right)^{-1} = \left(\frac{1}{2}\,\frac{\text{Clock Freq}}{N}\right)^{-1} = \left(\frac{1\text{MHz}}{2\,(4096)}\right)^{-1}$$

$$= \frac{8192}{1\times10^6}\ \text{sec} = 8.192\text{ms:}\quad \text{where N is the total number of codes for a given resolution}$$

Table III shows the relationship between the AD651 clock frequency and gate time for various degrees of resolution. In this application, 16 bits of resolution were desired using a 4MHz clock, making the required gate time 32.77ms.

GATING OFF A KNOWN INTERFERING SIGNAL

One source of error in analog-to-digital systems is an interfering signal that couples into the analog signal to be converted. For example, undesired coupling of powerline energy often appears as a sine wave riding on top of the DC level to be converted, resulting in an erroneous digital output. Since the frequency of this interfering sine wave is well-known (50 or 60Hz), the errors caused by the pickup can be integrated out by using a gating time equal to a multiple of the period of the sine wave. A replica of the interfering signal can be picked off of a nearby transformer and fed into a phase locked loop, a block diagram of which is shown in Figure 6. This loop provides two output signals – a high frequency clock (at a high harmonic of the interference) and a gating clock (at a lower harmonic of the interference).

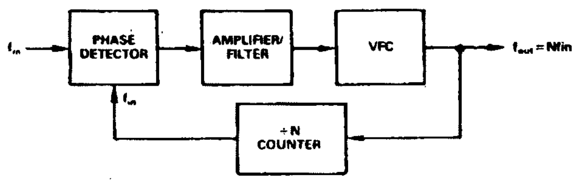

Figure 6. Phase Locked Loop

By using f_{OUT} as the AD651 clock source and by using f_{IN} from the divide by N counter as the count window source, a resolution of one part in 1/2 N is achieved, where N is the "÷N" of the counter. If the count window is level-triggered rather than edge-triggered (as with the 7208), then the resolution will be 1/4 N.

Figure 7 shows the hardware required to implement Figure 6. The MC4044 in Figure 7 contains both the phase detector and the amplifier/filter shown in the block diagram. The interfering signal was 60Hz and was converted into a TTL level signal to feed into the MC4044. Components R_1, R_2 and C_1 were selected to allow for a 50Hz or a 60Hz interfering signal. The error voltage from the MC4044 is then fed into the AD654, which is configured in a 0 to +1V input, 0 to 500kHz output mode. Because the 74LS393

Resolution	N	Clock	Conversion or Gate Time	Typ Lin	Comments
12 Bits	4096	81.92kHz	100ms	0.002%	50,60,400HzNMR
12 Bits	4096	2MHz	4.096ms	0.01%	
12 Bits	4096	4MHz	2.048ms	0.02%	
4 Digits	10000	200kHz	100ms	0.002%	50,60,400HzNMR
14 Bits	16384	327.68kHz	100ms	0.002%	50,60,400HzNMR
14 Bits	16384	1.966MHz	16.66ms	0.01%	60HzNMR
14 Bits	16384	1.638MHz	20ms	0.01%	50HzNMR
4 1/2 Digits	20000	400kHz	100ms	0.002%	50,60,400HzNMR
16 Bits	65536	655.36kHz	200ms	0.002%	50,60,400HzNMR
16 Bits	65536	4MHz	32.77ms	0.02%	

Table III. AD651 Clock Frequency and Gate Time Relationships for Various Resolutions

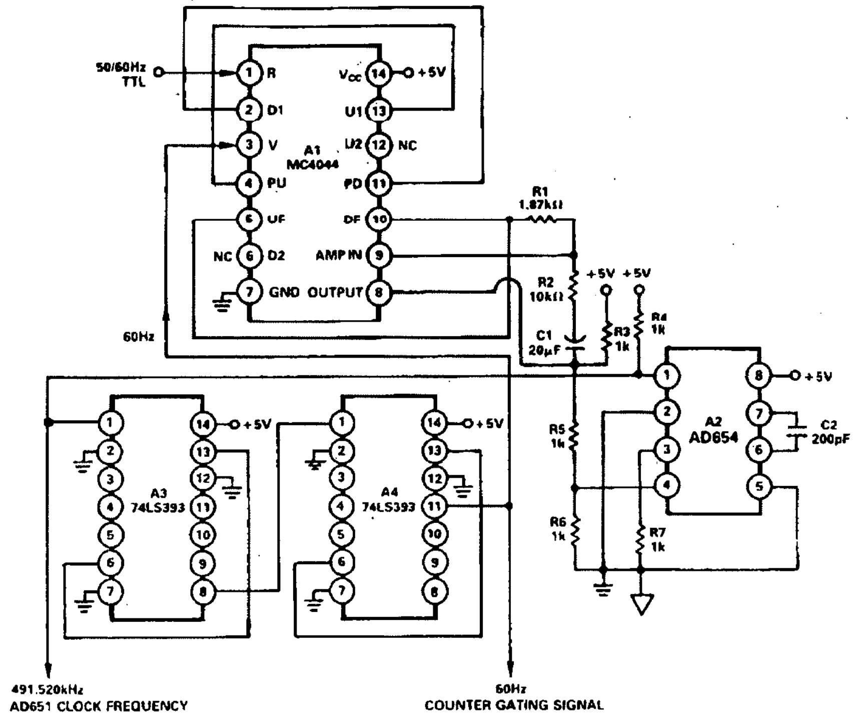

Figure 7. Gating Off a Known Interfering Signal

dual 4-bit binary counters are configured to provide a "÷N" of 8192, the output frequency of the AD654 will be 491520Hz. This signal is used as the clock for the AD651. The output of the second counter is 60Hz and is both fed back into the MC4044 and used as the frequency counter gate signal.

In this system it is also possible to get higher resolution by using Pins 10, 9 or 8 of A4 as the frequency counter gate signal. Pin 11, though, must be fed back into the MC4044. From Pin 1 of A3 to Pin 11 of A4, N = 8192 or 2^{13}. This will supply 12-bit resolution. Using Pins 10, 9 or 8 to provide the gate frequencies will supply resolutions of 13, 14 or 15 bits, respectively.

MC6801-AD650 BASED ANALOG-TO-DIGITAL CONVERSION

In some applications, it may be necessary to perform an analog-to-digital conversion with a microprocessor that either does not have an on-chip timer/counter, or does not have an available on-chip timer/counter. Using some additional hardware, it's still possible to count the output pules of a VFC for a fixed period of time and store this count in the microprocessor's RAM, when the user may access it at his convenience.

Figure 8 shows the circuitry necessary to perform the analog-to-digital conversion using the AD650 VFC and the MC6801 μP, and assumes that the on-chip counter is dedicated to another function. The MC6801 is an 8-bit, single-chip microcomputer with 2048 bytes of ROM, 128 bytes of RAM, a serial communications interface, and a three function programmable timer. The AD650 VFC is configured in the 0 to +10V input, 0 to 1MHz output mode.

The additional hardware necessary to count the output pulses of the AD650 includes two 74590 8-bit binary counters with output registers, one 4020B 14-stage binary counter, one 7474 dual D-type flip-flop with preset and clear, and one inverter of a 74LS04 hex inverter.

The 4020B and 7474 provide the timing signal which tells the counters when to start and stop counting. The input to the 4020B is tied to the 'E' Pin (Pin 40) of the MC6801. The signal at the 'E' Pin is simply the MC6801 external crystal frequency divided by four, or in this case 1.2288MHz. The 4020B divides the 1.2288MHz signal by 2^{14}, which results in a 75Hz signal being fed into Pin 3 of the 7474. The 7474 further divides this signal by four, to 18.75Hz. This signal will appear at Pin 9 depending upon the signal level at Pins 1 and 10. If Pins 1 and 10 are low, Pin 9 will be held at a TTL high. If Pins 1 and 10 are high, the 18.75Hz square wave will appear at Pin 9. When Pin 9 is high, the 74590 counters are disabled and no counting occurs; if the output of Pin 9 is a square wave, the counters will be enabled during the low level of the period and will thus count for 26.67ms. Note that it is Bit 4 of Port 1 (P14) that controls whether the counters will be enabled or not. If this bit is low, no count will occur. Note also that Pin 9 is connected to the external interrupt request ($\overline{\text{IRQ1}}$) pin of the MC6801 through an inverter. This produces two results when Pin 9 changes from low to high. First, the counters are disabled since $\overline{\text{CCKEN}}$ of A2 is tied to Pin 9. Secondly, the low-to-high transition is inverted and generates an interrupt request on the $\overline{\text{IRQ1}}$ line. The MC6801 then clears P14 to prevent another count from occurring before the 74590 counter values are read.

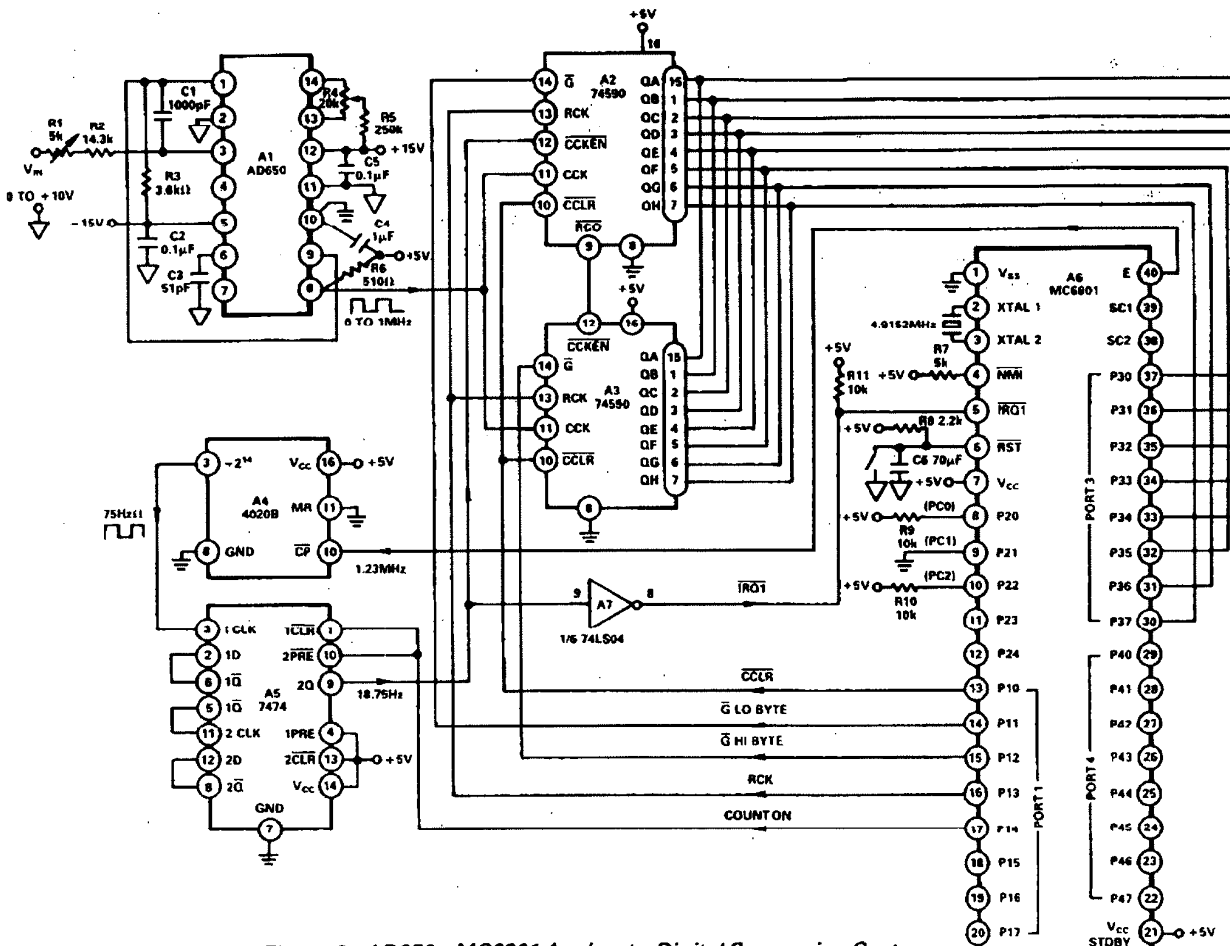

Figure 8. AD650–MC6801 Analog-to-Digital Conversion System

All of the counting events are controlled by the signal levels of the different bits of Port 1. By writing different values to Port 1 the 74590 counters are cleared, enabled, disabled, latched, and read. The values which control these functions are shown in Table IV below.

Port 1 Configuration

Event	P4	P3	P2	P1	P0	Hex
Clear Counters	0	0	1	1	0	06
Enable Counters	1	0	1	1	1	17
Disable Counters	0	0	1	1	1	07
Latch Data	0	1	1	1	1	0F
Output Hi Byte	0	1	1	0	1	0D
Output Lo Byte	0	1	0	1	1	0B

Table IV. Port 1 Event Control Values

Table V shows the software routine that controls the AD650 pulse counter routine. This routine sets the stack pointer, disables the interrupts, clears the 74590 counters and then enables the interrupts after a delay period (this will be discussed later). After the interrupt request is triggered by a low-to-high transition at Pin 9 of the SN7474, the program counter jumps to the interrupt routine. This routine disables any further interrupts, turns off the 74590 counters and reads the lo and high bytes,

```
                      ORG    0100            COUNT ROUTINE
8E  00C0   BEGIN      LDS    #$C0            Set Stack Pointer
0F                    SEI                    Disable Interrupts
86  06                LDAA   #$06            Clear Counter
97  02                STAA   $02
86  17                LDAA   #$17            Turn On Counter
97  02                STAA   $02
86  2F                LDAA   #$2F            Insert Delay >13.33ms To
C6  7F     AGN        LDBB   #$7F            Prevent Interrupt Error
5A         CNT        DECB
2E  FD                BGT    CNT
4A                    DECA
2E  F8                BGT    AGN
0E                    CLI                    Allow Interrupt
39                    RTS                    Return From Subroutine

                      ORG    FFF8
0080                  AIRQ1  FDB    LDCNT    Define Interrupt Routine
                                             Start Point

                      ORG    0080            INTERRUPT ROUTINE
0F         LDCNT      SEI                    Disable Interrupt
86  07                LDAA   #$07            Turn Off Counter
97  02                STAA   $02
86  0F                LDAA   #$0F            Latch Data In Counter
97  02                STAA   $02
86  0D                LDAA   #$0D            Output Counter Lo Byte
97  02                STAA   $02
96  06                LDAA   $06             Read Byte From Port 3
97  A7                STAA   $A7             Store in Location 00A7
86  0B                LDAA   #$0B            Output Counter Hi Byte
97  02                STAA   $02
96  06                LDAA   $06             Read Byte From Port 3
97  A6                STAA   $A6             Store In Location 00A6
86  07                LDAA   #$07            Turn Off Counter
97  02                STAA   $02
86  10                LDAA   #$10            Set The Interrupt Bit On The
9A  BA                ORAA   $BA             Stacked Condition Code Register
97  BA                STAA   $BA
3B                    RTI                    Return From Interrupt
```

Table V. MC6801 Pulse Count Routine

interrupt bit on the stacked condition code register is then set and the program counter returns from the interrupt routine.

Two areas in the software routine need to be discussed in further detail – the time delay before enabling the MC6801 interrupt and setting the interrupt bit on the stacked condition code register.

Figure 9 shows the waveforms that will appear around the time the 74590 counters are instructed to start counting by setting Pins 1 and 10 high.

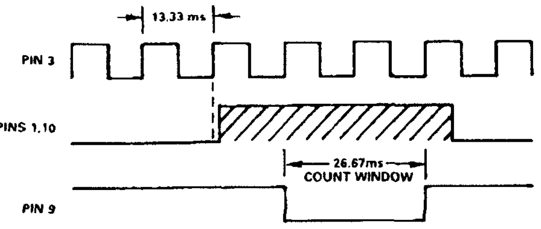

Figure 9. SN7474 Waveforms

Recall that Pin 9 will only output an 18.75Hz square wave when Pins 1 and 10 are held high. Pin 9 will not change state after Pins 1 and 10 go high until the next positive edge occurs at Pin 3. Figure 9 shows the worst case that will happen – Pins 1 and 10 go high just a few nanoseconds after Pin 3 sees a positive going edge. Pin 13.33ms. Also, recall that Pin 9 is fed to the IRFQ1 pin through an inverter, thereby setting the signal level low during the 13.33ms wait state. Since $\overline{\text{IRFQ1}}$ is level sensitive, allowing an interrupt during this wait state would cause a jump to the interrupt routine before the 26.67ms count window has been opened up. By inserting a delay greater than 13.33ms before allowing the interrupt, the interrupt is enabled during the count window (when Pin 9 is low, and thus the $\overline{\text{IRFQ1}}$ pin is high). This will ensure that the interrupt routine is accessed after the 26.67ms count window closes.

Setting the interrupt bit on the stacked condition code register is also important. After executing the CLI instruction in software, Bit 4 of the condition code register (CCR) is cleared. This enables the $\overline{\text{IRQ1}}$ interrupt. Once an interrupt request is detected (a low level at the $\overline{\text{IRQ1}}$ pin), the CCR is pushed onto the stack in its present state. A total of seven bytes are pushed on the stack and the last one *pushed is the CCR. Since the stack pointer was set at location* 00C0 the location of the CCR is 7 bytes down, or at location 00BA. By OR'ing 10H with the contents of location 00BA the interrupt is disabled. This will prevent the program counter from jumping into the interrupt routine again right after coming out of it, even though the level sensed at the $\overline{\text{IRQ1}}$ pin is low. If this bit were not set, the program counter would jump out of the interrupt routine, see that the interrupt enable bit was clear and that the level at the $\overline{\text{IRFQ1}}$ pin was low, and jump right back into the interrupt routine again.