AN1708

Single-Slope Analog-to-Digital (A/D) Conversion

By Stephen Ledford
CSIC Product Engineering
Austin, Texas

Introduction

The most common implementation for analog-to-digital (A/D) conversion among Motorola microcontrollers is the successive approximation (SAR) method. An alternative A/D conversion technique uses the single-slope A/D converter.

This application note explains the theory of operation of the single-slope A/D, applies the principles of operation to the single-slope A/D on the MC68HC705JP7, and provides software examples for the control of the A/D system found on this MCU. Comparisons are made with the SAR A/D where appropriate.

Typical A/D converters implemented with the SAR method have resolutions of eight bits and a conversion time of 10 to 16 clock cycles. The conversion rate is one of the advantages of the SAR A/D implementation. However, for many reasons, increasing the resolution of the SAR A/D converter becomes difficult above 8 bits.

The advantages of using the single-slope A/D converter are increased resolution, which in the case of the 68HC705JP7 is 12 bits, and the simplicity of the design. However, to obtain the increase in resolution, the rate of conversion is increased when compared to the SAR A/D. The single-slope A/D conversion utilizes several unique components: an

external ramp capacitor, a constant current source, a simple voltage comparator, and a free-running timer. Fundamentally, the conversion is represented as a difference in time. The conversion begins with the external ramp capacitor at V_{SS} and is charged by the constant current source until it reaches the same level as the sampled analog value. At this point, the voltage comparator transitions resulting in the capture of the free-running timer. The conversion value is the difference in the timer value from the start of the ramp to when it reaches the unknown analog value. Another point of difference with the SAR A/D is the converted value is in most cases not of immediate value and requires additional computation by the CPU for the result to be useful in system decisions and actions.

Theory of Operation

A typical SAR A/D system has several inputs, analog channels, voltage references, and in some cases a separate analog power to provide increased isolation to noise. The single-slope A/D converter on the MC68HC705JP7 has similar inputs. However, one additional component is necessary to support the conversion process: an external capacitor for charge integration. The external capacitor is one of the fundamental differences of the single-slope A/D converter design.

An examination of the SAR A/D helps identify the differences between it and the single-slope A/D converter. The basic components of the SAR A/D are an internal comparator, an approximation register, a state machine to provide control of the converter and an analog feedback path. A diagram of a simplified SAR A/D system is shown in **Figure 1**.

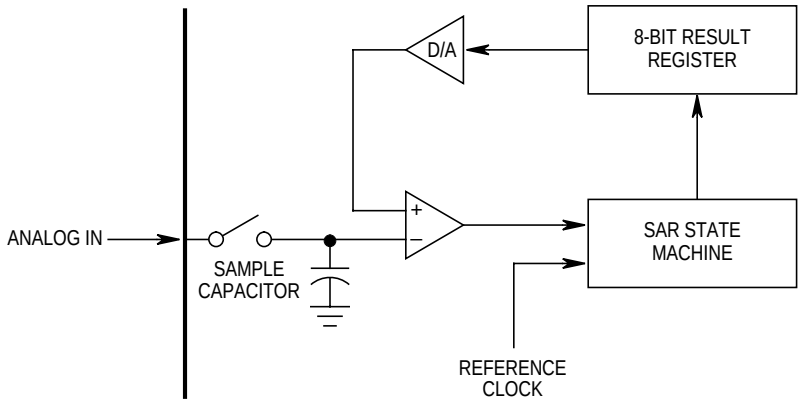

Figure 1. SAR A/D Converter Example

A conversion begins by the external analog value being sampled and stored on a storage capacitor. The state machine is initialized to begin a conversion along with the SAR register. The analog feedback path for this example is shown as a digital-to-analog converter for input back into the comparator. The SAR register typically is initialized where the most significant bit is set, representing a 1/2 scale value. The comparison is made and the result from the comparator is fed into the SAR state machine which is stepped through each state by the reference clock. Depending on the comparator value, the state machine would either invert the bit in the case of a high or leave the bit in the case of a low. A low would represent that the external analog value is higher than the current value in an 8-bit register, or, conversely, a high would represent a value lower than the SAR register. For its next operation, the state machine would set the next lowest bit in the SAR register and perform the same comparison operation. This process would be repeated until all bits in the SAR register had been completed. After the last bit is finished, the value in the SAR is the digital representation of the external analog value. The time to convert the value is the number of clock cycles to perform the conversion times the period of the clock. This method is sometimes referred to as an analog binary search. Issues with increasing the resolution of the SAR A/D primarily involve the accuracy and resolution of the analog feedback path.

As mentioned previously, the single-slope A/D has many similarities to the SAR A/D. These similarities can be seen in the conceptual diagram of a single-slope system in **Figure 2**.

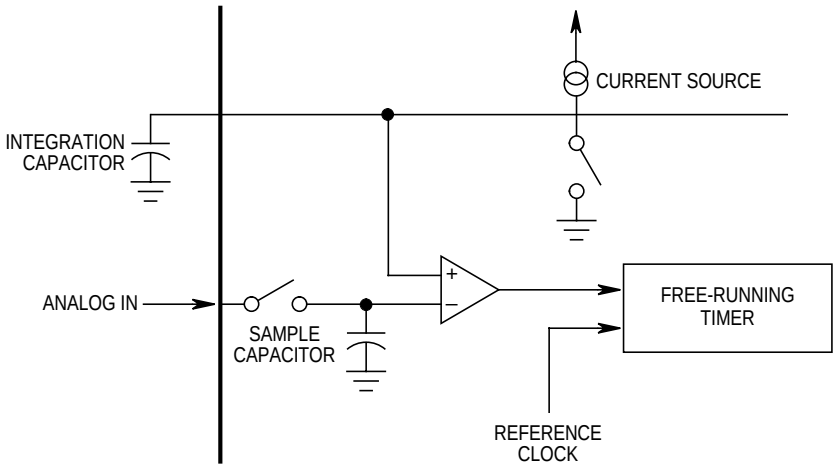

Figure 2. Single-Slope A/D Example

The main components of the single-slope system are the comparator, the constant current source for the external capacitor, and the free-running timer. A conversion begins by the external capacitor being discharged through the ground path. The analog signal to be converted either is held on the input continuously or is sampled on the internal storage capacitor for the duration of the conversion. The current source is used to charge the external capacitor. The charge transfer from the current source to the capacitor results in an increase in the capacitor voltage, which causes the comparator to transition from low to high when it reaches the level of the sampled analog voltage. The transition of the comparator is used to capture the value of the free-running timer, which is a digital representation of the sampled voltage.

The single-slope A/D conversion technique is very different from the more conventional SAR A/D, and a description of the theory of operation will assist in understanding how the capture of the timer translates to a conversion of the voltage.

In the single-slope A/D conversion, current is the transfer of charge in a given period of time and is represented as

$$i = dq / dt$$

In the case of a capacitor, the charge on the capacitor is proportional to the voltage across it, so that

$$q = C * v$$

Substituting this in the first current equation results in

$$i = dq / dt * C * v$$

Since the capacitor is a constant value, this becomes

$$i = C * dv / dt$$

and solving for the voltage is

$$dv = 1 / C * i * dt$$

Finally, integrating this equation results in

$$v(t) = 1 / C \int_{t0}^{t1} i(t)dt$$

For the integration of the voltage, the beginning and end of the integral represent the beginning and end of the A/D conversion sequence, from the discharge of the external capacitor to the time that the comparator transitions from low to high. This equation explains why the external capacitor is referred to as an integration capacitor due to the time integration of the current being sourced to the external capacitor. Since the current source is constant, the integral is simplified in practice to

$$v1–v0 = 1 / C * i * (t1–t0)$$

In addition, since the beginning of a conversion discharges the external capacitor to ground, the equation can be reduced even further to

$$v1 = 1 / C * i * (t1–t0) \quad \text{(equation 1)}$$

Since the values of the external capacitor and the current source remain constant, it can be seen that the external voltage is represented as a

difference in time from the start of the conversion to the end of the conversion. In this case, the difference is when the external integrating capacitor reaches the same voltage as the sampled analog value and the comparator transition captures the timer value. Solving equation 1 for the elapsed time gives

$$(t1–t0) = C * v1 / i$$

or

$$\Delta t = C * v1 / i \quad \text{(equation 2)}$$

The diagram of the conversion sequence in **Figure 3** will help visualize the operation.

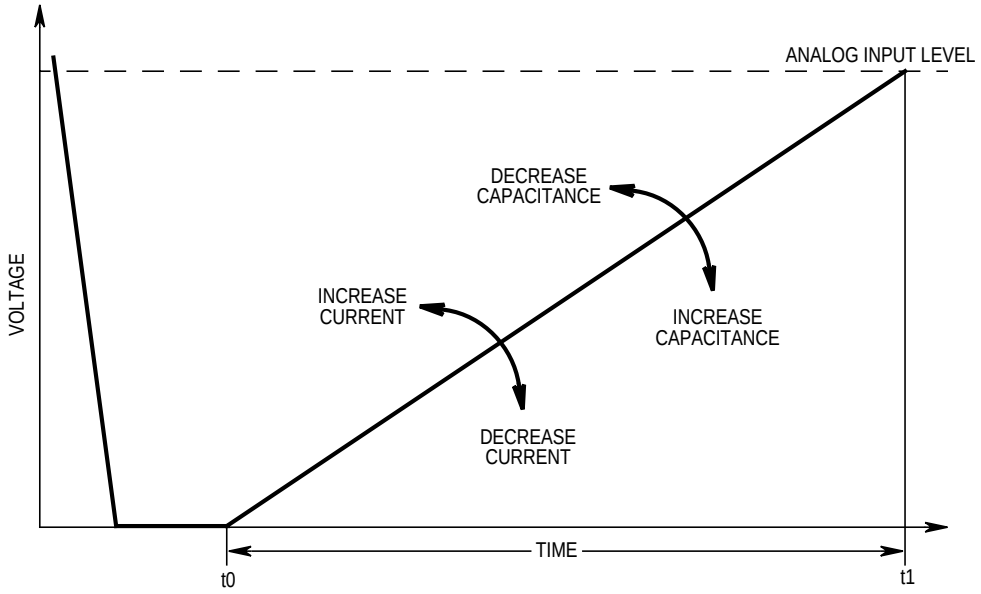

Figure 3. Relationship of Component Values to Conversion Time

Understanding the components that directly affect the conversion resolution and accuracy of the system is important. As described previously, the conversion of the analog value is represented as a difference in time. The amount of time required for the integrating capacitor to reach the external voltage is controlled by two primary variables: the external capacitor value and the current from the internal

current source. It is also important to understand the impact of the frequency of the timer on the resolution of the system.

The internal current source should be linear across the range of analog input values, and the user does not typically have control of the magnitude of this current. However, the external capacitor is the user's choice. By increasing the capacitor value the slope of the ramp voltage is decreased, thereby increasing the amount of time for the capacitor to reach the level of the analog value, to be converted. Conversely, decreasing the capacitor value increases the slope, thus decreasing the amount of time for the capacitor to reach the level of the analog value.

In the case of the SAR A/D converter, the resolution of the system is the number of bits in the SAR, as noted previously. Therefore, an 8-bit A/D converter has 256, or 2^8, discrete levels for the range of analog values to be converted. For the single-slope A/D converter, the analog value is represented as an elapsed time and thus a 12-bit resolution has 4096, or 2^{12}, discrete counts for the full range of voltage input. From this it can be seen how the frequency of the input clock to the timer directly affects the resolution of the converter. The maximum number of counts for the full-scale converted voltage needs to equal the amount of time for the integrating capacitor to ramp to the voltage. An example of this relationship is given in **Example 1**.

Example 1

The full-scale voltage to be converted is 3.0 V, the source current for the ramp capacitor is 100 µA, the ramp capacitor is 1.0 µF, and the target A/D resolution is 12 bits.

First, the elapsed time to ramp to the full-scale voltage is calculated using equation 2.

$$\Delta t = (1.0\ \mu F * 3.0\ V) / 100\ \mu A$$
$$\Delta t = 30{,}000\ \mu s$$

Using the elapsed time to calculate the input clock frequency of the timer,

$$clock\_period = \Delta t / count$$

The count is the effective resolution of the converter, which is 4096 or 2^{12}. Therefore, the calculated clock period is:

$$clock\_period = 30{,}000\ \mu s / 4096$$
$$clock\_period = 7324\ ns$$
$$or$$
$$frequency = 136.5\ kHz$$

By understanding that the conversion of the single-slope A/D is fundamentally a measurement of elapsed time to ramp a capacitor to the level of the unknown analog value, it can be seen why the single-slope A/D conversion speed is slower than the SAR A/D. Whereas an SAR A/D conversion is measured in microseconds, the single-slope A/D conversion primarily is measured in milliseconds. In addition, it can be seen from the previous example that as the target conversion resolution increases, the conversion time increases.

In practice, the current source and capacitance of the integrating capacitor have tolerances that vary from part to part and across temperature. The calculation of the previous example assumed that both the current source and capacitance are constant, whereas a typical capacitor can vary by as much as 50 to 80% and the current source tolerance typically would be ±10%. These variations can account for reductions in the resolution of a given set of measurements unless accounted for in the utilization of the A/D converter. Due to these variations, the single-slope A/D primarily is used in two different modes: ratiometric conversions and absolute value conversions. In either case, the CPU available on the MCU is utilized to perform calculations to determine how the converted value compares to other reference values or to calculate the absolute value of the measurement. An example of a ratiometric conversion can help the user understand the practice of using the single-slope A/D as well as accounting for the component variations and the impact on resolution.

Example 2

The full-scale voltage to be converted is 3.0 V, the source current for the ramp capacitor is 100 µA, the ramp capacitor is 1.0 µF and, the target A/D resolution is 12 bits. The tolerance on the ramp capacitor is ±50% across temperature and the current source varies by ±10%.

Using the calculation from **Example 1**, which utilized equation 2, note that an increase or decrease in the capacitor value of ±50% directly results in an increase or decrease in elapsed time of a similar percentage. Therefore, the integration time can either be a maximum of 45,000 µs or a minimum of 15,000 µs.

In addition to the capacitor variation, the current source variation needs to be accounted for, which is ±10% for this example. From equation 2, it can be seen that changes in the current source have an inverse relationship to the ramp time. Using the previous calculation of the extremes of the capacitor variation, the additional changes to the current source can be calculated. Therefore, the maximum integration time is 45,000 µs/0.9, in the case of a decrease of 10% in the current, or 50,000 µs and the minimum time would be 15,000/1.1, in the case of an increase of 10% in the current, or 13,636 µs.

The minimum integration time establishes the worst case value for resolution and should be used to calculate the clock period of the timer system. With a worst case time of 13,636 µs and a target resolution of 12 bits, the clock rate needs to be

$$clock\_period = 13{,}636\ \mu s / 4096$$
$$clock\_period = 329\ ns$$
$$or$$
$$frequency = 300.4\ kHz$$

In addition, since the amount of time for the ramp changes due to variations in capacitance or current, the timer count is not always a direct digital representation of the absolute value of the analog voltage. The only time that this would occur is when the component values used in the previous computations are exact. Because of this, a ratiometric method, or a comparison of one conversion to another conversion, is used to interpret the converted analog input. An example can assist in understanding this method.

Example 3

The full-scale voltage to be converted is 3.0 V, the source current for the ramp capacitor is 100 µA, the ramp capacitor is 1.0 µF, and the target A/D resolution is 12 bits. The tolerance on the ramp capacitor is ±50% across temperature and the current source varies by ±10%. A reference voltage of undetermined value is input on one channel of the A/D system and is used in the ratiometric calculations. The result of the conversion of the reference voltage is 3750. The analog input has a range of 0 V to 2.0 V.

From **Example 2**, it was calculated that the input frequency of the timer needs to be no less than 300 kHz to ensure a conversion resolution of 12 bits. The ramp time, as previously calculated in **Example 2**, can vary from 13,636 µs to 50,000 µs which would result in a timer count for a full-scale conversion of 3.0 V of 4096 to 15,015. It is given that the reference voltage conversion is 3750, less than the full scale conversion value. The analog input has a conversion value of 2000. The ratio of the two signals, reference over analog input, would be 3750/2000 or 1.875.

As long as the component values for the conversion of the reference voltage, the capacitance and current, are the same as for the conversion of the unknown analog value, the ratio of the two conversions will always be the same. In practice, the reference voltage should be converted immediately before the conversion of the unknown analog value in order to obtain the most accurate ratiometric result. This procedure would ensure that any gradual change over time in some or all of the component values of the measurement system would not impact its accuracy. The accuracy of the system will remain the same as long as the reference voltage and the analog input track each other across temperature or if the voltage reference is temperature compensated to remove variation across temperature. This method provides a means to eliminate the need for a fixed magnitude voltage reference and still obtain useful information about the analog inputs of the system.

An absolute value can be calculated for the converted analog voltage if the reference voltage in the system is known and is a constant value. From **Example 3**, a ratio of the reference to the converted analog voltage was calculated to be 1.875. If the voltage reference were a known value, for example 2.8 V, and is constant across temperature and

supply voltages, the absolute value of the analog system can be calculated.

$$analog\_input = reference / ratio$$
$$analog\_input = 2.8\ V / 1.875$$
$$analog\_input = 1.493\ V$$

It is important to note that the absolute value of the analog input can be calculated only if the reference voltage is stable across all variations. Otherwise, changes in the reference voltage will result in a loss of accuracy of the system's absolute voltage.

Features

These basic principles for a single-slope A/D converter can be applied to the operation of the A/D converter found on the MC68HC705JP7. The primary components of the MC68HC705JP7's A/D are: a 16-bit free-running timer with input capture, a low-power current reference, and an internal voltage comparator with an analog mux for selection of one of four external input sources. A detailed diagram of all the system components as well as the control registers can be found in the *68HC705JP7 General Release Specification*.

Several methods can cause the A/D system to perform conversions. They are:

1. Manual charge with manual discharge cycle counting
2. Manual charge with automatic discharge cycle counting
3. Automatic start of conversion from the timer overflow (TOF) with automatic discharge. Timer value value is captured by the input capture flag (ICF) of the timer.
4. Automatic start of conversion from the output compare flag (OCF) with automatic discharge. Timer value is captured by the input capture flag (ICF) of the timer.

Some special operation modes have been provided to improve implementation of the single-slope A/D in a system. In addition to the

four external analog inputs, there are two internal paths for conversion for V_{DD} and V_{SS}. The primary use of this would be if V_{DD} were used as the reference voltage in the system, in which case the reference would not occupy one of the four external analog inputs. The comparator inputs also can be swapped, or inverted, to provide a means to calculate any offset voltages in the comparator that would reduce the accuracy of the conversions. Finally, the input to the comparator can be divided by two internally to extend the input range of the conversions.

To begin, the equations used in the explanation of the theory of operation can be applied specifically to the MC68HC705JP7. Three basic equations were described in the theory of operation:

$$v1 = 1 / C * i * (t1–t0) \quad (1)$$

$$\Delta t = C * v1 / i \quad (2)$$

$$clock\_period = \Delta t / full\_count \quad (3)$$

The terms of the equations relate to various functions of the single-slope A/D on the MC68HC705JP7. In the first equation, the value v1 is the analog voltage being converted. The specification of the comparator on the MC68HC705JP7 states that the common-mode input range is from V_{SS} to V_{DD} –1.5 V. Therefore, the input to the A/D needs to be in this range or the internal voltage divider can be used to extend the range to as high as 2 * (V_{DD} –1.5 V) but still must be less than V_{DD} +0.5 V due to the input protection clamping diode. The clock period referred to in the third equation is the clock at the input to the 16-bit timer. Remembering that the MCU oscillator frequency is divided by eight before being routed into the timer is important. In addition, the pre-scaler to the timer can be utilized to achieve the proper frequency.

All of the software examples will require "tuning" based on the particular application requirements, resolution, and accuracy. The routines as written do not provide a particular conversion resolution but are intended as a framework for application coding and conversion management. From the examples and explanations provided previously, items such as the timer prescaler value and the choice of an external capacitor can be calculated and integrated in the application code.

A software polling routine, as mentioned previously, can be used to perform the A/D conversions. An advantage of a software polling routine is that a conversion still can be performed even if the timer is being used for other functions. However, some disadvantages are that the routine cannot be interrupted since the routine depends on cycle counting, the CPU cannot be placed in wait mode to reduce power consumption during the conversions when using the programmable timer, and the resolution of the conversion is more limited since the number of CPU cycles is larger than the programmable timer. The flowchart of the software polling routine for A/D conversion is shown in **Figure 5** and **Figure 6**.

The use of the programmable timer provides the most flexibility in the system as well as provides the highest resolutions for A/D conversions. The only bottle neck in the system might be if the programmable timer is needed for other functions within the system. Otherwise, the disadvantages noted for the software polling method are eliminated. The TOF/ICF method has the longest latency to perform a conversion since the conversion starts with a timer overflow. For example, if the timer count was $0010 when the main program called the subroutine, the timer would have to reach $FFFF before the conversion would begin, possibly several milliseconds. The OCF/ICF method eliminates the conversion latency but requires more code to manage the timer. Flowcharts for both methods are provided in **Figure 7**, **Figure 8**, and **Figure 9**.

The previously discussed routines for performing the A/D conversions all provide the converted result but do not perform any analysis or decisions on this data. Finally, an example of a main program that utilizes the OCF/ICF methods is shown in **Figure 4**. This is a basic program that performs a conversion on the reference channel, in this case V_{DD}, followed by a conversion of the analog input channel. The main program remains the same regardless of the conversion method chosen, with the exception of the calling subroutine to perform the conversion.

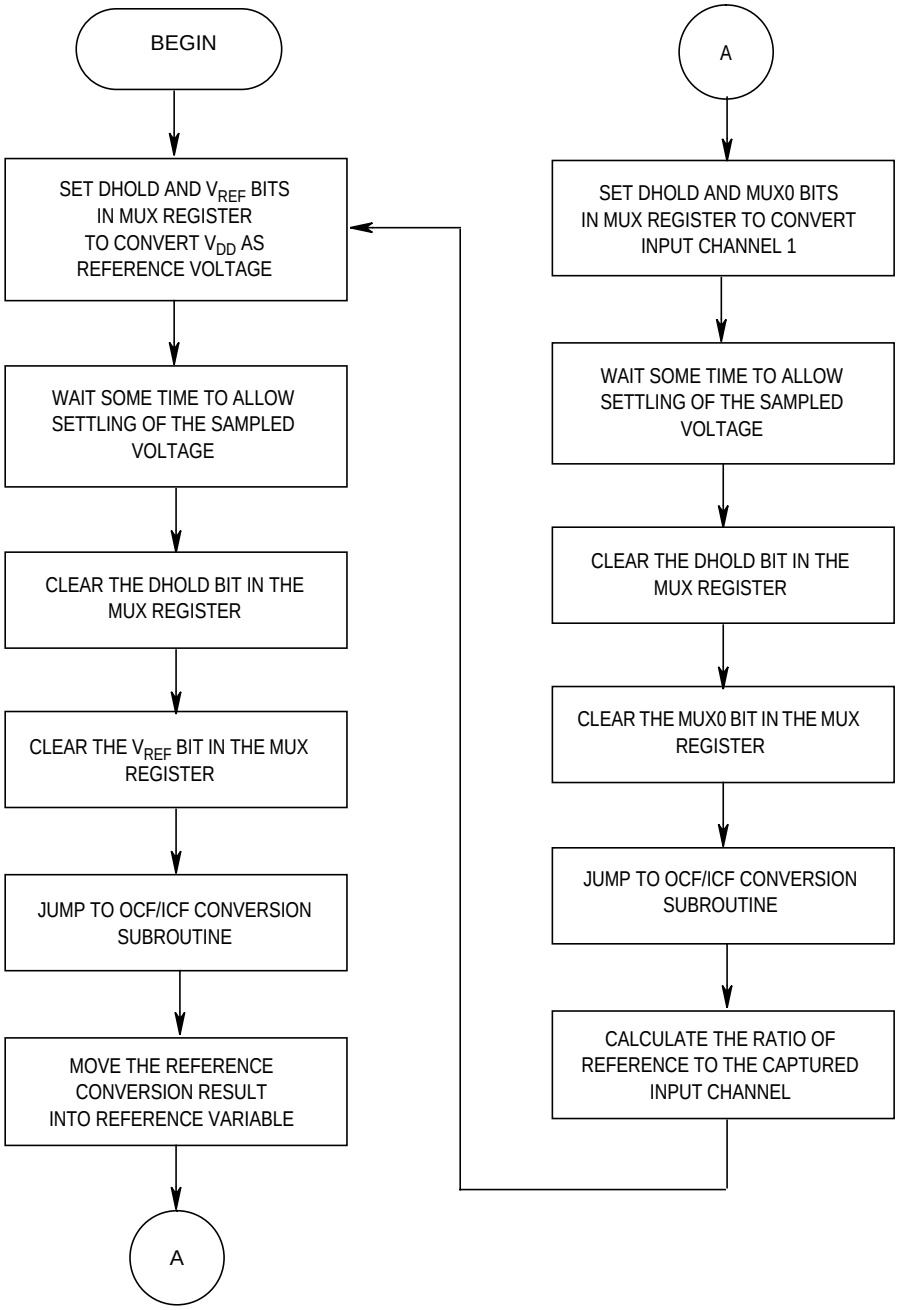

Figure 4. Main Program Using the OCF/ICF Conversion Method

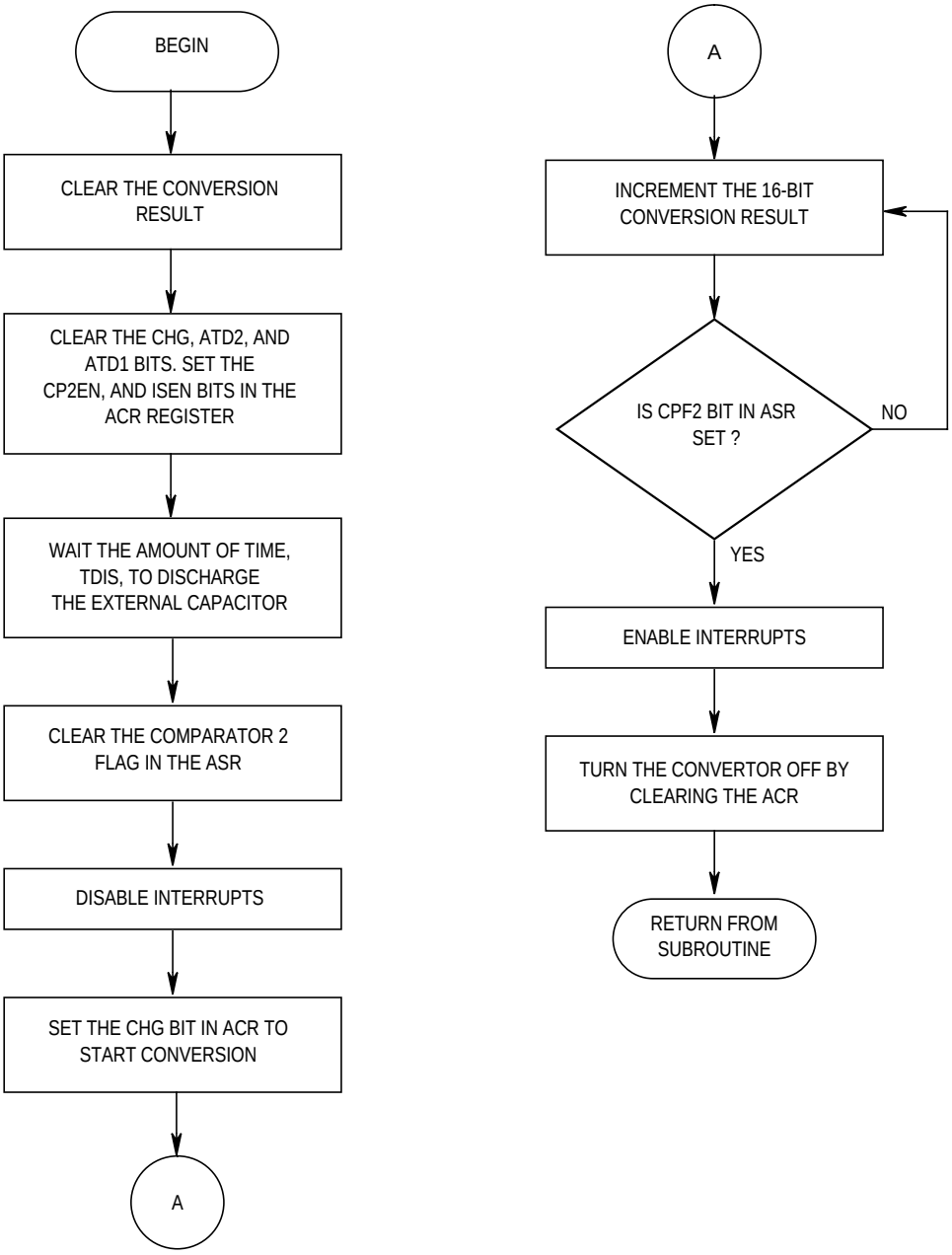

Figure 5. Manual Start/Manual Discharge Conversion Method Subroutine

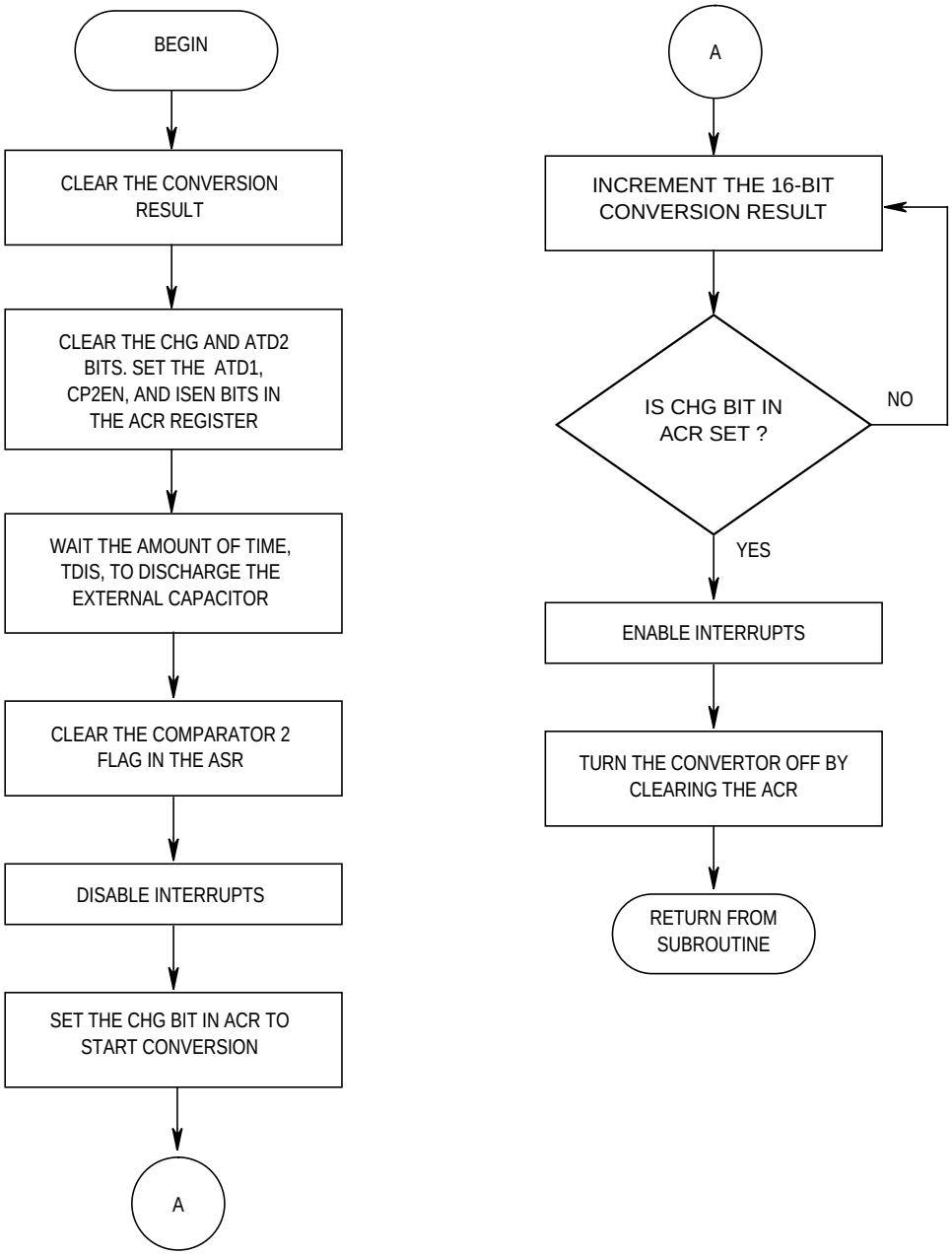

Figure 6. Manual Start/Automatic Discharge Conversion Method Subroutine

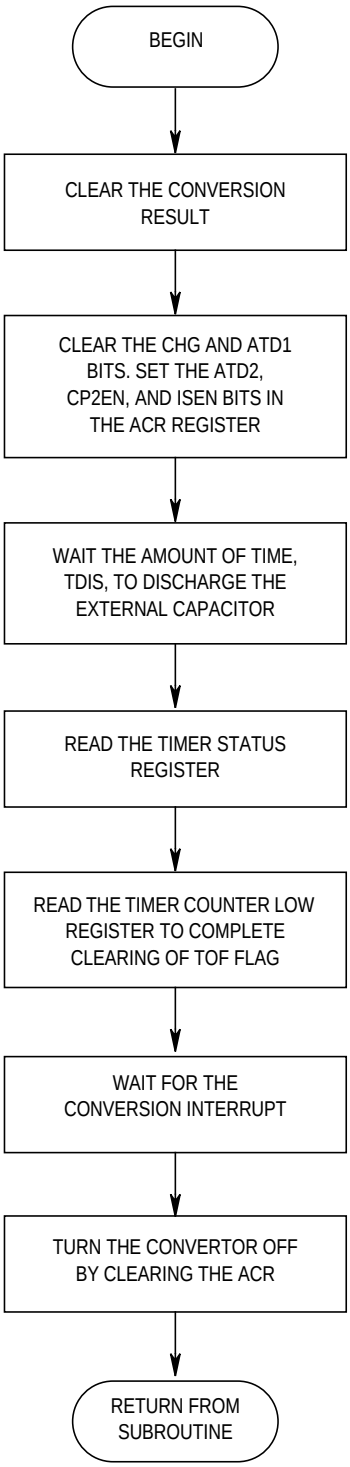

Figure 7. TOF/ICF Conversion Method Subroutine

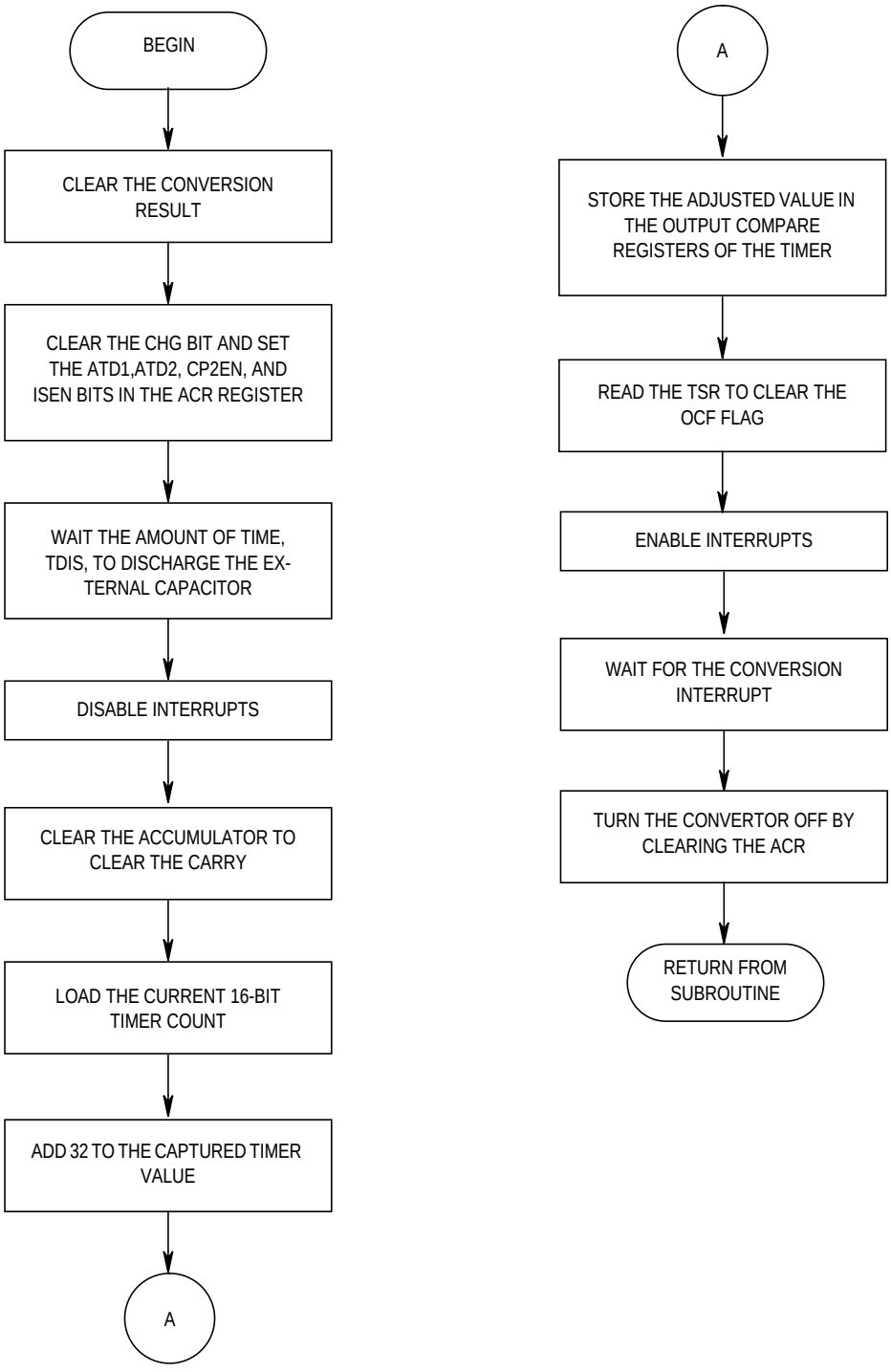

Figure 8. OCF/ICF Conversion Method Subroutine

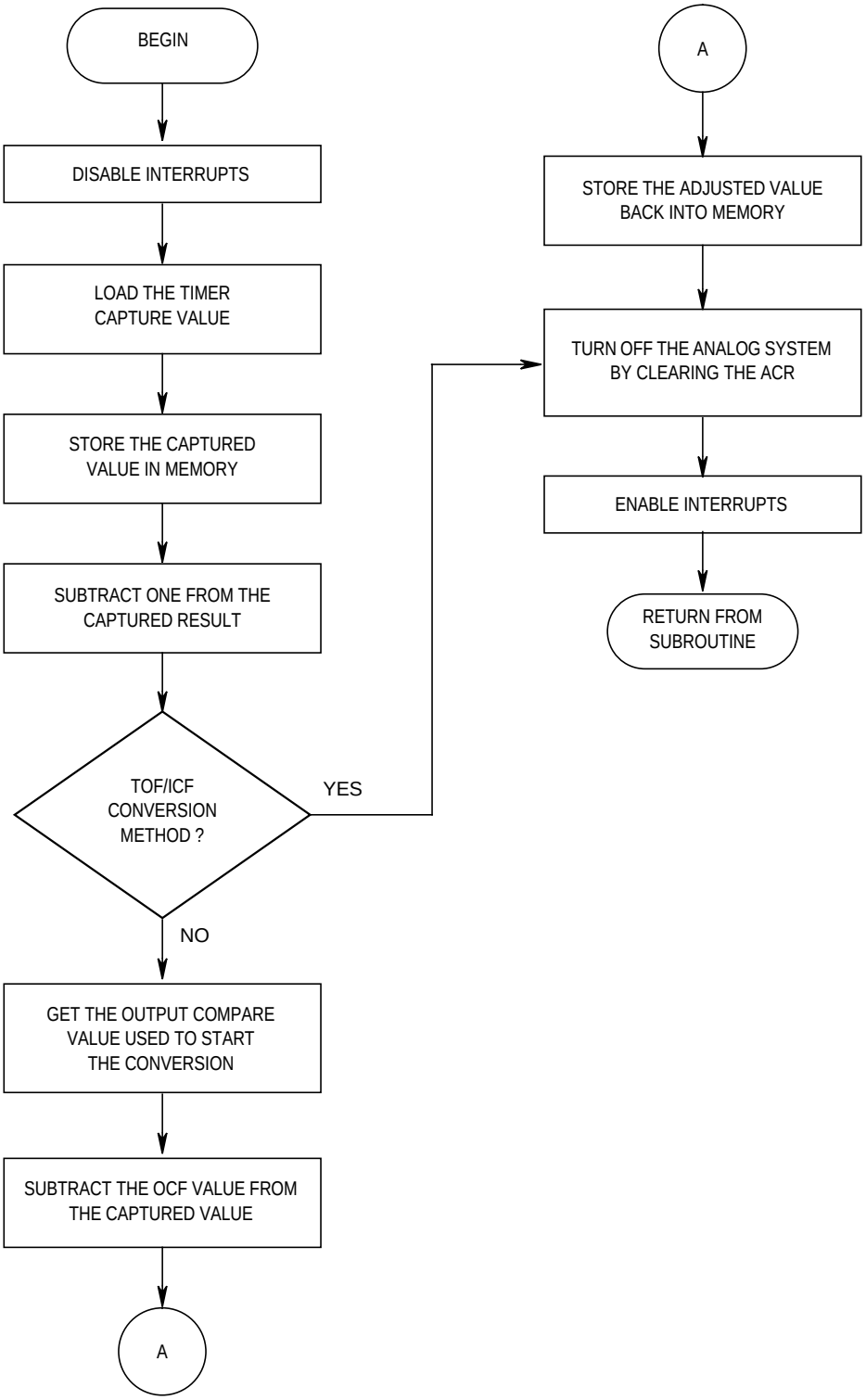

Figure 9. Conversion Interrupt Service Routine

Single-Slope Analog-to-Digital Conversion Code

```
*****************************************************************************
*
*       Filename: SSADCON.ASM
*       Revision: 1.00
*       Date: June 10,1996
*
*       Written By: Stephen Ledford
*                   Motorola CSIC Product Engineering
*
*       Assembled Under: Micro Dialects, Inc. MicroASM-6805
*
*                    *******************************
*                    *       Revision History      *
*                    *******************************
*
*       Revison 1.00    6/10/96 Original release
*
*****************************************************************************
*
*       Program Description:
*
*       This program performs conversions with the single-slope A/D
*       convertor system on the 68HC705JP7.
*
*****************************************************************************
*
*       Registers
*
AMUX    EQU     $03             ;Ananlog MUX control register
TCR     EQU     $12             ;Timer control register
TSR     EQU     $13             ;Timer status register
ICRH    EQU     $14             :Input capture high byte buffer
ICRL    EQU     $15             ;Input capture low byte buffer
TMRH    EQU     $18             ;Timer count upper byte register (read)
TMRL    EQU     $19             ;Timer count low byte register (read)
OCRH    EQU     $18             ;Output compare register high byte (write)
OCRL    EQU     $19             ;Output compare register low byte (write)
ACR     EQU     $1D             ;Analog System control resgister
ASR     EQU     $1E             ;Analog System status register
*
```

```
*       Start of main routine
*
        ORG     $50             ;RAM address space
*
REF     RMB     2               ;reference voltage conversion value
VAL     RMB     2               ;converted value storage location
TEMP    RMB     2               ;temporary storage space for processing
OCFVAL  RMB     2               ;output compare value used for OCF/ICF method
*
        ORG     $700            ;start of ROM/EPROM address space
*
*******************************************************************************
*
*       Main program routine
*
*       This is an example of a main program that uses the OCF/ICF method to perform
*       conversions and manage the analog subsystem.
*
*******************************************************************************
MAIN    EQU     *
*
*       First set up to convert the reference voltage which in this case is the
*       internal connection of VDD to the comparators.
*
        LDA     #$50            ;Set the DHOLD and VREF bits, clearing the MUX bits
        STA     AMUX            ;Store to the AMUX register
REFWAIT LDA     #$80            ;Load a loop counter to wait long enough for the sample
                                ;cap to settle to the reference voltage
        DECA                    ;Decrement the loop counter
        BNE     REFWAIT         ;Loop back until the count is complete
        BCLR    6,AMUX          ;Clear the sample and hold bit to avoid changes in
                                ;samples value
        BCLR    4,AMUX          ;Remove the VDD as the input source to the comparator
*
*       Now perform the conversion
*
        JSR     OCFCON          ;Jump to subroutine to perform an OCF/ICF conversion
        LDA     VAL             ;Get the high byte of the converted result
        STA     REF             ;Store it in the REFERENCE variable
        LDA     VAL+1           ;Get the low byte of the converted result
        STA     REF+1           ;Store it in the REFERENCE variable
*
```

```
*       Now select a regular input channel and convert it
*
        LDA     #$41            ;Set the DHOLD bit and select channel 1 as input
        STA     AMUX            ;Store to the AMUX register
SAMPLE  LDA     #$80            ;Load a loop counter to wait long enough for the sample
                                ;cap to settle to the reference voltage
        DECA                    ;Decrement the loop counter
        BNE     SAMPLE          ;Loop back until the count is complete
        BCLR    6,AMUX          ;Clear the sample and hold bit to avoid changes in
                                ;sampled value
        BCLR    0,AMUX          ;Deselect the input channel
*
        JSR     OCFCON          ;Jump to subroutine to perform an OCF/ICF conversion
*
*       Now there is a reference conversion and input channel conversion. Do the
*       ratio calculation.
*
        BRA     MAIN            ;Loop back to the top of the program
*
*******************************************************************************
*
*       Subroutine to perform a fully manual conversion by cycle counting.
*
*       This is a completely manual conversion method. The start of the
*       conversion is manual and the completion of the conversion is
*       manual. The conversion result is calculated by a software cycle
*       count loop. Since this is a cycle counting loop, this does not have
*       any other hardware dependencies, such as the timer, but gives the lowest
*       resolution for conversion as well as prevents other processing from
*       being performed.
*
*******************************************************************************
MANCON  EQU     *

*
*       Initialization
*
        CLR     VAL             ;Clear the conversion result upper byte
        CLR     VAL+1           ;Clear the conversion result lower byte
        LDA     #$05            ;Clear the CHG,ATD2, and ATD1 to configure for a manual
                                ;mode conversion. Set CP2EN and ISEN to start up system
        STA     ACR             ;Store the value in the ACR
*
```

```
*       Start the conversion sequence by first discharging the external cap
*
        LDA     #$FF            ;Load up a simple loop counter to wait time TDIS. This
                                ;value should be tuned based on system bus speed.
MANINI  DECA                    ;Decrement the loop counter
        BNE     MANINI          ;Loop back until the amount of wait time is complete
        BSET    5,ASR           ;Set the CPFR2 bit to make sure the flags are clear
*
*       Now begin the conversion by start of the charging on the ramp cap
*
        SEI                     ;Disable all interrupts as this would mess up count
        BSET    7,ACR           ;Set the CHG bit engaging the current source
MANCNT  INC     VAL+1           ;Increment the conversion value low byte
        BNE     MANHIGH         ;Branch around incrementing the high byte if not rolled
        INC     VAL             ;Increment the conversion value high byte if low rolled
MANHIGH BRCLR   7,ASR,MANCNT    ;Test the CPF2 flag and continue looping if not converted
        CLR     ACR             ;Clear the CHG bit, discharging the external cap and turn
                                ;the system off.
*
*       Conversion is complete. Return to main program
*
        CLI                     ;Enable interrupts again
        RTS                     ;Return to calling routine
*
*******************************************************************************
*
*       Subroutine to perform a partially manual conversion by cycle counting.
*
*       This is a partially manual conversion method. The start of the
*       conversion is manual and the completion of the conversion is
*       automatic. The conversion result is calculated by a software cycle
*       count loop. Since this is a cycle counting loop, this does not have
*       any other hardware dependencies, such as the timer, but gives the lowest
*       resolution for conversion as well as prevents other processing from
*       being performed.
*
*******************************************************************************
CYCCON  EQU     *
*
*       Initialization
*
        CLR     VAL             ;Clear the conversion result upper byte
        CLR     VAL+1           ;Clear the conversion result lower byte
        LDA     #$25            ;Clear the CHG,ATD2 and set ATD1 to configure for a semi-
manual
                                ;mode conversion. Set CP2EN and ISEN to start up system
        STA     ACR             ;Store the value in the ACR
*
```

```
*       Start the conversion sequence by first discharging the external cap
*
        LDA     #$FF            ;Load up a simple loop counter to wait time TDIS. This
                                ;value should be tuned based on system bus speed.
WAITINI DECA                    ;Decrement the loop counter
        BNE     WAITINI         ;Loop back until the amount of wait time is complete
        BSET    5,ASR           ;Set the CPFR2 bit to make sure the flags are clear
*
*       Now begin the conversion by start of the charging on the ramp cap
*
        SEI                     ;Disable all interrupts as this would mess up count
        BSET    7,ACR           ;Set the CHG bit engaging the current source
COUNT   INC     VAL+1           ;Increment the conversion value low byte
        BNE     NOHIGH          ;Branch around incrementing the high byte if not rolled
        INC     VAL             ;Increment the conversion value high byte if low rolled
NOHIGH  BRCLR   7,ASR,COUNT     ;Test the CPF2 flag and continue looping if not converted
                                ;No need to do a manual discharge as this was handled by
                                ;the convertor hardware.
        CLR     ACR             ;Turn the convertor off
*
*       Conversion is complete. Return to main program
*
        CLI                     ;Enable interrupts again
        RTS                     ;Return to calling routine
*
*******************************************************************************
*
*       Subroutine to perform a conversion by use of the TOF to ICF method.
*
*       This is only the setup routine for the TOF/ICF method. An ISR is also
*       required to get the result when the conversion is completed. This routine
*       requires the least amount of code intervention to perform and the best accuracy
*       possible, but the conversion can have the longest latency as the timer must
*       reach its full count before a conversion begins.
*
*******************************************************************************
TOFCON  EQU     *
*
*       Initialization
*
        CLR     VAL             ;Clear the conversion result upper byte
        CLR     VAL+1           ;Clear the conversion result lower byte
        LDA     #$4D            ;Clear the CHG,ATD1 and set ATD2 to configure for a
                                ;TOF/ICF
                                ;mode conversion. Set CP2EN and ISEN to start up system
                                ;as well as enabling the Analog interrupts by setting
CPIE
        STA     ACR             ;Store the value in the ACR
*
```

```
*       Start the conversion sequence by first discharging the external cap
*
        LDA     #$FF            ;Load up a simple loop counter to wait time TDIS. This
                                ;value should be tuned based on system bus speed.
TOFINI  DECA                    ;Decrement the loop counter
        BNE     TOFINI          ;Loop back until the amount of wait time is complete
        BSET    5,ASR           ;Set the CPFR2 bit to make sure the flags are clear
        LDA     TSR             ;Read the timer status register to clear the TOF
        LDA     TMRL            ;Read the low byte of the timer value to complete the
                                ;clear
*
*       The mode is configured to start at the TOF of the timer. Therefore wait
*       until an interrupt occurs and the timer value should be captured.
*
        WAIT                    ;Wait here until the analog interrupt occurs
                                ;Other processing can occur if necessary until the
                                ;interrupt occurs
        CLR     ACR             ;Turn the convertor off so as not to cause another
                                ;conversion and overrun of the result
*
*       Conversion is complete. Return to main program
*
        RTS                     ;Return to calling routine
*
*******************************************************************************
*
*       Subroutine to perform a conversion by use of the OCF to ICF method.
*
*       This is just the setup routine for the OCF/ICF method. An ISR is also
*       required to get the result when the conversion is completed. This routine
*       has a little more code required to support it, but has the lowest possible
*       conversion time and the best accuracy possible.
*
*******************************************************************************
OCFCON  EQU     *
*
*       Initialization
*
        CLR     VAL             ;Clear the conversion result upper byte
        CLR     VAL+1           ;Clear the conversion result lower byte
        LDA     #$6D            ;Clear the CHG and set ATD1,ATD2 to configure for a
OCF/ICF
                                ;mode conversion. Set CP2EN and ISEN to start up system
                                ;as well as enabling the Analog interrupts by setting
CPIE
        STA     ACR             ;Store the value in the ACR
*
```

```
*       Start the conversion sequence by first discharging the external cap
*
        LDA     #$FF            ;Load up a simple loop counter to wait time TDIS. This
                                ;value should be tuned based on system bus speed.
OCFINI  DECA                    ;Decrement the loop counter
        BNE     OCFINI          ;Loop back until the amount of wait time is complete
        BSET    5,ASR           ;Set the CPFR2 bit to make sure the flags are clear
*
*       The mode is configured to start at the OCF of the timer. Therefore wait
*       until an interrupt occurs and the timer value should be captured. Start
*       by configuring the output compare value for an OCF to occur.
*
        SEI                     ;Set the interrupt bit to prevent additional interrupts
        CLRA                    ;Clear the accumulator to clear the carry bit
        LDA     TMRH            ;Get the current value of the timer high byte
                                ;This is necessary to freeze the timer value
        LDA     TMRL            ;Load the current timer count low byte
        ADD     #$20            ;Add an offset to start the conversion sequence
        STA     OCFVAL+1        ;Store away for use later
        LDA     TMRH            ;Get the current value of the timer high byte
        ADD     #0              ;Add the carry if there was one
        STA     OCFVAL          ;Store the value for use later
        STA     OCRH            :Store in the output compare to start the count
        LDA     TSR             ;Read the TSR to allow clearing of the OCF
        LDA     OCFVAL+1        ;Get back the low byte value
        STA     OCRL            ;Store in the lower byte of the output compare register
*
*       Wait for the conversion to complete
*
        CLI
        WAIT                    ;Wait here until the analog interrupt occurs
                                ;Other processing can occur if necessary until the
                                ;interrupt occurs
        CLR     ACR             ;Turn the convertor off so as not to cause another
                                ;conversion and overrun of the result
*
*       Conversion is complete. Return to main program
*
        RTS                     ;Return to calling routine
*
*******************************************************************************
*
*       ISR for control of the conversion methods both TOF/ICF and OCF/ICF.
*
*       Once the ISR is complete, the program control goes back to the calling
*       subroutine to complete execution of the conversion. The conversion result
*       placed in the variable CONV which is a 16-bit value.
*
*******************************************************************************
```

```
CONISR EQU    *
       SEI                  ;Set the interrupt bit to avoid other IRQs
*
*      Get the result from the timer registers and place it in the variable CONV
*
       LDA    ICRH          ;Get the high byte of the timer counter
       STA    VAL           ;Store in the conversion result variable
       LDA    ICRL          ;Get the low byte of the timer counter
       SUB    #1            ;Must subtract 1 from the result since the timer
                            ;adds an extra tick to the counter
       STA    VAL+1         ;Store in the conversion result variable
       LDA    VAL           ;Get back the high byte of the converstion result
       SBC    #0            ;Subtract just with the carry to finish the
                            ;processing of the result
       STA    VAL           ;Store back the processed result
*
*      Check to see which kind of conversion is in process
*
       BRCLR  5,ACR,TOFISR  ;If the ATD1 bit is clear then this is a TOF/ICF
                            ;else this is a OCF/ICF conversion method.
*
*      Handle the OCF/ICF conversion result. Must subtract the OCF value to
*      to find the delta time between the start and stop of the counter
*
ICFISR LDA    OCFVAL+1      ;Load the value used to start the conversion
       SUB    VAL+1         ;Subtract from the captured timer value
       STA    VAL+1         ;Store back as the adjusted result
       LDA    OCFVAL        :Load the high byte of the start value
       SBC    VAL           ;Subtract with the carry from the captured value
       STA    VAL           ;Store back in the result register high byte
*
*      Handle the TOF/ICF conversion result. No post processing is required
*      since the start is from a counter of $0000 to the captured count.
*
TOFISR CLR    ACR           ;Power down the Analog system so as not to get any
                            ;errant conversions
*
*      Return from interrupt
*
       CLI                  ;Clear the interrupt bit to allow IRQs
       RTI                  ;Return from interrupt
*
*******************************************************************************
*
*      Vectors
*
*******************************************************************************
       ORG    $1FF2         ;Location of Analog system vector
ANALOG FCB    CONISR        ;Analog conversion ISR pointer
       ORG    $1FFE         ;Location of Reset vector
RST    FCB    MAIN          ;Reset vector pointer
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com