AVR401: 8-Bit Precision A/D Converter

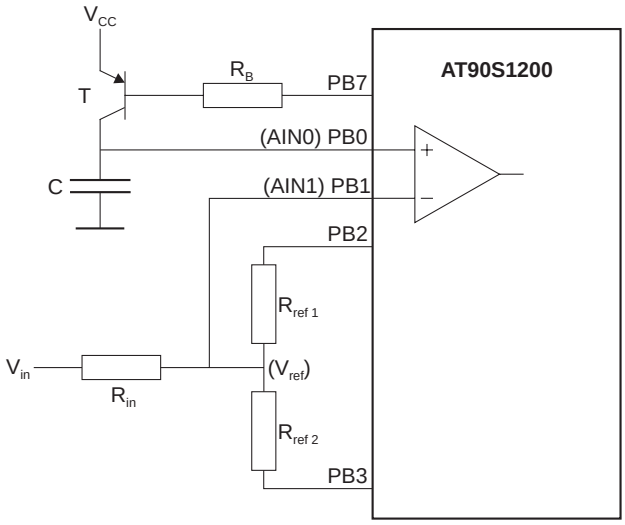
Features

- **Very Low Cost**
- **High Precision**
- **Auto-calibration Eliminates Component Inaccuracy**
- **Measures Voltages for 0 to V_{CC}**
- **Max Conversion Time: 1.1 ms**

Introduction

This application note describes how to perform a kind of dual slope A/D conversion with an AVR microcontroller. The converter is very low cost, requiring only six discrete components in addition to the AVR. Five of the controller pins are used (see Figure 1). This example is based on the AT90S1200 device, but any AVR device with a comparator can be used.

Figure 1. A/D Converter



Theory of Operation

The capacitor is charged with a constant current supplied by the transistor. The capacitor voltage will rise linearly. To discharge the capacitor, the AIN1-pin is set to output with a ‘0’ applied. A reference voltage at $V_{CC}/2$ is supplied by the resistor network R_{ref1} and R_{ref2}. When the PB1 and PB2-pins are configured as inputs, the reference is turned off, and the voltage level at the AIN1-pin will be the input voltage V_{in}. By setting the pins as outputs and applying a ‘0’ and a ‘1’, the level at the AIN1-pin will be $V_{CC}/2$ (if the resistors are of equal size). The input resistor R_{in} have to be at least 100 times higher than the reference resistors R_{ref1} and R_{ref2} to avoid measurement errors.

8-Bit Microcontroller

Application Note

0953A-A–9/97

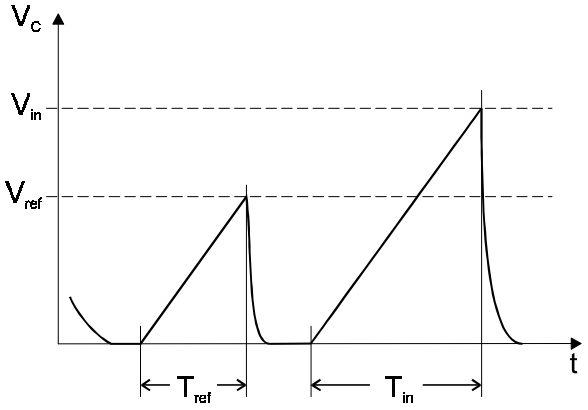
The algorithm used for the conversion is as follows:

1. Turn on the reference
2. Charge the capacitor until the reference voltage is reached. Measure the time needed for this, T_{ref}.
3. Turn off the reference and discharge the capacitor.
4. Charge the capacitor until the input voltage is reached. Measure the time needed for this, T_{in}.

The conversion cycle is shown in Figure 2.

The time measurement is performed by the Timer/Counter, which is expanded to nine bits by using the Timer/Counter Overflow interrupt.

Figure 2. Conversion cycle



Calculation

Suppose that V_{CC} is 5 volts. The relationship between the input voltage and the reference voltage is given by:

Equation 1 $$V_{in} = \frac{V_{ref} \times T_{in}}{T_{ref}}$$

The ideal output from the conversion is an 8-bit number, where 0 volts corresponds to zero and 5 volts is 255. The reference voltage $V_{CC}/2$ thus corresponds to 128. The equation can be re-written as:

Equation 2 $$V_{in} = \frac{T_{in} \times 128}{T_{ref}}$$

However, with inaccuracy in the reference resistors, the reference voltage may vary slightly. To compensate for this, a calibration can be performed by applying a known voltage at the input, and compare this to the reference. If the applied calibration voltage is exactly 2.5 volts, the reference voltage can be found by the equation

Equation 3 $$V_{ref} = \frac{T_{ref} \times V_{cal}}{T_{cal}} = \frac{T_{ref} \times 128}{T_{cal}}$$

The calibration cycle is executed by holding the PB7-pin low during power-up. The calibration voltage is then applied, and the PB7-pin is released. This starts calibration, and once performed, the value of the reference voltage is stored in EEPROM. During normal operation, the reference value is read from EEPROM, and the input voltage is calculated using Equation 1.

Configuration Example

As the resulting output is to be 8 bits, the timer should be of at least 9 bits to maintain the resolution. The components should be chosen so that the nominal time charging the capacitor up to V_{CC} is about 256 timer steps. In that way, inaccuracy in the component values and temperature changes are allowed, without causing the charging time to be longer than the maximum timer period, or too short, giving lower resolution. To achieve sufficient accuracy, a prescaler factor of 8 or higher should be used. The AT90S1200 Timer/Counter0 is of only 8 bits, so the 9th bit must be handled in software. The following example illustrates how the component values can be found.

First, decide which crystal frequency to operate at. With a 4 MHz crystal, the clock period is 250 ns. By setting the prescaler to CK/8, the timer is incremented every 2 ms. The maximum timer period with 9 bits is 512×2 ms = 1,024 ms. From this, we set $2 \times T_{ref}$ to 512 ms.

The charging of a capacitor whit a constant current is described by the equation:

Equation 4 $$\Delta V = \frac{I}{C} \times \Delta t$$

We can find the required current when the capacitor size, the time and the voltage difference is known:

Equation 5 $$I = \frac{\Delta V \times C}{\Delta t}$$

The capacitor will be charged up to V_{CC} = 5 V, and with a 0.22 mF capacitor, the transistor must supply a current of 2.15 mA. The R_B value is dependent upon the transistor's h_{FE}. For a BC558A pnp transistor, h_{FE} is in the range 125 to 250. This makes this transistor ideal for use, since any h_{FE} value in the specified range can be used. To make sure the full range in h_{FE} can be used, the average value, 188, is used in the calculations. The resulting base current is 11.4 µA.

The transistor is turned on by applying a '0' on the corresponding pin. At this current values, the transistor base-emitter voltage is about ÷0.1 V. The base resistor is found to be

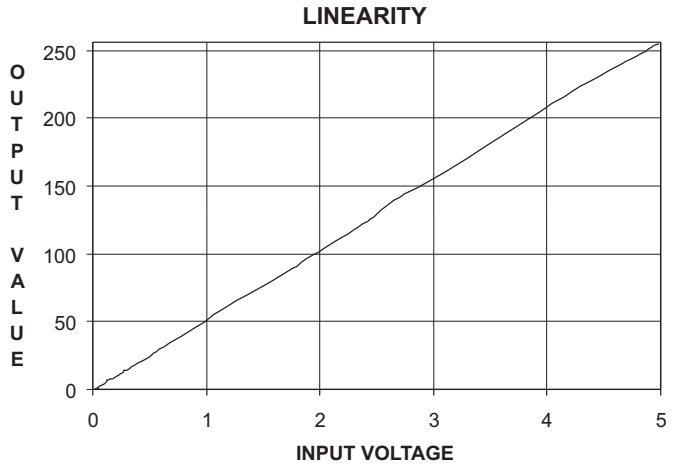
Equation 6 $$R_B = \frac{V_{CC} + V_{BE}}{I_B} = \frac{4.9V}{11.4\mu A} = 430k\Omega$$

The reference voltage is generated by the divider network R_{ref1} and R_{ref2}. The R_{in} has to be much larger than these two, so that the input voltage will not influence with the reference voltage. 100 kΩ for R_{in} and 1 kΩ for each of R_{ref1} and R_{ref2} is suitable.

The transistor should be connected to a pin as long away from the comparator inputs as possible. When a pin is switched, a noise spike appear at the adjacent pins. This will cause problems when measuring low voltages, as the noise spike might trigger the comparator before the capacitor voltage has reached the measured voltage.

Figure 3 shows measured linearity for a 4 MHz clocked application using the component values calculated in the above example.

Figure 3. Measured Linearity



Implementation

The software consist of several subroutines. The routines “reference” and “convert” handles the charging and timing. After they complete execution, the main program must perform the calculation needed. This is done using two subroutines that performs division and multiplication, «div17u» and «mul9». There are also two delays used by the other routines and the main program. They are used to discharge the capacitor completely and to generate a delay between each conversion.

“reference” Subroutine - Measures the Reference Voltage

The routine discharges the capacitor, turns on the transistor and charges the capacitor until the capacitor voltage is equal to the reference voltage. The time from the beginning of the charging and until the voltages are equal is measured. The capacitor is then discharged again. The charging time is used together with the charging time from the “convert” routine to calculate the input voltage.

This routine does not have to be called every time a conversion is performed, depending on variations in ambient temperature. Especially the parameter h_{FE} in the transistor is quite temperature dependent, so if the ambient temperature is varying, the subroutine will have to be executed frequently. In the example program, the «reference» routine is called each time a conversion is performed.

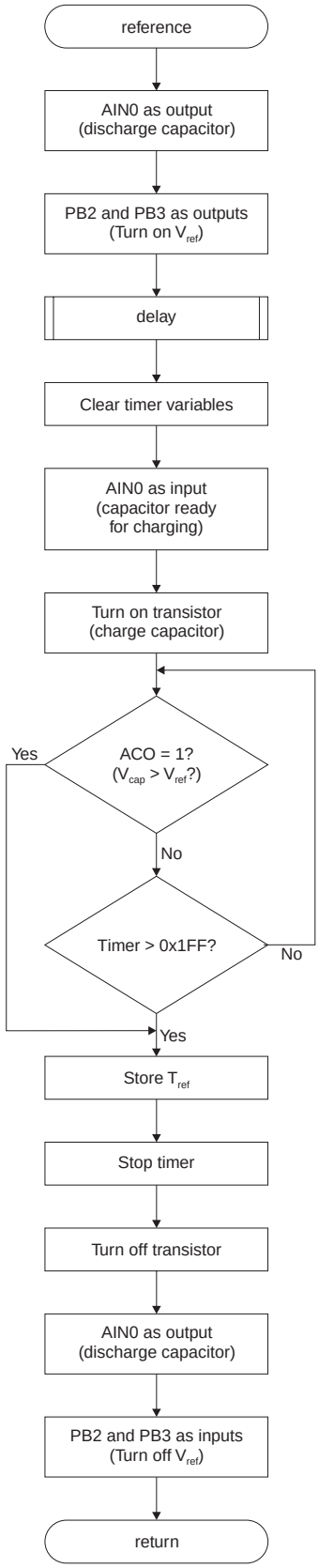
Table 1. «reference» Subroutine Performance Figures

Parameter	Value
Code Size	24 words
Execution cycles	Depends on the reference voltage.
Register Usage	Low registers :None High registers :2 Global :1

Table 2. «reference» Register Usage

Register	Input	Internal	Output
R17			“Tref” - Holds the time to reach the reference voltage.
R18		«TH» - High part of timer.	
R20		«temp»	

Figure 4. Flow Chart for “reference”



“input” Subroutine - Measures the Input Voltage

The routine turns on the transistor and charges the capacitor until the capacitor voltage is equal to the input voltage. Then capacitor is then discharged. The time needed to do this is measured and stored in T_{in}.

There should be a few microseconds delay between two conversion cycles, to ensure that the capacitor is completely discharged. In the example program, this is done by calling a delay routine.

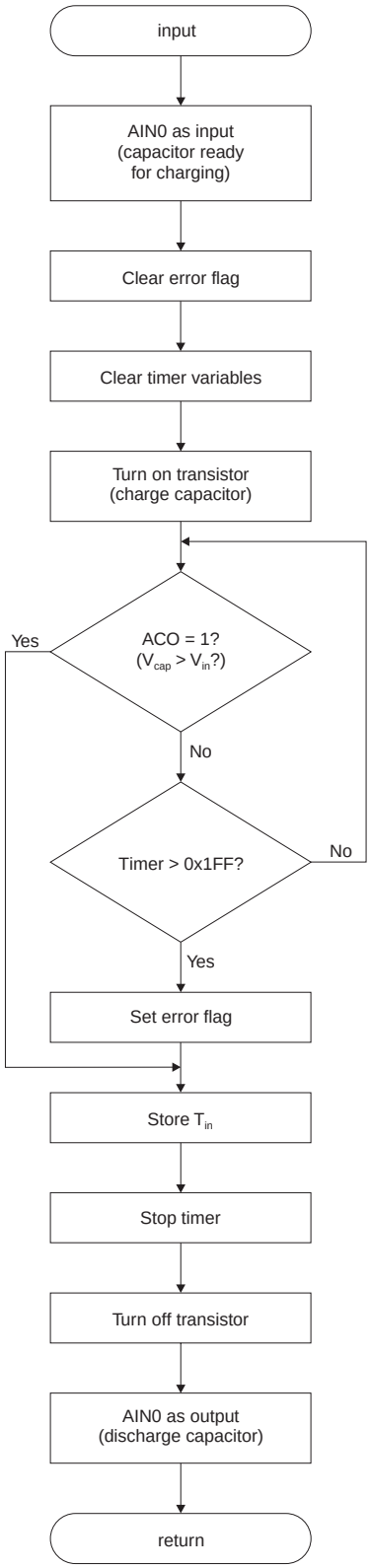
Table 3. «input» Subroutine Performance Figures

Parameter	Value
Code Size	19 words
Execution cycles	Depends on the input voltage
Register Usage	Low registers :2 High registers :None Global :1

Table 4. «input» Register Usage

Register	Input	Internal	Output
R14			“TinH” - High part of the input voltage charge time.
R15			“TinL” - Low part of the input voltage charge time.
R20		«temp»	

Figure 5. Flow Chart for “input”



“T0_int” Interrupt Service Routine

The only function for this routine is to increment the TH variable, so a 16-bit timer is created. Only 9 bits are used.

Table 5. “T0_int” interrupt Performance Figures

Parameter	Value
Code Size	2 words
Execution cycles	9 - including the reti instruction
Register Usage	Low registers :None High registers :None Global :1

«mpy9u» 9×8 Bit Multiplication

This routine performs a 9×8 bit multiplication. The 9-bit multiplier must be stored in the carry-flag (MSB) and the «mp9u» register. The multiplicand is stored in the “mc9u” register. The answer is placed in «C:m9uH:m9uL». The registers used for the result are the same as those used for the input to the division routine. The routine is based on the «mpy8u» multiplication routine described in application note AVR 200.

Table 6. «mpy9u» Subroutine Performance Figures

Parameter	Value
Code Size	11 words
Execution cycles	83
Register Usage	Low registers :3 High registers :None Global :1 Flags :C

Table 7. «mpy9u» Register Usage

Register	Input	Internal	Output
R0	«mc9u» - Multiplicand		
R1	«mp9u» - Multiplier		m9uL - Result low byte
R2			m9uH - Result high byte
C-flag	Multiplier, 9th bit		Result, 17th bit
R20		«temp» - Used as loop counter	

«div17u» 17/16 Bit Division

This routine performs a 17/16 bit division. The 17-bit dividend must be stored in the (C:didH:didL) variable, where the carry-flag is most significant. The divisor is stored in the (divH:divL) variable. The result is placed in (resH:resL) and the reminder in (remH:remL). The routine is based on the

«div16u» multiplication routine described in application note AVR 200.

Table 8. «div17u» Subroutine Performance Figures

Parameter	Value
Code Size	18 words
Execution cycles	209 min, 292 max.
Register Usage	Low registers :6 High registers None Global :1 Flags :C

Table 9. «div17u» Register Usage

Register	Input	Internal	Output
R1	«didL» - Low part dividend		«dresL» - Low part result
R2	«didH» - High part dividend		«dresH» - High part result
C-flag	17th bit of dividend		
R3	«divL» - Low part divisor		
R4	«divH» - High part divisor		
R5			«remL» - Low part reminder
R6			«remH» - High part reminder

Example Program

The included example program performs repeated conversions. First, the charging time for the reference is measured, then for the input voltage. The result is output to Port D and Port B pin 7 (MSB). The result is inverted before it is output, so active low LEDs can be connected to show the result. This conversion cycle is repeated in an endless loop.

If the PB4-pin is grounded at power-up, a calibration is performed. The user should apply 2.5 volts at the input before releasing the PB4-pin. The calibrated V_{ref} is stored in EEPROM, where it is fetched at every normal power-up.

Performance Figures

Table 10. Overall Performance Figures

Parameter	Value
Code Size	43 words - Conversion routines only (not mpy9u and div17u) 147 words - Complete application note
Register Usage	Low Registers :9 High registers :5 Pointers :None
Interrupt Usage	Timer/Counter 0 Interrupt
Peripheral Usage	Timer/Counter0 Analog Comparator Port B, pin 0 to 3 and pin 7 Port D, all pins (example program only) Port B, pin 4 (example program only)

The calibration routine can be skipped if only relative values are measured. The reference voltage is then assumed to be «128», which will also make the calculations easier.

The reference network can be substituted with a voltage reference to achieve even better accuracy. It is then possible to measure variations in V_{CC} by connecting it via a voltage divider network to the input.