AVR400: Low Cost A/D Converter

Features

- **Interrupt driven**
- **Code size: 23 Words**
- **Low use of external components**
- **Resolution: 6 bits**
- **Measurement range: 0 - 2V**
- **Runs on any AVR device with 8-bit timer/counter and analog comparator**

Introduction

This application note targets cost and space critical applications that need an ADC. It describes how to make a simple ADC with only two external components, a resistor and a capacitor. The design enables a very compact and inexpensive application.

Theory of Operation

Nearly all AVR microcontrollers feature an analog comparator which makes it easy to implement an ADC. The signal to be measured is connected to the inverted input, and a reference signal is connected to the non-inverting input. The reference signal is generated by charging a capacitor through a resistor. When the capacitor is being charged, the voltage across it will follow an exponential curve. If the voltage range to be measured is limited to $2/5*V_{CC}$, the exponential curve is a good approximation to a straight line. The voltage of the applied signal, UIN is found by measuring the time it takes for the voltage across the capacitor to rise above the applied voltage. By using one pin on port B to control the charging and discharging of the capacitor, only three port pins are used. A schematic diagram is found in Figure 1.

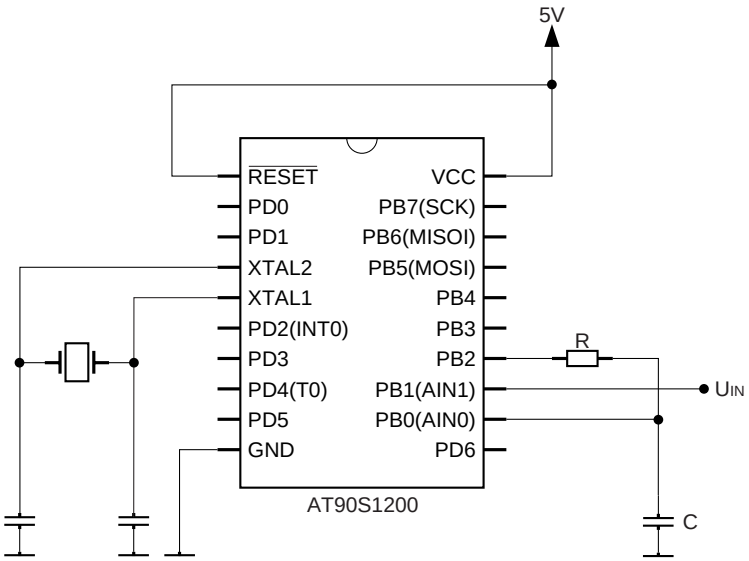

Figure 1. Circuit Diagram

8-Bit AVR Microcontroller

Application Note

0942A-A–8/97

The time constant of the R/C network must be tuned so that the following equation is satisfied:

$$\frac{512}{f} = -RC\ln\left(1-\frac{2}{5}\right) \Rightarrow \underline{RC = \frac{1002}{f}}$$

Component values for some typical oscillator frequencies are shown in Table 1 . If the time constant varies from this, it will cause errors in the result. This makes it necessary to use components with high accuracy in the RC-network. The voltage curve for the capacitor is shown together with a straight line in fig. 2. As the supply voltage is used as a reference, it must be stable during the conversion.

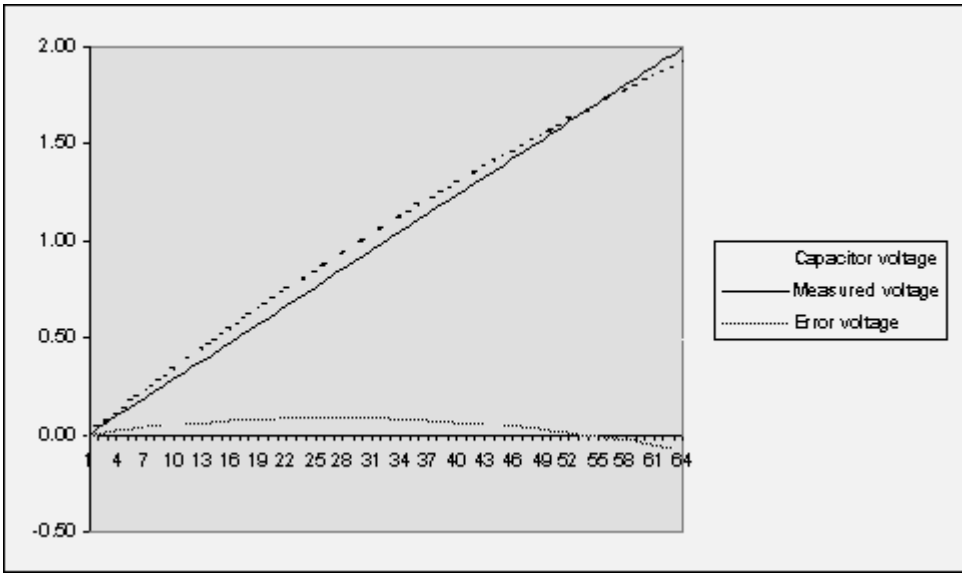

Figure 2. A/D converter linearity

Table 1. R/C Network Coomponent Values

XTAL (MHz)	1	2	4	6	8	10	12	14	16
R(kΩ)	100	33	30	30	27	100	56	47	160
C(nF)	10	15	8.2	5.6	4.7	1	1.5	1.5	0.39

To ensure proper operation, the capacitor must be discharged for at least 200µs between conversions. If the capacitor is not properly discharged, it will not be possible to measure low values. If the voltage input is larger than $2/5V_{CC}$, the converter will return the maximum value. This is accomplished by loading an offset value into the Timer/Counter0 register before conversion starts. The timer will give an overflow interrupt after 512 cycles (64*8). This is the time it takes for the voltage across the capacitor to reach $2/5V_{CC}$. If the voltage is within the operating range, an Analog Comparator interrupt will occur. Offset is subtracted from the measured value.

Implementation

The ADC uses Timer/Counter0 and the analog comparator interrupts. This frees the MCU resources during conversion.

Subroutine “convert_init” - ADC Initialization

This subroutine is called to initialize the ADC. This must be done before the ADC is used. By calling this subroutine, the comparator and timer interrupts are enabled, and the control pin is set as output. Then the 'SEI' instruction, which enables global interrupts, should be called to enable the A/D converter. By calling 'CLI', the A/D converter is disabled.

Table 2. “convert_init” Subroutine Performance Figures

Parameter	Value
Code size	6 words
Execution cycles	10, including the RET instruction
Register usage	• Low registers :None • High registers :1 • Pointers :None

Table 3. “convert_init” Register usage

Register	Input	Internal	Output
R16		"result" - Scratch register	

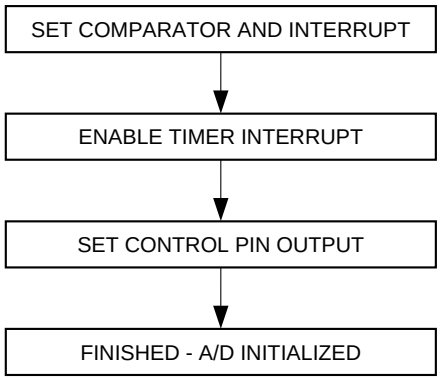

Figure 3. “convert_init” Flow Chart

“AD_convert” subroutine - start an A/D conversion

This routine is used to start an A/D conversion. It pre-loads the counter with 256-64 and starts counting up at the frequency XTAL/8. The conversion complete flag (the T-flag in the status register) is cleared, and the charging of the capacitor is started.

Table 4. “AD_convert” Subroutine Performance Figures

Parameter	Value
Code Size	7 words
Execution Cycle	10 (including RET)
Register Usage	• Low registers :None • High registers :1 • Pointers :None • Status flags :1

Table 5. “AD_convert” Register usage

Register	Input	Internal	Output
R16		"result" - Scratch register	
SREG			T-flag - This flag is used to indicate that a conversion is in progress.

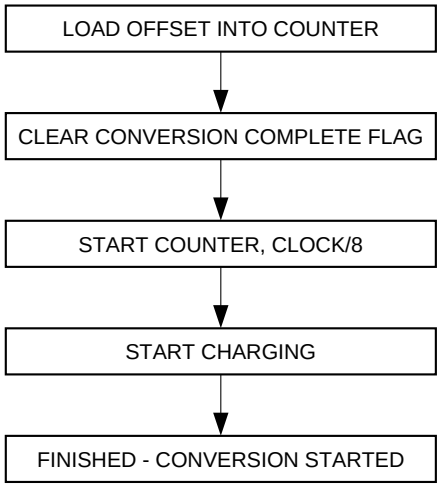

Figure 4. ADC Conversion Flow Chart

"ANA_COMP" Interrupt Handling Routine

This routine is executed when a conversion is complete. It loads the Timer/Counter0 value, stops the timer and sets the conversion complete flag (T-flag in SREG). The offset is then subtracted from the timer value. It is necessary to subtract 1 more than the offset, because the interrupt handling takes a minimum of 7 cycles.

Table 6. "ANA_COMP" Subroutine Performance Figures

Parameter	Value
Code Size	7 words
Execution Cycle	11 (including RETI)
Register Usage	• Low registers :None • High registers :2 • Pointers :None • Status flags :1
Interrupt Usage	Timer/Counter0 and Analog Comparator Interrupt

Table 7. "ANA_COMP" Register usage

Register	Input	Internal	Output
R16		"result" - Stores the timer value	"result" - Contains the result from the A/D conversion
R17		"temp" - Scratch register	
SREG			T-Flag - This flag is used to indicate that the conversion is finished

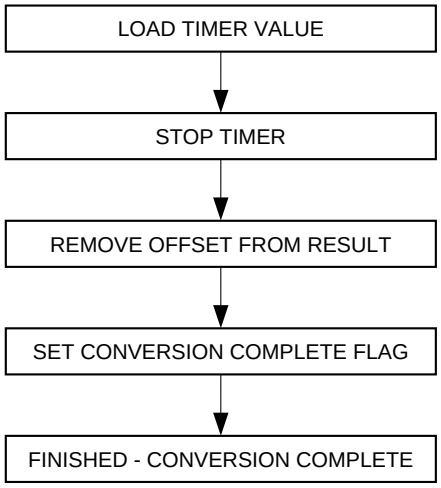

Figure 5. "ANA_COMP" Flowchart

Example Program

The example program that is included in this application note performs successive conversions, and presents the data as binary values on port B.

Table 8. Overall Performance Figures

Parameter	Value
Code Size	23 words - A/D converter routines only 37 words - complete with test program
Register Usage	• Low registers :None • High registers :2 • Pointers :None • Status flags :1
Interrupt Usage	Timer/Counter0 overflow interrupt Analog comparator interrupt
Peripheral Usage	Timer/Counter0 Analog Comparator (port B pin0 and pin1) Port B pin2 Port D (example program only)