AVR Freaks

DESIGN NOTE #021

AUTHOR: AVRFREAKS

KEYWORDS: ADC, ANALOG TO DIGITAL CONVERTER

Using the Built-in ADC in AVR

Introduction

Many AVR parts have a built-in Analog to Digital Converter (ADC). The ADC has a resolution of 10 bits, and, depending on which part is used, there are 6 to 8 channels available. Sampling rates up to 15.4 ksps, using full 10-bit resolution, are selectable.

This document provides an overview of the features and the use of the ADC in AVR circuits, including a code example for standard employment.

Overview

To not entirely copy the data sheet but rather give a more practical angle how to use the ADC, this document will focus on selected topics that are most often considered when using the built-in ADC of AVR parts. Not all available features are described, since some are not common to all parts, and also partly because some features are not quite so commonly utilized (this includes the noise canceling function).

The following topics have been considered to be most relevant when wanting to use an ADC:

- How to connect the pins related to the ADC (analog references).
- How to make an Analog to Digital conversion.
- How to use interrupt controlled sampling.
- How to verify conversion complete (polling the ADC).
- How to specify resolution/conversion speed.
- How to select which channels to use..

To make it easier to follow the subsequent descriptions of how to use the ADC, it is recommended to read this document with the block schematic of the ADC from the data sheet at hand. The example in this document relates to AT90S8535 but other parts are similar.

Analog References/Pin Connections

To be able to make the ADC work, the analog references must be connected. The analog references are providing the ADC with the boundaries of the voltage range that it should work within. The analog references are applied to specific pins on the part: AGND and AREF. The choice of analog references must be made within certain limits. The ADC in AVR parts can measure voltage levels between AGND and AREF. AGND should be connected to GND, and AREF should be connected to a stable analog voltage reference, which must be less or equal to AVCC (which should not vary more than 0.3V from V_{CC}).

Like most other modules in the AVR using I/O pins, the ADC uses pins which are alternatively used for general I/O when not used by the ADC. If the ADC is not used, the analog references should be connected to GND to ensure that they are not floating. An exception is AVCC, which should be connected to V_{CC} whenever PORTA is used as output. This is because PORTA is always is sourced by AVCC.

A Simple Analog to Digital Conversion

Before starting a conversion the ADC must be enabled. This is done by setting the ADEN bit in the ADCSR Register. The ADC can be run in two modes: Single Conversion mode or Free-running mode.

In Single Conversion mode the conversion is initiated manually, by setting the ADSC bit in the ADCSR Register.

In Free-Running mode the ADC is set up to start a new conversion immediately after the previous conversion is complete. To enable this feature, the ADFR bit in the ADCSR Register must be set. When this bit is set, the ADC will start a new conversion immediately when the old conversion has finished.

To verify that a conversion is complete, the ADIF bit in ADCSR can be inspected. When using Free-running mode, it is an advantage to let an interrupt routine handle the removal of the conversion result instead of polling the ADIF bit. Once the conversion is complete, the result is located in the ADCH/ADCL Registers. It is important to read ADCL first to ensure that valid data is read. By reading ADCL first, the ADCL and the ADCH Registers are “locked” and cannot be updated by the ADC until the ADCH has been read.

From the conversion is initiated to the result is in the ADCH/L 13/14 ADC cycles will pass. Extra time is used for the first conversion (see data sheet).

Conversion Resolution/Speed

Resolution and conversion speed is always important when using an ADC. It should be considered that the relation between ADC resolution and conversion speed are related. If a fast sampling rate is required, it could be necessary to use only 8-bit resolution or less, and if full 10-bit resolution is desired, “slower” sampling could be enforced (the example below also relates to this question).

Starting with the sampling rate: This is dependant/related to the AVR speed since the ADC clock is generated by prescaling the AVR clock. The prescaling factor is determined by bits 0 - 2 in the ADCSR Register, which provides the possible prescaling factors of 2, 4, 8, 16, 32, 64, and 128. When considering the sampling rate one should also consider that one conversion takes 13/14 ADC cycles. If running the AVR on 8 MHz one wants to sample at around 10 kSPS, a prescaling factor of 64 would be appropriate (making the sampling rate 9.6 kSPS).

Using Different Channels

As seen from the block schematics in the data sheet, the ADC can access multiple analog channels through the MUX. The ADMUX Register controls the MUX and the different input pins can be directed to the sample-hold circuit. The sample-hold circuit keeps the sampled voltage level stable while the conversion is made, using successive approximation. If the ADMUX Register is changed during a conversion, the channel will not be changed until a new conversion is started. This can be utilized with interrupt in Free-running. By changing the MUX channel when a conversion complete interrupt occurs it is possible to scan through the input channels.

Note that all channels to the ADC share one single ADC and that the sampling rate for each channel decreases for additional channels (the example below also relates to this question).

Example Using the ADC

Case: You are running an AT90S8535 at 8 MHz and require AD conversion in five channels at approximately 10 ksps per channel. How can the ADC be set up and used to support this operation?

With 10 kSPS the ADC needs approximately 130 kHz ADC clock (13 ADC clock cycles per conversion). For five channels the ADC have to work five times faster than with a single channel. The closest possible value to 650 kHz is 500 kHz which is obtained with a prescaling factor at 16.

An ADC clock at 500 kHz violates the limit in ADC clocking for full 10 bit resolution (ADC clock 200 kHz for 10-bit resolution). Considering the table regarding “ADC characteristics” section “Analog to Digital Converter” in the data sheet, the absolute accuracy would probably be 2 - 3 LSB. This means that the ADC can be considered to be a 9-bit ADC since the LSB is not reliable.

Using an interrupt is a practical way of collecting data and update the channel that are used.

```
/* Compiled using IAR embedded workbench ver. 1.51b */
#include "io8535.h"
#include "ina90.h"
#define FALSE 0
#define TRUE 1

/* Global variable */
unsigned char currentChannel = 0;
unsigned int newSample;
unsigned char newDateReady = FALSE;

/* Interrupt service rutines */
interrupt [ADC_vect] void ISR_ADC(void){
newSample = ADC;
  if(currentChannel<4)//Cycle trough the channels 0-4
    currentChannel ++;
  else
newSampleReady = TRUE;
    currentChannel = 0;
  ADMUX = currentChannel;//Update the ADMUX register
}

/* Sub-rutine */
void process_Data(void){
  newSampleReady = FALSE;
}   //Does in fact nothing...

/* Main function */
void main(void)
  {
  ADCSR |= (1<<ADPS2);     //Use prescale factor 16 -> ADC clock is 500kHz
  ADCSR |= (1<<ADFR);      //Enable free-running mode
  ADMUX = currentChannel;  //Initial channel selection
```

```
ADCSR |= (1<<ADIE);      //Enable ADC conversion complete interrupt
ADCSR |= (1<<ADEN);      //Enable the ADC
_SEI();                  //Enable global interrupts
ADCSR |= (1<<ADSC);      //Start first conversion in Free-running mode
                         //Now the conversions are running
                         //Channels are changed in interrupt service routine

while(1)
  {
  if (newSampleReady)    //If new sample available
    process_Data();      //Imagine that samples are processed
  }
}
```