Atmel AVR123: AT90PWM81/161 ADC
Conversion Optimization Versus Temperature

Features
- Optimization of ADC conversion results versus temperature
- Applicable to the Atmel® AT90PWM81/161 when using the internal \( V_{\text{REF}} \) for the ADC

1 Introduction

The AT90PWM81/161 features a 10-bit successive approximation ADC. The ADC is connected to a 15-channel analog multiplexer, which provides:

- 11 single-ended inputs which are referenced to 0V (GND)
- Two differential voltage input combinations, which come with a programmable gain stage, providing amplification steps of 14dB (5x), 20 dB (10x), 26 dB (20x), or 32dB (40x) on the differential input voltage before the A/D conversion. On the amplified channels, 8-bit resolution can be expected.

This application note explains how to re-adjust the ADC conversion results over the temperature.
2 Theory of operation – the ADC conversion

2.1 VREF control

The reference voltage for the ADC (VREF) indicates the conversion range for the ADC. It can be selected as either:

- AVCC,
- internal 2.56V reference,
- or the voltage present on the external AREF pin.

The internal reference VREF of 2.56V is generated, after multiplication, from the Bandgap voltage.

2.2 VREF calibration

This internal voltage reference is function of the temperature.

It is calibrated at factory @3V and ambient temperature within accuracy of ±1% of the 2.56V reference voltage. The result of this calibration is stored in the Signature Row. This Final Test “Amb.VREF” is loaded at address 0x1E (please also see the Atmel AT90PWM81/161 datasheet).

Still at factory, a reading of the VREF level is achieved at 105°C. This value is also stored in the Signature Row. This Final Test “Hot VRef” is loaded at address 0x1F.

2.3 ADC conversion

For single ended conversions, the conversion result is:

\[
\text{Read ADC} = \frac{V_{\text{IN}} \times 1023}{V_{\text{REF}}}
\]

where \( V_{\text{IN}} \) is the voltage on the selected input pin and \( V_{\text{REF}} \) the selected voltage reference.
3 The compensation method

The Atmel AT90PWM81/161 microcontroller offers an embedded temperature sensor. This feature can be used for runtime compensation of temperature drift in the voltage reference.

3.1 $V_{\text{REF}}$ versus temperature

In following example, we consider the default configuration of BGCRR Register which shifts the top of the $V_{\text{REF}}$ curve to the highest possible temperature. In this configuration; the higher the temperature is, the higher the $V_{\text{REF}}$.

![Figure 3-1. $V_{\text{REF}}$ versus temperature.](image)

**Notes:**
1. Amb $V_{\text{REF}}$ is not necessarily strictly equal to 2.56V depending on the calibration ($\pm1\%$ accuracy).
2. Amb $V_{\text{REF}}$ is the calibrated value at Factory = 2.56V @25°C with a $\pm1\%$ accuracy
   Hot $V_{\text{REF}}$ is the Read value at Factory @105°C.
3. Amb $V_{\text{REF}}$ and Hot $V_{\text{REF}}$ are stored in Signature Row during Test operation at Factory and can be read by Software:
4. Final Test Amb $V_{\text{REF}}$: is loaded in two bytes at Address 0x3D (High Byte) 0x3C (Low Byte).
5. Final Test Hot $V_{\text{REF}}$ (only for Read): is loaded in two bytes at Address 0x3F/0x3E.
   These constants are the hexadecimal value of the voltage in mV: for instance 0x0A00 represents the Hexadecimal value of 2560mV.

3.2 Temperature measurement

This implementation uses the measurement achieved with the embedded temperature sensor of the AT90PWM81/161.
If the temperature sensor has been selected, the temperature measurement formula is:

Temp (°C) = (((((ADCH << 8) | ADCL) - (273 + 25-TSOFFSET)) × TSGAIN)/128) + 25

TSGAIN and TSOFFSET are stored in the Signature Row during Test operation at Factory:

Temperature Sensor Offset: TSOFFSET is loaded in High Byte of Address 0x05
Device Temperature Sensor Gain: TSGAIN is loaded in High Byte of Address 0x07 (typical value is 0x80)

3.3 $\text{V}_{\text{REF}}$ recalculation

Between 25°C and 105°C, $\text{V}_{\text{REF}}$ curve versus temperature range can be extrapolated as a straight line (see Figure 3-1).

To improve overall $\text{V}_{\text{REF}}$ accuracy, the recalculated $\text{V}_{\text{REF}}$ can be calculated as following:

Re-calc. $\text{V}_{\text{REF}} = (A \times \text{Temp}) + B$

The known points of the straight line are:

Amb $\text{V}_{\text{REF}} = (A \times (25^\circ\text{C})) + B = \text{data stored in Signature Byte}$

Hot $\text{V}_{\text{REF}} = (A \times (105^\circ\text{C})) + B = \text{data stored in Signature Byte}$

A and B variables can be extracted from these two equations.

3.4 ADC measurement compensation

The ADC measurement result can be compensated with following formula:

Compensated ADC = Read ADC × Re-calc. $\text{V}_{\text{REF}} / 2.56$
4 Hardware configuration

4.1 AT90PWM81/161

Five parts have been tested. Their Fuse configuration is:

Extended = FD
High = D9
Low = CC

4.2 STK521

XTAL: 8MHz
Monitoring of $V_{\text{REF}}$: $A_{\text{REF}}$ output / $V_{\text{SS}}$
UART software output: PB0 connected to RS232 interface

4.3 Hyperterminal

UART Bitrate = 19200 bit/s
5 Software configuration

See Chapter 8: Code example (compiled with IAR).
6 Result of temperature measurement

The chart in Figure 6-1 confirms that, over the five tested parts, the difference of temperature measurement is + 4°C.

Figure 6-1. Temperature monitoring gap.
7 Results of $V_{\text{REF}}$ recalculation

The chart in Figure 7-1 provides the accuracy (%) of the recalculated $V_{\text{REF}}$ versus the real $V_{\text{REF}}$ over the temperature range:

$$\text{Accuracy} = \frac{(V_{\text{REF \ recalc}} - V_{\text{REF \ output \ monitoring}})}{V_{\text{REF \ output \ monitoring}}}$$

**Figure 7-1. Accuracy vs. temperature.**

These typical results confirm that over a temperature range of [+5°C to +125°C]; the accuracy of the recalculation is better than 1%. 
### 8 Code example

#### 8.1 C Main function

```c
//! Copyright (c) 2009 Atmel.
//! This program uses a loop to:
//! - monitor the temperature sensor
//! - calculate the V_{REF} which should be equal to the real internal V_{REF}

// INCLUDES
#include "config.h"
#include "iopwm81.h"
#include "my_print.h"

// DECLATATIONS
#define HIGHBYTE(v) ((unsigned char) (((unsigned int) (v)) >> 8));
#define LOWBYTE(v) ((unsigned char) (v));

int main(void)
{
    unsigned char gain, offset, temp, vref_amb_low, vref_hot_low, vref_amb_high,
    vref_hot_high, result;
    unsigned int vref_recalc, vref_amb, vref_hot, n;
    char g, i;
    float a, b;

    PORTB = 0x00;
    DDRB = 0xC7;
    ADCSRA |= 0x80; /* ADEN=1 */
    ADMUX |= 0x80; /* Vref=2.56V */
    ADMUX &= ~0x2F;
    ADMUX |= 0x0C; /* MUX to Temp sensor */
    ADCSRB = 0x80; /* ADC High speed + free running*/
    ADCSRA |= 0x04; /* prescaler /16 */
    ADCSRA |= (1<<ADSC); /* first conversion */
```
while (SPMEN==1);

asm("LDI R17,$00");/* Beginning of LPM sequence to read the Temp. sensor OFFSET in Signature Row */
asm("LDI R16,$05");
asm("MOV R31,R17");/* */
asm("MOV R30,R16");/* ;move address to z pointer (R31=ZH R30=ZL)*/
SPMCSR=0x21;
asm("LPM");/* ;Store program memory*/
asm("MOV R16, R0");/* ;Store return value (1byte->R16 register)*/
asm("OUT 0x1B, R16");/* ;Store return value (1byte->R16 register)*/
while (SPMEN==1);
offset=GPIOR2; /* return of Temp. sensor OFFSET in Signature Row */

asm("LDI R17,$00"); /* Beginning of LPM sequence to read the Temp. sensor GAIN in Signature Row */
asm("LDI R16,$07");
asm("MOV R31,R17");/* */
asm("MOV R30,R16");/* ;move adress to z pointer (R31=ZH R30=ZL)*/
SPMCSR=0x21;
asm("LPM");/* ;Store program memory*/
asm("MOV R16, R0");/* ;Store return value (1byte->R16 register)*/
asm("OUT 0x1A, R16");/* ;Store return value (1byte->R16 register)*/
while (SPMEN==1);
gain=GPIOR1; /* return of Temp. sensor GAIN in Signature Row */

asm("LDI R17,$00"); /* Beginning of LPM sequence to read the Vref. Amb.(low Byte) in Signature Row */
asm("LDI R16,$3C");
asm("MOV R31,R17");/* */
asm("MOV R30,R16");/* ;move adress to z pointer (R31=ZH R30=ZL)*/
SPMCSR=0x21;
asm("LPM");/* ;Store program memory*/
asm("MOV R16, R0");/* ;Store return value (1byte->R16 register)*/
asm("OUT 0x1A, R16");/* ;Store return value (1byte->R16 register)*/
while (SPMEN==1);
vref_amb_low=GPIOR1; /* return of Vref. Amb. Low Byte in Signature Row */

asm("LDI R17,$00") ;/*Beginning of LPM sequence to read the Vref. Amb.(High Byte) in Signature Row */
asm("LDI R16,$3D");
asm("MOV R31,R17");/* */
asm("MOV R30,R16");/* ;move adress to z pointer (R31=ZH R30=ZL)*/
SPMCSR=0x21;
asm("LPM");/* ;Store program memory*/
asm("MOV R16, R0");/* ;Store return value (1byte->R16 register)*/
asm("OUT 0x1A, R16");/* ;Store return value (1byte->R16 register)*/
while (SPMEN==1);
vref_amb_high=GPIOR1; /* return of Vref. Amb. High Byte in Signature Row */

asm("LDI R17,$00");;/*Beginning of LPM sequence to read the Vref. Hot(low Byte) in Signature Row */
asm("LDI R16,$3E");
asm("MOV R31,R17");/* */
asm("MOV R30,R16");/* ;move adress to z pointer (R31=ZH R30=ZL)*/
SPMCSR=0x21;
asm("LPM");/* ;Store program memory*/
asm("MOV R16, R0");/* ;Store return value (1byte->R16 register)*/
asm("OUT 0x1A, R16");/* ;Store return value (1byte->R16 register)*/
while (SPMEN==1);
vref_hot_low=GPIOR1; /* return of Vref. Hot Low Byte in Signature Row */

asm("LDI R17,$00");;/*Beginning of LPM sequence to read the Vref. Hot(High Byte) in Signature Row */
asm("LDI R16,$3F");
asm("MOV R31,R17");/* */
asm("MOV R30,R16");/* ;move adress to z pointer (R31=ZH R30=ZL)*/
SPMCSR=0x21;
asm("LPM");/* ;Store program memory*/
asm("MOV R16, R0");/* ;Store return value (1byte->R16 register)*/
asm("OUT 0x1A, R16");/* ;Store return value (1byte->R16 register)*/
while (SPMEN==1);
vref_hot_high = GPIO1; /* return of Vref. Hot High Byte in Signature Row */

vref_hot = (vref_hot_high * 256) + vref_hot_low;
vref_amb = (vref_amb_high * 256) + vref_amb_low;
a = (vref_hot - vref_amb) * 100;
a = a / 0x50;
b = vref_amb - ((a * 0x19) / 100);

while(1)
{
    while (ADIF == 0);
    ADCSRA |= 0x10; /* reset ADIF */
    temp = ADCL;
    result = (ADCH << 8);
    result = result | temp;
    result = result - (273 + 25 - offset);
    temp = gain / 128;
    result = result * temp;
    temp = result + 25;

    putchar(0x54); putchar(0x3D); print_hex(temp); /* T=... temperature measurement in hex format */
    for(i=1; i<100; i++); putchar(0x0D); for(i=1; i<100; i++); putchar(0x0A);

    vref_recalc = (((a * temp) / 100) + b); /* Vref. recalculated versus the temperature measurement */
    putchar(0x41); putchar(0x3D);
    print_hex(vref_amb_high);
    print_hex(vref_amb_low);
    for(i=1; i<100; i++); putchar(0x0D); for(i=1; i<100; i++); putchar(0x0A); /* A=... Vref Amb. in hex format */

    putchar(0x48); putchar(0x3D);
    print_hex(vref_hot_high);
    print_hex(vref_hot_low);
for(i=1;i<100;i++); putchar(0x0D); for(i=1;i<100;i++); putchar(0x0A); /* H=... Vref Hot. in hex format*/

putchar(0x52);putchar(0x3D);  /* R=... Vref recalculated in hex format*/
temp=HIGHBYTE(vref_recalc);
print_hex (temp);
temp=LOWBYTE(vref_recalc);
print_hex (temp);
}
for(i=1;i<100;i++);   putchar(0x0D);  for(i=1;i<100;i++);  putchar(0x0A); /* R=... Vref Recalculated in hex format*/
for(n=1;n<10000;n++)
{
   for(i=1;i<100;i++);/* Delay to improve display in Hyperterminal */
}
ADMUX |=0x0C;
ADCSRA |= (1<<ADSC); /* Starts a new conversion on Temp. sensor */
}

8.2 C Software UART function

#include "my_print.h"
void print_hex(unsigned char n)
{
    unsigned char i;
    i=n>>4;
    if(i>9) putchar(i-0x0A+'A');
    else putchar(i+'0');
    i=n&0x0F;
    if(i>9) putchar(i-0x0A+'A');
    else putchar(i+'0');
}

void print_int(unsigned int n)
{
    print_hex(n>>8);
    print_hex(n);
8.3 Assembler Soft_uart.s90

// include the register definitions for the used AT90PWM81/161 mcu
#include "iopwm81.h"

PUBLIC putchar
PUBLIC soft_uart_init

;***** Pin definitions

// TxD EQU 0   ;Transmit pin is PDx

;***** Global register variables

#define bitcnt r16   ;bit counter
#define temp  r17   ;temporary storage register
#define Txbyte r18   ;Data to be transmitted

RSEG CODE:CODE:NOROOT(1)

;*********************************************************************
;* "putchar"
;* This subroutine transmits the byte stored in the "Txbyte" register
;* The number of stop bits used is set with the sb constant
;*
;* Number of words :14 including return
;* Number of cycles :Depends on bit rate
;* Low registers used :None
;* High registers used :2 (bitcnt,Txbyte)
;* Pointers used :None
;*
;*********************************************************************

sb EQU 1  ;Number of stop bits (1, 2, ...)

putchar:
  cli
  ;ldi bitcnt, (9+sb) ;1+8+sb (sb is # of stop bits)
  mov r18, r16
  ldi r16, (9+sb);1+8+sb (sb is # of stop bits)
  com  ;Inverte everything
  sec  ;Start bit

putchar0:  brcc putchar1 ;If carry set
  cbi PORTB,TxD ; send a '0'
  rjmp putchar2 ;else

putchar1:  sbi PORTB,TxD ; send a '1'
  nop

putchar2:  rcall UART_delay ;One bit delay
rcall UART_delay

lsr  Txbyte  ;Get next bit
dec  bitcnt  ;If not all bit sent
brne putchar0  ; send next
          ;else
sei
ret  ; return

;*********************************************************************
;*
;* "UART_delay"
;*
;* This delay subroutine generates the required delay between the bits
;* transmitting and receiving bytes. The total execution time is set
;* by the
;* constant "b":
;*
;* 3•b + 7 cycles (including rcall and ret)
;*
;* Number of words :4 including return
;* Low registers used :None
;* High registers used :1 (temp)
;* Pointers used  :None
;*
;*********************************************************************

b EQU 63
UART_delay:
    ldi  temp,b
    nop
    ldi  r17,b
UART_delay1:  dec  temp
              brne UART_delay1
          ret

;***** Program Execution Starts Here

;***** Test program

soft_uart_init:
    sbi  PORTB,TxD  ;Init port pins
    sbi  DDRB,TxD
    ret

forever:
    ldi  r18,0x55  ;'U'
    rcall  putchar
    rjmp  forever

END
9 Application note revision history

Please note that the page numbers are referring to this document. The revision reference in this section is to the document revision.

9.1 Rev. 8270B – 01/12

1. AT90PWM161 is added.

9.2 Rev. 8270A – 09/10

1. Initial version.
# Table of contents

**Features** .................................................................................................................. 1  
**1 Introduction** ........................................................................................................ 1  
**2 Theory of operation – the ADC conversion** ......................................................... 2  
  2.1 V\textsubscript{REF} control ....................................................................................... 2  
  2.2 V\textsubscript{REF} calibration .................................................................................... 2  
  2.3 ADC conversion ................................................................................................... 2  
**3 The compensation method** .................................................................................. 3  
  3.1 V\textsubscript{REF} versus temperature .......................................................................... 3  
  3.2 Temperature measurement ................................................................................... 3  
  3.3 V\textsubscript{REF} recalculation .................................................................................. 4  
  3.4 ADC measurement compensation ........................................................................ 4  
**4 Hardware configuration** ..................................................................................... 5  
  4.1 AT90PWM81/161 .................................................................................................. 5  
  4.2 STK521 ................................................................................................................ 5  
  4.3 Hyperterminal ...................................................................................................... 5  
**5 Software configuration** ....................................................................................... 6  
**6 Result of temperature measurement** ................................................................. 7  
**7 Results of V\textsubscript{REF} recalculation** ............................................................. 8  
**8 Code example** ...................................................................................................... 9  
  8.1 C Main function ................................................................................................... 9  
  8.2 C Software UART function .................................................................................. 13  
  8.3 Assembler Soft_uart.s90 ...................................................................................... 14  
**9 Application note revision history** ...................................................................... 16  
  9.1 Rev. 8270B – 01/12 ............................................................................................... 16  
  9.2 Rev. 8270A – 09/10 ............................................................................................... 16  
**10 Table of contents** .............................................................................................. 17