AVR351: Runtime calibration and compensation of RC oscillators

8-bit AVR® Microcontrollers

Application Note

Features

- **Use of the Oscillator Sampling Interface in calibration.**
- **Slow RC oscillator frequency prediction.**
- **Ultra Low Power RC oscillator frequency measurement.**
- **Fast RC frequency calibration.**

1 Introduction

Due to frequency drift over temperature, the clock sources in AVR should be calibrated at operating temperature or runtime if system temperature changes over time, to achieve the best possible accuracy.

Several of the new battery-monitoring devices, e.g. ATmega16HVA, have a very accurate coulomb counting ADC, and thus need a precise time reference to achieve the best possible results. They also have protection circuitry, and thus need a low-power clock source to minimize power consumption. The oscillators, signature bytes, internal temperature sensor and the Oscillator Sampling Interface provide the ways to achieve this.

The oscillators in these devices are the Slow RC, Ultra Low Power RC and Fast RC Oscillator. The Slow RC oscillator has a very predictable behavior over temperature, and can thus be used for runtime calibration of the other oscillators. In some devices it is also used for the Coulomb Counting ADC. The Ultra Low Power oscillator has very low power consumption and enables the lowest possible system power consumption, thus making it ideal for battery protection. In some devices it is also used for the Coulomb Counting ADC. The last oscillator is the Fast RC oscillator that is routed to parts of the system concerned with the AVR core, e.g. core, flash, eeprom and timers.

Rev. 8053A-AVR-10/08

2 Theory of operation

Calibration of the clock sources in the AVR parts is done by either adjusting the frequency directly or measuring the actual frequency and compensating for it. This application note describes the properties and use of the various oscillators in battery monitoring devices, and how calibration is performed. The new battery-monitoring devices, e.g. ATmega16HVA have a module called Oscillator Sampling Interface that is used to simplify measurements of clock periods and calibration.

2.1 Oscillator Sampling Interface

The Oscillator Sampling Interface (OSI) is a module that helps measuring the period of a clock input against the system clock. The OSI pre-scales the selected input and outputs this signal. This pre-scaled clock output can be directly seen by reading the OSIST bit in OSICSR, but should be used to trigger the input capture function of a connected timer for greatest accuracy. In e.g. the ATmega16HVA the Slow RC or ULP RC oscillator can be selected as input to the OSI with 128x pre-scaling and Timer/Counter0 used to capture the period.

2.2 Slow RC oscillator

The Slow RC oscillator has a predictable frequency drift over temperature and provides an accurate clock as reference for calibration or for coulomb counting ADC. From production the following two parameters are known and stored in the signature row of the:

- Slow RC oscillator frequency at high temperature (usually 70ºC or 85ºC) in production test (Slow RC word).
- Slow RC oscillator temperature drift coefficient (Slow RC Temp Prediction word)

When knowing these parameters and by measuring the on-chip temperature, the actual Slow RC oscillator frequency can be determined runtime for any temperature within the operating area.

Using the calibration scheme presented in this application note, the Slow RC oscillator frequency can be determined with an error less than 1% over the specified temperature range. The Slow RC oscillator is also used for the coulomb counter in some devices.

2.2.1 Calculating Slow RC oscillator period at current temperature

The formula for calculation of Slow RC oscillator period in e.g. ATmega16HVA is given in Equation 2-1.

Equation 2-1. Actual clock period of Slow RC Oscillator.

$$T_{Slow\ RC}(\mu s) = \frac{Slow\ RC\ word - Slow\ RC\ Temp\ word \cdot (T - T_{HOT})/64}{1024}$$

2.3 Ultra Low Power RC oscillator

The Ultra Low Power (ULP) RC oscillator provides a very low power clock that is used for Battery Protection, Watchdog timer and Reset Logic, and could also be used for the Coulomb Counting ADC (CC-ADC) in some devices. Its frequency drifts with temperature and process, and needs to be compensated for. The recommended way is comparing the actual period with the calculated Slow RC oscillator reference using the fast RC to measure both periods and the calculate the ULP RC period from the Slow RC period.

The nominal period at a high temperature (usually 70°C or 85°C) is also stored in the signature row. This value gives a rough estimate of the ULP oscillator frequency and can be used without further compensation for less demanding modules, e.g. to determine battery protection timing.

2.3.1 Measuring Ultra Low Power oscillator period against Slow RC oscillator

Equation 2-2 shows the formula for measuring and calculating the ULP RC period with the Slow RC as reference.

Equation 2-2. Calculation of ULP RC Oscillator clock period.

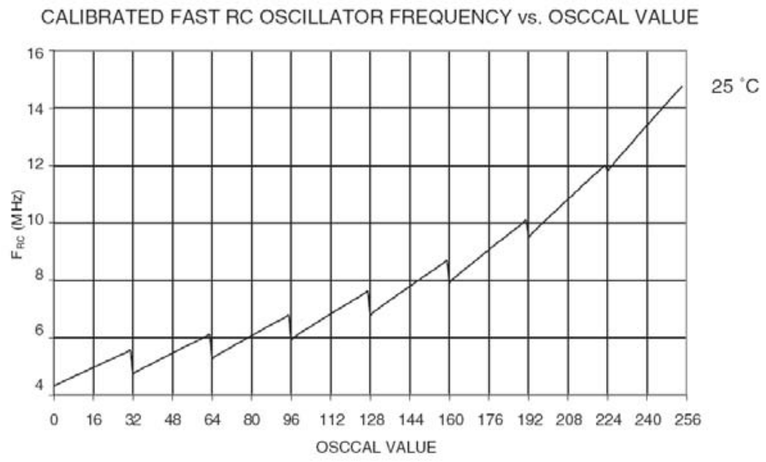
$$T_{ULP\ RC}\,(\mu s) = T_{Slow\ RC} \cdot \frac{\text{number of CPU cycles in } n \text{ prescaled ULP RC periods}}{\text{number of CPU cycles in } n \text{ prescaled Slow RC periods}}$$

2.4 Fast RC Oscillator

The Fast RC oscillator is the clock for the CPU core, RAM, Flash, EEPROM, VADC and other I/O-modules. At startup the calibration byte, FOSCCAL, is automatically loaded from the signature row into the FOSCCAL register. This is a calibrated value from Atmel production, and is guaranteed to give 1MHz ±4% at either 25°C or a high temperature (70ºC to 85°C) with default clock prescaler. The Fast RC oscillator frequency against FOSCCAL value for ATmega16HVA is plotted in Figure 2-1.

The Fast RC oscillator might need to be calibrated for some applications, e.g. asynchronous serial communication, especially if temperature is very different from production calibration or changing. This can e.g. be done runtime against the Slow RC oscillator as described in this application note. Note that the Fast RC oscillator frequency should under no circumstance be calibrated to more than 10% above the nominal frequency, since this may cause EEPROM and Flash access to fail.

Figure 2-1. ATmega16HVA Fast RC characteristics



The default FOSCCAL value is selected such that it is in the lower half of a segment. It is therefore sufficient to use that default segment and the one below to calibrate the oscillator frequency to 8MHz over the whole temperature range. To avoid a large frequency change when shifting between the two segments, a FOSC SEGMENT value is also stored in the signature row. This is the first FOSCCAL value giving a lower frequency than the lowest value in the default segment, and should be used when calibrating the Fast RC oscillator.

2.4.1 Runtime calibration of Fast RC against Slow RC oscillator using the Oscillator sampling interface

Equation 2-3. Fast RC oscillator period.

$$T_{Fast\ RC}(\mu s) = T_{Slow\ RC} \cdot \frac{128 \cdot n}{\text{number of CPU cycles in } n \text{ prescaled Slow RC periods}}$$

$$n = \text{number of prescaled Slow RC periods}$$

A typical calibration sequence incorporates measuring “number of CPU cycles in *n* prescaled Slow RC periods” and comparing this value to the reference that can be found by rearranging Equation 2-3 and inserting the calculated Slow RC oscillator period at the actual temperature and the desired Fast RC oscillator period. The FOSCCAL is then adjusted and a new measurement is done. This is repeated until either the best value is found or a sufficient accuracy is attained

This procedure is not very resource consuming from a CPU point-of-view since the OSI interface is used to trigger a Timer/Counter0 input capture. However, since the Fast RC oscillator must be running during the calibration, the chip cannot operate in the deepest sleep mode (Power Save). Hence, extended calibration beyond actual need may increase power consumption unnecessarily.

2.5 Internal temperature sensor

The ATmega16HVA have an internal temperature sensor that can be sampled with the voltage ADC. Routines for getting accurate temperature results have been included for demonstration, but the reader is directed to application note AVR353 for more details.

3 Implementation

This application note includes source code with routines for frequency calculation, measurement and calibration. It can be downloaded as a zip-archive from Atmel Web. The source contains code consisting of both functions for calibration and supporting routines. The code has source documentation (see section Source documentation), while this section documents the code at a higher level.

3.1 Signature bytes

The ATmega16HVA contains extensive signature row data. A file defining all signature bytes, together with a macro for reading signature bytes, is provided with the source code. In this application only the relevant bytes are used.

3.2 Temperature measurements

The Slow RC oscillator period calculation requires measuring on-chip or system temperature. In an actual battery application this would usually be part of the interrupt controlled VADC measurements, but in this application a function for sampling the internal temperature sensor is provided. To ensure accurate measurements, a function for calibrating the voltage reference with factory calibration from the signature row is provided. For more details on the VADC, temperature measurements and the voltage reference please see application note AVR353.

3.3 Oscillator Sampling Interface

The OSI is used to generate a capture event in Timer/Counter0, but to avoid using resources needed for ADC measurements and communications the Capture interrupt flag is polled instead of using actual interrupts. This can be rewritten to interrupt control if so is desired. A dedicated subroutine for measuring a number of CPU cycles in a number of OSI pre-scaled clocks (in this case 8) are included and used in the ULP RC Oscillator measurement and the Fast RC Oscillator Calibration.

3.4 Slow RC Oscillator period calculation

The Slow RC oscillator period is calculated by inputting on-chip temperature to the provided function, which uses Equation 2-1 to calculate the period except that the result is not divided by 1024. This increases the accuracy and enables the result to be used directly for calibrating the Fast RC oscillator without any scaling except if system frequency is different from 1 MHz. The resulting equation is shown in Equation 3-1.

Equation 3-1. Clock period of Slow RC Oscillator used.

$$T_{Slow\ RC}\left[\mu s * 1024\right] = Slow\ RC\ word\ - Slow\ RC\ Temp\ word\ \cdot (T - T_{HOT})/64$$

3.5 ULP RC Oscillator measurement

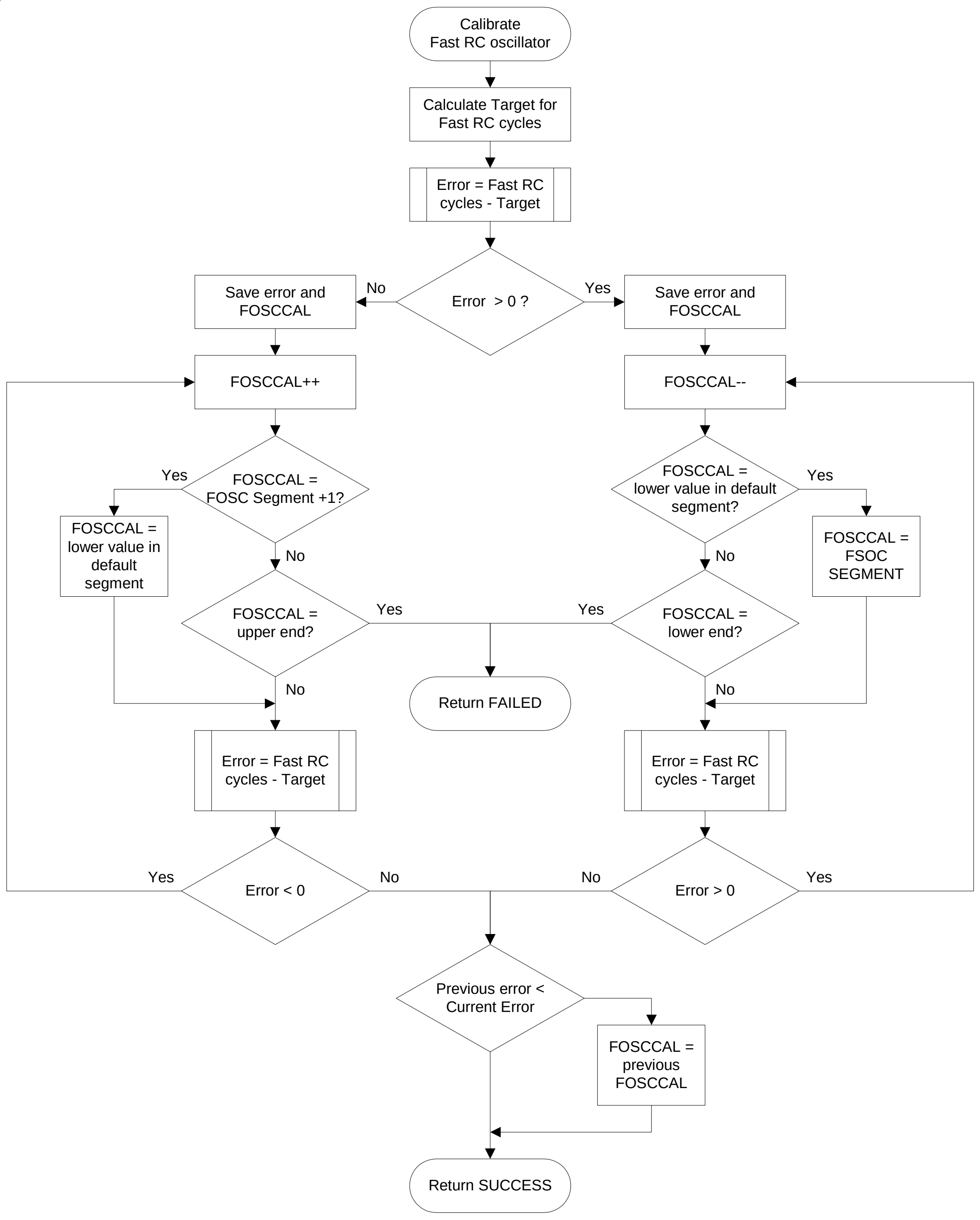
The period of the ULP RC oscillator is measured against the Slow RC oscillator with the OSI. Slow RC oscillator period is input to the ULP RC oscillator function, which then measures the period of both 8 OSI pre-scaled Slow RC and ULP RC oscillator periods. Finally, the ULP RC oscillator period is calculated from Equation 2-2, but as the Slow RC oscillator period is not divided by 1024 the ULP RC result is neither.

3.6 Fast RC Oscillator calibration

The fast RC oscillator is calibrated against the Slow RC oscillator. The function takes the Slow RC period as input and searches linearly from the current FOSCCAL value. Initially a search direction is determined and then the FOSCCAL is adjusted in that direction until the Fast RC frequency has passed the target. The error at that point is then compared to the error at the previous FOSCCAL value and the value giving the least error is selected and a success value returned. The FOSC SEGMENT value form the signature row is used to avoid a step in frequency when jumping from the upper to the lower segment and reverse. Since the two segments shall provide a sufficient search range to find a FOSCCAL value for all temperatures the search is aborted, the default FOSCCAL value loaded and a failure code returned if the upper or lower limits are encountered.

A flowchart for the function is shown in Figure 3-1.

Figure 3-1. Flowchart for the Fast RC oscillator calibration.



Calibrate Fast RC oscillator
Calculate Target for Fast RC cycles
Error = Fast RC cycles - Target
Error > 0 ?
No
Yes
Save error and FOSCCAL
Save error and FOSCCAL
FOSCCAL++
FOSCCAL--
FOSCCAL = FOSC Segment +1?
Yes
No
FOSCCAL = lower value in default segment
FOSCCAL = lower value in default segment?
Yes
No
FOSCCAL = FSOC SEGMENT
FOSCCAL = upper end?
Yes
No
FOSCCAL = lower end?
Yes
No
Return FAILED
Error = Fast RC cycles - Target
Error = Fast RC cycles - Target
Error < 0
Yes
No
Error > 0
No
Yes
Previous error < Current Error
FOSCCAL = previous FOSCCAL
Return SUCCESS

4 Source documentation

Please see `readme.html` provided with the code for complete source documentation. Complete complier information, settings, and device information are also provided there.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature