AVR1606: XMEGA Internal RC Oscillator Calibration

8-bit AVR® Microcontrollers

Application Note

Features

- **Adjustable RC frequency with +/-1% accuracy**
- **Support for all XMEGA™'s with tunable RC oscillator via JTAG interface**
- **Calibration using JTAGICE mkII and AVRONE for one time**
- **Tune RC oscillator at any operating voltage and temperature**
- **Firmware implementation for use during production programming**

1 Introduction

This application note describes a fast and accurate method to calibrate the internal RC oscillator. It offers an easily adaptable calibration firmware source code, which can be used with any XMEGA with internal tunable RC oscillator and JTAG interface. This firmware allows device calibration using the AVR® tools JTAGICE mkII and AVRONE.

The internal RC oscillator frequency can be calibrated to within +/-1% of the frequency specified in the datasheet for the XMEGA device. This feature offers great flexibility and significant cost savings compared to using an external oscillator.

The calibration performed in the Atmel factory is made at a fixed operating voltage and temperature (typically 85°C, 3V). As the frequency of internal RC oscillators are affected by both operating voltage and temperature, it may be desired to perform a secondary calibration, which matches the specific application environment. This secondary calibration can be performed to gain higher accuracy than the standard calibration offers, to match a specific operating voltage or temperature.

The calibration method described in this application note only takes a fraction of a second longer than reading the factory calibration byte from the signature row and writing it back to the to the device memory. Thus, the overall programming time is almost unaffected when performing calibration in the programming step in production.

In some systems it may be more beneficial to perform run-time calibration of the oscillator. This may be desirable in applications that need an accurate system clock over the entire temperature range and independent of operating voltage. In that case a 32.768 kHz watch crystal may offer a reliable and cost efficient solution. Automatic runtime calibration is covered by chapter 7 Runtime calibration

A Quick Start Guide is found at the end of this document.

Rev. 8277A-AVR-12/09

2 Theory of operation – the internal RC oscillators

In production the internal RC oscillators are mostly calibrated at 3V/85°C. Refer to Oscillator Characteristics in the datasheet of the individual devices for information about the operating voltage used during calibration. If a design's need for accuracy is +/-1% at other operating voltage and temperature beyond what can be offered by the standard calibration in factory by Atmel, it is possible to perform a secondary calibration of the RC oscillator. By doing this it is possible to obtain frequency accuracy within +/-1 at any operating voltage and temperature. A secondary calibration can thus be performed to improve or tailor the accuracy or frequency of the oscillator.

2.1 Clock selection

The XMEGA System Clock source is selectable from software, and can be changed during normal operation. Each oscillator option has a status flag that can be read from software to check that the oscillator is ready. After reset the XMEGA starts up running from the 2 MHz internal calibrated RC oscillator. An overview of the clock selection is available in the datasheets.

2.2 Internal calibrated RC oscillator overview

There are three internal calibrated RC oscillators on XMEGA, ranging from 32.768 kHz RC oscillator to a 32 MHz factory calibrated RC oscillator with auto calibration features. All calibrated oscillators can be used as main system clock.

The following sections provide an overview of the internal calibrated RC oscillators available in the XMEGA microcontrollers.

2.2.1 Calibrated 32.768 kHz RC Oscillator

This RC oscillator provides an approximate 32.768 kHz clock. A factory-calibrated value is written to the 32.768 kHz oscillator calibration register during reset to ensure that the oscillator is running within its specification. The calibration register can also be written from software for runtime calibration of the oscillator frequency. The oscillator employs a built in prescaler providing both a 32.768 kHz output and a 1.024 kHz output.

2.2.2 Calibrated 2 MHz RC Oscillator

This RC oscillator provides an approximate 2 MHz clock. The oscillator employs a Digital Frequency Looked Loop (DFLL) that can be enabled for automatic run-time calibration of the oscillator. A factory-calibrated value is written to the 2 MHz DFLL Calibration Register during reset to ensure that the oscillator is running within its specification. The calibration register can also be written from software for manual run-time calibration of the oscillator.

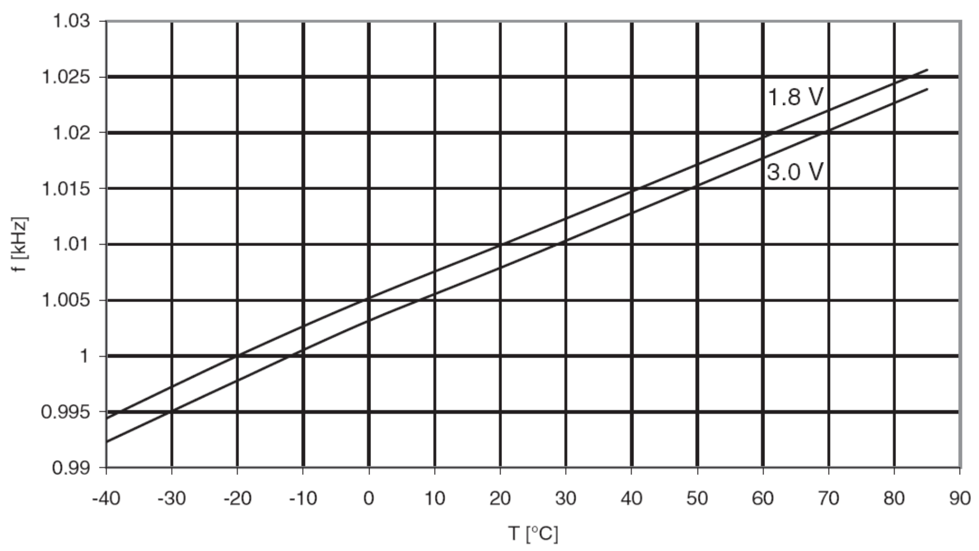
2.2.3 Calibrated 32 MHz RC Oscillator

This RC oscillator provides an approximate 32 MHz clock. The oscillator employs a Digital Frequency Locked Loop (DFLL) that can be enabled for automatic run-time calibration of the oscillator. A factory-calibrated value is written to the 32 MHz DFLL Calibration Register during reset to ensure that the oscillator is running within its specification. The calibration register can also be written from software for manual run-time calibration of the oscillator.

2.3 Oscillator characteristics

The frequency of the internal 32.768 kHz RC oscillator is depending on the temperature and operating voltage. An example of this dependency is seen in Figure 2-1, which shows the 1.024 kHz output frequency of the 32.768 kHz RC oscillator of the ATxmega128A1. As seen from the figure, the frequency increases with increasing temperature, and decreases slightly with increasing operating voltage. These characteristics will vary from device to device. For details on a specific device refer to its datasheet.

Figure 2-1. Internal 32.768 kHz oscillator Frequency vs. Temperature (1.024 kHz output)



All XMEGA devices with tunable 32.768 kHz RC oscillators have an RC32KCAL register for tuning the oscillator frequency. An increasing value in RC32KCAL will result in an increase in frequency. This information is very relevant when searching for the best calibration value to fit a given frequency.

The two built in Digital Frequency Locked Loops (DFLLs) in all XMEGA devices can be used to improve the accuracy of the 2 MHz and 32 MHz internal oscillators. The reference clock sources can be selected to be the internal 32.768 Hz RC oscillator or an external 32.768 kHz watch crystal. That means the 2 MHz and 32 MHz internal oscillators precision will be decided by the reference clock accuracy. When the DFLL is enabled it will count each oscillator clock cycle, and for each reference clock edge, the counter value is compared to the fixed ideal relationship between the reference clock and the oscillator frequency. If the internal oscillator runs too fast or too slow, the DFLL will decrement or increment the corresponding DFLL Calibration Register value by one to adjust the oscillator frequency slightly. For details refer to the XMEGA manual.

Knowing the fundamental characteristics of the RC oscillators, it is possible to make an efficient calibration routine that calibrates the RC oscillator to a given frequency, at any operating voltage and at any temperature with an accuracy of +/-1%.

2.4 Implementation of the calibration

This section is divided into a description of the calibration protocol and a description of the firmware. The protocol can be adapted into any test or programming tool to support calibration. The AVR tools JTAGICE mkII and AVRONE support the implemented calibration protocol. The usage of these tools to calibrate a device is described later.

The calibration support in JTAGICE mkII and AVRONE is at present only supported in the command-line version of the tools. The calibration is supported for JTAGICE mkII and AVRONE in AVR Studio® version 4.18 (or later). The newest release of AVR Studio can be downloaded from www.atmel.com/avrstudio.

The calibration support for STK600 will be present in future version.

2.5 Calibration protocol

The protocol for calibration uses TDI and TDO on the JTAG interface so that that the calibration can be used in production environment for a final product (or on PCB).

The basic concept is that the programmer generates the calibration clock (C-clock), and the device uses this as a reference to calibrate its internal RC oscillator. When the device has completed the calibration it signals "OK" to the programmer on the TDO line.

In the protocol of calibration, the XMEGA device should enable pull-up resistor on the TDI line, and the programmer (JTAGICE mkII and AVRONE) should enable pull-up resistor on the TDO line. Unfortunately the programmer is in many cases behind level converters, so the device sets the TDI line high also. This is done to ensure that noise is unlikely to corrupt the calibration.

The programmer can use 5120 C-cycles (cycles on the C-clock) as time-out period, as the calibration routine is guaranteed to be completed within this number of C-cycles.

The calibration procedure runs through the following steps:

1. The programmer writes the calibration firmware into the device and possibly enables the TDO pull-up, and releases the reset line. The JTAG disable bit in MCUCSR is set. The calibration clock is applied on the TDI line. The frequency of this is 32.768 kHz.
2. The device enables the internal pull-up on the TDI line, sets the TDO line high, and starts listening for the calibration clock on TDI.
3. When the device detects the calibration clock a binary search is used to find an RC32KCAL value that meets the criteria of 1% accuracy. If calibration fails the TDO line is set low and program flow goes to step 6.
4. The calibration value is stored in EEPROM (skipped if calibration fails).
5. TDO line is toggled 8 times / 4 cycles by the device. The toggling of the TDO line is performed on the falling edge of the clock on the TDI line (C-clock), but 5 to 10 CPU cycles delayed. In the case of failing calibration the TDO line is not toggled.
6. JTAG interface is re-enabled and the device goes into an infinite loop.

7. If the device does not have an EESAVE fuse, the programmer must read back the calibration byte from EEPROM, for later restoring when the calibration firmware has been erased from the Flash. If the device have an EESAVE fuse, this fuse can be set so that erasing the Flash does not also erase the EEPROM.

Please note it is necessary to copy the calibration byte from EEPROM or FLASH to the calibration register at run-time. A routine for this must therefore be implemented in the final firmware.

2.6 The calibration firmware

The calibration code is written in C, for the AVR Studio 4.18 / IAR 5.20 (or later) with the calibration package installed.

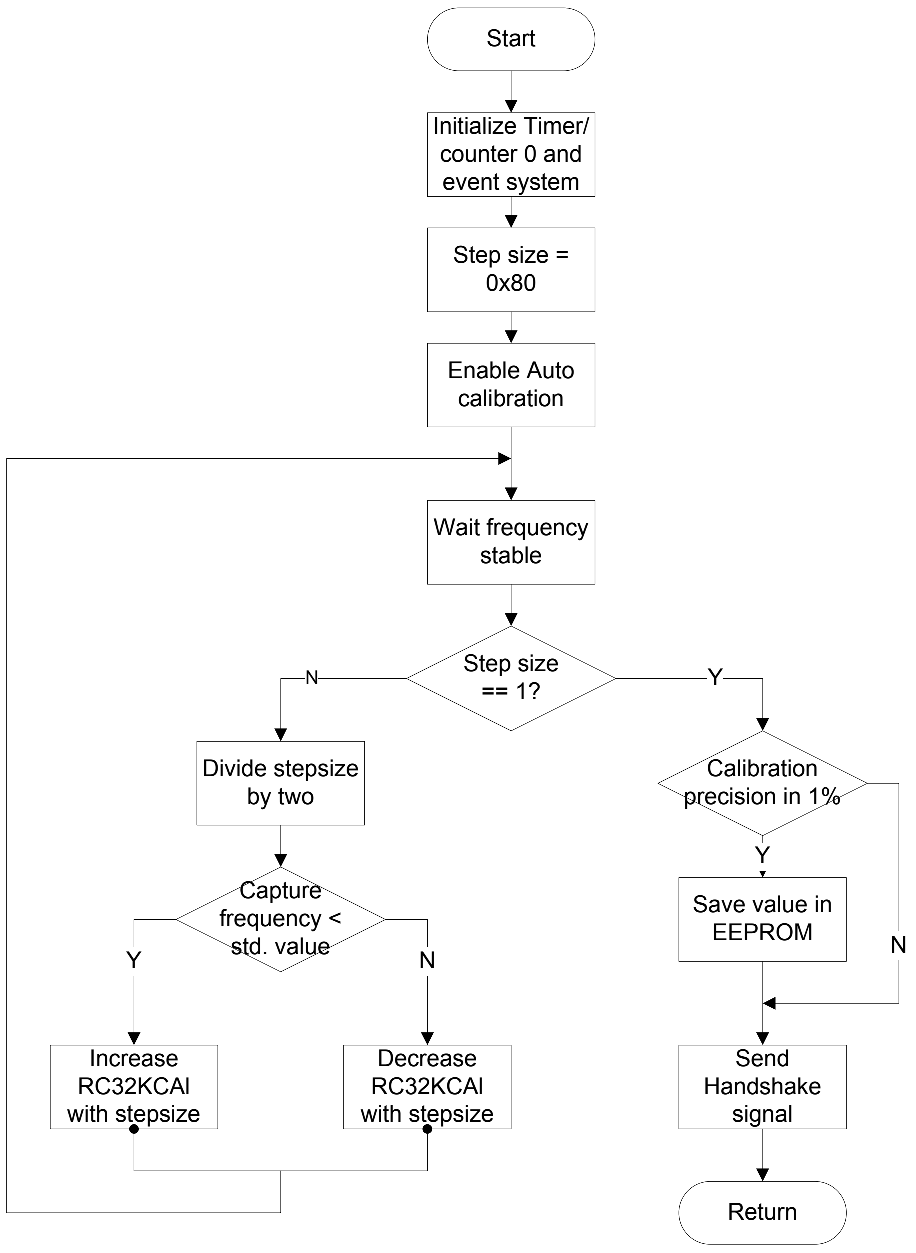
The root file refers to (includes) the following files:

1. The common calibration code “main.c”.
2. The driver of XMEGA event system code “event_system_driver.c”
3. The driver of XMEGA clock system code “clksys_driver.c”
4. The driver of XMEGA EEPROM code “eeprom_driver.c”
5. The driver of XMEGA timer/counter code “TC_driver.c”

2.6.1 Method for determining the oscillator frequency

The comparison between the Calibration clock (C-clock) and the internal RC oscillator are performed using the 16-bit Timer/Counter C0 (TCC0). It is used since it is be present in all devices that have tunable RC oscillator. The idea is to capture the frequency of C-clock cycles using XMEGA event system and compare the frequency to predefined limits. The C-frequency in the present implementation is given in the interface specific include file. The method for determining the oscillator frequency is described in the flowchart in Figure 2-2.

Figure 2-2. Flowchart of algorithm determining relationship between the C-clock and the internal oscillator frequency



Start
Initialize Timer/ counter 0 and event system
Step size = 0x80
Enable Auto calibration
Wait frequency stable
Step size == 1?
N
Y
Divide stepsize by two
Calibration precision in 1%
Y
N
Capture frequency < std. value
Y
N
Save value in EEPROM
Increase RC32KCAl with stepsize
Decrease RC32KCAl with stepsize
Send Handshake signal
Return

3 Calibrating with JTAGICE mkII or AVRONE

The source code of the calibration firmware and the batch file provided is made as an example of how to use JTAGICE mkII or AVRONE to perform calibration. The firmware needs few or no modifications to be used in other calibration systems.

3.1 Make the calibration firmware

To make the project in IAR EWAVR:

Add the .c files for the given example to your project. Select device type and enable bit definitions in I/O include files, optimization low for debug target and high for release, output format: ubrof8 for Debug and intel_extended for Release, select Normal DLIB as library.

To make the project in WinAVR:

Add the .c files (and .S files where applicable) for the given example to your project. Select device type, optimization low for debug target and high for release.

3.2 Using the command line tools

The calibration support in JTAGICE mkII and AVRONE is at present only supported in the command-line version of the tools (AVR Studio 4.18 or later). The software package that provides this support can be found at http://www.atmel.com/avr/. Please install this package for calibration support.

The package includes a new firmware for the AVR tools, which is required to enable calibration. The firmware upgrade is automatic when first connecting to the tool with AVR Studio 4.18 (or later) or manual as described in the AVR Studio help.

Batch files are provided along with the source code. These batch files show how the command line tools can be used to program the calibration code into the target device, perform the calibration and hence reprogram the device with the final firmware. The batch files are performing calibration of the ATxmega128A1 through JTAGICE mkII and AVRONE respectively. Please study these batch files and the AVR Studio integrated help to understand the use of JTAGICE mkII and AVRONE command line tools. Table 3-1 includes a list of the new commands to the .exe files that are related to the calibration operation.

Table 3-1. New oscillator calibration specific options in jtagice.exe

Command	**Description**
-Z [addr]	Read calibration byte from EEPROM memory. 'addr' is byte address. The read operation is performed before the "chip erase" is executed. Using '-S#' will re-write the value to flash or EEPROM after the chip erase.
-Y	Perform the oscillator calibration sequence. This command will override all other operations. The exe file will return an errorlevel 1 if it does not get the acknowledge signal from the target device.

4 Performance of the Calibration firmware

The code has been written with focus on efficiency: The entire calibration should be performed fairly quickly. The performance therefore depends on the size of the calibration firmware and the time it takes to complete the calibration.

The calibration routine is completed in less than 5120 calibration cycles. The shortest duration is however dependent on how fast the binary search algorithm can find a suitable RC32KCAL value, and the write time of the EEPROM. In the present implementation, using JTAGICEII.exe or AVRONE.exe, the calibration itself is completed in less than 1s.

5 Calibration Clock Accuracy

The accuracy of the calibration is highly dependent on the accuracy of the external calibration clock. The calibration clock frequency generated by the AVR tools may vary. It is therefore important to measure the exact frequency on the TDI pin on the JTAG interface and change it into the main.c file. Since resonators are dependent on both operating voltage and temperature, the calibration frequency should be measured when these parameters equals the conditions during calibration.

6 Quick Start Guide to Calibration of the internal RC using AVR Studio

To get started using the calibration feature in one of the device already supported one can follow steps below.

1. Download and unzip the source code for AVR1606 (any location can be used, here called \AVR1606\).
2. Download and install AVR Studio 4.18 (or later) from http://www.atmel.com/avr/
3. Open AVR Studio, create a new AVR GCC project called “rc_calib”, and add root source code file to the project.
4. Select a target device from AVR Studio <project> <configuration option>
5. Measure the frequency of the calibration clock with a frequency counter or an oscilloscope. This signal can be found on the TDI pin on JTAG interface. Change the line in the main.c file “#define CAL_REF_CLOCK_FREQ (xxxxx)” to reflect the measure frequency.
6. Compile the project to generate the hex binary file that should be programmed into the device.
7. Open the batch file corresponding to the tool, and edit the file to match the desired device, by changing the –datxmega128a1 argument to –d[target device].
   a) JTAGICE_mkII_rc_calib.bat for JTAGICE mkII with JTAG interface.
   b) AVRONE_rc_calib.bat for AVRONE with JTAG interface.
8. Please note that the reset line must be available.
9. Change the fuse setting to the desired setting. Make sure that the watchdog timer always on fuse is not set.
10. If the install path for AVR Studio differs from the one used in the batch file (the standard in English windows® versions), please change the path to the relevant .exe file.
11. For production calibration the @PAUSE command at successful calibration should be removed.
12. Save the batch file.
13. Connect the JTAGICE mkII or AVRONE to the target board. Power the tool and application. Make sure that the USB cable is attached between the tool and the PC.
14. Open a command shell window (a DOS prompt), navigate to the directory “\AVR1606\Batch file\”, and execute the batch file (JTAGICE_mkII_rc_calib.bat or AVRONE_rc_calib.bat). Or simply just run the batch file from a file browser.
15. Wait a short while for the calibration to complete.

The batch file can also be modified to program a custom firmware rather than the test.hex firmware after the calibration. Be aware that the new calibration value should be loaded into the RC32KCAL, DFLLRC32M and DFLLRC2M register at runtime by the firmware.

7 Runtime calibration using a 32.768 kHz reference clock

The XMEGA Clock System provides two Digital Frequency-locked Loops (DFLLs), one for the 2 MHz RC oscillator and one for the 32 MHz RC oscillator. The DFLLs can be configured individually to use either the internal 32.768 kHz RC oscillator or an external 32.768 kHz watch crystal as a reference for the calibration process.

Once enabled, a DFLL provides continuous calibration of its oscillator based on the clock reference. When entering sleep mode, the current state is frozen and the calibration loop continues from where it stopped when exiting from sleep mode again.

If a DFLL is disabled, the current calibration value for the oscillator will remain in effect until the DFLL is enabled again and the calibration process continues.

For more information please refer to the device datasheet and application note AVR1003.

Headquarters

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

International

Atmel Asia	***Atmel Europe***	***Atmel Japan***
Unit 1-5 & 16, 19/F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon Hong Kong Tel: (852) 2245-6100 Fax: (852) 2722-1369	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	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	***Technical Support***	***Sales Contact***
http://www.atmel.com/	avr@atmel.com	www.atmel.com/contacts

Literature Request
www.atmel.com/literature