AVR Freaks

DESIGN NOTE #017

AUTHOR: AVRFREAKS

KEYWORDS: INTERNAL RC OSCILLATOR, STK500, CALIBRATION, OSCCAL

Calibrating the Internal RC Oscillator Using the STK500

Introduction

Some AVR devices have the option of running from an internal RC Oscillator. This allows cutting down on component cost and even freeing up the XTAL pins for other use. The traditional drawbacks for internal RC Oscillators are low accuracy and stability.

However, in applications with medium or low requirements for frequency accuracy, it makes sense to use an internal RC Oscillator instead of an expensive external clock option.

The frequency of an RC Oscillator is determined by the values of the R and C. The geometries of the internal resistor and capacitor are very small, so even very small differences in process parameters and location on the wafer itself will create differences in these values. The result is therefore that the RC Oscillator frequency will vary from device to device. Calibrating the internal RC Oscillator is a way to compensate for process variations from one device to the other.

Most AVRs with internal RC Oscillator allow calibration of the internal RC Oscillator to achieve highest possible accuracy. For this reason, each and every AVR device with a tunable internal RC Oscillator run a calibration test and the correct calibration value is stored in each device before leaving the factory. By having a individual calibration value the frequency variation from device to device can be minimized. The frequency of the internal RC Oscillator of the AVR can be calibrated within 1% accuracy.

AVR Internal RC Oscillator Basics

The frequency of the internal RC Oscillator can be adjusted by writing different values to the OSCCAL Register. The OSCCAL Register is an 8-bit register used for adjusting/calibrating the internal RC Oscillator to a desired frequency. The value that must be written to the OSCCAL Register in order to get the correct frequency according to the data sheet is called the Oscillator Calibration Value (OSCVAL). OSCVAL is identified during testing in the factory and is stored in the device signature. The correct OSCVAL can therefore easily be identified without further testing of the devices, simply by reading the OSCVAL from the signature of the device using standard programming tools.

By increasing the OSCCAL Register value, the internal RC frequency will increase. In other words, writing 0 to the OSCCAL Register will give you the lowest possible frequency. Writing 255 to the register will give you the highest possible frequency. Note should be taken not to run the device to far off the “correct” frequency. In general it is not recommended to run the device more than 10% of the specified frequency. OSCVAL can vary from 0x00 to 0xFF and it will not always be possible to increase/decrease the OSCVAL Register value further.

AVR Internal RC Oscillators

When it comes to AVR's and internal clocking options, the AVR family can be divided into the following groups:

- Devices without internal RC clock option.
- Devices with non-calibratable internal RC Oscillator.
- Devices with manually calibratable internal RC.
- Devices automatically calibrated to 1 MHz.

The first group falls outside the scope of this document. The other groups do have internal RC, but implemented slightly different.

Non-calibratable Internal RC

This group contains only three devices: ATtiny11, AT90S1200, and AT90LS2343. These devices do have an internal RC Oscillator, but no OSCCAL Register. The actual frequency of the device may vary slightly from device to device. In many applications this does not matter. If the application requires a higher degree of accuracy, you need an external clock source or a device with a tunable internal RC.

Manually Calibratable Internal RC

This group contains five devices: ATtiny12, ATtiny15, ATtiny28, ATmega163, and ATmega323. These devices do have an OSCCAL Register. To tune them to correct frequency, this register should be loaded with the device specific OSCVAL. This value is unique for each device and stored inside the devices signature area. The next section will explain how to read and use this value to tune the device to correct frequency.

Devices Automatically Calibrated to 1MHz

In these automatically calibrated devices, the OSCVAL is loaded into OSCCAL at startup, and is "invisible" for the user. The OSCVAL can be read from the OSCCAL Register if needed.

This group contains the most recent AVR devices with internal RC clocking options. As a rule of thumb, all AVR devices where you can select the internal RC to be 1, 2, 4, or 8 MHz are of the automatically calibrated type. Be aware however that the OSCVAL automatically loaded into the OSCCAL Register is for 1 Mhz operation. If you want to run the device at 2, 4, or 8 MHz, the OSCCAL Register have to be loaded in run-time, as manually calibratable devices, to obtain the correct frequency. Actually the 1 MHz OSCVAL will calibrate the RC Oscillator quite close to other oscillator frequencies (2, 4, or 8 MHz), but to get it as close to the desired frequency as possible the correct OSCVAL should be loaded.

Manually Loading of the OSCCAL

If you are using a device without OSCCAL Register, this section does not apply, nor does it apply if you are using an AVR with automatically tuned RC Oscillator, and you just wish to use the supplied 1 MHz OSCVAL. If you on the other hand are using a device with manually calibrated RC Oscillator or you want to load an OSCVAL different from the on automatically loaded, then read on.

Reading the OSCCAL Value

The OSCCAL value is stored in the signature byte area of the device. This means that the OSCVAL is not directly accessible when the AVR is running since the contents of the signature bytes can only be accessed using a programming tool. Using a programming tool the OSCVAL can be read out and stored in a memory location accessible by the AVR while operating. This can be achieved by e.g., the STK500, AVRISP, or the JTAG ICE. The OSCVAL is then read from this memory location at run time and loaded into the OSCCAL Register to calibrate the internal RC Oscillator.

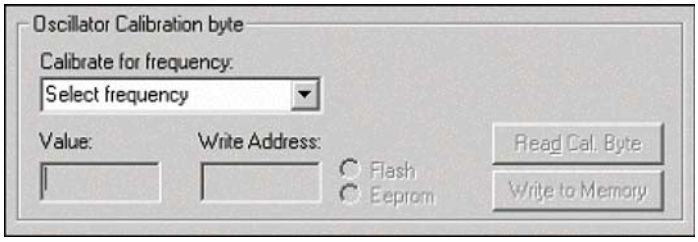
Reading out the OSCCAL Value Using STK500/AVRISP

To read out the OSCVAL value from a device, do the following:

1. Place the device in STK500 (take care to use correct socket and orientation).
2. Start AVR Studio®.
3. Start the STK500 software.
4. Select correct device.
5. Select the “Advanced” Tab.

Figure 1 shows the STK500 control dialog for reading and writing the OSCVAL from the signature.

Figure 1. STK500 Control Dialog – Grayed Out if Device Does not Support this Feature



It you have selected a device without calibratable internal RC, the “Oscillator Calibration Byte” options will be grayed out (see Figure 1). For devices where different Oscillator frequencies can be used, select desired frequency, and then press the “Read Cal Byte” to read this OSCVAL from the signature. It will appear in the “Value” text field. You have now read Oscillator calibration value and must decide where to store it – Flash or EEPROM.

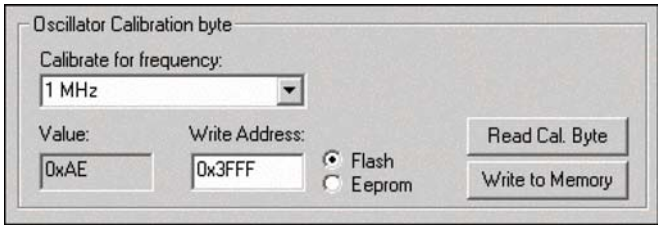
Writing OSCVAL to Memory

For the AVR to be able to update the OSCCAL Register, the OSCVAL must be accessible for the program. This can be achieved by writing the OSCCAL value to a memory location in Flash or EEPROM memory. Your program can then simply fetch the value from this location, and write it to the OSCCAL Register. Since the OSCVAL does not change, it is also possible to hardcode it into the code itself. This is often the most convenient way for prototype development. So where is the ideal location to store the OSCCAL value? In fact there are pros and cons for both Flash and EEPROM storage, so lets take a closer look at these options.

Storing the OSCVAL in Flash

Storing the OSCCAL value in Flash, note should be taken if using CRC check on the Program memory (the Flash). Since the OSCVAL may change from device to device, the CRC checksum must exclude the location of the OSCVAL to be common for all devices despite the variation of the OSCVAL.

Figure 2. STK500 Control Dialog



The “Write Address” must be given as “byte address”. Assume using an ATmega163 with 16k byte of memory. 16kB = 16,384 Bytes. This gives valid byte address range of 0 - 16,384b or 0 - 0x3FFF (hex). To store the OSCVAL in the last Flash location use “0x3FFF” as Write Address. See Figure 2 that shows the STK500 control dialog for reading OSCVAL from the signature and writing it to flash or eeprom.

Press “Write to Memory” button to execute the Flash write.

To load the value from flash into the OSCCAL Register the following code segment can be used:

```
LDI  ZL, low( FLASHEND * 2)
LDI  ZH, high( FLASHEND * 2)
LPM
OUT  OSCCAL, R0
```

The label “FLASHEND” is the word address to the last location in Flash (in this example it is 0x1FFF).

Storing the OSCVAL in EEPROM

Storing the OSCCAL value in EEPROM gives you the possibility of downloading a “fixed” code to the Flash, and then add an “individual” OSCVAL to a location in EEPROM. You are free to use whatever location you want, but as a rule of thumb do not use location 0 (0x00 hex). This is the first location to be trashed if the device is operating outside specifications (e.g., during Brown-out periods). The main EEPROM drawback is that it is possible to delete or modify an EEPROM location, so ensure that the EEPROM location holding the OSCVAL is not accessed by the code.

The “Write Address” must be given as “byte address”. Assume using an ATmega163 with 512 Byte of memory. This gives valid byte address range of 0 - 511. To store the OSCVAL in the last EEPROM location use “0x1FF” as Write Address.

Press “Write to Memory” button to execute the EEPROM write.

To load the value from EEPROM into the OSCCAL Register the following code segment can be used:

```
LDI  R16,  low( EEPROMEND )  ; Set up Address to EEPROM end
OUT  EEARL, R16
LDI  R16, high( EEPROMEND )
OUT  EEARH, R16
LDI  R16, ( 1 << EERE )      ; Give Read strobe
OUT  EECR, R16
IN   R16, EEDR               ; Write EEPROM data to OSCCAL
OUT  OSCCAL, R16
```

The label “EEPROMEND” is the byte address to the last location in EEPROM.

Hard-coding the OSCVAL Directly in Program

This is the sleekest approach, and thus the most difficult to pull of in a production environment. For prototype and development, this approach is common and basically means that you hard-code the OSCVAL into the assembly code (or C equivalent):

```
LDI  temp,   OSCVAL
OUT  OSCCAL, temp
```

OSCVAL is the OSCCAL value you get when pressing the “Read Cal Byte” in the STK500 Software.