DESIGN NOTE #018

AUTHOR: AVRFREAKS

KEYWORDS: INTERNAL RC OSCILLATOR, 32 KHZ CRYSTAL, CALIBRATION

Calibrating the Internal RC Oscillator

Introduction

This design note describes how to calibrate the internal RC Oscillator using an external 32.768 KHz crystal as input signal to an asynchronous Timer/Counter.

AVR devices with internal RC Oscillator and an OSCCAL Calibration Register can be calibrated very accurately by using an external signal with stable frequency. This calibration ca be useful if running the device at a temerature or supply voltage level different from what it is calibrated for by default (5V @ 25°C).

Overview

Note that external low-frequency crystals may require a stabilization period of up to one second after Power-up. Please consult the crystal data sheet for exact data.

The calibration is executed in a 5-cycle loop. In every loop a Counter is incremented, and the value of the 32 KHz Timer is read. After 10 ticks of the 32 KHz timer are completed, the frequency of the internal RC is compared with the Timer value. If the internal RC frequency is too low, a higher value is written to the OSCCAL Register. If the frequency is too high, a lower value is written. The loop is repeated 255 times to guarantee that OSCCAL gets the correct value.

The relation between the repeat count value and the frequency can be obtained from:

Loop cycles (5) x count value x RC period = external clk period x external ticks (10)

Calibrating the RC Oscillator to 1 MHz gives the following values:

5 x count value x 1 us(1 MHz) = 30.5 us (32.768 KHz) x 10

Which gives the count value of 61.

Below is a sample program in C language that carries out the calibration described above. Make sure to include the bit definitions and intrinsic instructions for the specific device.

```
//compiled using IAR EWA90 1.51b
#define FREQUENCY_1MHZ          61
#define FREQUENCY_1_8MHZ        112

void CalibrateInternalRc(void)
{
    unsigned char status = SREG;
    unsigned char cycles = 0xFF;
    unsigned char count;

    _CLI();
    do
    {
        count = 0;
        TIMSK &=~((1<<TOIE2)|(1<<OCIE2)); // Disable TC2 interrupt
        ASSR |= (1<<AS2);   // Set TC2 to be asynchronous from the CPU clock
                         // with a second external clock(32,768kHz)driving it
        TCNT2 = 0x00;
        TCCR2 = 0x01;       // Prescale the timer to be undivided from clock
source
        while (ASSR & 0x07)    // Wait until TC2 is updated
    ;
 do
        {
            count++;                    // Increment counter
        } while (TCNT2 < 10);           // Until 32KHz * 10 cycles = 305us
        if (count > FREQUENCY_1MHZ)
        {
            OSCCAL--;            // If count is more than 61 - decrease speed
        }
        if (count < FREQUENCY_1MHZ)   // 61 = 1MHz, 112.5 = 1.8432MHz
        {
            OSCCAL++;        // If count is less than 61 - increase speed
        }
    } while (--cycles);          // Calibrate for 255 cycles

    // Finished calibration, set up timer for 1 second interrupts

    TIMSK &=~((1<<TOIE2)|(1<<OCIE2)); // Disable TC2 interrupt
    TCNT2 = 0x00;
    ASSR |= (1<<AS2);      // Set TC2 to be asynchronous from the CPU clock
                        // with a second external clock(32,768kHz)driving it.
    TIMSK |= (1<<TOIE2);   // Set 8-bit TC2 Overflow Interrupt Enable
    TCCR2 = 0x05; // Prescale the timer to be clock source/128 to make it
       // exactly 1 second for every overflow to occur
    while (ASSR&0x07)      // Wait until TC2 is updated
    ;
/*  DDRD = 0x01;         // OPTIONAL debug
    PORTD |= 0x01; // Use this pulse to measure frequency on PD0, 2 us pulse
    PORTD &= ~0x01;
    PORTD |= 0x01;
*/  SREG = status;
}
```