Application Note 51

Title: **How to Trim RC oscillator on EM68xx**

Product Family: **8-bit Microcontroller**

Part Number: EM6812
Keywords: Microcontroller, RC oscillator, Trimming
Date: January 1, 2004

Introduction

This application note describes the implementation of a software algorithm used to trim the RC oscillator frequency.
This FLL (Frequency Locked Loop) algorithm is using a very precise frequency (Xtal oscillator) as reference to trim precisely the RC oscillator frequency.

EM68xx clock management overview

EM68xx include high (RC or external) and low (Xtal or external) frequency clock domain.

- RC High frequency clock domain provide two frequency range (10MHz – 1MHz) + RC trimming ±40% (8bit register) – RC divider (1,2,4,8)
- Xtal frequency clock domain is 32768Hz Xtal oscillator

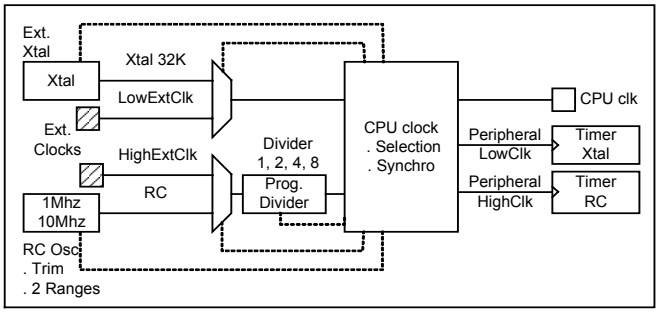

Figure 1: Clock management diagram

Principle

This Application Note is targetting a specific RC trimming frequency : 1MHz clock (500 Kinstruction/s). But the principle can be applied for any RC frequency available.
The reference Xtal is used to provide a precise and steady time base (frequency drift is very low for temperature variations).
To improve the accuracy of the RC setting, the measurement is done over several Xtal clock cycles (NbXtal).
The technique used here is start in parallel two Timers. One is running on RC clock, the other on the Xtal clock.
As soon as the Xtal time base unit is achieved, the Timers are stopped and the RC Timer value is tested. Depending on the remaining RC Timer value tested, the RC trimming register value is incremented or decremented until the goal is achieved.

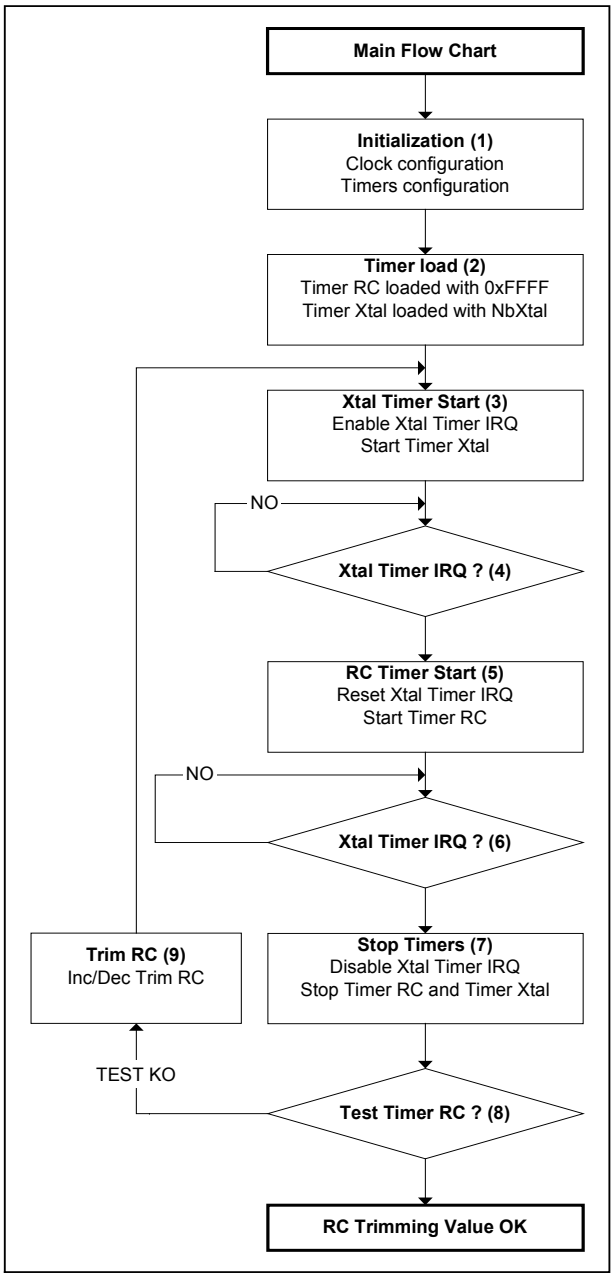

Figure 2: Flow-Chart diagram

Refer to the definition and details next page

Flow-Chart comments

The following comments handle the trimming of 1 MHz RC frequency.

❶ Initialization
It is necessary to define first the most suitable frequency range in order to get the highest trim area. In our case, the most evident setting is :
Default Clock RC = 1 MHz range (Freq. range = 0) . RC divider = 1 (RCDiv = 00)
Two Timers will be used :

- Timer RC (16 bit)
  Timer 1 (LSB) and Timer 2 (MSB) cascaded
  Input clock selected is CPU/2 = RC/2 (Refer to the datasheet for details)
  Zero Stop Mode
- Timer Xtal (16 bit)
  Timer 3 (LSB) and Timer 4 (MSB) cascaded
  Input clock selected is Xtal = 32768 Hz
  Auto reload Mode

❷ Timer load
The Timer Xtal is loaded to count Nbxtal cycles.
The value loaded is (Nbxtal –1). Refer to Figure 3.
The Timer RC is loaded with 0xFFFF.
The expected RC Timer remaining value must be evaluated.
The number of RC Timer cycles ExpectedRC corresponding to Nbxtal (64) cycles is given by :
ExpectedRC = 0xFFFF – (((FrequencyRC /2) * Nbxtal) / Frequency Xtal) = 0xFC2F

❸ Xtal Timer Start
Interrupt from Xtal Timer are enabled. Then, the Xtal Timer can be started. CPU is put in halt mode.

❹ Xtal Timer IRQ
As soon as the interrupt comes, the interrupt flag is cleared.

❺ RC Timer start
RC Timer is started. In the meantime Xtal Timer is loaded again and continue the countdown (auto reload mode). This step is used to set the synchronization between RC Timer and Xtal Timer.
Interrupt from Xtal Timer are still enabled. CPU is put in halt mode.

❻ Xtal Timer IRQ
As soon as the interrupt comes, the interrupt flag is cleared. This interupt routine is exactly similar to ❹ Xtal Timer IRQ.

❼ Stop Timers
Timers Xtal and RC are immediately stopped. Interrupt from Xtal Timer are disabled.

❽ Test Timer RC
The remaining value (TimerRCValue) of RC Timer is compared to the ExpectedRC value.
If TimerRCValue equals ExpectedRC the RC frequency is correctly set (no more Trim).
If TimerRCValue does not equal ExpectedRC. Trim register must be modified.
TimerRCValue > ExpectedRC => RC frequency too low - increment RC Trim register
TimerRCValue < ExpectedRC => RC frequency too high - decrement RC Trim register

❾ Trim RC
Modify Trim register. Restart from Step 3.

Notes

NbXtal choice : NbXtal < 2^{16} * ((Frequency Xtal) / (Frequency RC / 2))
The software routine described here could also be modified or improved :

- Implementing automatic RCDiv and FreqRange choice by frequency measurement.
- Dichotomy or other methods could be used to converge to the final RC Trim value.
- Use prescalers instead of Timers for Xtal time base to reduce Timer uses.

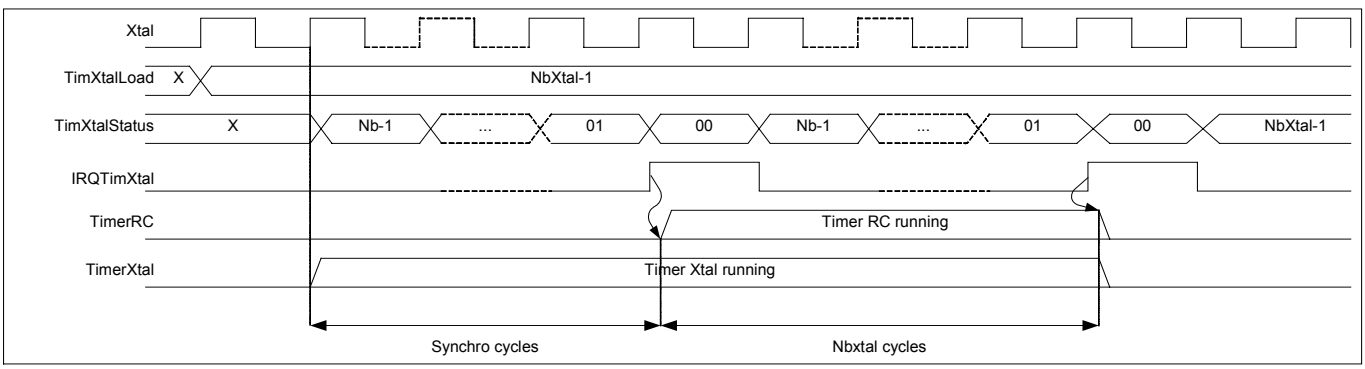

Figure 3: Timing diagram