
TN8000.09

Technical Note

D.F.L.L Digital Frequency Locked Loop

Author : Miguel Luis
For further information, please contact

XEMICS SA
E mail: info@xemics.com
Web: www.xemics.com

INTRODUCTION

A Digital Frequency Locked Loop (DFLL) is an algorithm used to precisely set an RC oscillator frequency using a precise clock source as reference.

The X8000 family needs these kind of algorithms when precise frequencies higher than 32 kHz are needed. A typical example where a DFLL is needed is an application that uses UART communications with baud rates higher than 2400 bauds.

To generate baud rates higher than 2400 bauds the RC oscillator is used, but this oscillator is not very precise: it depends on technology tolerances. For the same trimming value on two different chips, the resulting frequency can have a difference of $\pm 50\%$. Whereas a precise baud rate generator is needed to implement UART communications, having a precision of typically $\pm 5\%$ or less.

The XE8000 family has a built in XTAL oscillator that has a very high level of precision, depending on the quartz crystal. This will be the reference for the DFLL algorithm to trim the RC oscillator.

DFLL ALGORITHM PRINCIPLE

The algorithm described in this document is a dichotomy algorithm.

XE8000 OSCILLATORS

The XE8000 microcontroller family has two possible main clock sources:

- One XTAL oscillator (32768 Hz)
- One RC oscillator. This oscillator has two ranges :
 - 100 kHz to 1 MHz and 1 MHz to 10 MHz for XE88LC01, XE88LC03, XE88LC05.
 - 50 kHz to 0.5 MHz and 0.5 MHz to 5 MHz for XE88LC02, XE88LC06A.

XTAL Oscillator

When starting the XTAL oscillator, the oscillator generates a frequency that is stabilized after 32768 clock cycles. This oscillator is very stable even if there are temperature variations (XTAL dependent).

RC Oscillator

The RC oscillator is the default main clock for the XE8000 family. This oscillator starts very quickly as it only needs 8 clock cycles to obtain a stabilized frequency. However, this oscillator is temperature dependent and the default frequency is not precise.

The XE8000 RC oscillator trimming resolution is typically 1.4%.

The XE8000 RC oscillator is set up by a trimming word of 11 bits.

- **1 bit** selects the range of the RC oscillator: When this bit is 0 we have the low frequency range.
- **4 bits** for a coarse adjustment of frequency. With these four bits we can adjust 16 different frequencies.
- **6 bits** for a fine adjustment of the frequency chosen approximately by the 4 coarse bits and the range bit.

FREQUENCY MEASUREMENT

XE8000 Counters

The XE8000 family has four general 8 bit up/down counters that can be cascaded to form two 16 bits up/down counters.

For the DFLL frequency measurement the counters are used in 16 bit mode. Counter AB takes its clock source from the RC oscillator (RC Counter). Counter CD takes its clock source from the XTAL oscillator (XTAL Counter).

Figure 1 shows how the frequency measurement is done. The number of RC clock cycles M is counted during a fixed period of N XTAL cycles. The trimming bits of the RC oscillator are chosen so that the ratio between M and N corresponds to the ratio between the RC oscillator target frequency and the crystal oscillator frequency. This means:

$$M = N \cdot \frac{Freq_{RC}}{Freq_{XTAL}}$$

Equation 1

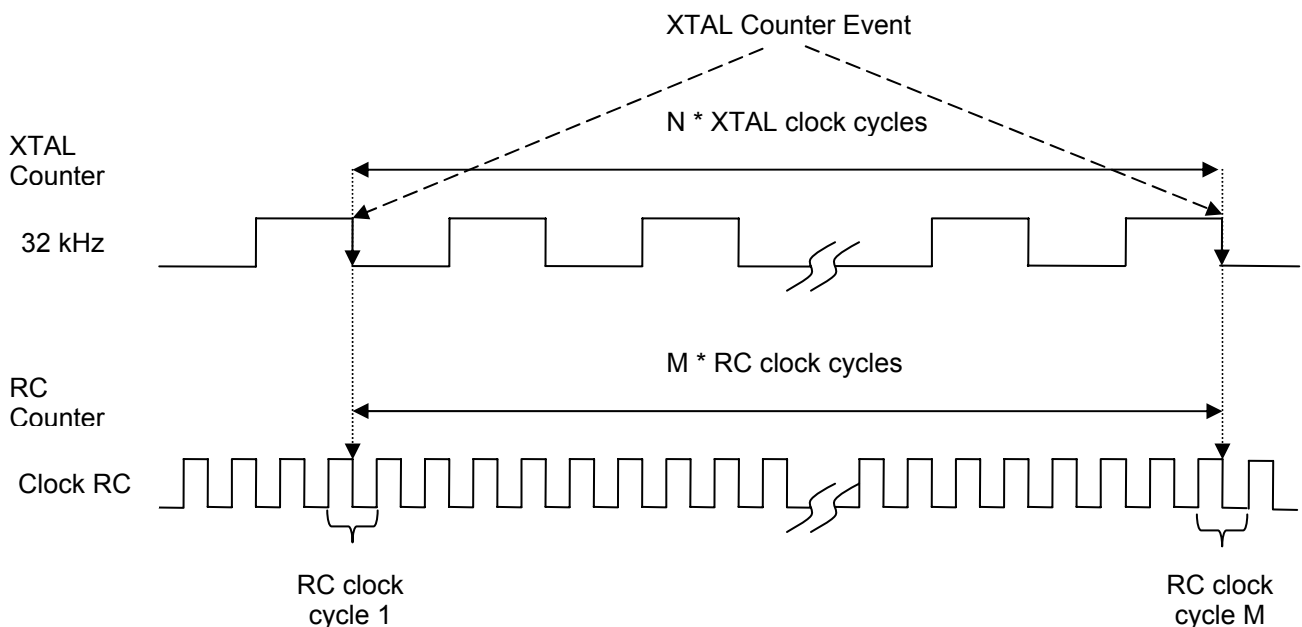


Figure 1

N value is chosen so that $M < 2^{16}$ in order to avoid counter overflow. Using Equation 1, this translates to:

$$N < 2^{16} \cdot \frac{XTAL_{Freq}}{RC_{Freq}}$$

Equation 2

Precision

The DFLL precision depends on the frequency measurement precision.

The frequency measurement uncertainty is 2 RC clock cycles as shown in the Figure 1. The start of the first XTAL clock cycle is anywhere during RC clock cycle 1 and the end of the Nth XTAL clock cycle is anywhere during the Mth RC clock cycle. Therefore, the measurement precision is the uncertainty of 2 RC cycles compared to the total number of counted RC cycles:

$$Precision_{acquisition} = \frac{2}{M}$$

Equation 3

Using Equation 1, this gives:

$$Precision_{acquisition} = \frac{2 \cdot Freq_{XTAL}}{N \cdot Freq_{RC}}$$

Equation 4

The final DFLL precision can of course not be more precise than half the RC oscillator frequency trimming step. The XTAL error is not considered because it is very small.

$$Precision_{DFLL} = \text{MAX} \left(\frac{RC_{Step}}{2}, Precision_{acquisition} \right) = \text{MAX} \left(\frac{1.4\%}{2}, \frac{2 \cdot Freq_{XTAL}}{N \cdot Freq_{RC}} \right)$$

Equation 5

Execution time

For each RC oscillator trimming bit, $2 \cdot N$ XTAL cycles are used (Equation 6), one time N for the actual measurement and one time N to do the calculations and rearm the counters.

$$Time_{Acquisition} = \frac{2 \cdot N}{Freq_{XTAL}}$$

Equation 6

The algorithm repeats this for each trimming bit of the RC oscillator, i.e. 11 times.

$$Time_{DFLL} = \frac{11 \cdot 2 \cdot N}{Freq_{XTAL}}$$

Equation 7

Examples

In the following examples we have chosen a multiple of 2 for the N value to simplify the division calculation by the microcontroller processor.

Measurement parameters:

N = 512
XTAL = 32768 Hz
RC = 1 MHz

$$Precision_{acquisition} = \frac{2 \cdot 32768}{512 \cdot 1 \cdot 10^6} = 128 \cdot 10^{-6}$$

Equation 8

Measurement parameters:

N = 512
XTAL = 32768 Hz
RC = 100 kHz

$$Precision_{acquisition} = \frac{2 \cdot 32768}{512 \cdot 100 \cdot 10^3} = 1.28 \cdot 10^{-3}$$

Equation 9

Measurement parameters:

N = 64
XTAL = 32768 Hz
RC = 1 MHz

$$Precision_{acquisition} = \frac{2 \cdot 32768}{64 \cdot 1 \cdot 10^6} = 1.03 \cdot 10^{-3}$$

Equation 10

Measurement parameters:

N = 64
XTAL = 32768 Hz
RC = 100 kHz

$$Precision_{acquisition} = \frac{2 \cdot 32768}{512 \cdot 100 \cdot 10^3} = 10.3 \cdot 10^{-3}$$

Equation 11

Measurement Algorithm

Figure 2 shows the flow chart to implement the frequency measurement function.

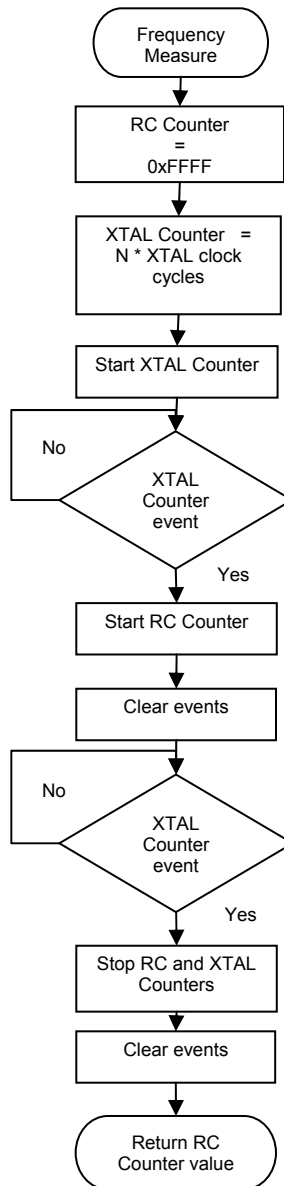


Figure 2

DICHOTOMY ALGORITHM

The following lines describe the DFLL algorithm.

1. Sets a variable with the bit position to be written to the RC trimming registers. The algorithm starts by setting the eleventh bit (MSB) of bit position variable to 1. The algorithm will run until the bit position variable is 0.
2. The algorithm writes the RC trimming registers with the bit position variable value.
3. A small delay is generated to let the RC frequency stabilize.
4. A frequency measurement is done.
5. The frequency measurement result is then compared to the frequency that DFLL should reach.
6. If the result is smaller than the frequency that DFLL should reach then the bit set in step 2 is erased. If the result is greater the bit set in step 2 is kept.
7. The bit position is shifted right by 1.
8. Jumps to step 2 until bit position variable is 0
9. Difference between the measured frequency and the wanted frequency is calculated.
10. If difference is positive the RC oscillator fine trimming value is incremented, otherwise the value is decremented.
11. A small delay is generated to let the RC frequency stabilize.
12. A frequency measurement is done.
13. If the new absolute Difference between the frequencies is greater than the old absolute difference then we set the Fine trimming value to the old value.

While running the DFLL algorithm, the RC frequency can go further than the maximum allowed frequency for the processor and Flash memory. To avoid this problem the CoolRISC FREQ instruction is used. At the beginning of the algorithm we divide the processor frequency at least by 2 and at the end we remove the division.

Figure 3 shows the DFLL algorithm flow chart.

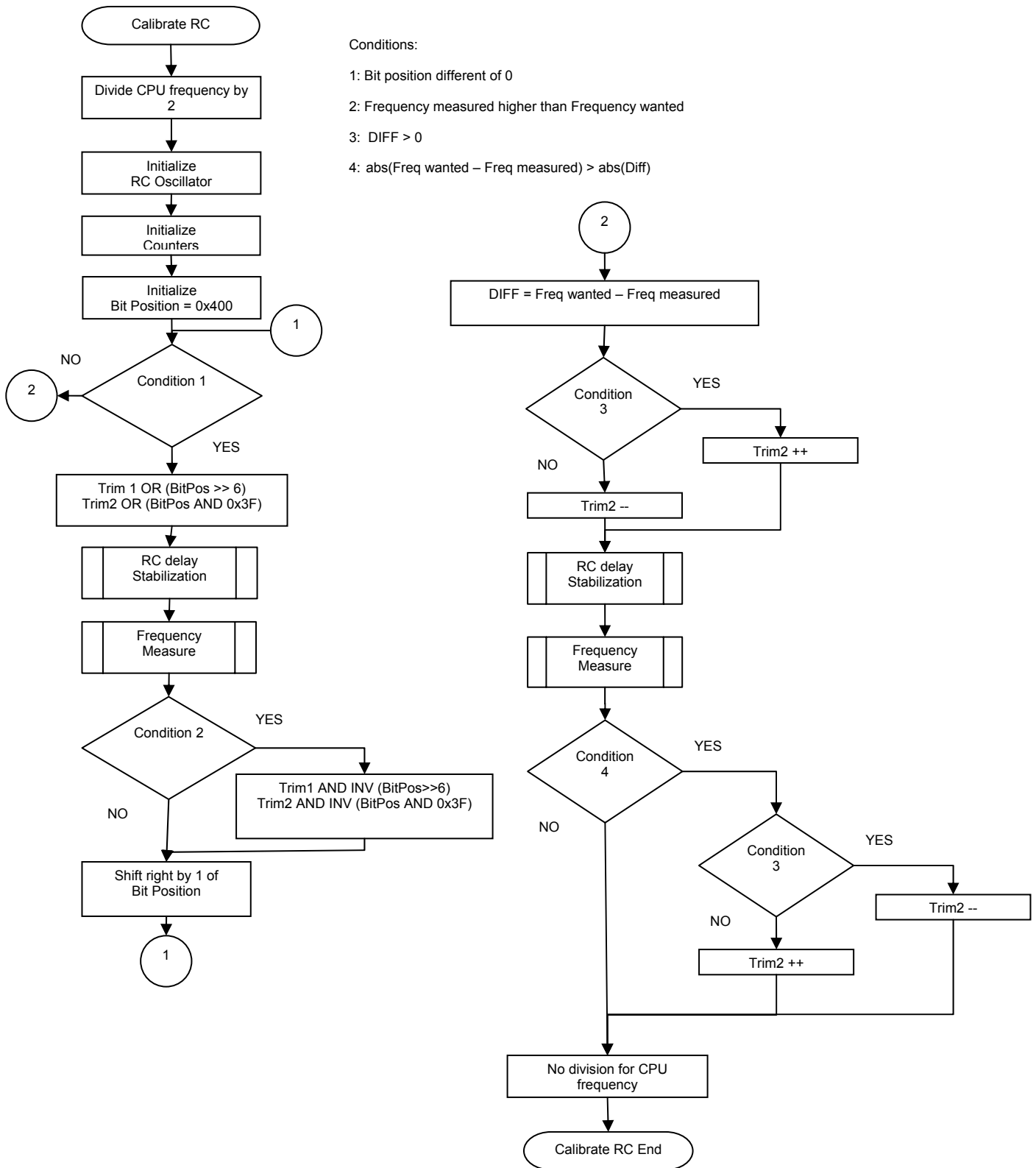


Figure 3

CONCLUSION

The algorithm (frequency measurement and Dichotomy) presented in this document is one of the ways to implement a DFLL.

Indeed the frequency measurement can be done in different ways, for example by using the counter's capture function or instead of using a 16 bits counter for the XTAL oscillator one can use the 1 Hz or 128 Hz event. The Dichotomy is not the only possible algorithm. For example a ramp approximation algorithm can also be implemented.

The DFLL described in this document should only be used for first time RC oscillator initialization. Indeed this is an algorithm that can take long time to reach the targeted frequency, depending on the acquisition time of the frequency measurement.

The RC oscillator frequency jumps around quite a lot during the loop locking. Once the loop is locked, it might be better to implement a new function that adapts to slow variations by incrementing/decrementing the RC oscillator fine trimming code.

If the target application is designed to be in an environment where the temperature changes are very high, one needs to implement a function that adjusts the frequency while the target application is running. If the target application does something only from time to time, one can call the algorithm described in this document to adjust the RC frequency.