AN1324
APPLICATION NOTE

CALIBRATING THE RC OSCILLATOR OF THE ST7FLITE0 MCU USING THE MAINS

INTRODUCTION

The ST7FLITE0 microcontroller contains an internal RC oscillator which can be trimmed to a specific frequency with an accuracy of 1%. The oscillator frequency has to be calibrated by software using the RCCR (RC Control Register). The value entered in the RCCR will switch on a corresponding number of resistors that will modify the oscillator frequency. Whenever the ST7FLITE0 microcontroller is reset, the RCCR is restored to its default value (FFh), so each time the device is reset, you have to load the calibration value in the RCCR. There are predefined calibration values stored in memory (refer to section 7.1 in the ST7FLITE0 datasheet) You can load one of these values in the RCCR if one of the operating conditions matches that in your application. Otherwise, you can define your own value, store it in EEPROM or any non-volatile memory and load it in the RCCR register after each reset. However, if any of the external conditions (temperature or voltage, for instance) change too drastically, the stored value may no longer produce the required 1% accuracy. One solution is to recalculate the RCCR value after each reset, based on an external reference.

The purpose of this application note is to present a software solution using the frequency of the European standard mains (220V/50Hz) as a timebase to adjust the internal RC oscillator of the ST7FLITE0 to 1 MHz (1%). The same approach can also be used for the US mains standard (110V/60Hz).

The basic software takes less than 160 ms to calibrate the oscillator and uses less than 90 bytes of program memory and five bytes of RAM for its simplest version. These RAM bytes can be freed for other purposes when the calibration is done. Another example using averages is given in this application note. This can be useful with noisy mains

This application note also contains the diagram of a low cost circuit which converts the mains into a 5 volt power supply and protects the microcontroller from overcurrent on the input connected to the mains.

Rev. 2

1 CALIBRATION SOFTWARE

1.1 SOFTWARE PRINCIPLE

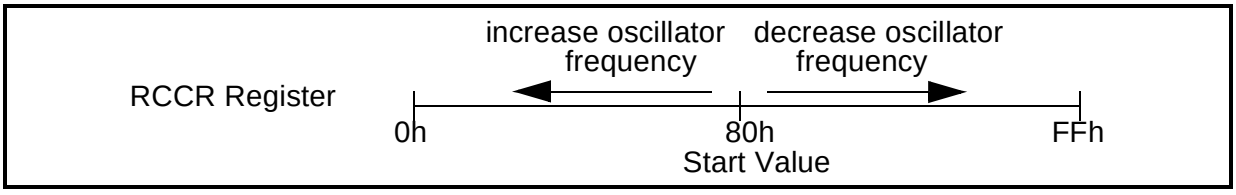
The software algorithm, described in the following flowchart (see Figure 3), uses the mains frequency as a timebase. This timebase allows the microcontroller to test if the RC oscillator frequency is above or below 1 MHz and repeatedly transforms it by dichotomous analysis so that in 7 iterations the RCCR is set to the optimum value.

As the timer speed depends on the RC oscillator frequency, it is easy to determine if the oscillator is too fast or too slow. The counted value can be obtained by the following equation:

$$countedvalue = \frac{f_{cpu}}{32 \times f_{mains}}$$

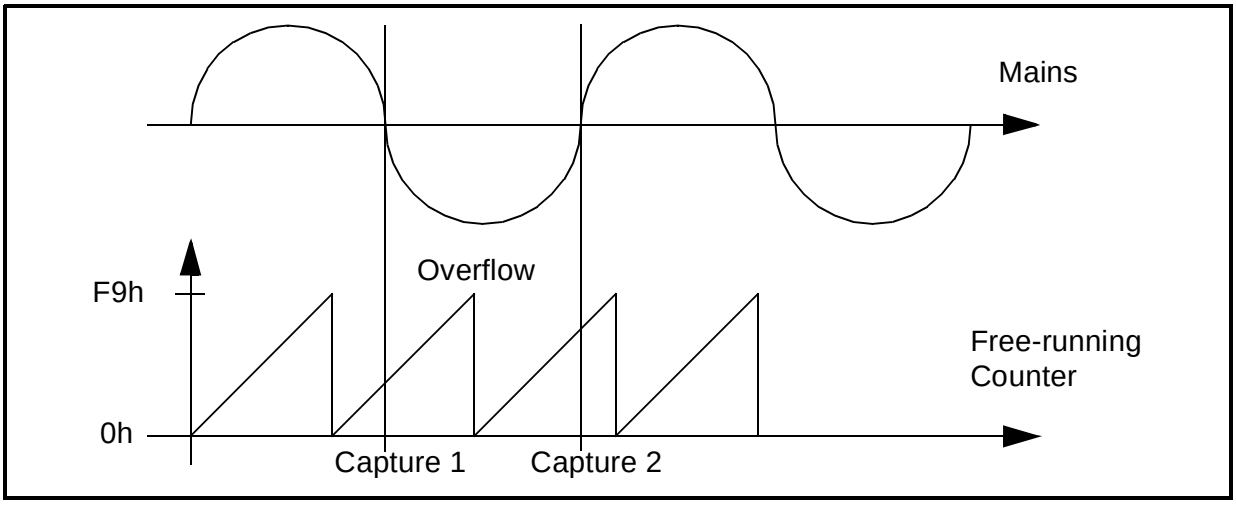
Since the frequency of the counter is the frequency of the oscillator divided by 32, if the oscillator is at 1 MHz, the result of the count between two edges (which have a 10 ms interval), is 138h for the European standard (220V/50Hz). For the US standard (110V/60Hz) the right value is 104h. Since the goal of the software is to set the RC oscillator frequency to 1 MHz it means obtaining 138h as the result of the count. So if the result of the count is greater than 138h, it means that the frequency is too high so the program increases the value of RCCR in order to decrease the RC oscillator frequency. And if the result is less than 138h, the RCCR is decreased in order to increase the RC oscillator frequency.

Figure 1. Dichotomous Analysis of RCCR Value



The RCCR register is set to 80h initially by the program, then the dichotomization starts by adding or subtracting 40h and after each iteration the result is divided by two, so that after 7 iterations the value of RCCR is set with an accuracy of one bit.

Figure 2. Using the Timer Input Capture to Measure the Mains Frequency



To measure the frequency, the software uses the Lite Timer input capture (LTIC) so that on each edge of the mains the value of the free running counter is stored as shown in Figure 2. Then the microcontroller calculates the elapsed time between the two edges of the mains. This time is given by the following equation:

$$time = nbover \times F9h + capture2 - capture1$$

where nbover represents the number of counter overflows during the measurement, capture 1 and capture 2 are the values captured on the free running counter when an edge occurs on the mains and F9h is the overflow value of the free running counter.

If the RC oscillator frequency is equal to 1 MHz, the result time will be 138h for European standard (220V/50Hz) or 104h for US standard mains (110V/60Hz), so these are the reference values.

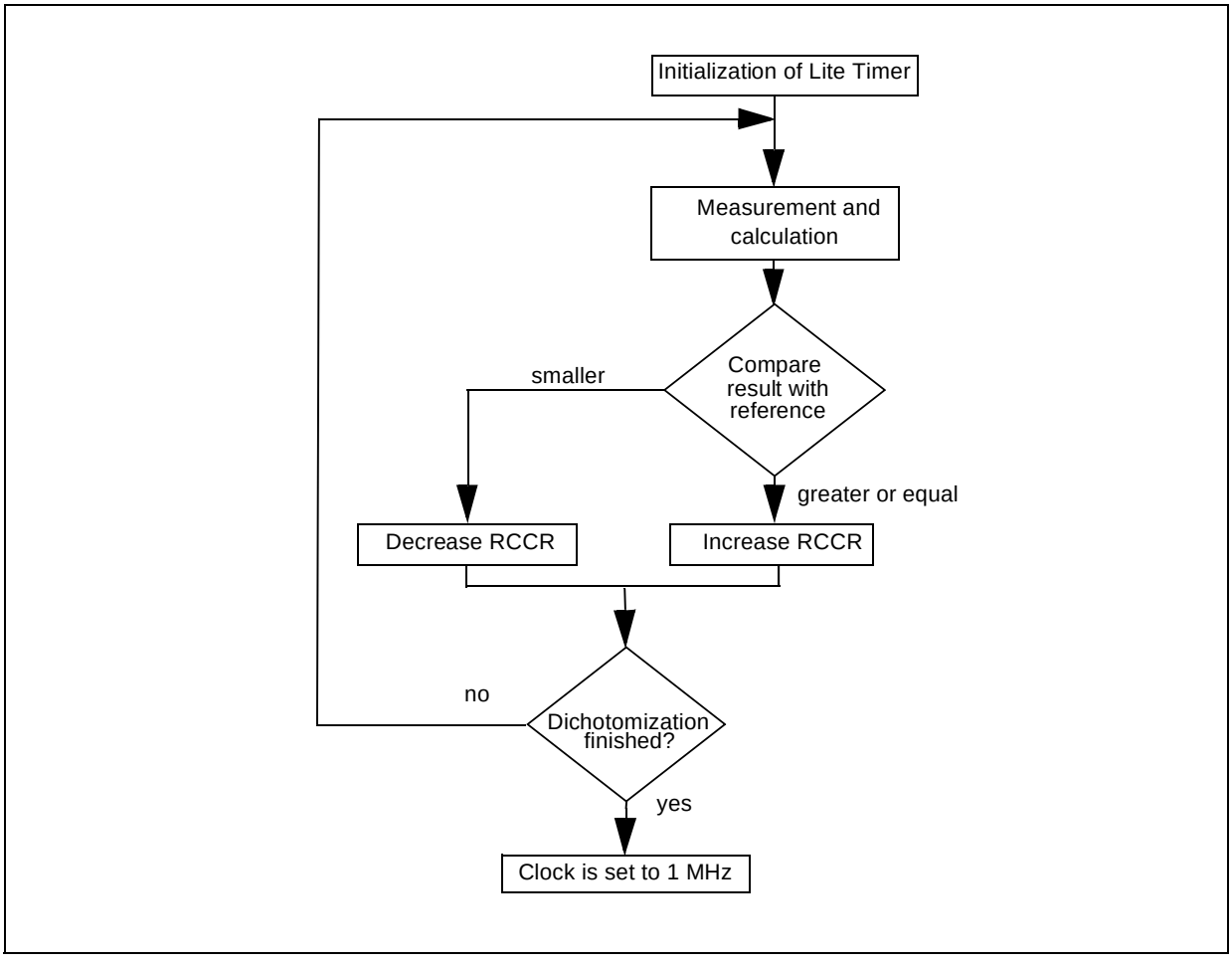
This measurement result is compared to the reference value and, depending on the result of the comparison, the microcontroller adds to or subtracts from the current RCCR value.

1.2 BASIC VERSION

In this version the measurement is done only once for each dichotomization step. This allows the calibration software to be light and fast. It requires only 90 bytes of program memory and 5 bytes of RAM during calibration. The calibration takes less than 160 ms to be completed

The software works as shown in the following flowchart. The assembly code and a more detailed flowchart can be found in Section 4.

Figure 3. Basic software flowchart



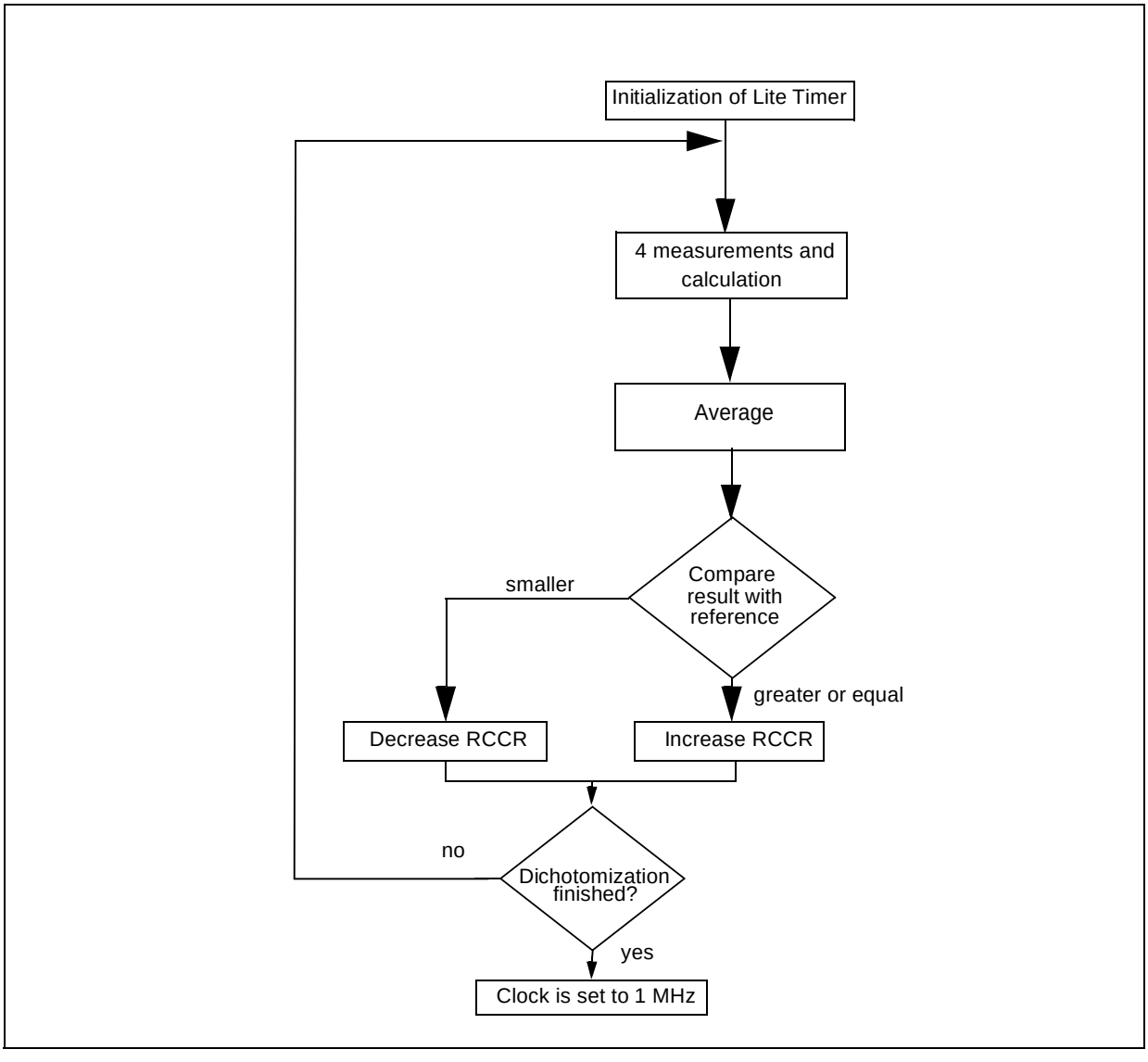
1.3 AVERAGE VERSION

This version uses the method described in Section 1.1 except it performs four measurements and uses their average for each dichotomization step. It is useful when the mains is noisy. For instance, when a motor starts it generates a tension pick and this can be considered as a mains edge.

This version is safer than the basic one but it requires more resources. It uses 136 bytes of program memory and 11 bytes of RAM during calibration. The calibration takes less than 560 ms to be completed.

The average version works as shown in the following flowchart. The assembly code can be found in Section 4.

Figure 4. Average software flowchart



2 POWER SUPPLY AND TIMEBASE DELIVERY CIRCUIT

The following figures show circuits which will provide 5V DC to the ST7LITE0 and protect the input capture from overcurrent. If no power supply is needed, the only component to keep is the resistor on the LTIC input, which is mainly to protect from overcurrent.

2.1 BASIC CIRCUIT

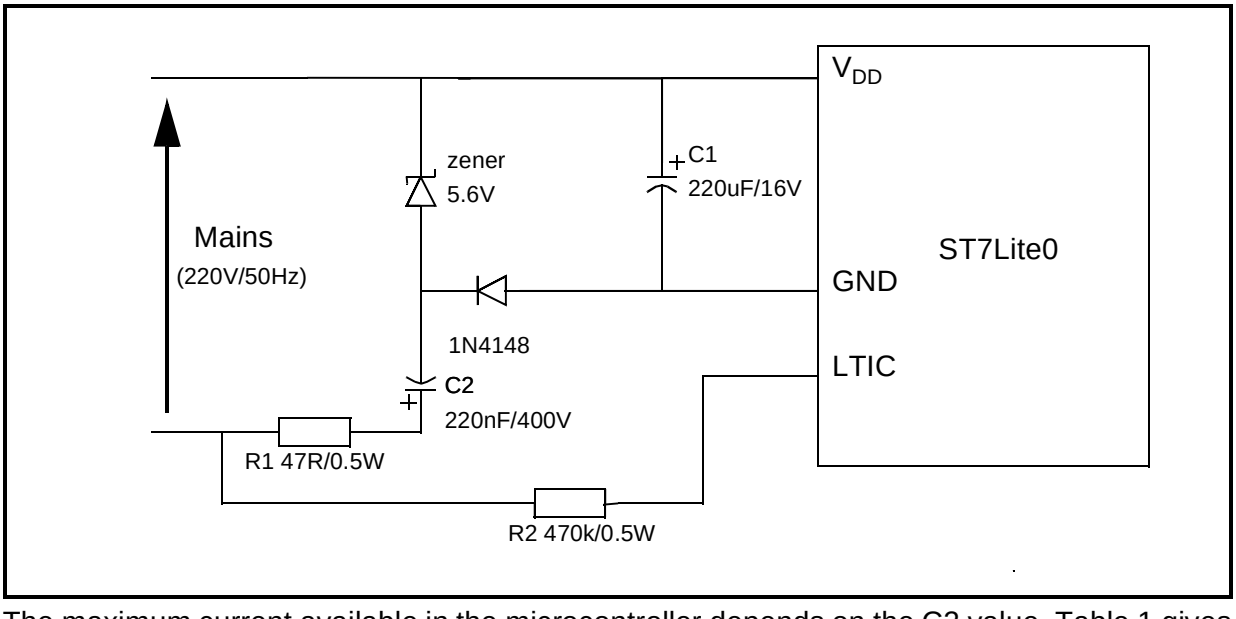
This circuit contains a capacitive power supply which converts the 220V/50Hz of the mains, as well as the 110V/60Hz of the US mains, into 5V DC.

Warning: be aware that this kind of power supply can't be used if there are big current variations.

It also inputs 220V/50Hz to the Lite Timer Input Capture pin (LTIC/PA0) protected by resistor R2.

The incoming alternating signal on the LTIC input pin is 220V/50Hz. Because of the clamping diode on the input of the ST7FLITE0, the input signal can be considered as a 0-5V square signal.

Figure 5. Power supply and timebase delivery circuit diagram



The maximum current available in the microcontroller depends on the C2 value. Table 1 gives the maximum average current versus the capacitor value. The average current follows the equation below:

$$Imax = Vmax \cdot 2 \cdot f \cdot C$$

In the case above, C2 is equal to 220nF so the available current is limited to 4.9 mA in the case of a European mains. To have the same current levels in the case of the US mains (110V/60Hz), C2 must be multiplied by two. A 440nF capacitor will limit the current to 4.9 mA.

For the US standard, R2 must be divided by two in order not to limit the current too much on the LTIC input. A 220k resistor is enough in this case.

Table 1. Maximum MCU Current

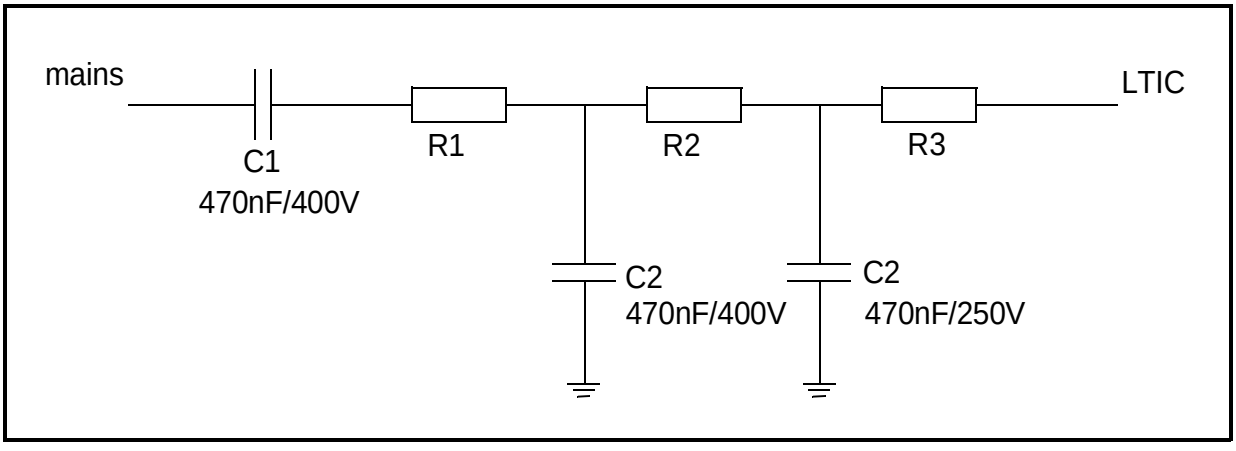
CAPACITOR C2	MAXIMUM CURRENT
220nF	4.9mA
330nF	7.3mA
470nF	10.4mA
680nF	15mA
1uF	22.1mA

2.2 HARDWARE PROTECTION

To prevent bad measurements due to noisy mains, a filter can be added between the mains and the input capture of the LITE0. The following figure shows one example of a filter. This filter is a pass band centered on the mains frequency in order to reject all frequency which could be understood by the microcontroller as a mains edge.

Be aware that this is just a second order filter and that this may not be enough if the mains is really noisy. Any kind of filter can be added on the LTIC.

Figure 6. Band pass filter



The pass band filter above must be tuned to mains frequency. The value of the resistors for this filter is given in the table below.

Table 2. resistors values

resistors	50Hz/220V	60Hz/110V
R1	6.8K/0.5W	5.6K/0.5W
R2	6.8K/0.5W	5.6K/0.5W
R3	470K/0.5W	220K/0.5W

3 CONCLUSION

This system allows you to have a power supply for the microcontroller and an auto adjustable clock set to 1MHz with an accuracy of 1% whatever the external conditions.

This solution also offers the advantage of being less expensive than a solution with a transformer and requires less space.

It requires a small amount of space in program memory (less than 90 bytes) in its smallest version.

4 SOFTWARE EXAMPLES

A zip file attached to this application note contains the complete software of this calibration method.

4.1 SINGLE ALTERNANCE

This version perform only one count between two edges and changes the value of the RCCR according to this measurement. This can lead to bad tuning if there is noise on the reference signal.

4.1.1 main program

```
;All the bytes from locations 80h to 85h are used by this software to store values or as control
registers but they can be reused safely after the clock has been set.

;dichotomy value
.value                  equ     $81             ;this byte contains the value which will be added
or subtracted to/from the RCCR last value at the end of each round
;capture values
.capture1               equ     $82
.capture2               equ     $83     ;these two bytes contain the two values of the counter
captured on the edge of the mains, they are used to calculate the time elapsed between the two
edges
;number of overflows
.nbover                 equ     $84     ;this byte contains the number of counter overflows
during the measurement
;control register
.cr                     equ     $85     ;this byte is used as a control register for the measure-
ment. Its bits allow or not the interrupts and show which step of the count is the current one.
.strtstp                equ     1       ;this is set to start the count and reset to stop it
.overflow               equ     2       ;this bit is set when the first capture has occurred. It
allows the overflows to be counted

.main

        bset            MCCSR, #1       ;output clock enable. You can remove this line if you do
not want to check the clock

        ld              A, #$80         ;value containing the value which will be
        ld              value, A        ;add or subs to/from RCCR during the dichotomy
        ld              RCCR, A         ;RCCR is set to the middle of its range of value

next    clr             nbover          ;clear the byte containing the number of timer overflow
        clr             cr              ;clear the byte use as control register for the count
        ld              A, LTICR        ;clear the ICF bit

        rim                             ;interrupts enable
        bset            LTCSR, #7       ;enable input capture interrupt
        bset            cr,#strtstp     ;set the start-stop bit of cr: count can start
count   btjt            cr, #strtstp, count; wait for the end of count
        clr             LTCSR           ;lite timer interrupts disable

        srl             value           ;dichotomy value divided by 2

        ld              A,#$F9          ;these lines calculate this equation:
        ld              X,nbover        ;
        mul             X,A             ;(nbover*$F9)+ capture2 - capture1
        add             A, capture2     ;
```

```
        jrnc            nocarry       ;this equation is calculated with 16 bits
        inc             X             ;
nocarry sub             A, capture1   ;MSB are in register X
        jrnc            noneg         ;
        dec             X             ;and LSB in register A.

noneg   cp              X, #$01       ;if mains frequency is 50Hz the reference value is $138
        jrmi            minus         ;if it is 60Hz the reference is $104.the program first
        jreq            compare       ;compares MSB with $01 and then compare LSB with
        jp              plus          ;$38 for 50Hz and $04 for 60Hz. if the calculated
compare cp              A, #$38       ;value is smaller than the reference the program jump to
        jrmi            minus         ;minus to decrease RCCR else it increase RCCR

plus    ld              A, RCCR
        add             A, value      ;add value if counted value is greater than ref
        jp              new

minus   ld              A, RCCR
        sub             A, value      ;subtract value if Y is smaller

new     ld              RCCR,A        ;enter the new value in RCCR

        btjf            value, #0, next;stop after 7 rounds

loop    jp              loop
```

4.1.2 input capture interrupt

```
        ld              A, LTICR              ;load captured value in A
        btjt            LTCSR, #4, finish     ;test if it is first or second capture
        bset            LTCSR, #4             ;allow timebase interrupt in order to count the
number of overflows
        ld              capture1, A           ;captured value is stored in capture1
        jp              endit1

finish  ld              capture2, A           ;if it is the second capture, captured value is
stored in capture2
        clr             cr                    ;clear cr to end the count
endit1  iret
```

4.1.3 timebase interrupt

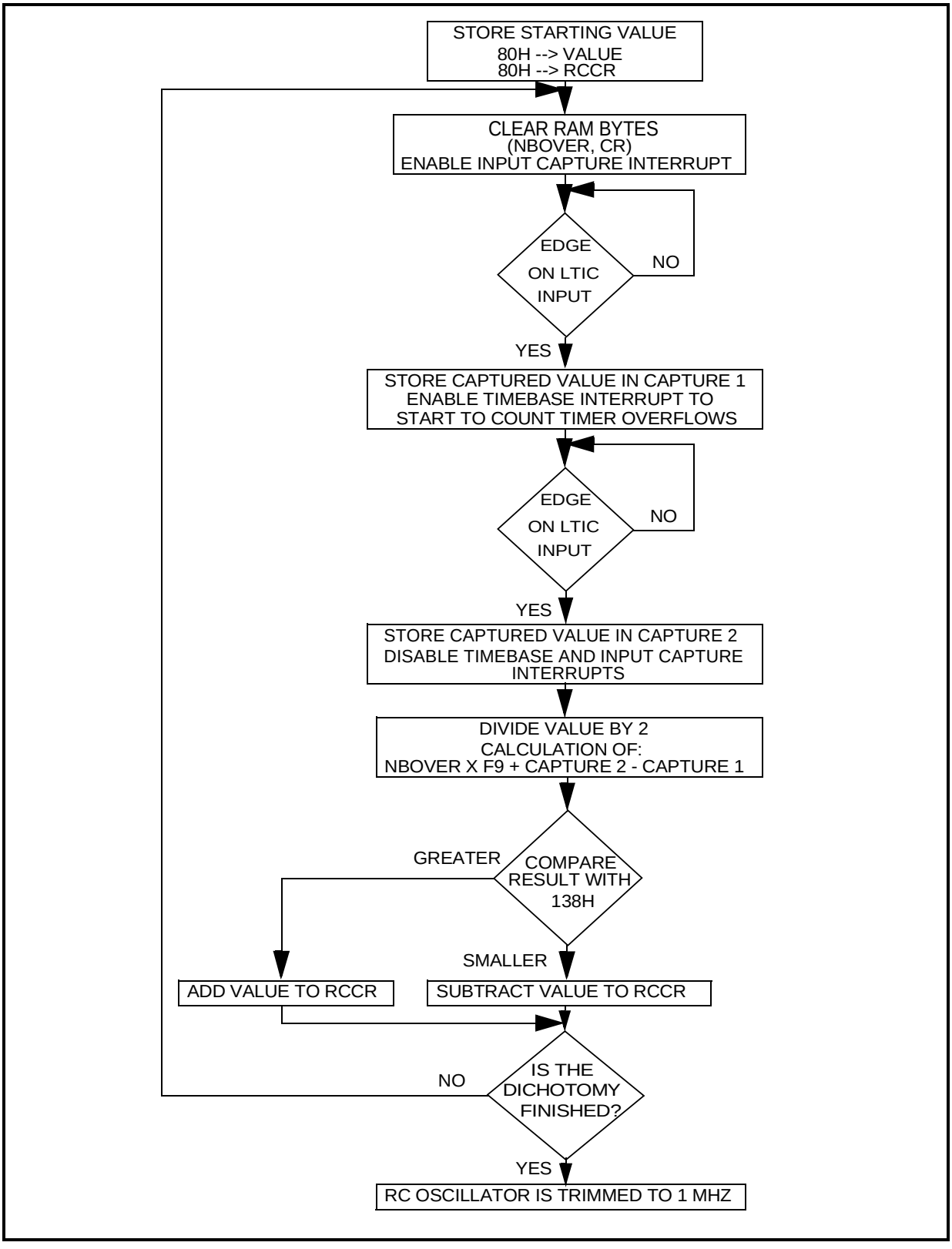
```
        ld              A, LTCSR              ;clear TB bit
        inc             nbover                ;increment number of overflows
endit2  iret
```

4.1.4 writing in eeprom

To store final value of RCCR in EEPROM, add theses lines after disabling the timer interrupts in the main program.

```
        ld              RCCR, A
        bset            EECSR,#1              ;start to enter value in the EEPROM
        ld              $1003,A               ;load value of the RCCR in EEPROM
        bset            EECSR,#0              ;start to write in the EEPROM
wait    btjt            EECSR,#0,wait         ;wait for the end of writing in EEPROM
```

4.1.5 Detailed basic version software flowchart



4.2 AVERAGE VERSION

This version perform the count between two edges four times and changes the value of the RCCR according to the average of these measurements. This method allows to perform a better tune of the RC oscillator.

4.2.1 main program

```
;All the bytes from locations 80h to 8Bh are used by this software to store values or as control
registers but they can be reused safely after the clock has been set.

;dichotomy value
.value                  equ     $81             ;this byte contains the value which will be added
or subtracted to/from the RCCR last value at the end of each round
;capture values
.capture1               equ     $82
.capture2               equ     $86     ;these bytes contain the values of the counter captured
on the edge of the mains, they are used to calculate the time elapsed between the two edges
;number of overflows
.nbover                 equ     $8A     ;this byte contains the number of counter overflows
during the measurement
;control register
.cr                     equ     $8B     ;this byte is used as a control register for the measure-
ment. Its bits allow or not the interrupts and show which step of the count is the current one.
.strtstp                equ     1       ;this is set to start the count and reset to stop it
.overflow               equ     2       ;this bit is set when the first capture has occurred. It
allows the overflows to be counted

.main

        bset            MCCSR, #1       ;output clock enable. You can remove this line if you do
not want to check the clock frequency

        ld              A, #$80         ;value containing the value which will be
        ld              value, A        ;add or subs to/from RCCR during the dichotomy
        ld              RCCR, A         ;RCCR is set to the middle of its range of value

next    clr             nbover          ;clear the byte containing the number of timer overflow
        clr             cr              ;clear the byte use as control register for the count
        ld              A, LTICR        ;clear the ICF bit

        clr             Y
capture rim                             ;interrupts enable
        bset            LTCSR, #7       ;enable input capture interrupt
        bset            cr,#strtstp     ;set the start-stop bit of cr: count can start
count   btjt            cr, #strtstp, count; wait for the end of count
        clr             LTCSR           ;lite timer interrupts disable
        inc             Y
        cp              Y,#$4           ;repeat the capture four time to make an average
        jrne            capture

        srl             value           ;dichotomy value divided by 2

        clr             Y
        ld              A,#$F9          ;these lines calculate this equation for the four
        ld              X,nbover        ;measures:
        mul             X,A             ;(nbover*$F9)+ capture2 - capture1
calcul  add             A, (capture2,Y);
        jrnc            nocarry         ;this equation is calculated with 16 bits
        inc             X               ;
```

```
nocarry sub     A, (capture1,Y);MSB are in register X
        jrnc    noneg          ;
        dec     X              ;and LSB in register A.
noneg   inc     Y
        cp      Y,#$4
        jrne    calcul

        srl     A              ;these lines calculate the average of the last four
        srl     X              ;measures by dividing their total by 4. It is done by
        jrnc    carry1         ;two consecutive right shift on the 16 bit result.
        add     A,#$80
carry1  srl     A
        srl     X
        jrnc    carry2
        add     A,#$80

carry2  cp      X, #$01        ;if mains frequency is 50Hz the reference value is $138
        jrmi    minus          ;if it is 60Hz the reference is $104.the program first
        jreq    compare        ;compares MSB with $01 and then compare LSB with
        jp      plus           ;$38 for 50Hz and $04 for 60Hz. if the calculated
compare cp      A, #$38        ;value is smaller than the reference the program jump to
        jrmi    minus          ;minus to decrease RCCR else it increase RCCR

plus    ld      A, RCCR
        add     A, value       ;add value if counted value is greater than ref
        jp      new

minus   ld      A, RCCR
        sub     A, value       ;subtract value if Y is smaller

new     ld      RCCR,A         ;enter the new value in RCCR

        btjf    value, #0, next;stop after 7 rounds

loop    jp      loop
```

4.2.2 input capture interrupt

```
        ld      A, LTICR              ;load captured value in A
        btjt    LTCSR, #4, finish     ;test if it is first or second capture
        bset    LTCSR, #4             ;allow timebase interrupt in order to count the
number of overflows
        ld      capture1, A           ;captured value is stored in capture1
        jp      endit1

finish  ld      capture2, A           ;if it is the second capture, captured value is
stored in capture2
        clr     cr                    ;clear cr to end the count
endit1  iret
```

4.2.3 timebase interrupt

```
        ld      A, LTCSR              ;clear TB bit
        inc     nbover                ;increment number of overflows
endit2  iret
```

4.2.4 Detailed average version software flowchart