

# AN10849

## LPC1700 RTC hardware auto calibration

Rev. 01 — 1 July 2009

Application note

### Document information

Info	Content
<b>Keywords</b>	RTC, Hardware Auto Calibration, LPC1700, Graphic LCD
<b>Abstract</b>	Using the LPC1700 RTC's auto calibration feature

**Revision history**

Rev	Date	Description
01	20090701	Initial revision

**Contact information**

For additional information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

## 1. Introduction

The Real-time Clock (RTC) oscillator is used to keep track of time (1 Hz) that is used to maintain calendars and clocks. By using a set of dedicated time and alarm registers, it is possible to implement many features that are seen in ordinary (1Hz) clocks.

NXP's LPC1700 RTC features a dedicated 32 kHz ultra-low power (< 1µA) design to support battery powered systems. Along with its 20 byte battery backed storage and its dedicated calendar and clock registers, it includes hardware auto calibration support that allows for a +/-1 second per day accuracy.

### 1.1 Why the RTC calibration is needed

While using a RTC, it is usually desired to maintain accurate timing over a long period. Minor timing differences between the actual RTC oscillator output and the perceived day-to-day time is compounded over a long period. A general workaround to this problem is to have the RTC periodically synchronized with a reference time. Differences in the RTC oscillator's output frequency are mainly caused by external factors such as crystal characteristics and types, trace capacitance, and in particular temperature.

In an attempt to account for timing differences RTCs have some sort of external trimmers to get their oscillations as close as possible to the commonly used 32.768 kHz. This frequency is usually chosen because a counter can count exactly 32768 (2<sup>15</sup>) cycles per second.

A key feature of the LPC1700 RTC block is hardware support for automatic time adjustment calibration, making it possible to keep a +/-1 second per day accuracy.

### 1.2 LPC1700 RTC hardware auto calibration feature

The LPC1700 series' RTC peripheral contains a hardware auto calibration counter that will make a 1 second adjustment to the RTC's clock registers. The adjustment itself depends on the value and the direction specified in the CALIBRATION register.

**Table 501. Calibration register (CALIBRATION - address 0x4002 4040) bit description**

Bit	Symbol	Value	Description	Reset value
16:0	CALVAL	-	If enabled, the calibration counter counts up to this value. The maximum value is 131, 072 corresponding to about 36.4 hours. Calibration is disabled if CALVAL = 0.	NC
17	CALDIR	1	Backward calibration. When CALVAL is equal to the calibration counter, the RTC timers will stop incrementing for 1 sec.	NC
		0	Forward calibration. When CALVAL is equal to the calibration counter, the RTC timers will jump by 2 sec.	
31:12			Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

**Fig 1. Calibration register**

The calibration counter increments once every second until it matches the calibration value. It then resets itself and performs the specified “Forward” or “Backward” adjustment of the time counter registers. The calibration value (CALVAL) is a 17-bit value so that it can maintain a +/- 1 second per day accuracy.

A “Backward” adjustment keeps the same value in the RTC’s clock registers for one second longer, whereas a “Forward” adjustment will increment the time counter registers by two seconds.

### 1.3 Determining the calibration direction and value

A recommended method for determining the calibration value is to use the CLKOUT feature to unintrusively observe the RTC oscillator frequency under the conditions it is to be trimmed for, and calculating the number of clocks that will be seen before the time is off by one second.

**Note:** The RTC oscillator frequency varies with temperature.

The CLKOUT feature can be multiplexed out on P1.27. The CLKOUTCFG register should be enabled to select the RTC with a CLKOUT divider of 1 (CLKOUTCFG = 0x104).

With this configuration it is possible to measure the RTC frequency directly on an oscilloscope.

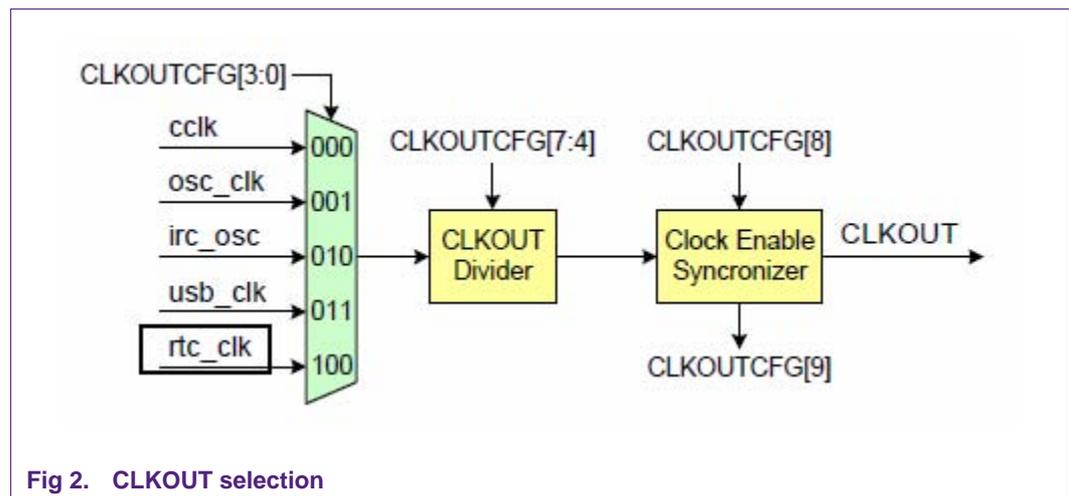


Fig 2. CLKOUT selection

Once the actual RTC frequency is known, CALVAL and CALDIR can be determined using the equations below.

### 1.3.1 Checking if auto calibration should be enabled

The calibration value must be less than  $(2^{17} - 1)$ : 0x0 – 0x1FFFF.

Hence, if the actual RTC oscillating frequency is close enough to the ideal 32.768 kHz, it may be too large to be represented using 17 bits.

Hence if the actual RTC frequency is within

$$\pm \frac{32768.000Hz}{(2^{17} - 1)} \cong \pm 0.250Hz$$

of the ideal 32.768 kHz, then the auto calibration cannot be used.

Incidentally, if the RTC oscillating frequency is that close to the ideal 32.768 kHz so that the calibration value is larger than 0x1FFFF, then a 1 second correction would not be needed for at least every 36 hours.

### 1.3.2 Calibration value (CALVAL)

$$CALVAL = \frac{32768.000}{|32768.000 - f(Hz)|}$$

$f(Hz)$ : The RTC frequency (Hz) observed on the CLKOUT pin.

If the CALVAL is larger than 0x1FFFF the auto calibration should be disabled.

### 1.3.3 Calibration direction (CALDIR)

CALDIR can be determined by evaluating the actual RTC frequency:  $f(Hz)$

If less than 32767.750 Hz, then a “Forward” correction is required (CALDIR = 0).

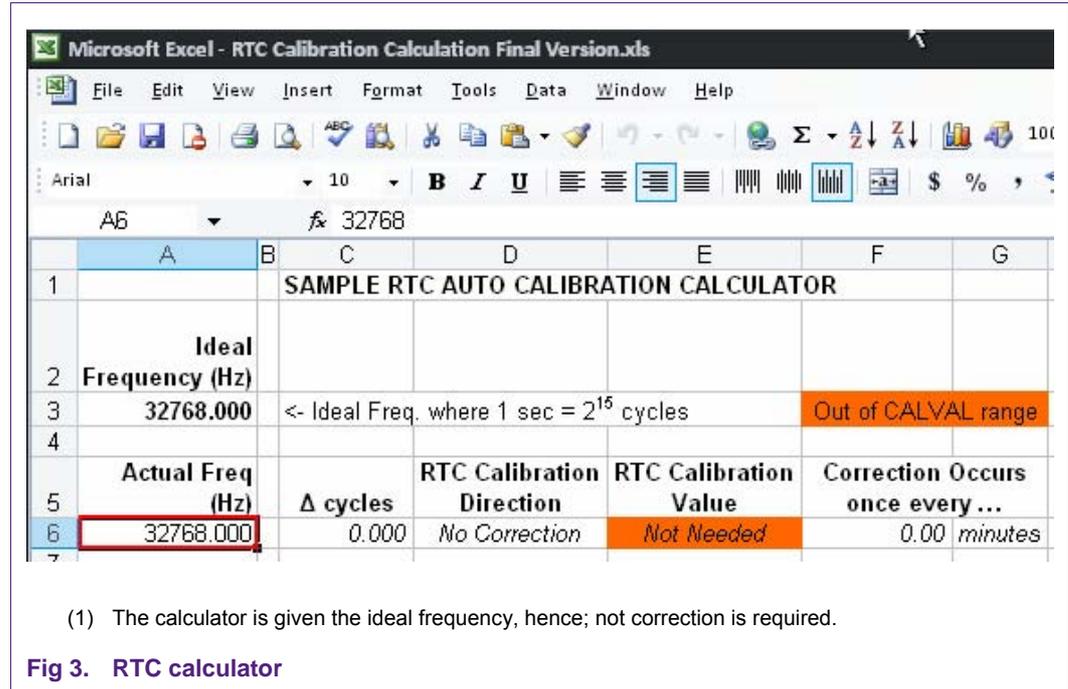
If larger than 32768.250 Hz, then a “Backward” correction is required (CALDIR = 1).

Otherwise no corrections are needed and calibration counter can be disabled (CCALEN = 1 in the CCR register).

1.3.4 Auto calibration calculator

As an additional reference, a RTC calibration calculator is included with the sample code project. The RTC calculator is in a form of an Excel document and includes a simple calculator and some sample reference calculations.

All that is needed is the actual frequency of the RTC oscillator. The calculator will then show the calibration value (CALVAL) and direction (CALDIR).



**Note:** The orange cell coloring indicates that calculated RTC calibration value cannot be represented using 17 bits.

2. RTC calibration demo

2.1 Requirements

The project demo was created using Keil’s μVision3 v3.7 and tested on the MCB1700 board. The sample code is CMSIS (Cortex Microcontroller Software Interface Standard) compliant.

**Note:** A modification to the “LPC17xx.h” header file may have to be performed if the RTC is not counting. See the [Known Issues](#) section.

2.2 Objective

The purpose of this demo is to show how the calibration counter performs a corrective action to the RTC’s clock registers using a known calibration value. The calibration value and calibration direction can be determined by using the formulas shown above.

To illustrate the auto calibration feature, both Timer 0 and RTC are displayed onto the MCB1700’s graphic LCD and onboard LEDs.

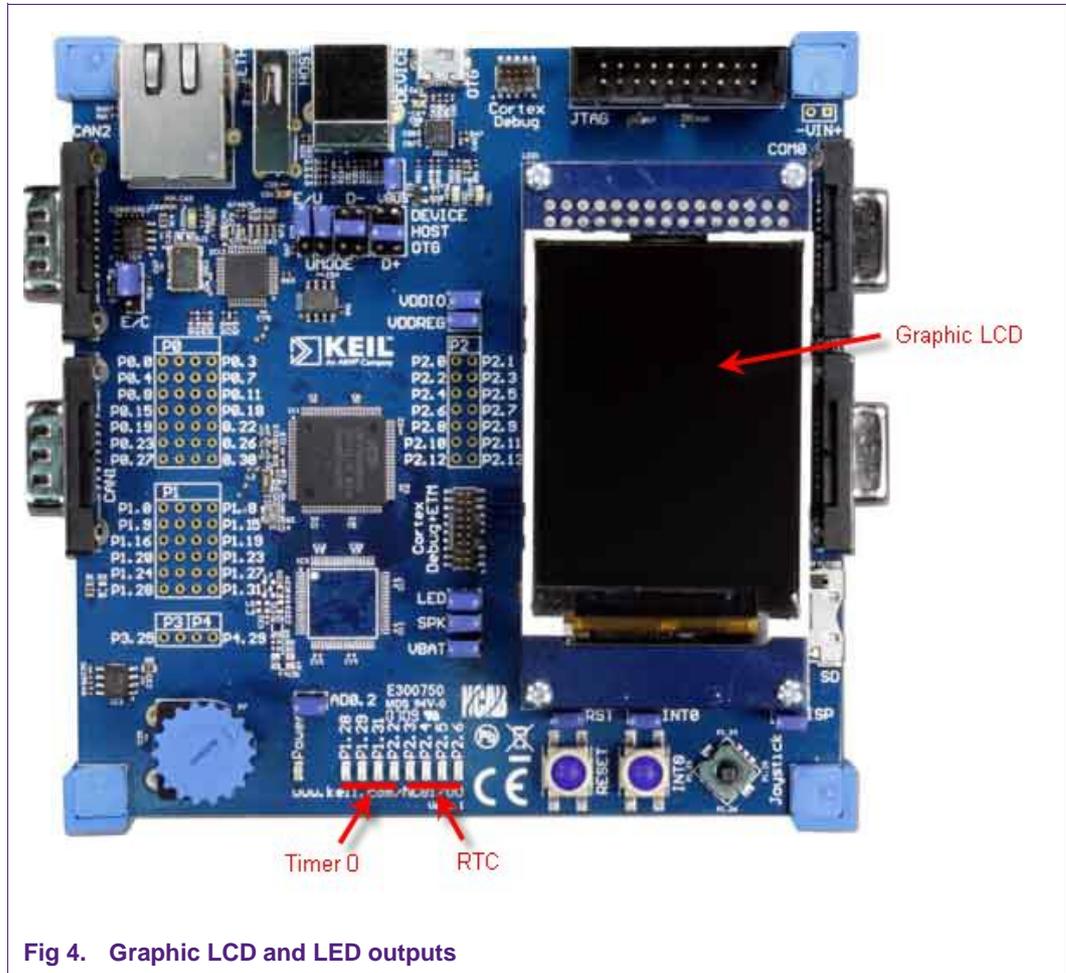


Fig 4. Graphic LCD and LED outputs

### 2.2.1 Graphic LCD output

The graphic LCD will output the real time generated by Timer 0 and by the RTC. It also indicates when the RTC’s auto calibration will be performed and in what direction.

### 2.2.2 LED output

Both the lower 4-bit value of the “SECONDS” time counter register is outputted onto four of the eight LEDs (P2.3-P2.6). The other four LEDs (P1.28, P1.29, P1.31, and P2.2) show the lower 4-bit value contents of Timer 0’s counter, which is also incremented once per second.

### 2.2.3 Demo specific

Both RTC and Timer 0 start with a value of 0 and increment once every second. As the calibration counter matches CALVAL, the auto calibration feature will adjust the RTC time whereas Timer 0 continues to increment normally.

For demonstration purposes,  $ACTUAL\_RTC\_FREQ = 32768000 * 1.3$  so that CALVAL is calculated to equal to 3 and CALDIR is set “Backwards”. This CALVAL is too small for practical use and serves only as a visual for the demo! Other sample values for ACTUAL\_RTC\_FREQ are also provided in the demo project. See [Fig 6](#).

## 2.3 Code design

### 2.3.1 Timer 0

The LPC1700 is configured such that the main oscillator is clocking Timer 0. Timer 0 is divided down to 18 MHz; hence Timer 0's prescaler is set to 18000000. As a result, Timer 0 will increment once every second (1 Hz).

```

5 void init_timer(BYTE timer_num, uint32_t TimerInterval)
6 {
7 // NVIC_InitTypeDef NVIC_InitStructure;
8
9 if ( timer_num == 0 )
10 {
11 timer0_counter = 0;
12 TIMO->MRO = TimerInterval;
13 TIMO->MCR = 2;           /* Disable Match Register Interrupts
14                          and Reset on Match */
15 TIMO->PR = 18000000;     /* TIMO is running @ 18 MHz, so increment TIMO every 18000000 cycles */
16
17 }

```

- (1) It is important to set the prescale to 18000000 so that Timer 0 increment once per second.

Fig 5. Timer 0 Initialization

### 2.3.2 RTC

The RTC is already clocked by the RTC oscillator so it just needs to be reset and enabled.

To determine the CALVAL and CALDIR bits, we provide the actual RTC frequency into `init_calibration_val_dir()` function. This will calculate and enable the calibration value and corrective direction bits.

```

088 void InitRTC(void) {
089
090     /* Disable RTC Clock, Reset Clock, Disable Calibration */
091     RTC->CCR = ( !CLKEN | CTRST );
092
093     /* Set Calibration value and direction */
094     init_calibration_val_dir(ACTUAL_RTC_FREQ);
095
096     /* Disable Clock Reset */
097     RTC->CCR &= ~CTRST;
098
099     /* Clear RTC DOY, HOUR, MIN and SEC registers */
100     RTC->DOY = 0;
101     RTC->HOUR = 0;
102     RTC->MIN = 0;
103     RTC->SEC = 0;
104
105     return;
106 }

```

Fig 6. RTC initialization

### 2.3.3 Sample RTC frequency calibration values

For the initial sample code demo, we will set ACTUAL\_RTC\_FREQ to  $(32768000 * 1.3)$  so that every 3 seconds the RTC auto calibration will perform the corrective action. This value can be determined by using the formulas provided above.

```

16  /* PUBLIC DEFINITIONS */
17  /* Definitions needed to calculate the calibration value and direction */
18  #define IDEAL_RTC_FREQ      32768000    //in mHz (milli Hz)
19  //#define ACTUAL_RTC_FREQ   32768000    //Ideal freq test
20  /* For the RTC demo we choose to make an adjustment every 3 seconds */
21  #define ACTUAL_RTC_FREQ 32768000 * 1.3 //in mHz Backwards every 3 seconds
22  //#define ACTUAL_RTC_FREQ 32768000 * 0.7 //in mHz Forwards every 3 seconds
23  //#define ACTUAL_RTC_FREQ 32768000 - 251 //in mHz closest allowable forward correction
24  //#define ACTUAL_RTC_FREQ 32768000 + 251 //in mHz closest allowable backward correction

```

(1) Sample definitions located in "rtc.h"

Fig 7. Actual RTC frequency definitions used for the DEMO

### 2.3.4 Optional LED output

Set "TIMER\_0\_VALUE\_OUTPUT" to 0 so that instead of displaying the actual 4-bit value of Timer 0, it will instead toggle the P1.28 LED every second.

```

16  /* Test 4 bit LED outputs */
17  #define LED_TEST_ENABLE      0
18
19  #define TIMER_0_VALUE_OUTPUT  1

```

Fig 8. Optional Output definitions

### 2.3.5 Graphic LCD and LED refreshing

Now that the RTC and Timer 0 both increment once per second, we can then let the CPU poll and display the two counter values onto the graphic LCD and LEDs.

```

157 while(1){
158     #if TIMER_0_LED_VALUE_OUTPUT
159         /* Display # of seconds from Timer0 to the upper nibble */
160         output_4_bit_value(TIM0->TC % 60 , 1);
161     #else
162         /* Toggle P1.28 each second */
163         output_4_bit_value(TIM0->TC << 3, 1);
164     #endif
165     /* Display RTC's SECONDS register onto the lower nibble */
166     output_4_bit_value(RTC->SEC , 0);
167
168     /* Convert integer values to char array format and output onto the Graphics LCD */
169     /* Timer 0 */
170     sprintf((char *)temp, "TIMO: %2.2d:%2.2d:%2.2d:%2.2d", (TIM0->TC/86400)%356,
171                                                     (TIM0->TC/3600)%24,
172                                                     ((TIM0->TC)%3600)/60,
173                                                     ((TIM0->TC)%3600)%60);
174     GLCD_displayStringLn(Line5, temp);
175
176     /* RTC */
177     sprintf((char *)temp, "RTC : %2.2d:%2.2d:%2.2d:%2.2d", RTC->DOY,
178                                                     RTC->HOUR,
179                                                     RTC->MIN,
180                                                     RTC->SEC);
181     GLCD_displayStringLn(Line7, temp);
182
183 } /* END WHILE(1) */

```

Fig 9. Graphic LCD and LED refreshing

In this demo, you should see the RTC being corrected every 3 seconds in the “Backward” direction. When a “Backward” correction is performed the RTC counter will keep its value for one additional second.

### 3. Known Issues

A correction in the “LPC17xx.h” header file may have to be performed if the RTC is not incrementing properly. This header file is included with Keil uVision 3.7 tool chain.

The following needs to be added to “LPC17xx.h”:

```

414 /*----- Real-Time Clock (RTC) -----*/
415 typedef struct
416 {
417     __IO uint8_t ILR;
418     uint8_t RESERVED0[3];
419     uint32_t EXTRA_BUFFER_NEEDED_HERE;
420     __IO uint8_t CCR;
421     uint8_t RESERVED1[3];
422     __IO uint8_t CIIR;
423     uint8_t RESERVED2[3];
424     __IO uint8_t AMR;

```

(1) A fix may have to be done in the RTC\_TypeDef structure.

Fig 10. Correction in “LPC17xx.h” header file in Keil’s uVision 3.7

## 4. Legal information

### 4.1 Definitions

**Draft** — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

### 4.2 Disclaimers

**General** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected

to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

### 4.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

## 5. Contents

<b>1.</b>	<b>Introduction .....</b>	<b>3</b>
1.1	Why the RTC calibration is needed .....	3
1.2	LPC1700 RTC hardware auto calibration feature .....	3
1.3	Determining the calibration direction and value..	4
1.3.1	Checking if auto calibration should be enabled..	5
1.3.2	Calibration value (CALVAL) .....	5
1.3.3	Calibration direction (CALDIR) .....	5
1.3.4	Auto calibration calculator .....	6
<b>2.</b>	<b>RTC calibration demo .....</b>	<b>6</b>
2.1	Requirements .....	6
2.2	Objective .....	6
2.2.1	Graphic LCD output .....	7
2.2.2	LED output .....	7
2.2.3	Demo specific.....	7
2.3	Code design.....	8
2.3.1	Timer 0 .....	8
2.3.2	RTC.....	8
2.3.3	Sample RTC frequency calibration values .....	9
2.3.4	Optional LED output.....	9
2.3.5	Graphic LCD and LED refreshing.....	9
<b>3.</b>	<b>Known Issues .....</b>	<b>10</b>
<b>4.</b>	<b>Legal information .....</b>	<b>11</b>
4.1	Definitions .....	11
4.2	Disclaimers.....	11
4.3	Trademarks .....	11
<b>5.</b>	<b>Contents.....</b>	<b>12</b>

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.



© NXP B.V. 2009. All rights reserved.

For more information, please visit: <http://www.nxp.com>  
 For sales office addresses, email to: [salesaddresses@nxp.com](mailto:salesaddresses@nxp.com)

Date of release: 1 July 2009  
 Document identifier: AN10849\_1