

F²MC-16FX FAMILY
16-BIT MICROCONTROLLER
ALL SERIES

CLOCK CALIBRATION

APPLICATION NOTE

Revision History

Date	Issue
2006-12-11	First Version; MWi
2007-07-22	Updated example software; HPi
2007-09-24	Added USART calibration; HPi
2007-09-27	Updated USART calibration chapter; HPi

This document contains 18 pages.

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH restricts its warranties and its liability for **all products delivered free of charge** (e.g. software include or header files, application examples, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Microelectronics Europe GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Microelectronics Europe GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Microelectronics Europe GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Microelectronics Europe GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Microelectronics Europe GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Microelectronics Europe GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Microelectronics Europe GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Microelectronics Europe GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Microelectronics Europe GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Microelectronics Europe GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect

Contents

REVISION HISTORY	2
WARRANTY AND DISCLAIMER	3
CONTENTS	4
1 INTRODUCTION	5
1.1 Key Features	5
2 CLOCK CALIBRATION	6
2.1 Functionality	6
2.2 Block Diagram	6
2.3 Timing Diagram	6
2.4 Registers	7
2.4.1 Clock Calibration Unit Control Register (CUCR)	7
2.4.2 Clock Calibration Duration Timer Data Register (CUTD)	7
2.4.3 Clock Calibration Timer Data Register (CUTR)	7
3 CALIBRATION AND MEASUREMENT	8
3.1 Interpreting Measurement Results	8
3.2 Calibration of the Real Time Clock	8
4 CLOCK CALIBRATION EXAMPLES	9
4.1 Calibration Measurement of the RC or Sub Clock	9
4.2 Calibrating the Real Time Clock	11
4.3 Calibrating the USART	15
5 ADDITIONAL INFORMATION	18

1 Introduction

This application note describes the functionality of the Clock Calibration function and gives an example.

1.1 Key Features

- Measurement of the following clocks against the Main Clock (CLKMC)
 - RC Clock (CLKRC) (100 kHz, 2 MHz)
 - Sub Clock (CLKSC)

2 Clock Calibration

THE BASIC FUNCTIONALITY OF THE CLOCK CALIBRATION FUNCTION

2.1 Functionality

With the Clock Calibration the RC or Sub Clock oscillation clocks a counter for a given duration value. In this duration time the calibration timer counter is clocked by the Main Clock (CLKMC).

This allows to trim other resources, which have reload counter prescalers (Real Time Clock, USART, etc.).

2.2 Block Diagram

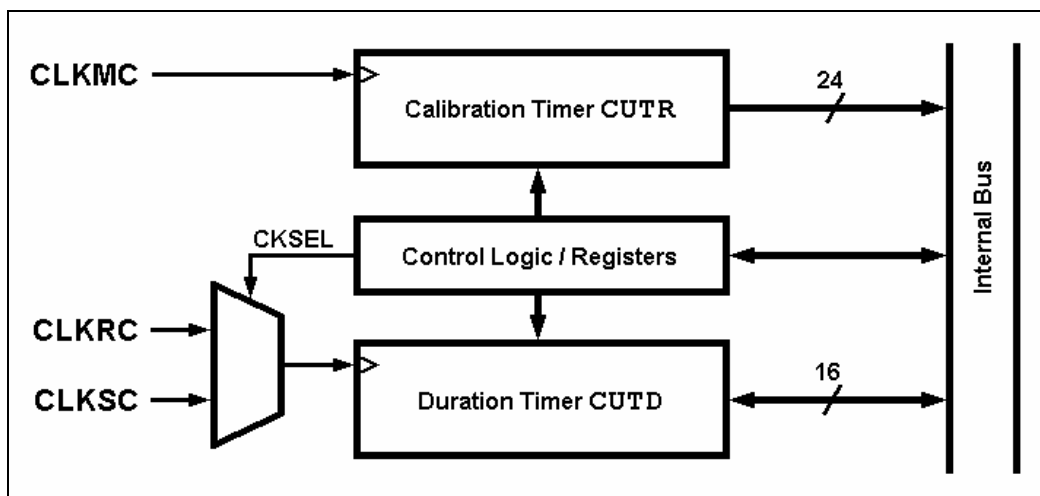


Figure 2-1: Clock Calibration block diagram

2.3 Timing Diagram

The following timing diagram shows the functionality of the Clock Calibration unit:

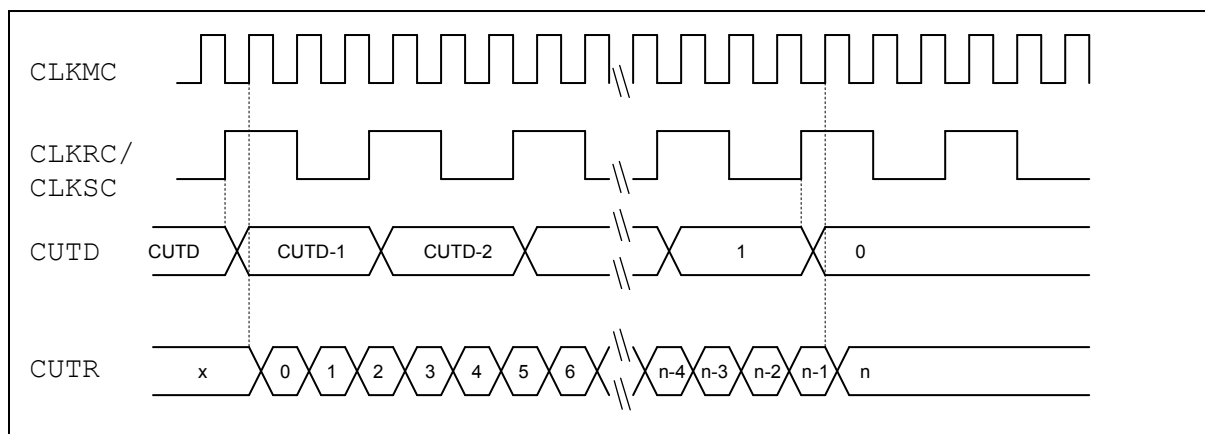


Figure 2-2: Timing diagram Clock calibration unit

Please note, that the length of the clock pulses are not drawn in a proportional manner.

2.4 Registers

2.4.1 Clock Calibration Unit Control Register (CUCR)

The CUCR consists of the following bits:

Bit No.	Bit Name	Initial Value	Value	Description
7	<i>Reserved</i>	X	0	Always write "0" to this bit
6, 5	-	0	-	<i>Unused bits</i>
4	STRT	0	0	No Calibration
			1	Start Calibration
3	-	0	-	<i>Unused bit</i>
2	CKSEL	0	0	Sub Clock (CLKSC) selected for calibration
			1	RC Clock (CLKRC) selected for calibration
1	INT	0	0	No calibration / Calibration ongoing Write: Clear bit
			1	Calibration finished
0	INTEN	0	0	Interrupt disabled
			1	Interrupt enabled

2.4.2 Clock Calibration Duration Timer Data Register (CUTD)

This 16-bit register contains the duration interval for the duration timer. This timer is clocked by the RC or Sub Clock as selected by CUCR: CKSEL

2.4.3 Clock Calibration Timer Data Register (CUTR)

This 24-bit register is clocked by the Main Clock (CLKMC). Its counting is gated by the duration of the clock to be calibrated.

3 Calibration and Measurement

THIS CHAPTER SHOWS HOW TO CALCULATE CALIBRATION DATA

3.1 Interpreting Measurement Results

Assume the 32.768 kHz sub clock should be measured with an ideal Main Clock of 4MHz. The duration in `CUTD` was set to a value of `0x2000 = 8192`, which should represent 0.25s at an ideal sub clock oscillation.

After measurement, the value of `CUTR` is e. g. `0x0F4002`. What does this mean?

The Calibration Timer has counted to `0x0F4002 = 999426` in the duration time given by the sub clock. This represents (at ideal 4 MHz Main Clock reference) a duration time of $999426 / 4000000 = 0.24986s$. Thus the sub clock oscillation is a bit to fast.

3.2 Calibration of the Real Time Clock

Assume the Real Time Clock is clocked by the Sub Clock oscillator with the same frequency deviation given in 3.1.

At an ideal Sub Clock of 32.768 kHz the Sub Second Register (`WTBR`) would have the reload value 8192 (= `0x2000`). Because the oscillation was measured as too fast, the new value can be calculated as:

$$WTBR_{CALIBRATED} = \frac{f_{RTC} \cdot f_{MAIN} \cdot t_{INTERVAL}}{4s^{-1} \cdot CUTR}$$

f_{RTC} :	Clock source of Real Time Clock
f_{MAIN} :	Main Clock (CLKMC)
$t_{INTERVAL}$:	Interval time given by <code>CUTD</code>

With the example values above the calculation is:

$$WTBR_{CALIBRATED} = \frac{32768s^{-1} \cdot 4000000s^{-1} \cdot 0.25s}{4s^{-1} \cdot 999426} = 8196.70$$

The new calibrated Sub Second Register value is then: 8197

4 Clock Calibration Examples

EXAMPLES FOR THE CLOCK CALIBRATION USAGE

4.1 Calibration Measurement of the RC or Sub Clock

The following example shows, how to measure the RC or sub clock. No interrupts are used.

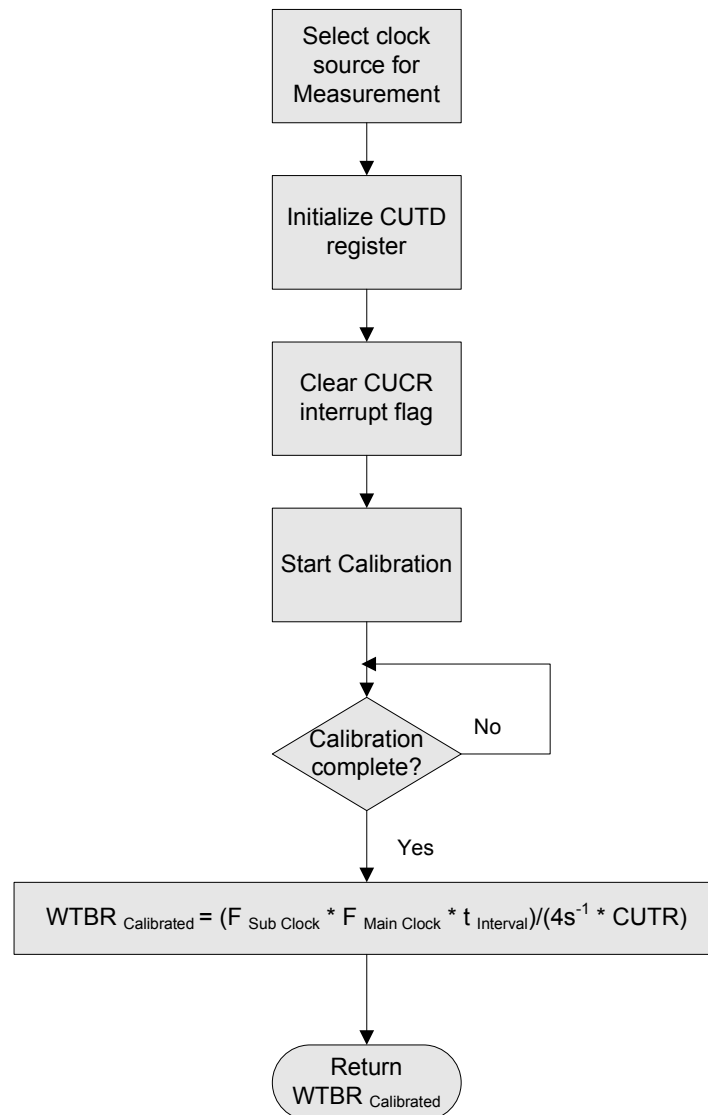


Figure 4-1: Flowchart for Calibration Measurement

```
/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*-----*/

#include "mb96347rw.h"

#define MAIN_CLOCK 4000000 // Main Clock in Hz
#define SUB_CLOCK 32768 // Sub Clock in Hz
#define RC_CLOCK 100000 // RC clock in Hz
#define INTERVAL 0.5 // Measurement duration in s
#define CUTD_VALUE (INTERVAL * SUB_CLOCK)

unsigned long ClockMeasure(unsigned int duration)
{
    float calibrate;
    unsigned long wtbr_cal2;
    CUCR = 0x00; // measure sub clock, no interrupts
    //CUCR = 0x04; // measure RC clock, no interrupts

    CUTD = CUTD_VALUE;
    CUCR_INT = 0;
    CUCR_STRT = 1; // start calibration measurement
    while (!CUCR_INT); // wait for end of measurement

    CUCR_INT = 0; // clear INT flag (is set even if no interrupts used)

    calibrate = 0.5 + (((float) SUB_CLOCK * (float) MAIN_CLOCK
        * (float) INTERVAL) / ((float) 4 * (float) CUTD));
    wtbr_cal2 = (unsigned long) calibrate; //0.5 is added for rounding

    return wtbr_cal2; // return measurement value
}
```

4.2 Calibrating the Real Time Clock

The following example shows how to calibrate the Real Time Clock clocked by the 32 kHz Sub Clock.

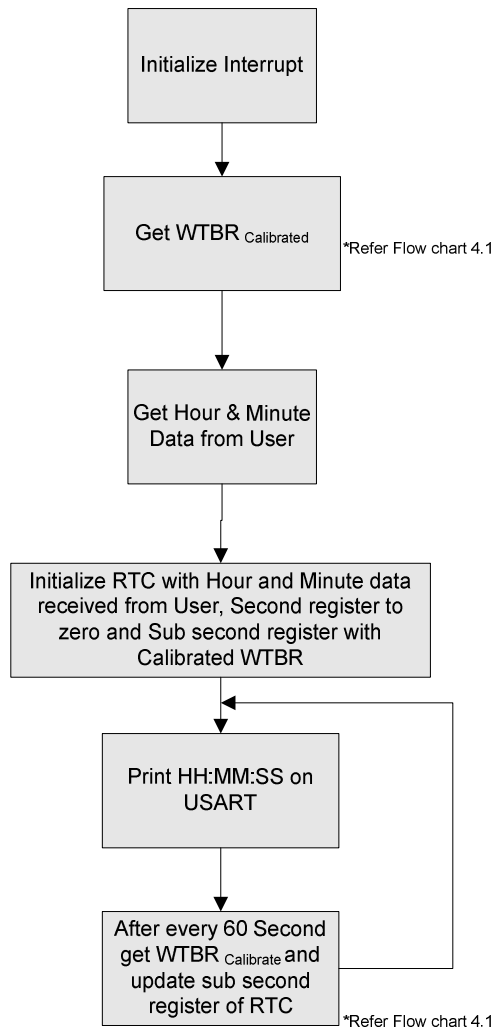


Figure 4-2: Flowchart for RTC Calibration

```

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS. FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY ERRORS OR */
/* ELIGIBILITY FOR ANY PURPOSES. */
/* (C) Fujitsu Microelectronics Europe GmbH */
/*-----*/

#include "mb96347rw.h"

#define MAIN_CLOCK 4000000 // Main Clock in Hz
#define SUB_CLOCK 32768 // Sub clock in Hz
#define INTERVAL 0.5 // Measurement duration in s
#define CUTD_VALUE (INTERVAL * SUB_CLOCK)
#define CUTR_IDEAL (unsigned long) (INTERVAL * MAIN_CLOCK)

#define SCT 0x47 // irq all 4s

#define UPDATE_INTERVAL 15 // 4s * 15 = 60s
#define WEIGHT (float)100
#define T_INTERVAL 60

char update_count;
unsigned long cal;
double dt, w;

void InitRTC(unsigned char hour, unsigned char minute, unsigned long subsecond)
{
    WTCR_ST = 0; // Stop RTC
    WTCR_INTE0 = 0; // No Interrupts
    WTCR_INTE1 = 0;
    WTCR_INTE2 = 0;
    WTCR_INTE3 = 0;
    //WTCR_INTE4 = 0;
    WTCR_OE = 1; // Output

    WTBRO = (0xFFFF & subsecond); // Set calibrated prescaler value
    WTBRI = (subsecond >> 16);
    WTHR = hour;
    WTMR = minute;
    WTSR = 0;
    WTCR_UPDT = 1; // Update RTC

    while(!SSR0_RDRF); // wait for "RETURN"

    WTCR_ST = 1; // Start RTC

    (volatile) RDR0; // Flush RDR0
}

unsigned long ClockMeasure(unsigned int duration)
{
    float calibrate;
    unsigned long wtbr_cal2;

    CUTD = duration;
    CUCR_INT = 0;
    CUCR_STRT = 1;
    while (!CUCR_INT);

    CUCR_INT = 0;

```

```

    calibrate = 0.5 + (((float) SUB_CLOCK * (float) MAIN_CLOCK
                      * (float) INTERVAL) / ((float) 4 * (float) CUTR));
    wtbr_cal2 = (unsigned long) calibrate;

    return wtbr_cal2;
}

void InitSCT(void)
{
    SCTCR = SCT;
}

void wait(long a)
{
    long i;

    for (i = 0; i < a; i++)
        __wait_nop();
}

void printdigits(unsigned char val)
{
    unsigned int val2;

    val2 = val / 10;
    Putch0(val2 + 0x30);

    val = val - (val2 * 10);
    Putch0(val + 0x30);
}

void printtime(void)
{
    printdigits(WTHR);
    Putch0(':');
    printdigits(WTMR);
    Putch0(':');
    printdigits(WTSR);
    Puts0("\n");
}

void main(void)
{
    unsigned char hh, hl, mh, ml, hour, minute;

    DDR00 = 0xFF;
    PDR00 = 0x01; // indicate start

    InitIrqLevels();
    set_il(7); // allow all levels
    EI(); // globally enable interrupts

    w = WEIGHT;
    dt = 0;

    WTCKSR = 0x01; // Sub clock
    CUCR_CKSEL = 1; // Sub clock for calibration unit
    cal = ClockMeasure(CUTD_VALUE);

    PDR00 = 0x02;

    InitUart0(416); // 19200 bps @ 8 MHz

    PDR00 = 0x03;

    Puts0("\nRTC/RC clock self calibration\n");
    Puts0("=====\n");
    Puts0("CUTR: 0x");
}

```

```

    printlong(CUTR);
    Puts0("\nCUTR: ");
    printdeclong(CUTR);
    Puts0("\nSub Second: 0x");
    printlong(cal);
    Puts0("\nSub Second: ");
    printdeclong(cal);
    Puts0("\n\nEnter Time (HHMM) <RETURN>: ");

    (volatile) RDR0; // Flush RDR0
    hh = Getch0();
    Putch0(hh);
    hl = Getch0();
    Putch0(hl);
    mh = Getch0();
    Putch0(mh);
    ml = Getch0();
    Putch0(ml);

    hour = (hl & 0x0F) + 10 * (hh & 0x0F);
    minute = (ml & 0x0F) + 10 * (mh & 0x0F);

    hour = (hl & 0x0F) + 10 * (hh & 0x0F);
    minute = (ml & 0x0F) + 10 * (mh & 0x0F);

    InitRTC(hour, minute, cal);

    InitSCT();

    Puts0("\n");

    while(1)
    {
        printtime();
    }
}
// Recalibration of RTC
__interrupt void ISR_SC(void)
{
    SCTCR = SCT; // clear irq

    if (update_count >= UPDATE_INTERVAL)
    {
        update_count = 0;

        dt += (((double)CUTR / (MAIN_CLOCK / 2)) * SUB_CLOCK / cal
              / 4 * (double)T_INTERVAL) - (double)T_INTERVAL;

        cal = ClockMeasure(CUTD_VALUE);
        cal += dt * WEIGHT;

        WTCR_INT0 = 0;
        while(!WTCR_INT0); // wait for second

        WTBR0 = (0xFFFF & cal); // Set calibrated prescaler value
        WTBR1 = (cal >> 16);
    }

    update_count++;
}

```

4.3 Calibrating the USART

As per RS232 standard, the maximum variance allowed for baud rate is 5%.

As per data sheet, for chip to chip, value of internal RC oscillator may vary from 1 MHz to 4 MHz for CLKRC of 2 MHz or it may vary from 50 kHz to 200 kHz for CLKRC of 100 kHz.

If we assume that CLKRC is operating at typical frequency of 2 MHz and CLKRC is selected as clock source for CLKS1 and intern to CLKP1 than BGR_n register value is calculated as...

$$BGR_n = \left[\frac{CLKRC}{9600} - 1 \right]$$

$$\therefore BGR_n = \left[\frac{2000000}{9600} - 1 \right]$$

$$\therefore BGR_{n \text{ Typ}} = 207$$

If we set the BGR_n equal to 207 and if the actual frequency of CLKRC is 2.5 MHz than resulting baud rate will be

$$BaudRate = \left[\frac{CLKRC}{BGR_n + 1} \right]$$

$$\therefore BaudRate = \left[\frac{2500000}{207 + 1} \right]$$

$$\therefore BaudRate = 12019$$

This is far beyond the acceptable variance by RS232 standard.

Considering above facts, when CLKRC is selected as clock source for CLKS1 and intern to CLKP1, one can use clock calibration unit to find the actual frequency of CLKRC and can set BGR_n register value to match the required baud rate.

Let us consider an example; here we will assume that clock source CLKMC which is input to calibration timer is stable and its frequency is exactly 4 MHz.

Now we know that when we initialise $CUTD$ to 0x0000 and start measurement, and when an underflow occurs, the measurement will take $(0xFFFF + 1) * T_{CLKRC}$ seconds.

Let us assume that for the above measurement $CUTR$ counts up to 0x0019999.

From this information, we can find out value of CLKRC as follow

$$\begin{aligned} \frac{CUTD}{CLKRC} &= \frac{CUTR}{CLKMC} \\ \therefore \frac{65536}{CLKRC} &= \frac{104857}{4 \text{ MHz}} \\ \therefore CLKRC &= \frac{65536 * 4 \text{ MHz}}{104857} \\ \therefore CLKRC &= 2.500014 \text{ MHz} \end{aligned}$$

One point of interest over here is, consider that when CUTD reaches to zero and when STRT is reset indicating end of calibration process, it may happen that Main Clock cycle is already finished 90% of its time and was about to increment CUTR, so CUTR instead of counting 104858 has counted 104857 cycles. (CUTR count is incremented at the end of Main clock cycle).

If we recalculate, using above formula we will get CLKRC as follow

$$\begin{aligned} \frac{CUTD}{CLKRC} &= \frac{CUTR}{CLKMC} \\ \therefore \frac{65536}{CLKRC} &= \frac{104858}{4 \text{ MHz}} \\ \therefore CLKRC &= \frac{65536 * 4 \text{ MHz}}{104858} \\ \therefore CLKRC &= 2.499990 \text{ MHz} \end{aligned}$$

Now from above calculation, as we know the RC clock frequency, let us find out the BGR_n register value for baud rate of 9600:

$$\begin{aligned} BGR_n &= \left[\frac{CLKRC}{9600} - 1 \right] \\ \therefore BGR_n &= \left[\frac{2500014}{9600} - 1 \right] \text{ or } BGR_n = \left[\frac{2499990}{9600} - 1 \right] \\ \therefore BGR_n &= 259 \text{ or } BGR_n = 259 \end{aligned}$$

Let us also do reverse calculation to find out what is exact baud rate if we select BGR_n value to 259.

$$BaudRate = \left[\frac{CLKRC}{BGR_n + 1} \right]$$

$$\therefore BaudRate = \left[\frac{2500014}{259 + 1} \right] \text{ or } BaudRate = \left[\frac{2499990}{259 + 1} \right]$$

$$\therefore BaudRate = 9615 \text{ or } BaudRate = 9615$$

Here actual baud rate is 0.15% more than the required one, which is within acceptable limits of RS232 Standards.

Also from the above calculation, the error due to one cycle count miss of $CUTR$ will not affect the final baud rate, so one can safely ignore it.

Considering $CLKRC$ may vary from 1 MHz to 4 MHz and the fact that for asynchronous serial communication BGR_n must be set equal to or more than four; Maximum and minimum baud rate that can be set is calculated as follows

$$BaudRate = \left[\frac{CLKRC}{BGR_n + 1} \right]$$

$$\therefore BaudRate = \left[\frac{100000}{4 + 1} \right] \text{ or } BaudRate = \left[\frac{4000000}{4 + 1} \right]$$

$$\therefore BaudRate = 200\text{kbps} \text{ or } BaudRate = 800\text{kbps}$$

So for all practical purposes where $CLKRC$ is not known in advance one can still start with a baud rate of 200kbps.

One more fact to be considered while calibrating USART is that $CLKRC$ frequency is a temperature dependent.

5 Additional Information

Information about FUJITSU Microcontrollers can be found on the following Internet page:

<http://mcu.emea.fujitsu.com/>

The software examples related to this application note is:

96340_rtc_clkcal_sc_autocorr-v10

It can be found on the following Internet page:

http://mcu.emea.fujitsu.com/mcu_product/mcu_all_software.htm