

User Guide

How to use the LSE clock security system to improve clock system robustness Introduction

The purpose of this document is to help users understand the security monitoring function of LSE clocks, improve the robustness of the clock system, improve solution security performance, and reducing the development time and complexity.

This document is only applicable to NSING MCU products. Currently, the supported product include N32L43x, N32L40x, and N32G43x series.

Content

1	Introduction	3
1.1	Overview.....	3
1.2	Introduction to LSE Clocks	3
1.3	Introduction to LSI Clocks	3
1.4	Introduction to LSE Clock Security System (LSECSS)	3
1.5	Applicability	3
2	Hardware Environment	4
2.1	Demo Function	4
2.2	Hardware Platform.....	5
3	Demo Explain	6
3.1	Demo Process	6
3.2	Demo Analyze	7
3.2.1	USART1 Log Serial Port Initialization.....	7
3.2.2	Enable LSE	8
3.2.3	Enable LSI	9
3.2.4	Enable LSE-CSS Monitor.....	10
3.2.5	Enter STOP2	11
3.2.6	Initialize LPTIM and LPUART	12
4	User Guide	13
4.1	Reset MCU	13
4.2	Generate LSE Fault.....	14
4.3	LPTIMER Switch Clock Source.....	15
4.4	MCU Wakes Up From STOP2	16
4.5	MCU Enter STOP2 Sleep Mode.....	16
4.6	MCU Periodically Woken Up.....	17
4.7	LSE Recover	18
4.8	LPTIMER Switches The Clock Source to LSE	18
5	Version History	19
6	Disclaimer	20

1 Introduction

1.1 Overview

In some application scenarios, LSE failures may occur. Here, a method is proposed to use LSE-CSS to monitor LSE failures. When a failure occurs, the system clock can be switched from LSE to LSI, avoid the LSE failure causing the system stop running.

1.2 Introduction to LSE Clocks

The LSE crystal is a 32.768KHz low-speed external crystal or ceramic resonator. It provides a low-power and accurate clock source for real-time clocks or other timing functions. The LSEEN bit in the Low Power Domain Control Register (RCC_LDCTRL). The LSERD in the Low Power Domain Control Register (RCC_LDCTRL) indicates whether the LSE crystal oscillator is stable. During the startup phase, the LSE clock signal is not released until this bit is set by hardware. If enabled in the clock interrupt register (RCC_CLKINT), an interrupt request can be generated.

1.3 Introduction to LSI Clocks

The LSI RC can provide clocks for IWDG and AWU in the STOP2 and STANDBY mode. The LSI clock frequency is about 40KHz. The LSI RC can be enabled or disabled by the LSIEN bit in the Control/Status register (RCC_CTRLSTS). The LSIRD bit in the Control/Status register (RCC_CTRLSTS) indicates whether the low-speed internal oscillator is stable. During the startup phase, the clock is not released until this bit is set by hardware. If enabled in the clock interrupt register (RCC_CLKINT), an LSI interrupt request can be generated.

1.4 Introduction to LSE Clock Security System (LSECSS)

The LSE clock security system is activated by enabling the LSECLKSSEN bit in the Low Power Domain Control Register (RCC_LDCTRL). The LSECLKSSEN bit can be cleared by a hardware reset or RTC software reset or after detection of an LSE fault. When the LSE and LSI are enabled and ready, the LSECLKSSEN bit must be enabled after configuring the RTCSEL to select the RTC clock source. If an LSE failure is detected, LSE will no longer be provided to the RTC, but the RTCSEL bits will not be modified by hardware to switch the RTC clock source. In STANDBY mode, an LSE clock failure will trigger a wakeup. In other modes, an interrupt can be generated to wakeup, and then software can clear the LSECLKSSEN bit and turn off the LSE, and changing the clock source of the RTC, and other measures to ensure application security. The frequency of the LSE oscillator must be higher than 30KHz to prevent LSECSS false detection.

1.5 Applicability

This Demo is only applicable to N32L43x, N32L40x, N32G43x series MCU, supports KEIL5 platform.

[SDK-VER 1.1.0]

Release Date: 2021-11-30

2 Hardware Environment

2.1 Demo Function

This demo mainly shows the developer how to switch the peripheral clock source to the LSI when the LSE fails, the system monitors the LSE clock, waits for the LSE to recover, and then switches the peripheral clock source from the LSI to the LSE.

2.2 Hardware Platform

Figure 2-1 Hardware Platform

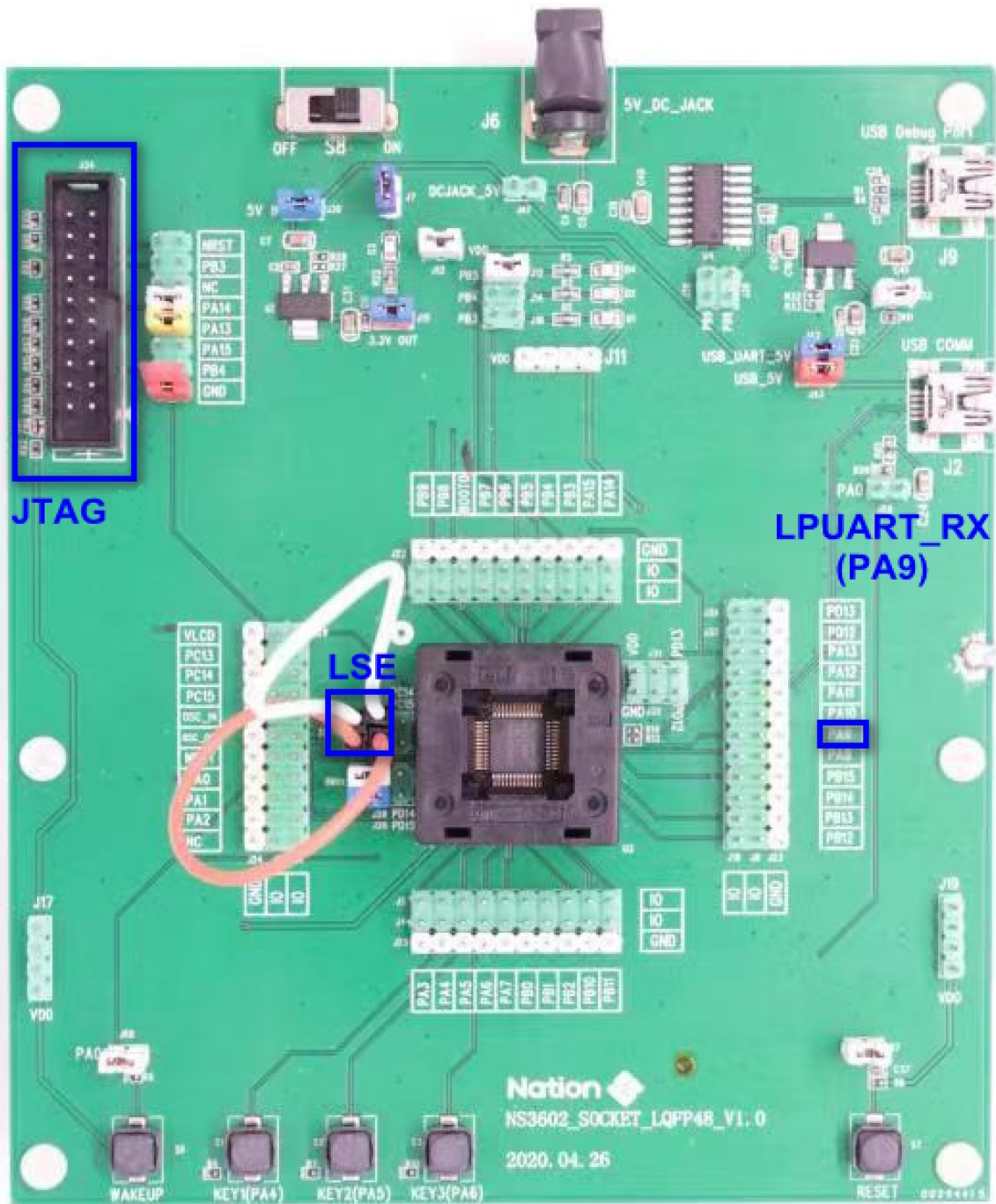


Table 2-1 Hardware List

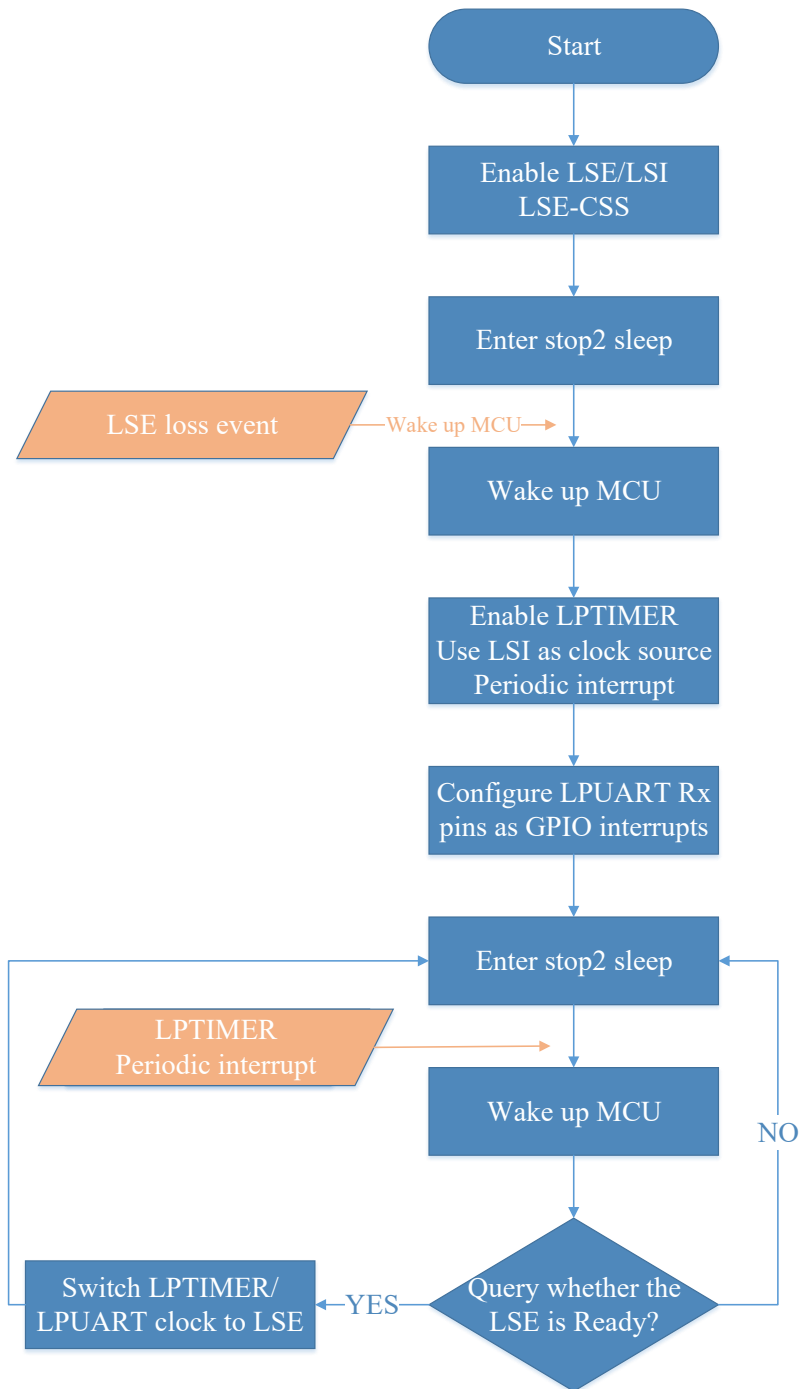
No.	Resource	Illustrate	Remark
1	NS3602_SOCKET_LQFP48_V1.0	Nations LQFP48 package test socket	

2	N32G435CBL7	MCU	
---	-------------	-----	--

3 Demo Explain

3.1 Demo Process

Figure 3–1 Demo Flowchart



3.2 Demo Analyze

3.2.1 USART1 Log Serial Port Initialization

Figure 3–2 USART1 Serial Port Initialization

```

70
71
72 /**
73  * @brief Main program
74  */
75 int main(void)
76 {
77     LEDInit(LED4_PORT, LED4_PIN);
78     /* Enable PWR Clock */
79     RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
80     /* should reinit after sysclk changed */
81     log_init();
82     log_info("\r\n MCU Reset \r\n");
83     USART1(Print serial port initialization)
84     User_EnableLse ();
85     User_EnableLsi ();
86
87     /* Enable Clock Security System(CSS):
88      * this will generate an NMI exception
89      * when LSE clock fails */
90     RCC_EnableLSEClockSecuritySystem(ENABLE);
91     User_ConfigLseCSSInt ();
92     /* EXTI19_TAMPER_IRQn configuration */
93     EXTI19_TAMPER_IRQn_Configuration();
94
95     /* Enable the DBG_STOP to keep debug in low power */
96     DBG_ConfigPeriph(DBG_STOP, ENABLE);
97     User_EnterSTOP2_Fun();
98
99     USER_LPTIM_Init();
100    User_LPUART_Init();
101    Delay(1000);
102    while(1)
103    {
104        Delay(100);
105        User_EnterSTOP2_Fun();
106    }
107 }
108
109
110
111

```

3.2.2 Enable LSE

Figure 3–3 User_EnableLse() Function

```

main.c
71
72 /**
73  * @brief Main program
74  */
75 int main(void)
76 {
77     LEDInit(LED4_PORT, LED4_PIN);
78     /* Enable FWR Clock */
79     RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_FWR, ENABLE);
80     /* Should reinit after sysclk changed */
81     log_init();
82     log_info("\r\nMCU Reset.\r\n");
83
84     User_EnableLse();
85     User_EnableLsi();
86
87     /* Enable Clock Security System(CSS):
88     ... this will generate an NMI exception
89     ... when LSE clock fails */
90     RCC_EnableLSEClockSecuritySystem(ENABLE);
91     User_ConfigLseCSSInt();
92
93     /* EXTI19_TAMPER_IRQn configuration */
94     EXTI19_TAMPER_IRQn_Configuration();
95     /* Enable the DBG_STOP to keep debug in low power */
96     DBG_ConfigPeriph(DBG_STOP, ENABLE);
97     User_EnterSTOP2_Fun();
98
99     USER_LPTIM_Init();
100    User_LPUART_Init();
101    Delay(1000);
102    while(1)
103    {
104        Delay(100);
105        User_EnterSTOP2_Fun();
106    }
107 }
    
```

```

main.c User_LSE_LSI_Config.c
19  * DISCLAIMED. IN NO EVENT SHALL NATIONS BE LIABLE FOR ANY DIRECT
20  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCL
21  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
22  * OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON AN
23  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INC
24  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
25  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
26  *
27  *
28
29 /**
30  * @file User_LSE_LSI_Config.c
31  * @author Nations
32  * @version v1.0.0
33  *
34  * @copyright Copyright (c) 2019, Nations Technologies Inc. All r
35  */
36
37
38 #include "n32143x.h"
39 #include "User_LSE_LSI_Config.h"
40
41 void User_EnableLse(void)
42 {
43     /* Enable the LSE_OSC32_IN_PC14
44     ... LSI is turned off here to ensure that only one clock is tu
45     ... RCC_EnableLsi(DISABLE);
46     ... RCC_ConfigLse(RCC_LSE_ENABLE);
47     ... while (RCC_GetFlagStatus(RCC_LDCTRL_FLAG_LSERD) == RESET);
48 }
49
50
51
52 void User_EnableLsi(void)
53 {
54     /* Enable the LSI_OSC */
55     RCC_EnableLsi(ENABLE);
56     while (RCC_GetFlagStatus(RCC_CTRLSTS_FLAG_LSIRD) == RESET);
57 }
    
```

Enable LSE

3.2.3 Enable LSI

Figure 3–4 User_EnableLsi() Function

```

main.c
70
71
72 /**
73  * @brief Main program
74  */
75 int main(void)
76 {
77     LEDInit(LED4_PORT, LED4_PIN);
78     /* Enable FWR Clock */
79     RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_FWR, ENABLE);
80     /* Should reinit after sysclk changed */
81     log_init();
82     log_info("\r\n MCU Reset.\r\n");
83
84     User_EnableLse();
85     User_EnableLsi();
86
87     /* Enable Clock Security System (CSS) :
88      * this will generate an NMI exception
89      * when LSE clock fails */
90     RCC_EnableLSEClockSecuritySystem(ENABLE);
91     User_ConfigLseCSSInt();
92
93     /* EXTI19_TAMPER_IRQn configuration */
94     EXTI19_TAMPER_IRQn_Configuration();
95     /* Enable the DBG_STOP to keep debug in low power */
96     DBG_ConfigPeriph(DBG_STOP, ENABLE);
97     User_EnterSTOP2_Fun();
98
99     USER_LPTIM_Init();
100    User_LPUART_Init();
101    Delay(1000);
102    while(1)
103    {
104        Delay(100);
105        User_EnterSTOP2_Fun();
106    }
107 }
    
```

```

User_LSE_LSI_Config.c
20 /* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLU
21 /* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS O
22 /* OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
23 /* LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCL
24 /* NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF T
25 /* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
26 .....
27
28
29 /**
30  * @file User_LSE_LSI_Config.c
31  * @author Nations
32  * @version v1.0.0
33  *
34  * @copyright Copyright (c) 2019, Nations Technologies Inc. All ri
35  */
36
37
38 #include "n32l43x.h"
39 #include "User_LSE_LSI_Config.h"
40
41 void User_EnableLse(void)
42 {
43     /* Enable the LSE-OSC2_IN-PC14
44     * LSI is turned off here to ensure that only one clock is tur
45     * RCC_EnableLsi(DISABLE);
46     * RCC_ConfigLse(RCC_LSE_ENABLE);
47     * while (RCC_GetFlagStatus(RCC_LDCTRL_FLAG_LSERD) == RESET);
48 }
49
50
51
52 void User_EnableLsi(void)
53 {
54     /* Enable the LSI-OSC */
55     RCC_EnableLsi(ENABLE);
56     while (RCC_GetFlagStatus(RCC_CTRLSTS_FLAG_LSIRD) == RESET);
57 }
    
```

3.2.4 Enable LSE-CSS Monitor

Figure 3–5 RCC_EnableLSEClockSecuritySystem() Function

```

70 |
71 |
72 | /**
73 |  * @brief Main program
74 |  */
75 | int main(void)
76 | {
77 |     LEDInit(LED4_PORT, LED4_PIN);
78 |     /* Enable PWR Clock */
79 |     RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
80 |     /* should reinit after sysclk changed */
81 |     log_init();
82 |     log_info("\r\n MCU Reset \r\n");
83 |
84 |     User_EnableLse();
85 |     User_EnableLsi();
86 |
87 |     /* Enable Clock Security System (CSS) :
88 |      * this will generate an NMI exception
89 |      * when LSE clock fails */
90 |     RCC_EnableLSEClockSecuritySystem(ENABLE);
91 |     User_ConfigLseCSSInt();
92 |     /* EXTI19_TAMPER_IRQn configuration */
93 |     EXTI19_TAMPER_IRQn_Configuration();
94 |
95 |     /* Enable the DBG_STOP to keep debug in low power */
96 |     DBG_ConfigPeriph(DBG_STOP, ENABLE);
97 |     User_EnterSTOP2_Fun();
98 |
99 |     USER_LPTIM_Init();
100 |     User_LPUART_Init();
101 |     Delay(1000);
102 |     while(1)
103 |     {
104 |         Delay(100);
105 |         User_EnterSTOP2_Fun();
106 |     }
107 | }

```

```

1613 | /**
1614 |  * @brief Enables or disables the LSE Clock Security System.
1615 |  * @param Cmd new state of the LSE Clock Security System.
1616 |  * This parameter can be: ENABLE or DISABLE.
1617 |  */
1618 | void RCC_EnableLSEClockSecuritySystem(FunctionalState Cmd)
1619 | {
1620 |     /* Check the parameters */
1621 |     assert_param(IS_FUNCTIONAL_STATE(Cmd));
1622 |     *(_IO_uint32_t*)LDCTRL_LSECLKSSEN_BB = (uint32_t)Cmd;
1623 | }

```

```

61 | void User_ConfigLseCSSInt(void)
62 | {
63 |     RCC->CLKINT |= (1 << 25);
64 | }
65 |

```

3.2.5 Enter STOP2

Figure 3–6 User_EnterSTOP2_Fun() Function

```

main.c
65 ..... /* Request to enter STOP2 mode */
66 ..... FWR_EnterSTOP2Mode(FWR_STOPENTRY_WFI, FWR_CTRL3_RAM1RET);
67 ..... log_info("\n MCU Wakeup From STOP2 Mode \n");
68 }
69
70
71
72 /**
73  * @brief Main program
74  */
75 int main(void)
76 {
77 ..... LEDInit(LED4_PORT, LED4_PIN);
78 ..... /* Enable PWR Clock */
79 ..... RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
80 ..... /* should reinit after sysclk changed */
81 ..... log_init();
82 ..... log_info("\n MCU Reset \n");
83 .....
84 ..... User_EnableLse();
85 ..... User_EnableLsi();
86 .....
87 ..... /* Enable Clock Security System (CSS):
88 ..... this will generate an NMI exception
89 ..... when LSE clock fails */
90 ..... RCC_EnableLSEClockSecuritySystem(ENABLE);
91 ..... User_ConfigLseCSSInt();
92 ..... /* EXTI19_TAMPER_IRQn configuration */
93 ..... EXTI19_TAMPER_IRQn_Configuration();
94 .....
95 ..... /* Enable the DBG_STOP to keep debug in low power */
96 ..... DBG_ConfigPeriph(DBG_STOP, ENABLE);
97 ..... User_EnterSTOP2_Fun();
98 .....
99 ..... USER_LPTIM_Init();
100 ..... User_LPUART_Init();
101 ..... Delay(1000);
102 ..... while(1)
103 ..... {
104 ..... Delay(100);
105 ..... User_EnterSTOP2_Fun();
106 ..... }
107 }
108
62 void User_EnterSTOP2_Fun(void)
63 {
64 ..... log_info("\n MCU Goto STOP2 Mode \n");
65 ..... /* Request to enter STOP2 mode */
66 ..... FWR_EnterSTOP2Mode(FWR_STOPENTRY_WFI, FWR_CTRL3_RAM1RET);
67 ..... log_info("\n MCU Wakeup From STOP2 Mode \n");
68 }

```

Enter stop2 sleep mode

3.2.6 Initialize LPTIM and LPUART

Figure 3–7 USER_LPTIM_Init() and User_LPUART_Init() Function

```

72 /**
73  * @brief Main program
74  */
75 int main(void)
76 {
77     LEDInit(LED4_PORT, LED4_PIN);
78     /* Enable FWR Clock */
79     RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_FWR, ENABLE);
80     /* should reinit after sysclk changed */
81     log_init();
82     log_info("\r\n MCU Reset \r\n");
83     User_EnableLse();
84     User_EnableLsi();
85
86
87     /* Enable Clock Security System (CSS) :
88     * this will generate an NMI exception
89     * when LSE clock fails */
90     RCC_EnableLSEClockSecuritySystem(ENABLE);
91     User_ConfigLseCSSInt();
92     /* EXTI19_TAMPER_IRQn configuration */
93     EXTI19_TAMPER_IRQn_Configuration();
94
95     /* Enable the DBG_STOP to keep debug in low power */
96     DBG_ConfigPeriph(DBG_STOP, ENABLE);
97     User_EnterSTOP2_Fun();
98
99     USER_LPTIM_Init();
100    User_LPUART_Init();
101    Delay(1000);
102    while(1)
103    {
104        Delay(100);
105        User_EnterSTOP2_Fun();
106    }
107 }

```

```

68 void User_LPUART_Init(void)
69 {
70     LPUART_InitType LPUART_InitStructure;
71     /* Configure the GPIO ports */
72     GPIO_Configuration();
73
74     /* System Clocks Configuration */
75     RCC_Configuration(RCC_LPUARTCLK_SRC_LSE);
76
77     /* LPUART configuration */
78     LPUART_DeInit();
79     LPUART_StructInit(&LPUART_InitStructure);
80     LPUART_InitStructure.BaudRate = 9600;
81     LPUART_InitStructure.Parity = LPUART_PE_NO;
82     LPUART_InitStructure.RtsThreshold = LPUART_RTSTH_FIFOFU;
83     LPUART_InitStructure.HardwareFlowControl = LPUART_HFCTRL_NONE;
84     LPUART_InitStructure.Mode = LPUART_MODE_RX | LPUART_MODE_TX;
85     /* Configure LPUART */
86     LPUART_Init(&LPUART_InitStructure);
87 }

```

```

67 void USER_LPTIM_Init(void)
68 {
69     /* Enable interrupt */
70     LPTIMNVIC_Config(ENABLE);
71     RCC_ConfigLPTIMClk(RCC_LPTIMCLK_SRC_LSE);
72     RCC_EnableRETPeriphClk(RCC_RET_PERIPH_LPTIM, ENABLE);
73
74     LPTIM_SetPrescaler(LPTIM, LPTIM_PRESCALER_DIV4);
75     LPTIM_EnableIT_CMPM(LPTIM);
76     /* config lptim ARR and compare register */
77     LPTIM_Enable(LPTIM);
78     LPTIM_SetAutoReload(LPTIM, 65000);
79     LPTIM_SetCompare(LPTIM, 60000);
80     LPTIM_StartCounter(LPTIM, LPTIM_OPERATING_MODE_CONTINUOUS);
81 }

```

Initialize
LPTIM/
LPUART

4 User Guide

4.1 Reset MCU

Figure 4–1 Reset MCU

```

/**
 * @brief Main program
 */
int main(void)
{
    LEDInit(LED4_PORT, LED4_PIN);
    /* Enable PWR Clock */
    RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
    /* should reinit after sysclk changed */
    log_init();
    log_info("\r\n MCU Reset \r\n");

    User_EnableLse();
    User_EnableLsi();

    /* Enable Clock Security System(CSS):
     * this will generate an NMI exception
     * when LSE clock fails */
    RCC_EnableLSEClockSecuritySystem(ENABLE);
    User_ConfigLseCSSInt();
    /* EXTI19_TAMPER_IRQn configuration */
    EXTI19_TAMPER_IRQn_Configuration();

    /* Enable the DBG_STOP to keep debug in low power */
    DBG_ConfigPeriph(DBG_STOP, ENABLE);
    User_EnterSTOP2_Fun();

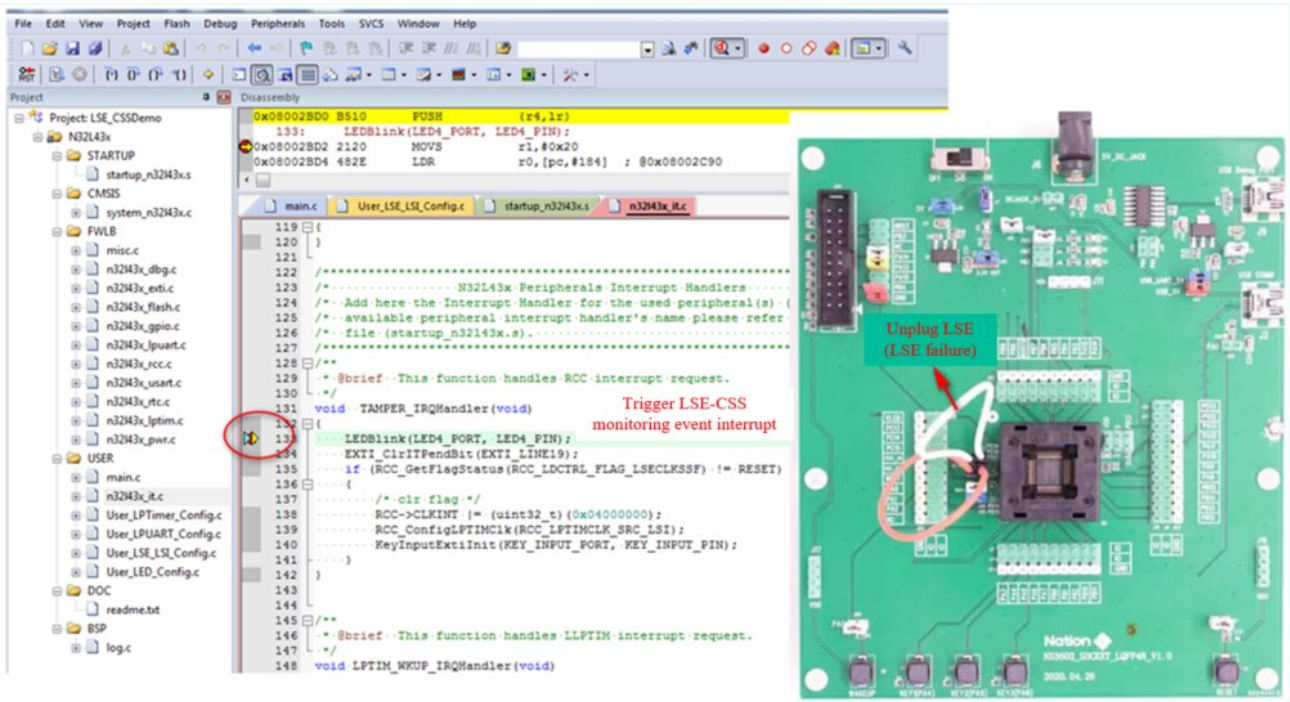
    USER_LPTIM_Init();
    User_LPUART_Init();
    Delay(1000);
    while(1)
    {
        Delay(100);
        User_EnterSTOP2_Fun();
    }
}
    
```

MCU Reset
MCU Goto STOP2 Mode

Downloader reset MCU MCU enters stop2 mode

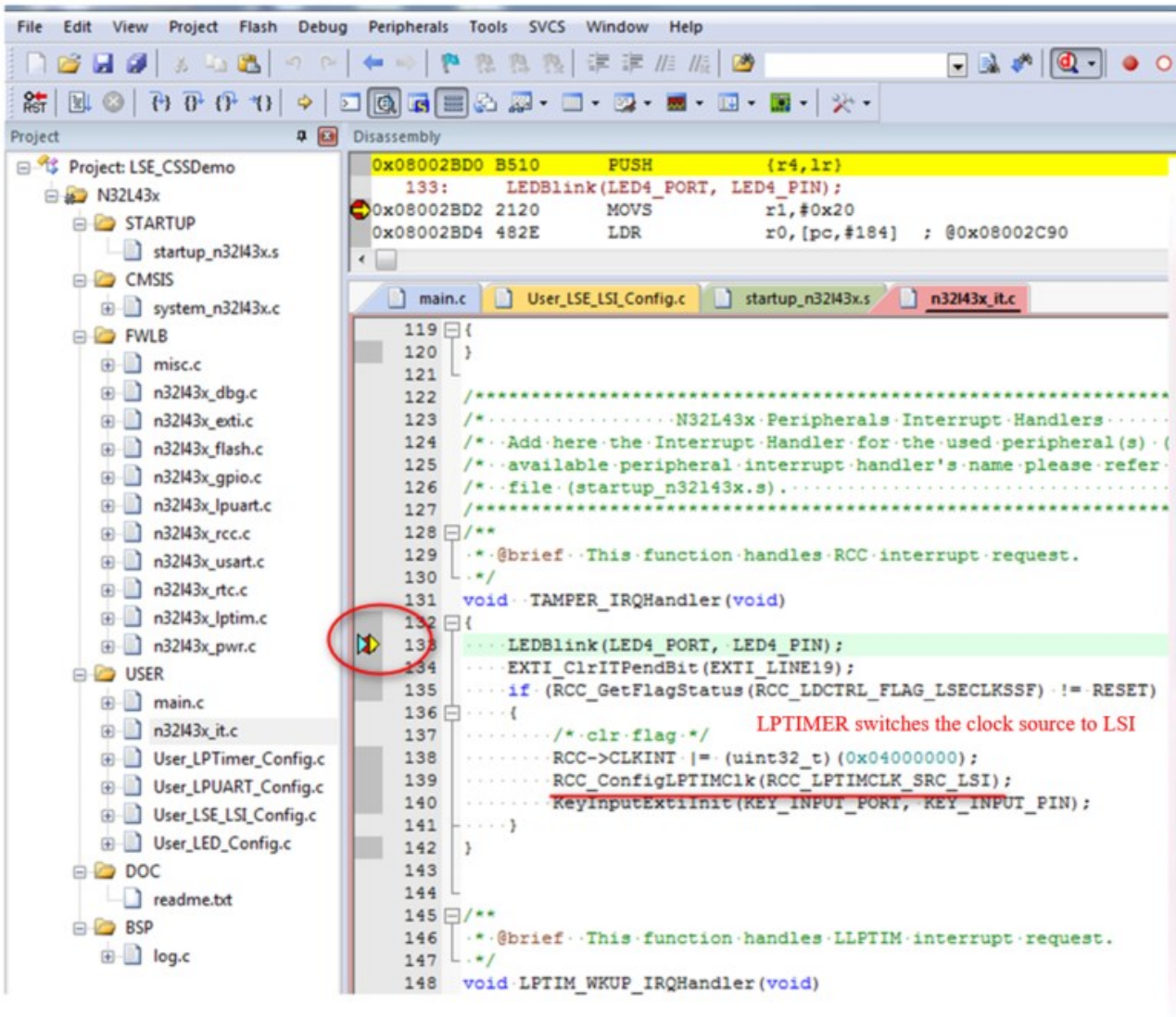
4.2 Generate LSE Fault

Figure 4–2 Generate LSE Fault



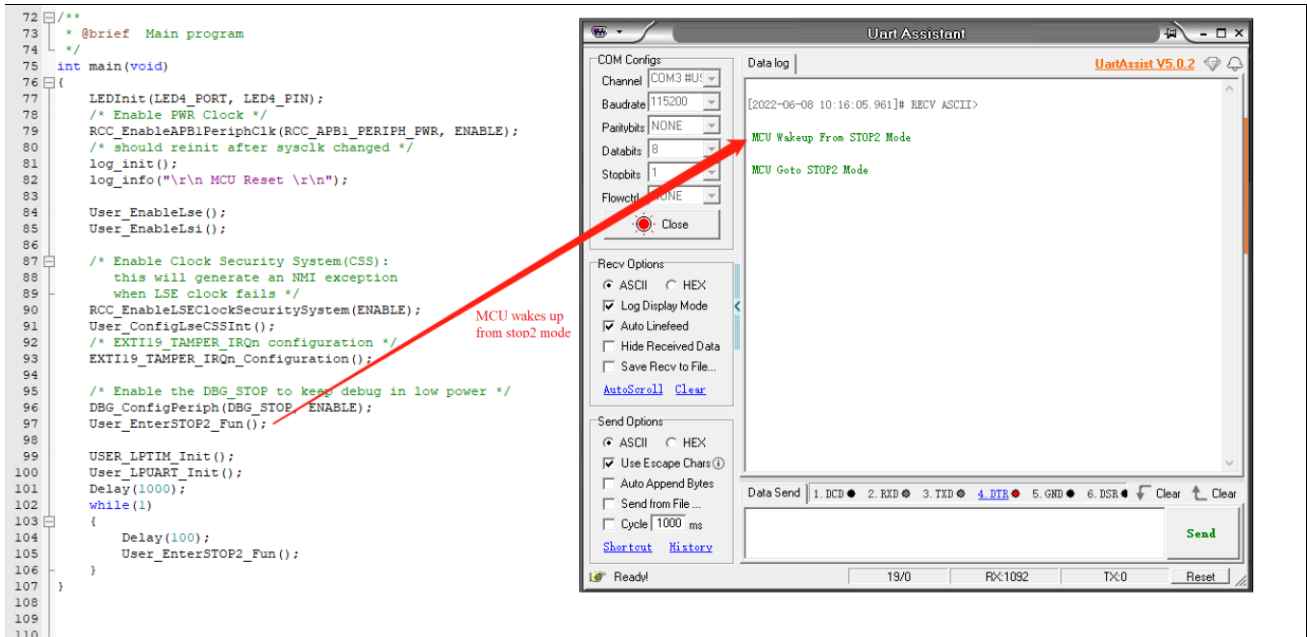
4.3 LPTIMER Switch Clock Source

Figure 4–3 LPTIMER Switch Clock Source



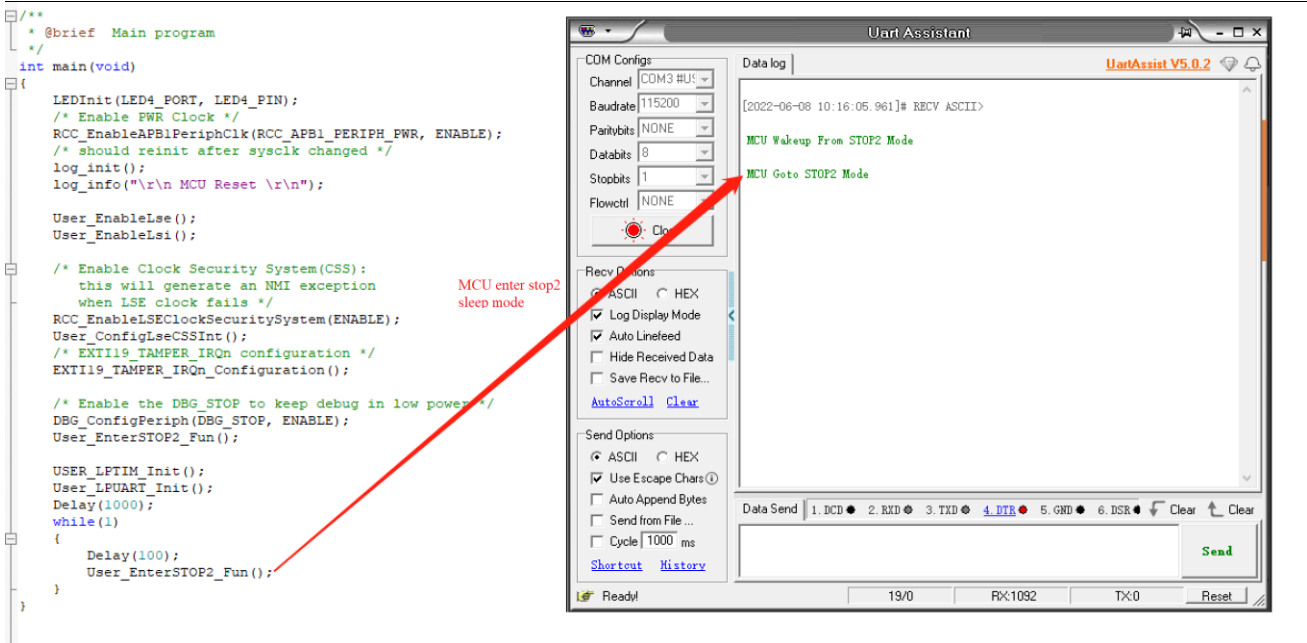
4.4 MCU Wakes Up From STOP2

Figure 4-4 MCU Wake Up from STOP2 Mode



4.5 MCU Enter STOP2 Sleep Mode

Figure 4-5 MCU Enter STOP2 Mode



4.6 MCU Periodically Woken Up

Figure 4-6 MCU Periodically Woken Up

The image shows a code editor on the left and a Uart Assistant window on the right. The code in the editor is as follows:

```

/**
 * @brief Main program
 */
int main(void)
{
    LEDInit(LED4_PORT, LED4_PIN);
    /* Enable PWR Clock */
    RCC_EnableAPB1PeriphClk(RCC_APB1_PERIPH_PWR, ENABLE);
    /* should reinit after sysclk changed */
    log_init();
    log_info("\r\n MCU Reset \r\n");

    User_EnableLse();
    User_EnableLsi();

    /* Enable Clock Security System(CSS):
    this will generate an NMI exception
    when LSE clock fails */
    RCC_EnableLSEClockSecuritySystem(ENABLE);
    User_ConfigLseCSSInt();
    /* EXTI19_TAMPER_IRQn configuration */
    EXTI19_TAMPER_IRQn_Configuration();

    /* Enable the DBG_STOP to keep debug in low power */
    DBG_ConfigPeriph(DBG_STOP, ENABLE);
    User_EnterSTOP2_Fun();

    USER_LPTIM_Init();
    User_LPUART_Init();
    Delay(1000);
    while(1)
    {
        Delay(100);
        User_EnterSTOP2_Fun();
    }
}
    
```

A red box highlights the `Delay(100);` and `User_EnterSTOP2_Fun();` lines in the `while(1)` loop. A red arrow points from this box to the Uart Assistant window. A red text annotation says "MCU is periodically woken up by LPTIMER".

The Uart Assistant window shows the following data log:

```

[2022-06-08 10:25:30.236]# RECV ASCII>
MCU Wakeup From STOP2 Mode

MCU Goto STOP2 Mode

[2022-06-08 10:25:38.158]# RECV ASCII>
MCU Wakeup From STOP2 Mode

MCU Goto STOP2 Mode

[2022-06-08 10:25:46.111]# RECV ASCII>
MCU Wakeup From STOP2 Mode

MCU Goto STOP2 Mode

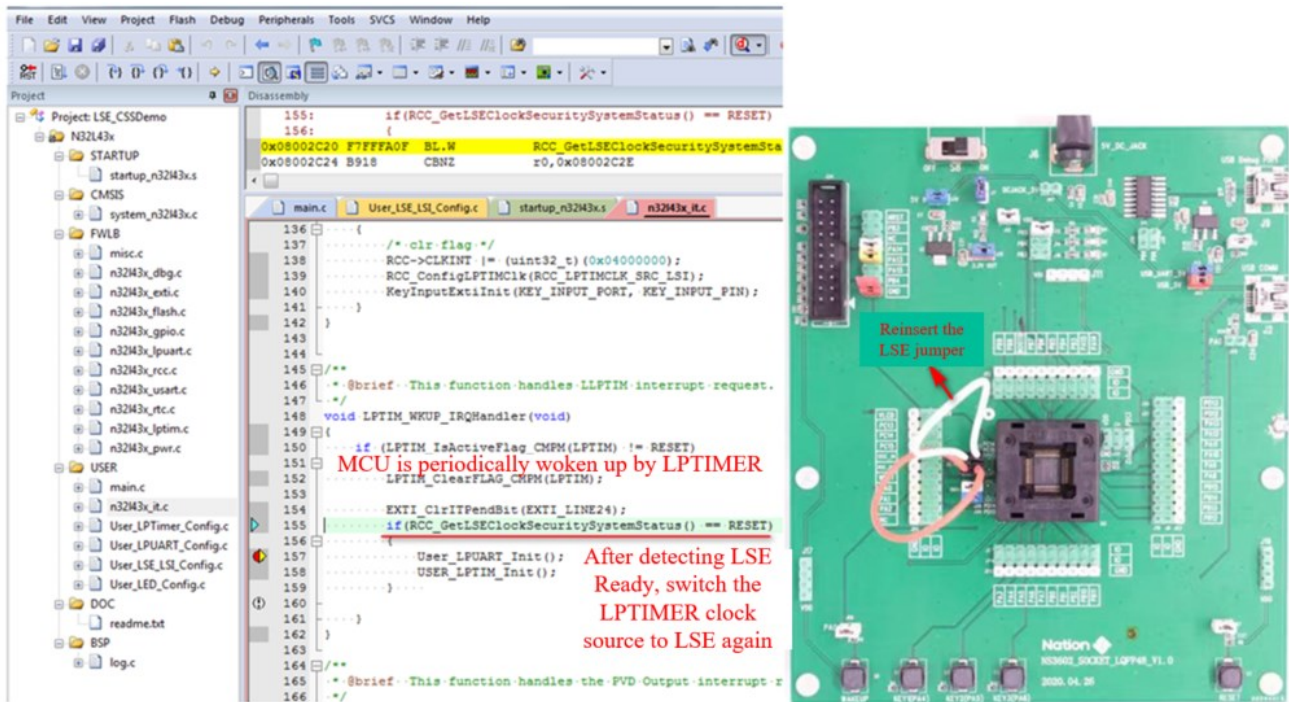
[2022-06-08 10:25:54.043]# RECV ASCII>
MCU Wakeup From STOP2 Mode

MCU Goto STOP2 Mode
    
```

The Uart Assistant window also shows COM Configs (Channel: COM3 #HS, Baudrate: 115200, Paritybits: NONE, Databits: 8, Stopbits: 1, Flowctrl: NONE) and Recv Options (ASCII selected, Log Display Mode checked, Auto Linefeed checked, Hide Received Data unchecked, Save Recv to File... unchecked, AutoScroll checked, Clear button).

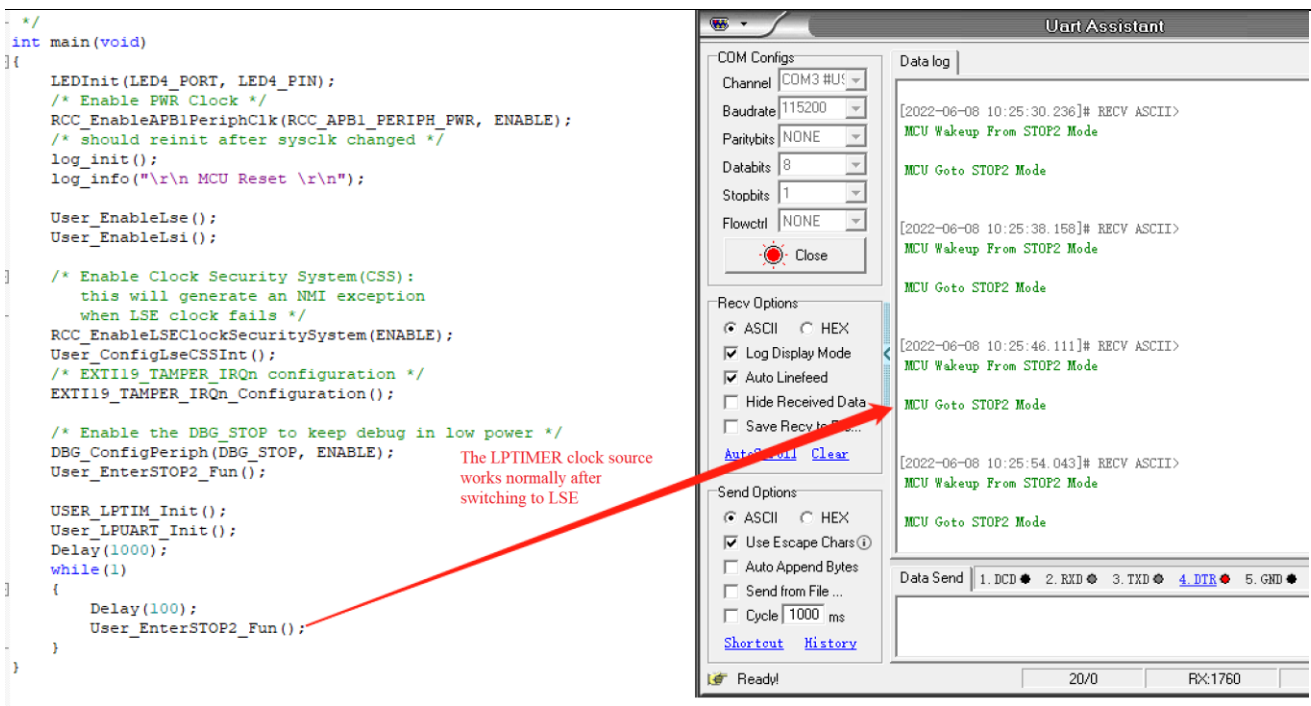
4.7 LSE Recover

Figure 4-7 LSE Recover



4.8 LPTIMER Switches The Clock Source to LSE

Figure 4-8 LPTIMER Switches the Clock Source to LSE



5 Version History

Version	Date	Changes
V1.0	2020.11.30	Initial release

6 Disclaimer

This document is the exclusive property of NSING TECHNOLOGIES PTE. LTD.(Hereinafter referred to as NSING). This document, and the product of NSING described herein (Hereinafter referred to as the Product) are owned by NSING under the laws and treaties of Republic of Singapore and other applicable jurisdictions worldwide. The intellectual properties of the product belong to Nations Technologies Inc. and Nations Technologies Inc. does not grant any third party any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only. NSING reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NSING and obtain the latest version of this document before placing orders. Although NATIONS has attempted to provide accurate and reliable information, NATIONS assumes no responsibility for the accuracy and reliability of this document. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NATIONS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product.

NATIONS Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, 'Insecure Usage'. Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to supporter sustain life. All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage Any express or implied warranty with regard to this document or the Product, including, but not limited to. The warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law. Unless otherwise explicitly permitted by NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.

