AN13057
Understanding Cap Bank in LPC55(S)xx

Rev. 0 — 16 November 2020

Application Note

by: NXP Semiconductors

Contents

1 How to read the application note

All LPC55(S)xx products have two crystal oscillators: A 16 MHz crystal oscillator and a 32 KHz crystal oscillator.

Each crystal oscillator has one embedded capacitor bank, which is always enabled and its capacitance can be adjusted in certain range (IEC equivalent capacitive load: 6 - 10 pF). The capacitor bank can be used as integrated load capacitors for the crystal oscillators, which helps save external load capacitors of the crystal and hence the system BOM cost.

This application note provides a short introduction to this capacitor bank or **Cap Bank**, and describes how to apply this function to the actual design.

2 Advantage of embedded Cap Bank

The advantages of the embedded Cap Bank are as follows:

- Conserves component area on the PCB.
- Saves Bill of Materials (BOM).
- Maintains a Capacitive Load (CL) between 6 - 10 pF (IEC equivalent).
- Relies on simple APIs to configure the Capacitor Banks based on the crystal Capacitive Load (CL) and measured PCB parasitic capacitances on the XIN and XOUT pins.

3 Understanding Cap Bank

3.1 Cap Bank block diagram

The following figure shows the equivalent internal Cap Bank block with a standard crystal connection diagram.

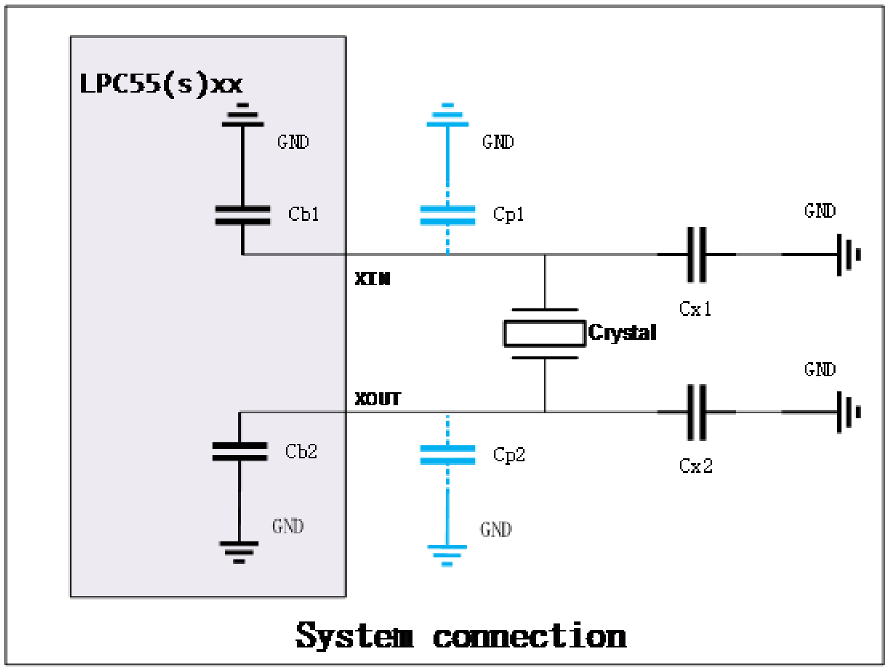


Figure 1. Cap Bank block diagram

- Cb1, Cb2: Total equivalent capacitor including capacitor bank, package, and pad parasitic capacitors in MCU in single ended mode (i.e., on single crystal pin)
- Cp1, Cp2: Equivalent PCB parasitic capacitances on the PCB board in single ended mode
- Cx1, Cx2: User external crystal load capacitors used in single ended mode (i.e., on each crystal pin)

The single ended total capacitive load seen by the external crystal:

CLtot_se1= Cb1 +Cp1+Cx1; (tot is short for total)

CLtot_se2= Cb2 +Cp2+Cx2;

The total IEC equivalent capacitive load seen by the external crystal (or CL) is the series of both single ended total capacitive loads:

CLtot_IEC= CL = (CLtot_se1* CLtot_se2)/(CLtot_se1+ CLtot_se2).

3.2 Cap Bank specification in LPC55(S)xx

- Cb1 and Cb2 are equal.
- Cb1 or Cb2 is 12pF - 20pF, so the range of IEC equivalent crystal load capacitance value for internal Cap Bank is 6pF - 10pF.
- The reset value (default value) of Cb1 or Cb2 is 12pF, so the reset IEC equivalent crystal load capacitance is 6pF.
- All these values are guaranteed at factory across the device working temperature range.

4 Use cases

Crystal Load capacitance (CL) value can be easily found in the crystal data sheet. For example, 6pF, 8pF, 10pF, 12pF, and so on. The aforementioned CLtot_IEC , total IEC equivalent capacitive load needed for crystal is equal to CL.

Firstly, CLtot_IEC (or CL) needs to be converted into single ended total capacitive value as follows:

CLtot_se1 = CLtot_se2 = 2*CLtot_IEC = 2* CL

Secondly, the parasitic capacitances of the PCB (Cp1 and Cp2) related to the crystal on the PCB board need be measured.

Then the following results are calculated:

Cb1 + Cx1 = CLtot_se1 – Cp1 = CL – Cp1

Cb2 + Cx2 = CLtot_se2 – Cp2 = CL – Cp2

The following sections shows different configurations of Cb1/2 and Cx1/2 for different use cases.

4.1 Use case 1: CL (or CLtot_IEC) = 8pF, Cp1 = 2pF, Cp2 = 3pF

Assume a crystal with load capacitance (CL) of 8pF is used, then

Cb1 + Cx1 = CLtot_se1 – Cp1 = CL – Cp1 = 2* 8pF – 2pf = 14pF

Cb2 + Cx2 = CLtot_se2 – Cp2 = CL – Cp2 = 2* 8pF – 3pf = 13pF;

Because the range of Cb1 and Cb2 values are 12pF - 20pF, which already covers the above calculated expected capacitance 14pF/13pF, the Cx1 and Cx2 values can be zero. In other words, there is no need to add Cx1 and Cx2 on PCB. This can save components and PCB area.

Therefore, just configure Cb1 = 14pF, Cb2 = 13pF. Because Cb1 and Cb2 are equal in LPC55(S)xx, the maximum value between them is taken normally.

Therefore, Cb1 = Cb2 = 14pF.

4.2 Use case 2: external crystal load capacitor needed

Assume a crystal with load capacitance (CL) of 15pF is used, then

CLtot_IEC = CL = 15pF, Cp1 = 2pF, Cp2 = 2pF

Cb1 + Cx1 = CLtot_se1 – Cp1 = 2* 15pF – 2pf = 28pF

Cb2 + Cx2 = CLtot_se2 – Cp2 = 2* 15pF – 2pf = 28pF

Because the range of Cb1 and Cb2 values are 12pF - 20pF, which is below the above calculated expected capacitance 28pF, the Cx1 and Cx2 are needed. In other words, Cx1 and Cx2 must be added on PCB.

In this case, the recommended values of Cb1 and Cb2 are taken as 16pF normally,

Therefore, Cx1/x2 can be derived as follows:

Cb1 = Cb2 = 16pF

Cx1 = Cx2 = 28pF – 16pF = 12pF

NOTE

If Cap Bank is not configured, Cb1 and Cb2 keep the default value, which is 12pF.

In this case, Cx1/x2 can be derived as follows:

Cb1 = Cb2 = 12pF

Cx1 = Cx2 = 28pF – 12pF = 16pF

5 Capacitor Banks API description

Capacitor banks API routines are used to configure the Cap Bank values of Cb1 and Cb2.

For user convenience, it only requires users to provide Cbp, Cp1, and Cp2.

Here, Cbp is the actual IEC equivalent load capacitance from crystal point of view **without** extra capacitors Cx1, Cx2, that is, Cbp is:

Cbp = (Cb1+Cp1) * (Cb2+Cp2)/((Cb1+Cp1) + (Cb2+Cp2))

1. If Cbp = CL, it does not need to add external crystal load capacitor Cx1 and Cx2. That is, if there is no need to add external crystal load capacitor Cx1 and Cx2 per use case 1 in section 4.1 (i.e., CL is within the range of 6 - 10 pF), then the Cbp is equal to total IEC equivalent capacitive load CLtot_IEC or the crystal load capacitance CL:

   Cbp = CLtot_IEC = CL;

   See the equivalent diagram below.

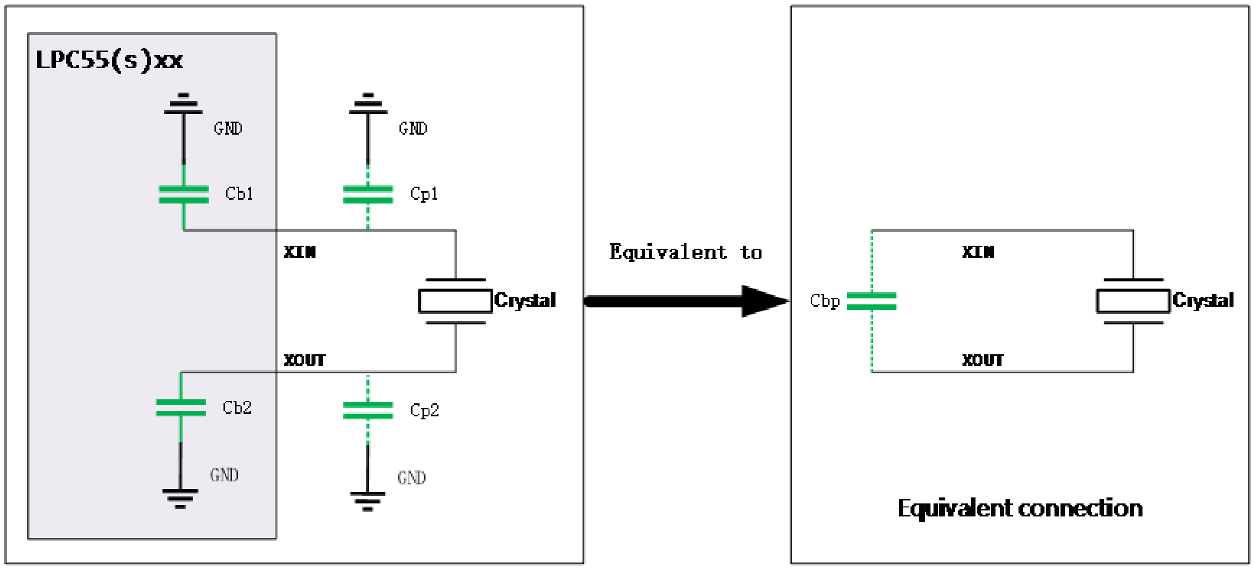


Figure 2. Equivalent diagram

2. If Cbp < CL, then the external crystal load capacitance Cx must be added. That is, if there is need to add Cx1 and Cx2 per use case 2 in section 4.2, and Cbp + Cx = CLtot_IEC = CL, Where Cx = (Cx1* Cx2)/ (Cx1+ Cx2), then the external load capacitance Cx can be derived by: Cx = CL – Cbp.

   Assume Cx1 = Cx2, then Cx1 = Cx2 = 2*Cx = 2*(CL – Cbp).

   In this case, it is recommended that the internal Cap Bank value Cb1, Cb2 = 16 pF.

   See the equivalent diagram below.

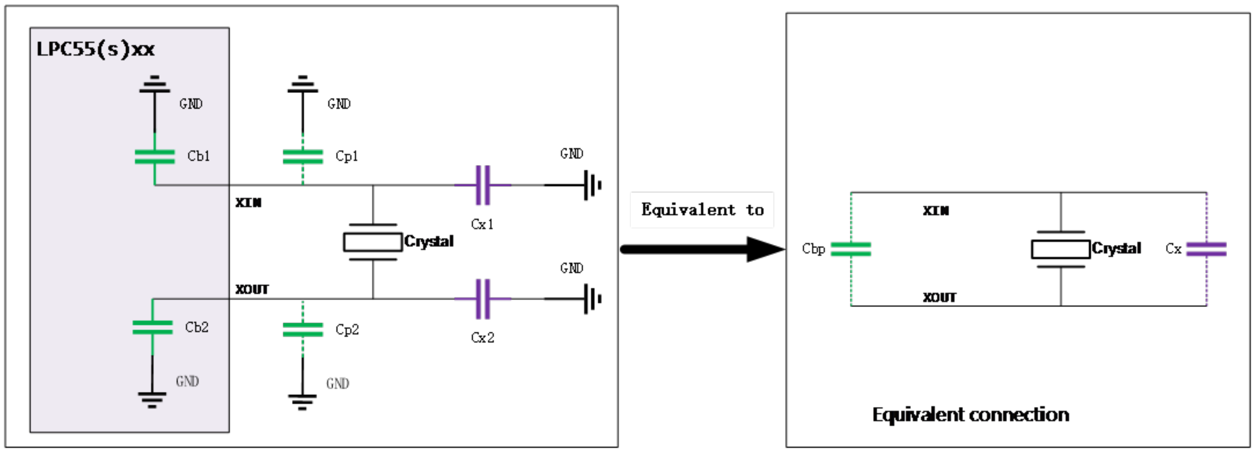


Figure 3. Equivalent diagram

Cp1 and Cp2 are related to the board, the user needs to provide their values to the Cap Bank API function as the second and third parameters after measurement.

CL can be found in the crystal data sheet, so Cbp can be easily derived as mentioned previously and passed to the Cap Bank API routine as the first parameter.

5.1 API routines

There is only one routine for each crystal oscillator as follows.

Table 1. API routines

Function prototype	API description
void **POWER_Xtal16mhzCapbankTrim**(int32_t pi32_16MfXtalIecLoadpF_x100, int32_t pi32_16MfXtalPPcbParCappF_x100, int32_t pi32_16MfXtalNPcbParCappF_x100);	This API configures the internal Capacitor Bank of the 16 MHz crystal oscillator, where parameters: pi32_16MfXtalIecLoadpF_x100: **Cbp** (pF) x 100; pi32_16MfXtalPPcbParCappF_x100: **Cp1** (pF) x 100; pi32_16MfXtalNPcbParCappF_x100: **Cp2** (pF) x 100;
void **POWER_Xtal32mhzCapbankTrim**(int32_t pi32_32kfXtalIecLoadpF_x100, int32_t pi32_32kfXtalPPcbParCappF_x100, int32_t pi32_32kfXtalNPcbParCappF_x100);	This API configures the Capacitor Bank of the 32 KHz crystal oscillator, where parameters: pi32_32kfXtalIecLoadpF_x100: **Cbp** (pF) x 100; pi32_32kfXtalPPcbParCappF_x100: **Cp1** (pF) x 100; pi32_32kfXtalNPcbParCappF_x100: **Cp2** (pF) x 100;

In each routine, there are three parameters needed (Cbp, Cp1, Cp2). The unit of parameters is one hundredth pF.

For example, if

Cbp = 8.5pF, Cp1 = 2pF, Cp2 = 3pF.

Then, the call to API should be POWER_Xtal16mhzCapbankTrim (850,200,300).

5.2 API usage examples

This section explains the API usage examples described in the device’s user manual.

5.2.1 16 MHz Crystal Oscillator

5.2.1.1 Example 1: 8pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 3pF PCB parasitic capacitance on XOUT pin.

According to the conditions of this example, CLtot_IEC, Cp1/p2 are:

CLtot_IEC = CL = 8pF, Cp1 = 2pF, Cp2 = 3pF.

Because CL is within the range of 6 - 10 pF, it does not need to add external crystal load capacitor Cx1 and Cx2, and the Cbp is equal to CL = 8pF.

Therefore, the call to Cap Bank API routine should be: XTAL_16mhz_capabank_trim(8 * 100, 2 * 100, 3 * 100).

NOTE

Cb1 should be equal to Cb2 and be set to the maximum value of Cb1 and Cb2.

The system connection as follows:

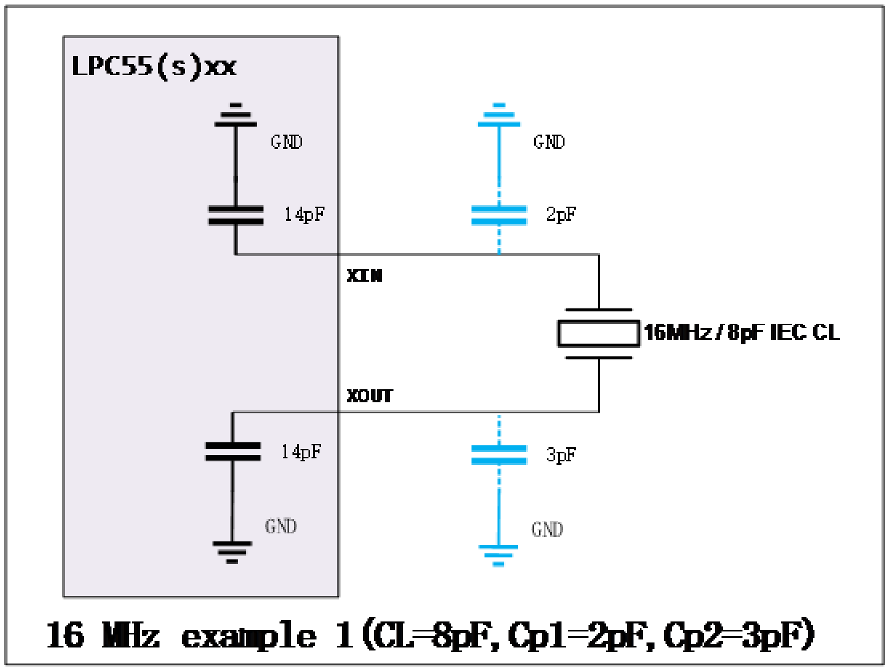


Figure 4. System connection

5.2.1.2 Example 2: 15pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 2pF PCB parasitic capacitance on XOUT pin.

According to the conditions of this example, CLtot_IEC, Cp1/p2 are:

CLtot_IEC = CL = 15pF, Cp1 = 2pF, Cp2 = 2pF.

Because CL exceeds 10 pF, external crystal load capacitance Cx must be added.

In another words, the user needs to add Cx1 and Cx2 per use case 2 in Section 4.2.

Cbp + Cx = CLtot_IEC = CL;

Then, the external load capacitance Cx can be derived by:

Cx = CL – Cbp;

Assume Cx1 = Cx2, then Cx1 = Cx2 = 2*Cx = 2*(CL – Cbp)

In this case, it is recommended that the internal Cap Bank value Cb1, Cb2 = 16pF.

Therefore:

Cbp = (Cb1+Cp1) * (Cb2+Cp2)/((Cb1+Cp1) + (Cb2+Cp2)) = 9pF.

Cx1 = Cx2 = 2*Cx = 2*(CL – Cbp)=12pF.

The call to Cap Bank API routine should be: XTAL_16mhz_capabank_trim(9 * 100, 2 * 100, 2 * 100);

The system connection as follows:

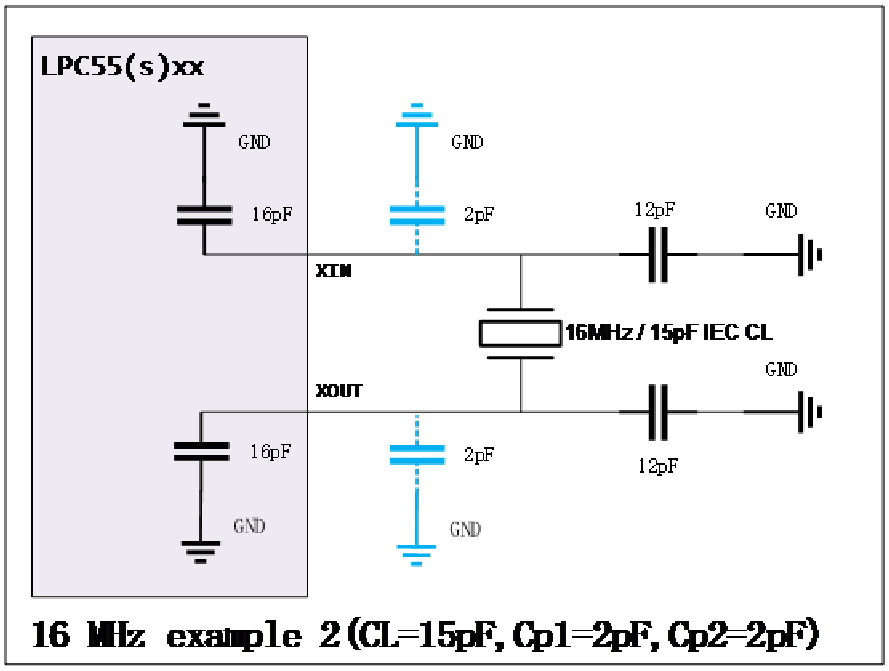


Figure 5. System connection

5.2.2 32 KHz Crystal Oscillator

5.2.2.1 Example 3: 8pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 3pF PCB parasitic capacitance on XOUT pin.

This is the same as example 1 aforementioned.

Therefore, the call to Cap Bank API routine should be: XTAL_32kHz_capabank_trim(8 * 100, 2 * 100, 3 * 100);

it docs not need to add extra capacitors on PCB.

The crystal system connection is as follows:

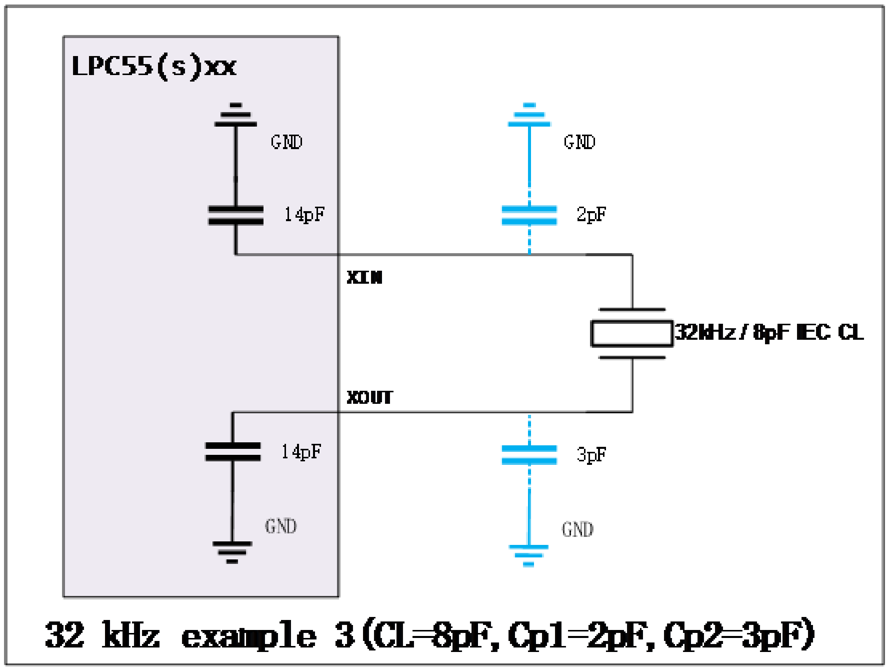


Figure 6. Crystal system connection

5.2.2.2 Example 4: 15pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 2pF PCB parasitic capacitance on XOUT pin.

This is the same as example 2 aforementioned.

Therefore, the call to Cap Bank API routine should be: XTAL_32kHz_capabank_trim(9 * 100, 2 * 100, 2 * 100);

And the user needs to add extra capacitors on PCB: Cx1 = Cx2 = 12pF.

The crystal system connection is as follows:

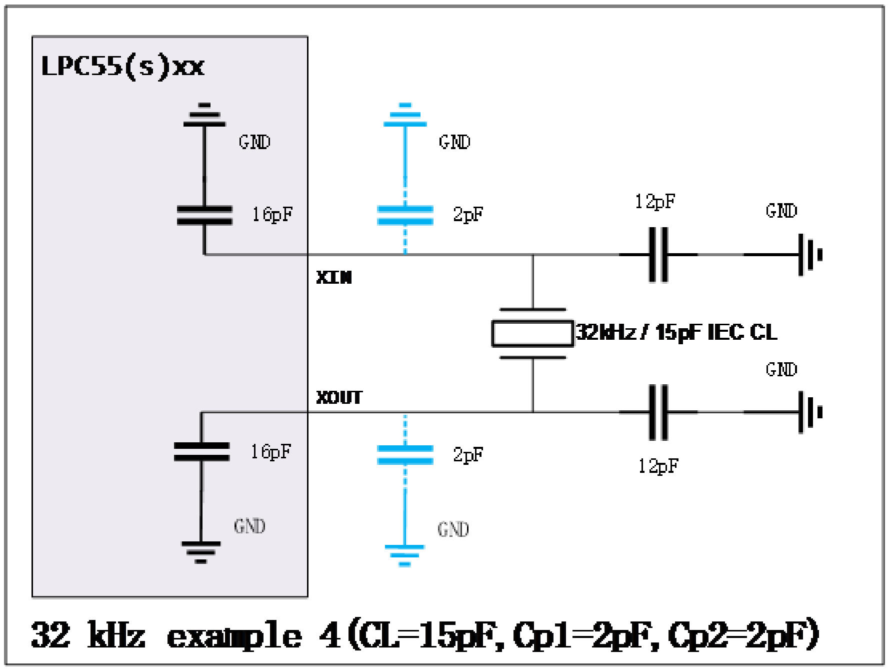


Figure 7. Crystal system connection

6 SDK example

Here is an example code that shows how to configure the internal Cap Bank, and how to fine tune it to achieve the best accuracy of crystal oscillator output on the existing PCB.

6.1 Hardware preparation

There are two external crystal oscillators on the board: 16 MHz and 32.768 KHz. This application note uses 32.768 KHz as an example to describe how to configure the Cap Bank function.

The part number of the 32k crystal on the EVK board is NX3215SA-32.768K-STD-MUS-2. It can be concluded from the manual that its IEC Capacitance Load (CL) is 12.5 pF, which does exceed the Cap Bank trim range of LPC5500 (6 – 10 pF).

Therefore, Cbp < CL, and the external crystal load capacitance Cx must be added. In another words, the user needs to add Cx1 and Cx2 per use case 2 in Section 4.2, and

Cbp + Cx = CLtot_IEC = CL;

Where Cx = (Cx1* Cx2)/ (Cx1+ Cx2).

Then, the external load capacitance Cx can be derived by:

Cx = CL – Cbp;

Assume Cx1 = Cx2, then Cx1 = Cx2 = 2*Cx = 2*(CL – Cbp).

In this case, it is recommended that the internal Cap Bank value Cb1, Cb2 = 16 pF, and Cp1 = 1pF, Cp2 = 1pF.

Therefore,

Cbp = (Cb1+Cp1) * (Cb2+Cp2)/((Cb1+Cp1) + (Cb2+Cp2)) = 8.5pF.

Cx1 = Cx2 = 2*Cx = 2*(CL – Cbp)=8pF.

The call to the Cap Bank API routine should be: XTAL_16mhz_capabank_trim(850, 100, 100);

That is, user needs to replace C78 and C79 with 8pF capacitors on the board as the following picture shows.

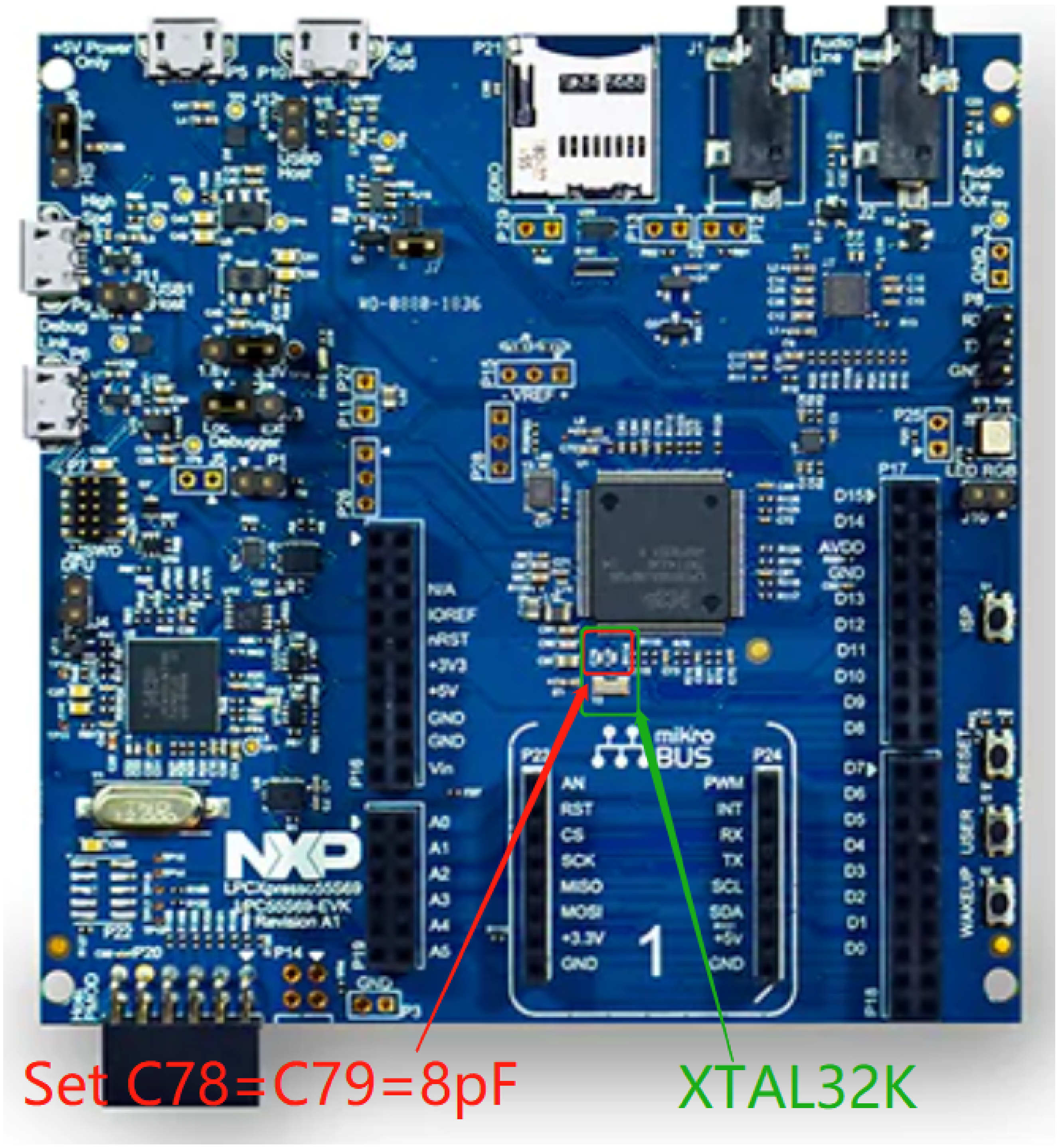
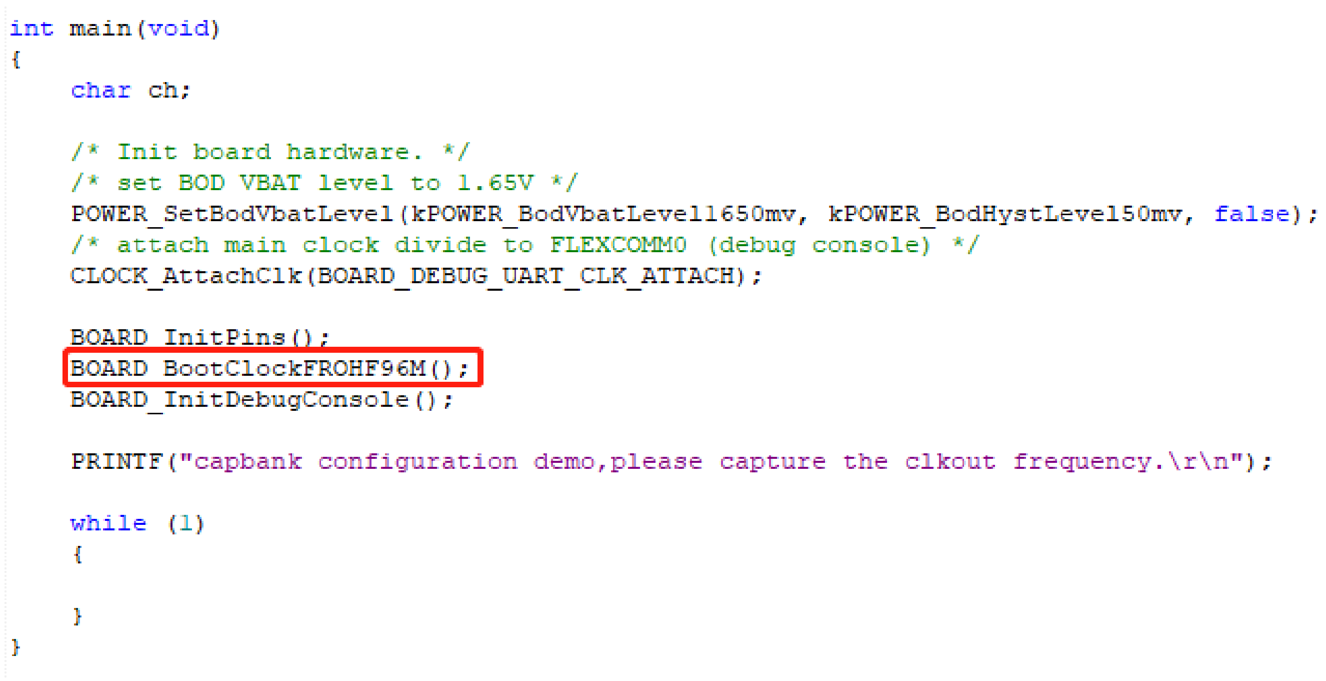


Figure 8. Replacing C78 and C79 with 8pF capacitors

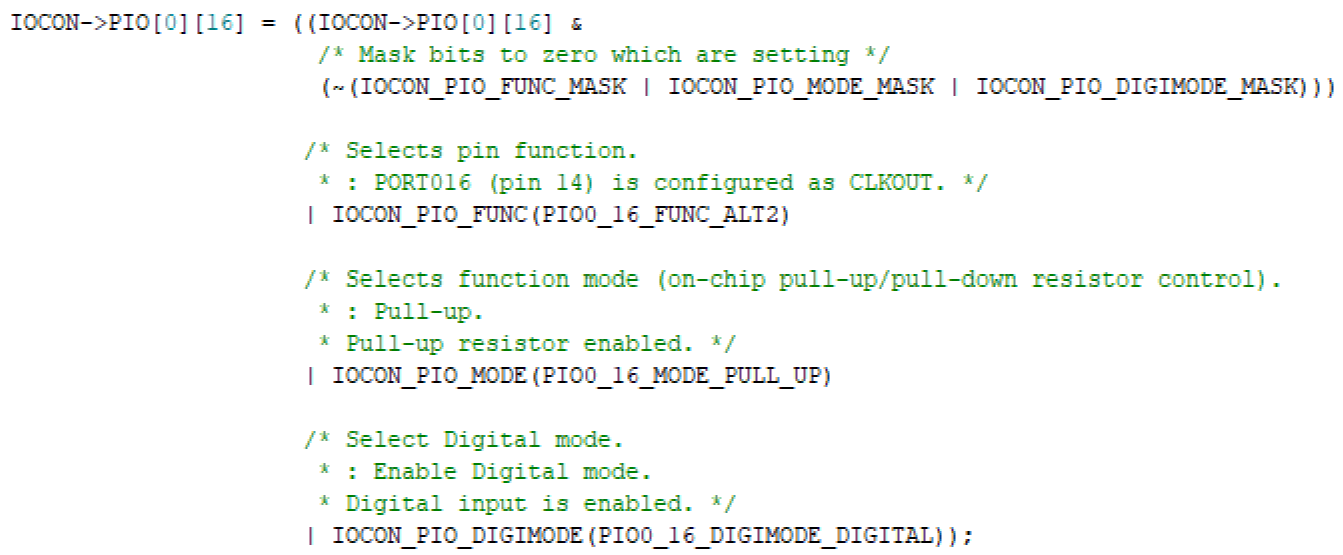
6.2 Software configuration

To configure the Cap Bank and easily check the results, users need to configure 32KHz Crystal Oscillator (XTAL32K) and CLKOUT function, output the trimmed clock directly to the clkout pin, and measure the result with a high-precision instrument.

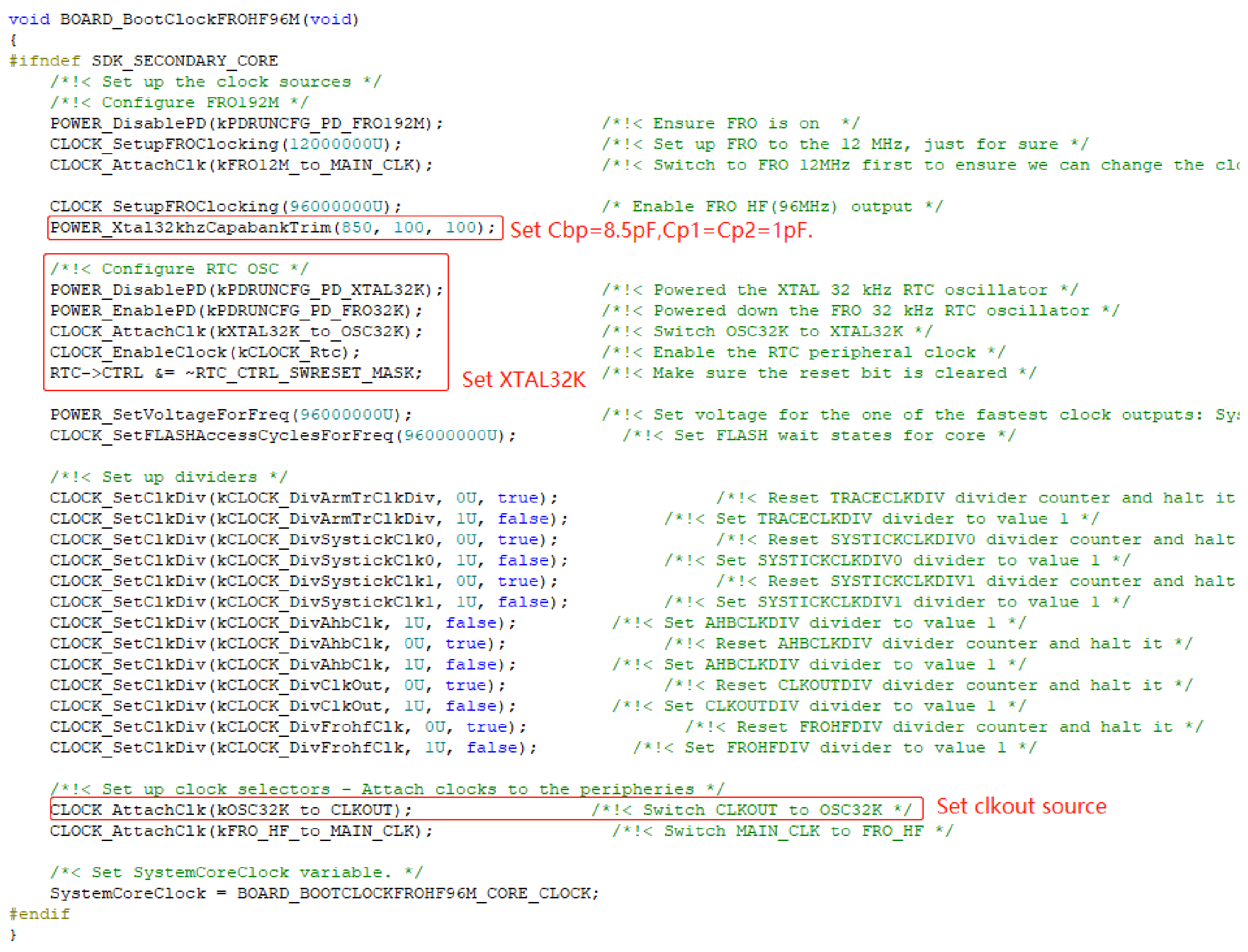
The following code snippet shows how to configure the Cap Bank and route the crystal oscillator output to the CLKOUT pin:



```
int main(void)
{
    char ch;

    /* Init board hardware. */
    /* set BOD VBAT level to 1.65V */
    POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
    /* attach main clock divide to FLEXCOMM0 (debug console) */
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);

    BOARD_InitPins();
    BOARD_BootClockFROHF96M();
    BOARD_InitDebugConsole();

    PRINTF("capbank configuration demo,please capture the clkout frequency.\r\n");

    while (1)
    {

    }
}
```



```
IOCON->PIO[0][16] = ((IOCON->PIO[0][16] &
                      /* Mask bits to zero which are setting */
                      (~(IOCON_PIO_FUNC_MASK | IOCON_PIO_MODE_MASK | IOCON_PIO_DIGIMODE_MASK)))

                     /* Selects pin function.
                      * : PORT016 (pin 14) is configured as CLKOUT. */
                     | IOCON_PIO_FUNC(PIO0_16_FUNC_ALT2)

                     /* Selects function mode (on-chip pull-up/pull-down resistor control).
                      * : Pull-up.
                      * Pull-up resistor enabled. */
                     | IOCON_PIO_MODE(PIO0_16_MODE_PULL_UP)

                     /* Select Digital mode.
                      * : Enable Digital mode.
                      * Digital input is enabled. */
                     | IOCON_PIO_DIGIMODE(PIO0_16_DIGIMODE_DIGITAL));
```



```
void BOARD_BootClockFROHF96M(void)
{
#ifndef SDK_SECONDARY_CORE
    /*!< Set up the clock sources */
    /*!< Configure FRO192M */
    POWER_DisablePD(kPDRUNCFG_PD_FRO192M);                /*!< Ensure FRO is on  */
    CLOCK_SetupFROClocking(12000000U);                    /*!< Set up FRO to the 12 MHz, just for sure */
    CLOCK_AttachClk(kFRO12M_to_MAIN_CLK);                 /*!< Switch to FRO 12MHz first to ensure we can change the cl

    CLOCK_SetupFROClocking(96000000U);                    /* Enable FRO HF(96MHz) output */
    POWER_Xtal32khzCapabankTrim(850, 100, 100);

    /*!< Configure RTC OSC */
    POWER_DisablePD(kPDRUNCFG_PD_XTAL32K);                /*!< Powered the XTAL 32 kHz RTC oscillator */
    POWER_EnablePD(kPDRUNCFG_PD_FRO32K);                  /*!< Powered down the FRO 32 kHz RTC oscillator */
    CLOCK_AttachClk(kXTAL32K_to_OSC32K);                  /*!< Switch OSC32K to XTAL32K */
    CLOCK_EnableClock(kCLOCK_Rtc);                        /*!< Enable the RTC peripheral clock */
    RTC->CTRL &= ~RTC_CTRL_SWRESET_MASK;                  /*!< Make sure the reset bit is cleared */

    POWER_SetVoltageForFreq(96000000U);                   /*!< Set voltage for the one of the fastest clock outputs: Sy
    CLOCK_SetFLASHAccessCyclesForFreq(96000000U);           /*!< Set FLASH wait states for core */

    /*!< Set up dividers */
    CLOCK_SetClkDiv(kCLOCK_DivArmTrClkDiv, 0U, true);               /*!< Reset TRACECLKDIV divider counter and halt it
    CLOCK_SetClkDiv(kCLOCK_DivArmTrClkDiv, 1U, false);          /*!< Set TRACECLKDIV divider to value 1 */
    CLOCK_SetClkDiv(kCLOCK_DivSystickClk0, 0U, true);               /*!< Reset SYSTICKCLKDIV0 divider counter and halt
    CLOCK_SetClkDiv(kCLOCK_DivSystickClk0, 1U, false);          /*!< Set SYSTICKCLKDIV0 divider to value 1 */
    CLOCK_SetClkDiv(kCLOCK_DivSystickClk1, 0U, true);               /*!< Reset SYSTICKCLKDIV1 divider counter and halt
    CLOCK_SetClkDiv(kCLOCK_DivSystickClk1, 1U, false);          /*!< Set SYSTICKCLKDIV1 divider to value 1 */
    CLOCK_SetClkDiv(kCLOCK_DivAhbClk, 1U, false);          /*!< Set AHBCLKDIV divider to value 1 */
    CLOCK_SetClkDiv(kCLOCK_DivAhbClk, 0U, true);                /*!< Reset AHBCLKDIV divider counter and halt it */
    CLOCK_SetClkDiv(kCLOCK_DivAhbClk, 1U, false);          /*!< Set AHBCLKDIV divider to value 1 */
    CLOCK_SetClkDiv(kCLOCK_DivClkOut, 0U, true);                /*!< Reset CLKOUTDIV divider counter and halt it */
    CLOCK_SetClkDiv(kCLOCK_DivClkOut, 1U, false);          /*!< Set CLKOUTDIV divider to value 1 */
    CLOCK_SetClkDiv(kCLOCK_DivFrohfClk, 0U, true);                /*!< Reset FROHFDIV divider counter and halt it */
    CLOCK_SetClkDiv(kCLOCK_DivFrohfClk, 1U, false);          /*!< Set FROHFDIV divider to value 1 */

    /*!< Set up clock selectors - Attach clocks to the peripheries */
    CLOCK_AttachClk(kOSC32K_to_CLKOUT);                 /*!< Switch CLKOUT to OSC32K */
    CLOCK_AttachClk(kFRO_HF_to_MAIN_CLK);                 /*!< Switch MAIN_CLK to FRO_HF */

    /*< Set SystemCoreClock variable. */
    SystemCoreClock = BOARD_BOOTCLOCKFROHF96M_CORE_CLOCK;
#endif
}
```

6.3 Demo

To see the demo result, a frequency meter such as HP 5313A Universal Counter shall be used.

Here assume Cp1=Cp2=1pF. Then the call to Cap Bank API is as follows:

POWER_Xtal32khzCapabankTrim(Cbp*100, 100, 100); by default Cbp = 8.5pF;

Now you can configure first parameter of this routine (**Cbp**) to different values other than the default value (8.5pF). Then rebuild and run the code above.

You can see crystal output frequency changes in the frequency meter with different Cbp as follows

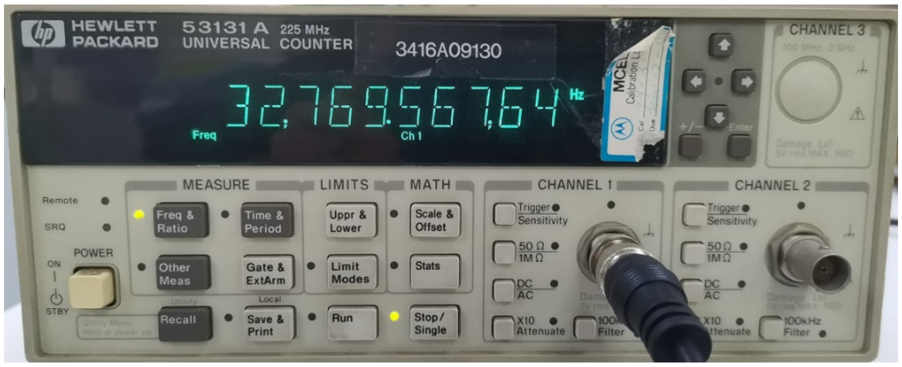


Figure 9. Crystal output frequency (1)

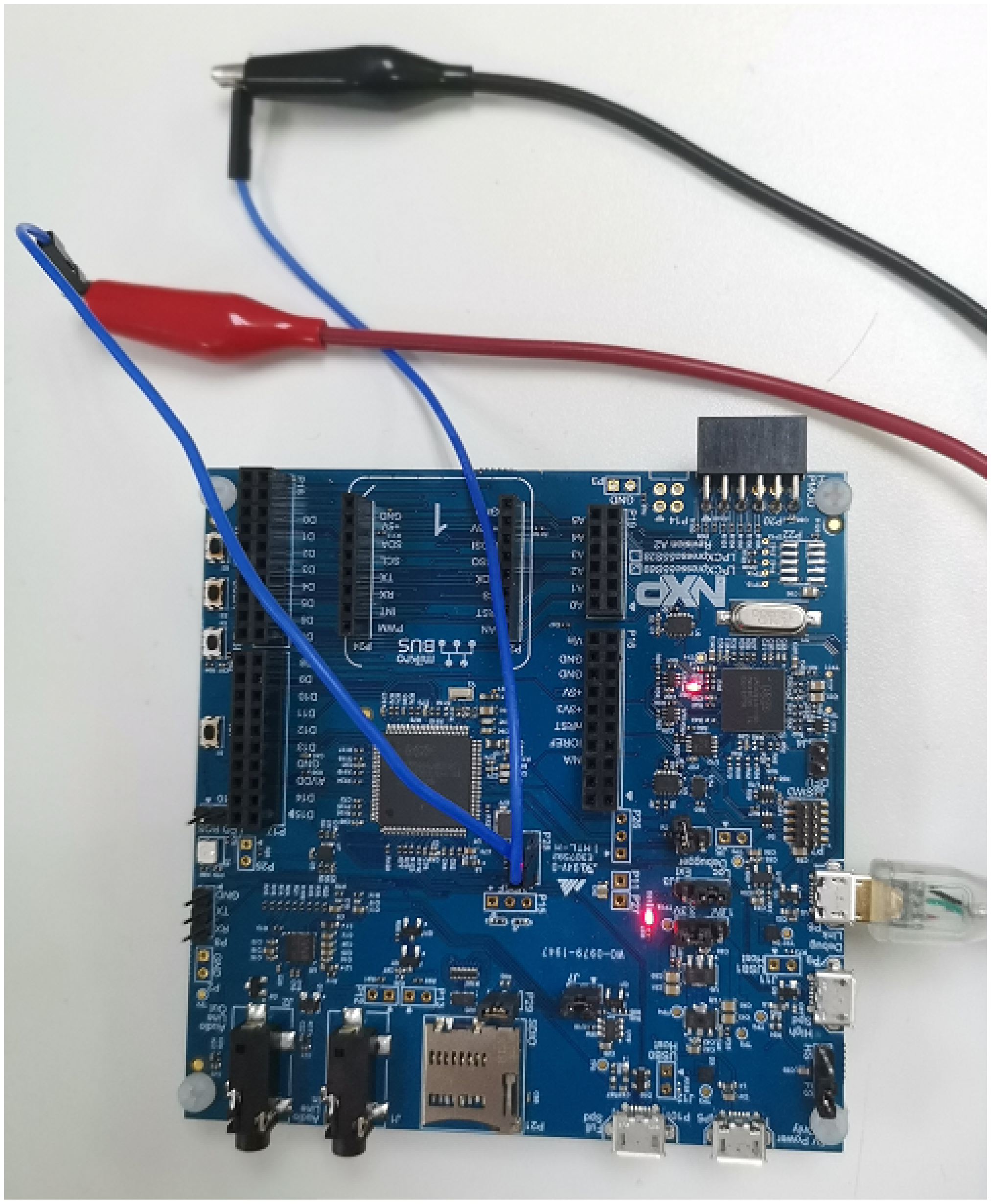


Figure 10. Crystal output frequency (2)

The following table summarizes the experimental results for different Cbp with same 32KB crystal on board (Cx1=Cx2=8pF).

Table 2. Experimental results for different Cbp with same 32KB crystal

Experiment Number	Code	Cbp value	Frequency output (Hz)	Target (Hz)	Error (ppm)
1	POWER_Xtal32khzCapabankTrim(**600**, 100, 100);	6 pF	32769.773	32768.000	54.108
2	POWER_Xtal32khzCapabankTrim(**700**, 100, 100);	7 pF	32769.159	32768.000	35.370
3	POWER_Xtal32khzCapabankTrim(**800**, 100, 100);	8 pF	32768.583	32768.000	17.792
4	POWER_Xtal32khzCapabankTrim(**850**, 100, 100);	8.5 pF	32768.360	32768.000	10.986
4	POWER_Xtal32khzCapabankTrim(**900**, 100, 100);	9 pF	32768.146	32768.000	4.456
5	POWER_Xtal32khzCapabankTrim(**1000**, 100, 100);	10 pF	32767.724	32768.000	8.423

Based on the test result above, Cbp = 9pF can achieve the best accuracy of the oscillator output frequency toward the target frequency 32768 Hz, which is only 4.5 PPM deviation.

7 Conclusion

The original intention of the built-in Cap Bank is to allow users to reduce BOM costs and PCB area. The original intention of Cap Bank's API function is to make it easy for users to configure the internal Cap Bank. These Cap Banks are always enabled, and their capacitance values have a certain adjustable range. In most cases, it is not exceeded. If external crystal’s IEC load capacitance (CL) exceeds this range, please correctly understand the meaning of the first parameter(Cbp) of the API and calculate it according to the calculation methods shown in the aforementioned example 2 and example 4.

When designing a PCB board, ensure that the PCB traces are symmetrical, that is, Cp1=Cp2. Use SDK version 2.8.0 or newer, which can correctly support Cap Bank configuration.

Date of release: 16 November 2020

Document identifier: AN13057