AN14512

How to Trim the Clock on MCX A Series

Rev. 1.1 — 16 July 2025 Application note

Document information

Information	Content
Keywords	AN14512, MCXA, Trim clock, SCG, FIRC, SIRC
Abstract	This application note introduces the SCG module in MCX A series and describes how to trim the clock on MCX A series.

1 Introduction

The MCX A series microcontrollers, powered by the Arm Cortex-M33, are general-purpose MCUs designed to address a wide range of applications with scalable device options, low power, and intelligent peripherals. MCX A series provide multiple clock sources, including System Oscillator Clock (SOSC), Slow Internal Reference Clock (SIRC, FRO12M), Fast Internal Reference Clock (FIRC, FRO192M), and Real-Time Oscillator Clock (ROSC). The System Clock Generator (SCG) module can be used to control these clocks, where FIRC and SIRC support manual trimming and auto trimming functions. Under open loop conditions, the accuracy of FIRC and SIRC is ±3 %. In applications requiring higher accuracy, the auto trimming function can be used to improve clock accuracy. When auto trimming is enabled, FIRC and SIRC can achieve ±0.25 % and ±0.6 % accuracy respectively. This application note introduces the SCG module in MCX A series and describes how to trim the clock on MCX A series.

2 System Clock Generator (SCG) overview

The SCG module provides the main clock and other peripheral clocks for the MCU. The SCG takes clock inputs from various sources and generates the clocks that the MCU requires.

2.1 Function

Figure 1 shows the block diagram of the SCG module.

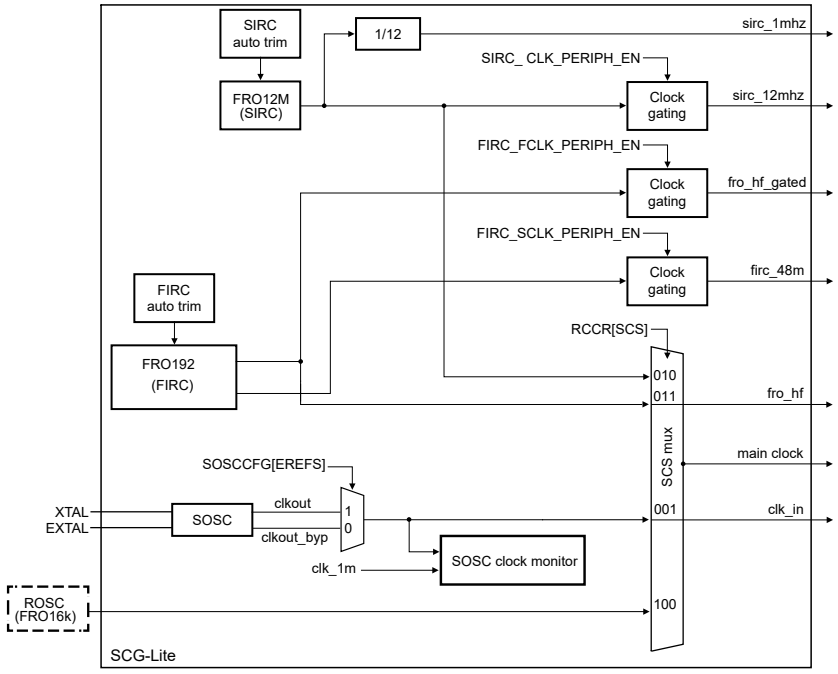

Figure 1. SCG block diagram

The system boots up from the FIRC source. The system clock can be switched among the following clock sources: FIRC, SIRC, SOSC, and ROSC.

- FIRC (FRO192M)
  - FIRC can output 192/96/64/48 MHz clock by register configuration.
  - FIRC can limit output frequency by frequency phantom.

  - FIRC with trim capability.
- SIRC (FRO12M)
  - SIRC can output 12 MHz clock and 1 MHz clock divided from it.
  - SIRC with trim capability.
- SOSC
  - An external crystal is required, ranged from 8 MHz to 50 MHz.
- ROSC
  - ROSC is sourced from the FRO16K clock in VBAT block.

3 Software design

This section describes the steps for trimming the clock and provides sample code. The code is based on FRDM-MCXA153 SDK 2.16.1 and uses FRDM-MCXA153 as the test platform.

3.1 Manual trimming

The FIRC clock is calibrated before the chip is delivered, and the trim values are recorded in the FIRC Trim register (FIRCTRIM) after reset. The user can manually trim the clock using the FIRCTRIM, as shown in Figure 2.

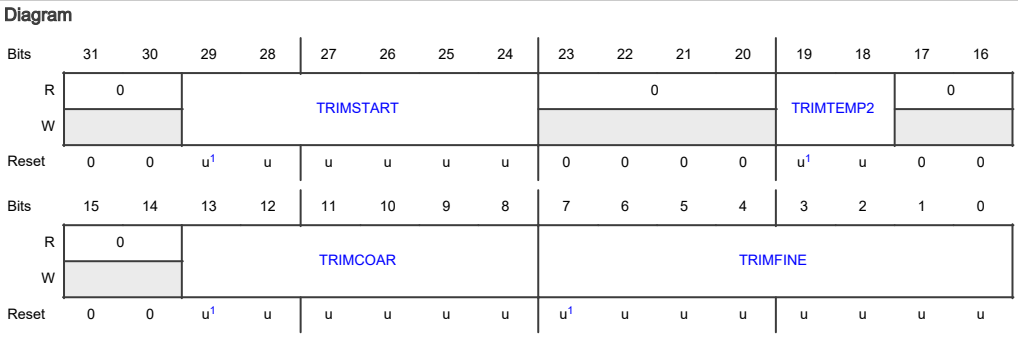

Figure 2. FIRC trim register

There are two kinds of trim accuracy:

- TRIMCOAR bits are used to coarsely trim the FIRC Clock to within approximately ±3.2 % of the target frequency.
- TRIMFINE bits are used to fine trim the FIRC Clock to within approximately ±0.25 % of the target frequency.

Note: *For TRIMCOAR, increasing the value makes the frequency lower. The frequency change rate near the center is about 2.5 % per bit, and the higher the value, the smaller the change rate. For TRIMFINE, increasing the value makes the frequency higher. The frequency change rate near the center is about 0.06 % per bit, and the higher the value, the smaller the change rate.*

To manual trim the FIRC clock, follow the steps below:

1. Read the value from FIRCTRIM and record it.
2. Write **0x5A5A** to `TRIM_LOCK[TRIM_LOCK_KEY]` and **1** to `TRIM_LOCK[TRIM_UNLOCK]` to unlock trim registers.
3. Add the trim value to the value recorded in Step 1.
4. Write **0x5A5A** to `TRIM_LOCK[TRIM_LOCK_KEY]` and **0** to `TRIM_LOCK[TRIM_UNLOCK]` to lock the trim registers.

The core code for FIRC manual trimming is as follows:

```
typedef enum
{
  TRIM_COARSE,
  TRIM_FINE
} trim_type;

/**
 * @brief     app_ FircManualTrim  Manual TrimFIRC clock
 * @param  type     TRIM_COARSE about 2.5%, TRIM_FINE about 0.06%
 *                 value   FIRCTRIM + value
 * @return   NULL
*/
void app_FircManualTrim(trim_type type,char value)
{
  uint32_t trim_value  = SCG0 -> FIRCTRIM;
  uint8_t  trim_coarse = (uint8_t)(trim_value >> 8);
  uint8_t  trim_fine   = (uint8_t)trim_value;
  PRINTF("%x\r\n", trim_value);
  /* UNLOCK the trim */
  SCG0 -> TRIM_LOCK = (SCG_TRIM_LOCK_TRIM_LOCK_KEY(0x5A5A)) |
 (SCG_TRIM_LOCK_TRIM_UNLOCK(1));
  if (type == TRIM_COARSE)   // coarse trim mode, use coarse trim register
  {
    trim_coarse = trim_coarse + value;
    SCG0 -> FIRCTRIM = (trim_value & 0xFFFF0000) | (trim_fine & 0xff) | ((trim_coarse &
 0x3f) << 8);
  }
  else if(type == TRIM_FINE) // fine trim mode, use fine trim register
  {
    trim_fine = trim_fine + value;
    SCG0 -> FIRCTRIM = (trim_value & 0xFFFF0000) | (trim_fine & 0xff)  | ((trim_coarse &
 0x3f) << 8);
  }
  trim_value  = SCG0 -> FIRCTRIM;
  /* LOCK the trim */
  SCG0 -> TRIM_LOCK = (SCG_TRIM_LOCK_TRIM_LOCK_KEY(0x5A5A)) |
 (SCG_TRIM_LOCK_TRIM_UNLOCK(0));
  PRINTF("%x\r\n", trim_value);
}
```

3.2 Auto trimming

The accuracy of FIRC and SIRC is ±3 % (Ta = -40 °C ~ 125 °C), in some use cases, a higher precision clock is required, in which case the auto trimming function is useful. After auto trimming, FIRC and SIRC can achieve ±0.25 % and ±0.6 % accuracy respectively.

An external crystal is required as a reference source to trim the clock.

To auto trim the FIRC clock, follow the steps below:

1. Enable the SOSC clock.
2. Wait until the SOSC clock is valid.
3. Write **2** to `FIRCTCFG[TRIMSRC]` to select SOSC as the auto trim clock source.
4. Write **7** to `FIRCTCFG[TRIMDIV]` to divide SOSC to 1 MHz (8 MHz external crystal).
5. Write **0** to `FIRCCSR[LK]` to unlock the FIRCCSR.
6. Write **1** to `FIRCCSR[FIRCTREN]` to enable the auto trim.
7. Write **1** to `FIRCCSR[FIRCTRUP]` to enable the update.
8. Read `FIRCCSR[FIRCVLD]` until it returns **1**, indicating that the FIRC is valid.
9. Read `FIRCCSR[FIRCERR]` to ensure that it returns **0**.

10. Read `FIRCCSR[TRIM_LOCK]` until it returns **1**.
11. Write **1** to `FIRCCSR[LK]` to lock the FIRCCSR.

The core code for FIRC auto trimming is as follows:

```
/**
 * @brief     app_FircAutoTrim   Run Auto Trim to trim the clock using external crystal
 * @param   NULL
 * @return    NULL
*/
void app_FircAutoTrim()
{
 CLOCK_SetupExtClocking(8000000U);              // Enable the 8MHz external crystal
 SCG0 -> FIRCTCFG |= SCG_FIRCTCFG_TRIMSRC(2);   // Select the external crystal(SOSC) as the source
 SCG0 -> FIRCTCFG |= SCG_FIRCTCFG_TRIMDIV(7);   // Divide the SOSC to 1MHz
 SCG0 -> FIRCCSR  &= ~SCG_FIRCCSR_LK(1);        // Unlock the FIRCCSR register
 SCG0 -> FIRCCSR  |= SCG_FIRCCSR_FIRCTREN(1);   // Enable auto trim
 SCG0 -> FIRCCSR  |= SCG_FIRCCSR_FIRCTRUP(1);   // Enable update
 /* Until it returns 1, indicating FIRC is valid */
 while(!(SCG0 -> FIRCCSR & SCG_FIRCCSR_FIRCVLD_MASK));
 /* Until it returns 0, indicating no error */
 while(  SCG0 -> FIRCCSR & SCG_FIRCCSR_FIRCERR_MASK);
 /* Until it returns 1, indicating FIRC auto trim locked to target frequency range */
 while(!(SCG0 -> FIRCCSR & SCG_FIRCCSR_TRIM_LOCK_MASK));
 SCG0 -> FIRCCSR  |= SCG_FIRCCSR_LK(1);          // Lock the FIRCCSR register
}
```

To auto trim the SIRC clock, follow the steps below:

1. Enable the SOSC clock.
2. Wait until the SOSC clock is valid.
3. Write **2** to `SIRCTCFG[TRIMSRC]` to select SOSC as the auto trim clock source.
4. Write **7** to `SIRCTCFG[TRIMDIV]` to divide SOSC to 1 MHz (8 MHz external crystal).
5. Write **0** to `SIRCCSR[LK]` to unlock the SIRCCSR.
6. Write **1** to `SIRCCSR[SIRCTREN]` to enable the auto trim.
7. Write **1** to `SIRCCSR[SIRCTRUP]` to enable the update.
8. Read `SIRCCSR[SIRCVLD]` until it returns **1**, indicating that the SIRC is valid.
9. Read `SIRCCSR[SIRCERR]` to ensure that it returns **0**.
10. Read `SIRCCSR[TRIM_LOCK]` until it returns **1**.
11. Write **1** to `SIRCCSR[LK]` to lock the SIRCCSR.

The core code for SIRC auto trimming is as follows:

```
/**
 * @brief   app_SircAutoTrim   Run Auto Trim to trim the clock using external crystal
 * @param   NULL
 * @return  NULL
*/
void app_SircAutoTrim()
{
 CLOCK_SetupExtClocking(8000000U);              // Enable the 8MHz external crystal
 SCG0 -> SIRCTCFG |= SCG_SIRCTCFG_TRIMSRC(2);   // Select the external crystal(SOSC) as the source
 SCG0 -> SIRCTCFG |= SCG_SIRCTCFG_TRIMDIV(7);   // Divide the SOSC to 1MHz
 SCG0 -> SIRCCSR  &= ~SCG_SIRCCSR_LK(1);        // Unlock the SIRCCSR register
 SCG0 -> SIRCCSR  |= SCG_SIRCCSR_SIRCTREN(1);   // Enable auto trim
 SCG0 -> SIRCCSR  |= SCG_SIRCCSR_SIRCTRUP(1);   // Enable update
 /* Until it returns 1, indicating SIRC is valid */
 while(!(SCG0 -> SIRCCSR & SCG_SIRCCSR_SIRCVLD_MASK));
 /* Until it returns 0, indicating no error */
 while(  SCG0 -> SIRCCSR & SCG_SIRCCSR_SIRCERR_MASK);
 /* Until it returns 1, indicating SIRC auto trim locked to target frequency range */
 while(!(SCG0 -> SIRCCSR & SCG_SIRCCSR_TRIM_LOCK_MASK));
 SCG0 -> SIRCCSR  |= SCG_SIRCCSR_LK(1);         // Lock the SIRCCSR register
}
```

Note: *Configure the required peripherals after trimming the clock.*

3.3 Clock output

The clock output function is a useful feature for clock observing. Some pins can be configured as clock output pin, refer to the reference manual SYSCON section for more detail. Take the MCXA153 as an example, configure `P0_6`, `P3_6`, or `P3_8` as the clock output pin. After configuring the pin, set the `SYSCON` and `MRCC` registers as follows:

1. Write **0** to `CLKUNLOCK[UNLOCK]` to unlock the clock configuration registers.
2. Write **1** to `MRCC_CLKOUT_CLKSEL[MUX]` to select **FIRC_DIV** as the clock source (when measuring SIRC, write **0** to `MRCC_CLKOUT_CLKSEL[MUX]`).
3. Write **15** to `MRCC_CLKOUT_CLKDIV[DIV]` to set the divider value to **16** (when measuring SIRC, write **11** to `MRCC_CLKOUT_CLKSEL[MUX]`).
4. Write **1** to `CLKUNLOCK[UNLOCK]` to lock the clock configuration registers.

The core code for clock output configuration is as follows:

```
  /* PORT3: Peripheral clock is enabled */
  CLOCK_EnableClock(kCLOCK_GatePORT3);
  /* PORT3 peripheral is released from reset */
  RESET_ReleasePeripheralReset(kPORT3_RST_SHIFT_RSTn);

  const port_pin_config_t port3_8_config = {/* Internal pull-up resistor is enabled */
                        kPORT_PullUp,
                        /* Low internal pull resistor value is selected. */
                        kPORT_LowPullResistor,
                        /* Fast slew rate is configured */
                        kPORT_FastSlewRate,
                        /* Passive input filter is disabled */
                        kPORT_PassiveFilterDisable,
                        /* Open drain output is disabled */
                        kPORT_OpenDrainDisable,
                        /* Low drive strength is configured */
                        kPORT_LowDriveStrength,
                        /* Normal drive strength is configured */
                        kPORT_NormalDriveStrength,
                        /* Pin is configured as CLKOUT */
                        kPORT_MuxAlt12,
                        /* Digital input enabled */
                        kPORT_InputBufferEnable,
                        /* Digital input is not inverted */
                        kPORT_InputNormal,
                        /* Pin Control Register fields [15:0] are not locked */
                        kPORT_UnlockRegister};
  /* PORT3_8 is configured as CLK_OUT */
  PORT_SetPinConfig(PORT3, 8U, &port3_8_config);

/**
 * @brief    app_ClockOut   clock output set, to observe clock signal
 * @param    NULL
 * @return   NULL
*/
void app_ClockOut()
{
 SYSCON -> CLKUNLOCK          = SYSCON_CLKUNLOCK_UNLOCK(0);      // Unlock the register
 MRCC0  -> MRCC_CLKOUT_CLKSEL = MRCC_MRCC_CLKOUT_CLKSEL_MUX(1);  // FIRC_DIV
 MRCC0  -> MRCC_CLKOUT_CLKDIV = MRCC_MRCC_CLKOUT_CLKDIV_DIV(15); // 1/16 Frequency
 SYSCON -> CLKUNLOCK          = SYSCON_CLKUNLOCK_UNLOCK(1);      // Lock the register
}
```

4 Test on Board

To test the clock trimming functions, the FRDM-MCXA153 board is used, as shown in Figure 3. The board ships with an onboard debugger. Use a USB-C cable to connect the board to the computer via J15 for downloading and debugging. In this test, `P3_8` is used as the clock output pin. Connect the `P3_8` (J3-11 on board) and GND to the oscilloscope probe, and capture the output signal.

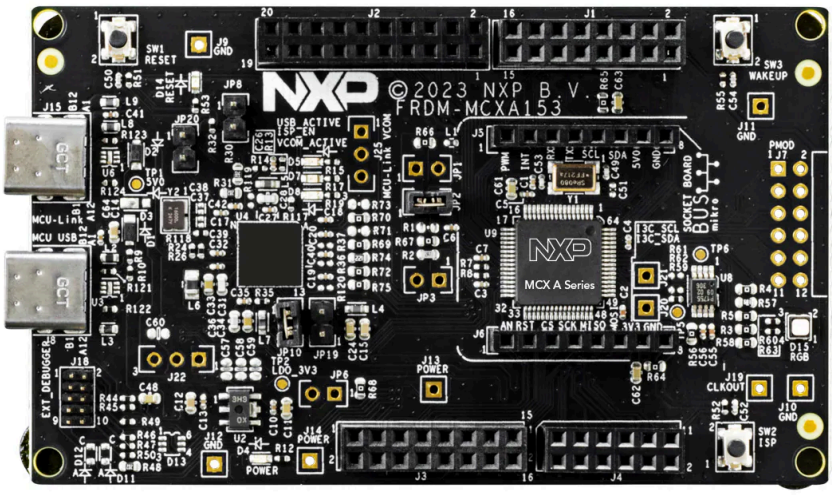

Figure 3. FRDM-MCXA153 board

4.1 Test result

When measuring the FIRC frequency, write **15** to `MRCC_CLKOUT_CLKDIV[DIV]` to set the divider value to **16**. In this test, set FIRC to 96 MHz and output the clock through `P3_8`. Before trimming, the clock output shows below, the frequency is about 6.045 MHz, about 1/16 of 96 MHz, as shown in Figure 4.

Figure 4. FIRC clock output before trimming

If **1** is added to `FIRCTRIM[TRIMCOAR]`, the frequency is about 5.911 MHz, and the frequency decreases by about 2.2 %, as shown in Figure 5.

Figure 5. FIRC clock output after FIRCTRIM[TRIMCOAR] + 1

If **-1** is added to `FIRCTRIM[TRIMCOAR]`, the frequency is about 6.183 MHz, and the frequency increases by about 2.3 %, as shown in Figure 6.

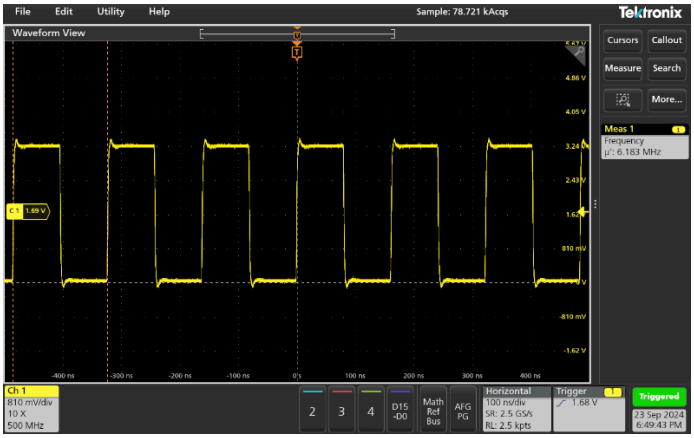

Figure 6. FIRC clock output after FIRCTRIM[TRIMCOAR] - 1

If **4** is added to `FIRCTRIM[TRIMFINE]`, the frequency is about 6.059 MHz, and the frequency increases by about 0.23 %, as shown in Figure 7.

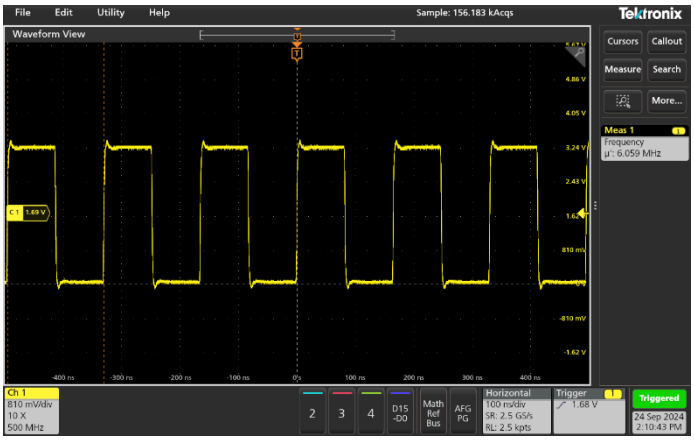

Figure 7. FIRC clock output after FIRCTRIM[TRIMFINE] + 4

If **-4** is added to `FIRCTRIM[TRIMFINE]`, the frequency is about 6.032 MHz, and the frequency decreases by about 0.22 %, as shown in Figure 8.

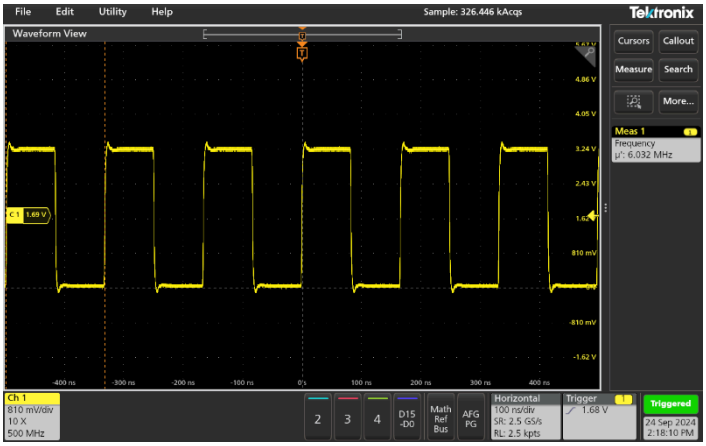

Figure 8. FIRC clock output after FIRCTRIM[TRIMFINE] - 4

After running the FIRC auto trimming function, the frequency is about 6.000 MHz, as shown in Figure 9. The accuracy of the FIRC clock has been improved.

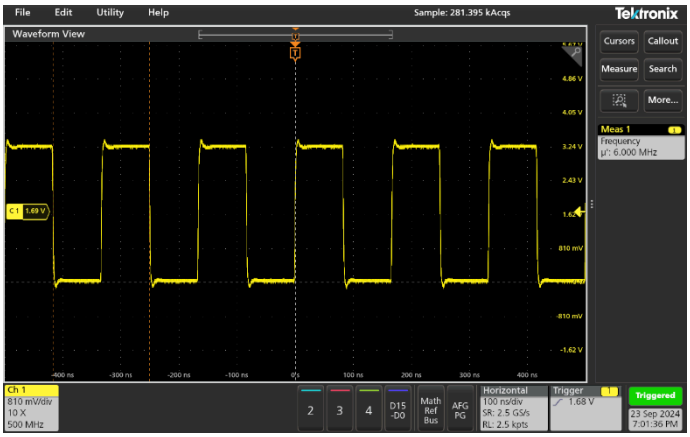

Figure 9. FIRC clock output after auto trimming

When measuring the SIRC frequency, write **11** to `MRCC_CLKOUT_CLKDIV[DIV]` to set the divider value to **12**. In this test, SIRC is set to 12 MHz and output the clock through `P3_8`. Before trimming, the clock output shows below, and the frequency is about 1.007 MHz, about 1/12 of 12 MHz, as shown in Figure 10.

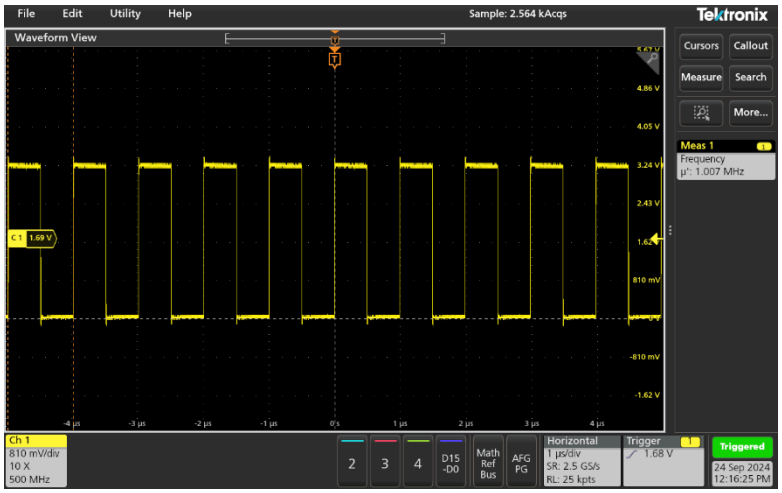

Figure 10. SIRC clock output before trimming

After running the SIRC auto trimming function, the frequency is about 1.000 MHz, as shown in Figure 11. The accuracy of the SIRC clock has been improved.

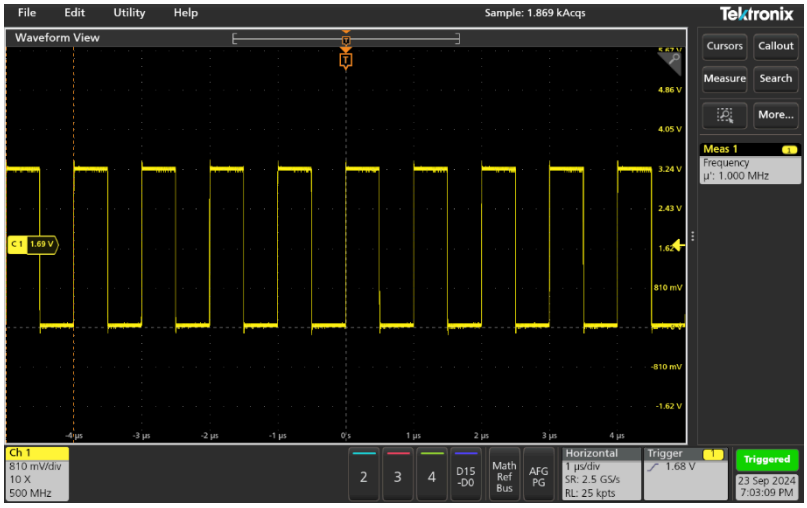

Figure 11. SIRC clock output before auto trimming

5 Conclusion

This application note provides some information and code to help the user trim the clock on MCX A series and do a test to verify the clock output.

6 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14512 v.1.1	16 July 2025	Updated the code in Section 3.3
AN14512 v.1.0	21 November 2024	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that 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 in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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 life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at 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.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

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 competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

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

NXP — wordmark and logo are trademarks of NXP B.V.

Contents