AN14681

MAU Usage to Accelerate Mathematical Operations

Rev. 1.0 — 16 July 2025 Application note

Document information

Information	Content
Keywords	AN14681, MCX A series, MAU, FRDM-MCXA346
Abstract	This document provides information, code snippets, and tips that help you to accelerate the calculations using MAU.

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 A346 series provide a Mathematical acceleration unit (MAU), which accelerates the mathematical operations, such as include sine, cosine, arctangent, square root, reciprocal, and reciprocal square root. Applications such as Motor Control and Sensor Data Analysis are benefited from MAU's acceleration.

2 MAU operation overview

To operate MAU, perform the following steps:

1. Set the bit 9 in the register `MRCC_GLB_RST2` and `MRCC_GLB_CC2` before starting the MAU operation. These bits control the MAU's clock and reset operation.
2. To operate the MAU correctly, it is essential to know the MAU's memory model.

MAU on the MCX A346 device has two memory spaces:

- Direct memory space (0x4010_8000 to 0x4010_87FF): This memory space contains the programming registers shown in Table 1.
  These registers control the behavior of the MAU and store the calculation results.

Table 1. Registers in direct memory space

Offset	Register	Width	Access
10 h	System Control (SYS_CTLR)	32	RW
14 h	General Exception Status Interrupt Enable (GEXP_STATUS_IE)	32	RW
18 h	General Exception Status (GEXP_STATUS)	32	RW
30 h	Operation Control (OP_CTRL)	32	RW
38 h	Result Status Interrupt Enable (RES_STATUS_IE)	32	RW
3C h	Result Status (RES_STATUS)	32	RW
40 h	Result Register 0 (RES0)	32	RW
44 h	Result Register 1 (RES1)	32	RW
48 h	Result Register 2 (RES2)	32	RW
4C h	Result Register 3 (RES3)	32	RW

- Indirect memory space (0x4010_8800 to 0x4010_8FFF): This memory space access is decoded into the MAU commands and data operands.
  Table 2 shows how the access address is decoded into commands.

Table 2. MAU Command Decode Table

BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
1	Data Type		Result Select		MAU Operation Code					Reserved	

Also, Table 2 shows how the address is decoded into the following 3 parameters:

- Data Type: 0b00 UINT, 0b01 INT, 0b10 Fixed Point, 0b11 Floating Point
- Result select: 0b00 RES0, 0b01 RES1, 0b10 RES2, 0b11 RES3
- MAU operation Code: 5b00001 y=1/x, 5b00010 y=sqrt(x), 5b00011 y=1/sqrt(x), 5b01000 y=cos(πx), 5b01001 y=sin(πx), 5b01100 y=arctan(x)/π

The following steps show how to perform a square root operation:

Note:

The input used in the example is `UINT` and the result is stored in `RES2`.

1. Determine the address according to the Table 2 and Reference Manual.
   From the given example, the write address should be:

```
0x40108000 (MAU base address) + 0x90C (0b1_00_10_00010_00, Data Type=UINT,
Result Select=2, MAU OpCode=2) = 0x4010_890C
```

2. Write the data to the determined address with selected width.
   If data is `UINT16`, then use instruction `STRH` or `uint16_t` pointer.
3. Read the result from the selected result register.
   In the given example, `RES2` is selected. Therefore, read the `UINT16` value from the following address:

```
0x40108000 (MAU base address) + 0x40 (RES0 register offset) = 0x40108040
```

2.1 Example

```
/*! @brief MAU data type. */
typedef enum _mau_data_type
{
    kMAU_DT_UINT = 0,
    kMAU_DT_INT = 1,
    kMAU_DT_Q1X = 2,
    kMAU_DT_FLOAT = 3
} mau_data_type_t;
/*! @brief MAU result register. */
typedef enum _mau_result
{
    kMAU_RES0 = 0,
    kMAU_RES1 = 1,
    kMAU_RES2 = 2,
    kMAU_RES3 = 3
} mau_result_t;
/*! @brief MAU calculation code. */
typedef enum _mau_mopc
{
    kMAU_MOPC_BYPASS = (0U),
    kMAU_MOPC_RECIP = (1U),
    kMAU_MOPC_SQRT = (2U),
    kMAU_MOPC_SQRT_RECIP = (3U),
    kMAU_MOPC_COS = (8U),
    kMAU_MOPC_SIN = (9U),
    kMAU_MOPC_ATAN = (12U)
} mau_mopc_t;
#define MAU_MATH_PI         (3.1415926535898f)
#define MAU_DT_SET(dt)      ((dt) << 9U)
#define MAU_RES_SET(res)    ((res) << 7U)
#define MAU_MOPC_SET(mopc)  ((mopc) << 2U)
#define MAU_INDIRECT_ADDR(base, dt, ds, mopc) ((base) | 0x800U | MAU_DT_SET(dt)
 | MAU_RES_SET(ds) | MAU_MOPC_SET(mopc))
```

Definitions are provided in the SDK header. To perform the calculation easily, use the APIs from SDK.

An example calculation is given below:

```
*(volatile uint16_t *)
(MAU_INDIRECT_ADDR(MAU0, kMAU_DT_UINT, kMAU_RES2, kMAU_MOPC_SQRT)) = x;
uint16_t y = *(volatile uint16_t *)MAU0->RES2;
```

3 Performance

MAU is a simple mathematical accelerator. It accelerates sine, cosine, square root, reciprocal, and reciprocal square root operations, and supports various data types and widths. MAU is very efficient based on its zero configuration feature and pipeline feature. Theoretically, MAU calculation takes only 3 cycles.

Table 3 shows the performance comparison in CMSIS-DSP compliant API.

Table 3. MAU performance comparison

Operation	CPU cycles with MAU	CPU cycles with CM33 Core	Performance improvement
arm_sin_f32	10	52	4.2x
arm_sin_q31	10	39	2.9x
arm_sin_q15	10	33	2.3x
arm_cos_f32	10	57	4.7x
arm_cos_q31	10	40	3.0x
arm_cos_q15	10	33	2.3x
arm_sin_cos_f32	14	123	7.7x
arm_sin_cos_q31	9	247	26.4x
arm_sqrt_f32	14	26	0.8x
arm_sqrt_q31	12	62	4.1x
arm_sqrt_q15	12	53	3.4x

The following example shows how the CMSIS-DSP compliant API is defined:

```
float32_t arm_cos_F32(float32_t x)
{
    MAU_REG_FLOAT(MAU_INDIRECT_ADDR(MAU0_BASE, kMAU_DT_FLOAT, kMAU_RES0, kMAU_MOPC_COS)) = x * (1 / MAU_MATH_PI);
    return MAU_REG_FLOAT((uint32_t)(&MAU0->RES0));
}
```

4 Achieving better performance

This section describes how to achieve a better performance with MAU.

4.1 Pipelined calculation

MAU has 4 result registers. Therefore, MAU can perform 4 calculations parallelly.

To achieve a better performance, `pipelined` calculation is performed.

The following code snippet shows how to perform a pipelined calculation:

Note: *A CMSIS-DSP compliant function `arm_vector_sin_f32` is implemented in this example.*

```
void arm_vector_sin_q31(q31_t *pSrc, q31_t *pDst, uint32_t blockSize)
{
        const q31_t *pIn = pSrc;
        uint32_t blkCnt = blockSize >> 2U;
        while (blkCnt > 0U)
        {
                MAU_REG_Q31(MAU_INDIRECT_ADDR(MAU0_BASE, kMAU_DT_Q1X, kMAU_RES0, kMAU_MOPC_SIN)) = *pIn++ * 2;
                MAU_REG_Q31(MAU_INDIRECT_ADDR(MAU0_BASE, kMAU_DT_Q1X, kMAU_RES1, kMAU_MOPC_SIN)) = *pIn++ * 2;
                MAU_REG_Q31(MAU_INDIRECT_ADDR(MAU0_BASE, kMAU_DT_Q1X, kMAU_RES2, kMAU_MOPC_SIN)) = *pIn++ * 2;
                MAU_REG_Q31(MAU_INDIRECT_ADDR(MAU0_BASE, kMAU_DT_Q1X, kMAU_RES3, kMAU_MOPC_SIN)) = *pIn++ * 2;
                *pDst++ = MAU_REG_Q31((uint32_t)(&MAU0->RES0));
                *pDst++ = MAU_REG_Q31((uint32_t)(&MAU0->RES1));
                *pDst++ = MAU_REG_Q31((uint32_t)(&MAU0->RES2));
                *pDst++ = MAU_REG_Q31((uint32_t)(&MAU0->RES3));blkCnt--;
        }
        blkCnt = blockSize % 0x4U;
        while (blkCnt > 0U)
        {
                MAU_REG_Q31(MAU_INDIRECT_ADDR(MAU0_BASE, kMAU_DT_Q1X, kMAU_RES0, kMAU_MOPC_SIN)) = *pIn++ * 2;
                *pDst++ = MAU_REG_Q31((uint32_t)(&MAU0->RES0));
                blkCnt--;
        }
}
```

Note: *Read from the MAU result register is not after write to the MAU indirect address immediately.*

As mentioned in Section 3, the MAU calculation takes 3 cycles and the read operation from the result register is stalled before the calculation is completed. Therefore, the read result register after writing the indirect address stalls the CPU immediately. To avoid these waiting cycles, perform the other tasks.

Figure 1 shows a pipelined operation, which can have a maximum 1.5x times faster performance than a normal operation.

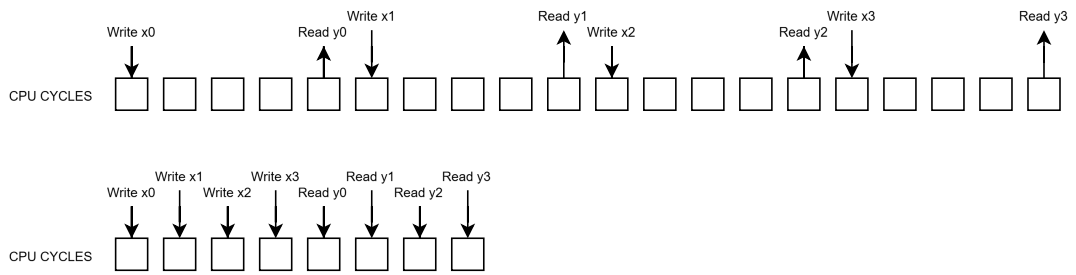

Figure 1. Performance comparison between pipelined operation and normal operation

4.2 Override result data type

MAU result data type can be overridden to save the CPU cycles on data type conversion.

To control the data type override function, use the register `OP_CTRL`.

Figure 2 shows the register `OP_CTRL` definition.

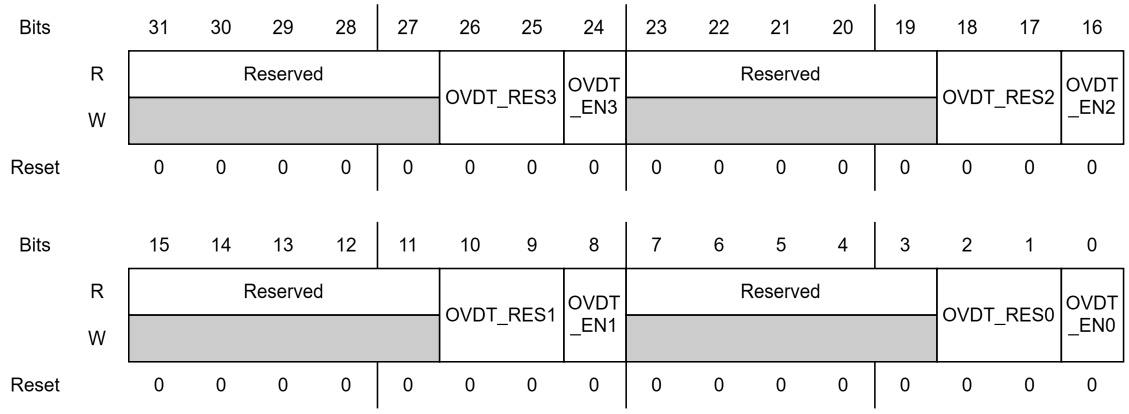

Figure 2. OP_CTRL register

`OP_CTRL` contains four parts. Each part has an `OVDT_ENn` field, which controls whether the data type overridden function is enabled.

The field `OVDT_RESn` controls to which data type, the result is overridden.

The following code snippet shows how to use the data type overridden function:

```
mau_config_t mauCfg;
MAU_GetDefaultConfig(&mauCfg);
MAU_Init(MAU0, &mauCfg);
MAU0->OP_CTRL = MAU_OP_CTRL_OVDT_EN_RES0(0b1) | MAU_OP_CTRL_OVDT_RES0(0b11);
MAU_REG_UINT32(MAU_INDIRECT_ADDR(MAU0, kMAU_DT_UINT, kMAU_RES0, kMAU_MOPC_SQRT))
 = x;
float y = *(volatile float *)MAU0->RES0;
```

5 Note about the source code in the document

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 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.

6 Revision history

Table 4 summarizes the revisions done to this document.

Table 4. Revision history

Document ID	Release date	Description
AN14681 v.1.0	16 July 2025	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.

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.

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.

Suitability for use in industrial applications (functional safety) — This NXP product has been qualified for use in industrial applications. It has been developed in accordance with IEC 61508, and has been SIL-classified accordingly. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

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.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

Contents

Date of release: 16 July 2025
Document identifier: AN14681