AN14377

Continuous SRAM Address Usage on MCXA15x

Rev. 1.0 — 26 July 2024 Application note

Document information

Information	Content
Keywords	AN14377, MCXA, continuous SRAM, remap
Abstract	This application note introduces how to configure and use the SRAM X0 Alias to form a continuous SRAM address.

1 Introduction

Configurable continuous SRAM address is a new and beneficial feature on the MCXA series. Continuous SRAM address offers the following benefits:

- Convenient for DMA operation leveraging the continuous SRAM address mapping.
- Suitable for applications that require a large continuous SRAM region, such as graphic display.

This application note considers MCXA156 as an example to demonstrate how to configure and use a continuous SRAM address.

2 Memory map and architecture

This section describes the memory map and memory architecture of MCXA156.

2.1 Memory map

Table 1 shows the memory map of MCXA156, which can also be found in the attachment of the Reference Manual. The address of SRAM X0 Alias follows the end of SRAM B2, offering a continuous address space with SRAM A0, A1, A2, A3, B0, B1, and B2. As a result, there are a total of 128 KB SRAMs with continuous address.

Table 1. Memory map

Start address (hex)	End address (hex)	Size (KB)	Description
Code bus memory			
04000000	04001FFF	8	SRAM X0 (Slave Port 0)
04002000	04002FFF	4	SRAM X1 (Slave Port 0)
System RAM			
20000000	20001FFF	8	SRAM A0 (Slave Port 1)
20002000	20005FFF	16	SRAM A1 (Slave Port 1)
20006000	20007FFF	8	SRAM A2 (Slave Port 1)
20008000	2000FFFF	32	SRAM A3 (Slave Port 1)
20010000	20017FFF	32	SRAM B0 (Slave Port 3)
20018000	2001BFFF	16	SRAM B1 (Slave Port 3)
2001C000	2001DFFF	8	SRAM B2 (Slave Port 3)
2001E000	**2001FFFF**	**8**	**SRAM X0 Alias**

Note: *The SRAM X0 Alias region is reserved by default.*

2.2 Memory architecture

Figure 1 shows the memory architecture of MCXA156. To access the SRAM X0 Alias region address, enable the corresponding CPU0_SBUS, DMA0, and USB0 bits of the AHB matrix remap control (remap) register in the SYSCON module. As a result, the CM33 System Bus, DMA, and USB FS can access up to 128 KB continuous address. After enabling the bits of remap, both SRAM X0 (0x04000000-0x04001FFF) and SRAM X0 Alias (0x2001E000-0x2001FFFF) region can be accessed. When the SRAM X0 Alias address is accessed, the address gets translated to the corresponding SRAM X0 address. Then the user can access the SRAM X0.

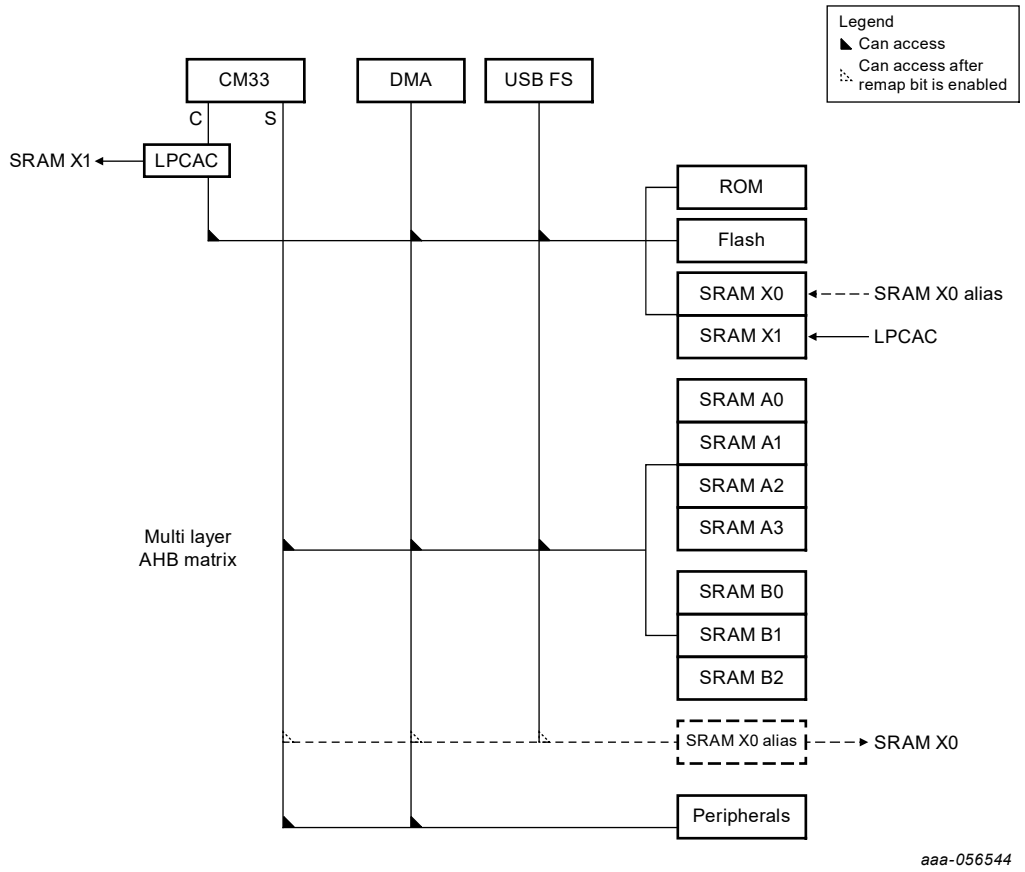

Figure 1. Memory architecture

Note: *CM33 accesses SRAM X0 Alias region through the System Bus and accesses SRAM X0 region through the Code Bus.*

3 SRAM X0 Alias usage limitations

This section lists the limitations on SRAM X0 Alias usage as follows:

- Enable the corresponding remap bit before accessing the SRAM X0 Alias region.
- Consider the value of the stack pointer (SP).

The user usually wants to define a continuous SRAM address in the linker file and allocate the stack region at the end of the SRAM. The stack pointer is 32-bit and the stack region must be 32-bit aligned, so the initial stack pointer address is usually the defined SRAM end address plus one.

If the user wants to use the SRAM X0 Alias region, the initial stack pointer has to be 0x20020000. The ROM checks the stack pointer before jumping to the extended bootloader.

For the MCXA156 extended bootloader, the valid stack pointer region is 0x20000000 to 0x2001FFFF and 0x04000000 to 0x04002FFF, so 0x20020000 is an invalid address. This is the stack pointer limitation of SRAM X0 Alias usage.

4 Workarounds

This section describes the two workarounds to configure continuous SRAM address.

4.1 Workaround to allocate the STACK space to form continuous SRAM address

To allocate the STACK space at the end of a valid stack pointer region and enable remap bits, perform the following steps:

1. Set the initial stack pointer to 0x2001FFFC, which is a valid stack pointer address for ROM validation and is 32-bit aligned.
2. The SRAM space from 0x2001FFFD to 0x2001FFFF can be used for data to check the safety of a stack pointer.
3. Then, enable the corresponding remap bit before using the stack.

4.2 Workaround to cheat the ROM to validate the SP value successfully to form a continuous SRAM address

To cheat the ROM and form a continuous SRAM address, perform the following steps:

1. Set the first word of the vector table to a valid stack pointer value.
2. Then, enable the corresponding remap bit.
3. Set the SP register to the desired value before using the stack.

Note: *The SP value in the vector table is only for ROM checking, not for the real SP value. Therefore, take care when using the SP value in the vector table in the application code.*

The following sections describe the detailed steps of this workaround for different IDEs.

4.2.1 MCUXpresso IDE

To modify the project code to implement the continuous SRAM address in the MCUXpresso IDE, perform the following steps:

1. Modify the size of SRAM to include the SRAM X0 Alias region. Also, modify the location and size of SRAMX to reserve the SRAM X0 region to ensure data security, as shown in Figure 2.

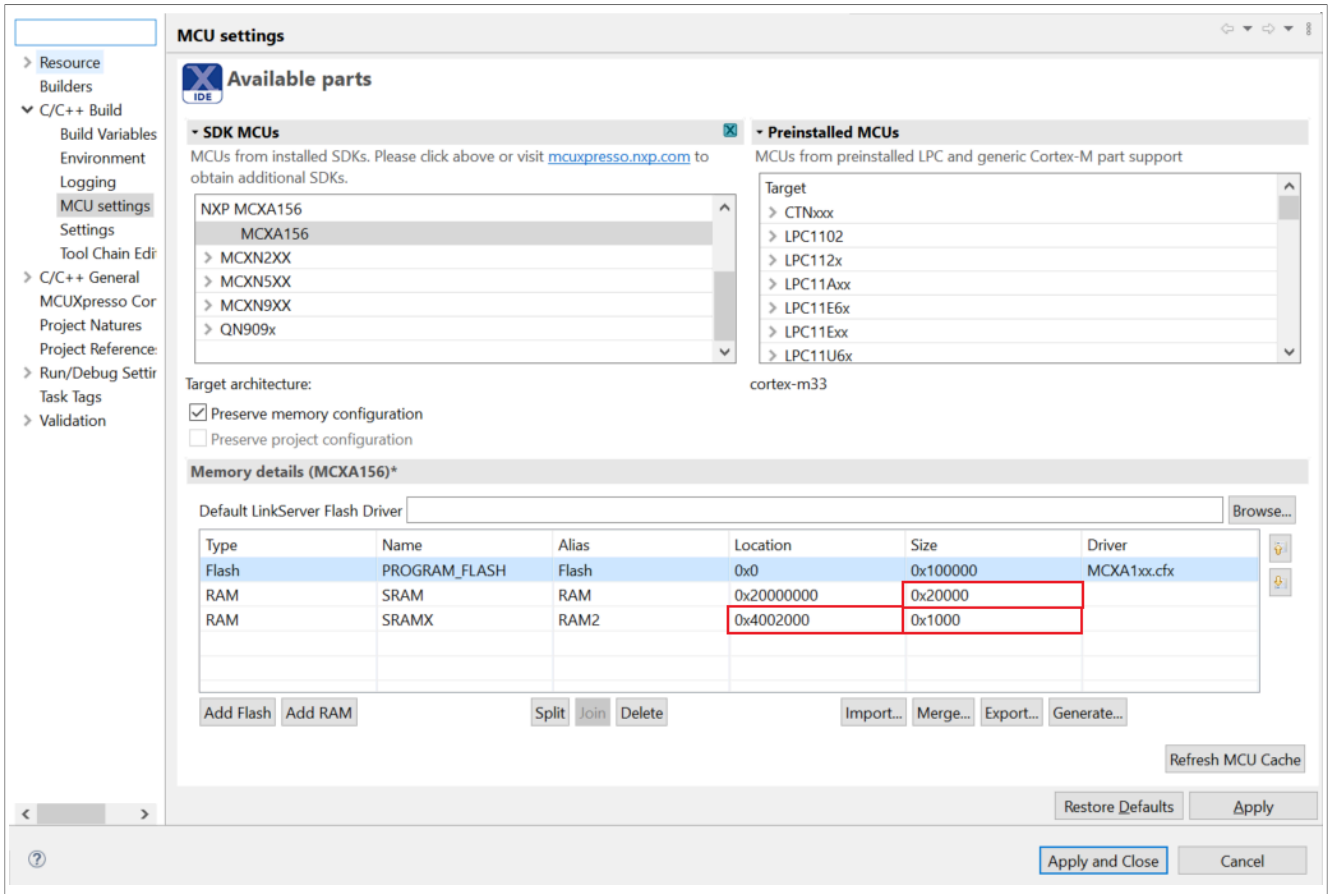

Figure 2. SRAM settings in MCUXpresso IDE

2. Set the initial stack pointer to a valid value for the ROM, as shown in Figure 3.

```
__attribute__ ((used, section(".isr_vector")))
void (* const g_pfnVectors[])(void) = {
    // Core Level - CM33
    &_vStackBase,                   // Set initial stack pointer to a valid value for ROM
    ResetISR,                       // The reset handler
    NMI_Handler,                    // NMI Handler
    HardFault_Handler,              // Hard Fault Handler
```

Figure 3. Set the first word of the vector table in startup_mcxa156.c

3. Enable the corresponding remap bit and set the SP register to the desired value before using the stack, as shown in Figure 4.

```
__attribute__ ((naked, section(".after_vectors.reset")))
void ResetISR(void) {
    // Disable interrupts
    __asm volatile ("cpsid i");
    // Config VTOR & MSPLIM register
    __asm volatile ("LDR  R0, =0xE000ED08  \n"
                    "STR  %0, [R0]         \n"
                    "LDR  R1, [%0]         \n"
                    "MOVS R0, #21          \n"// Enable Remap
                    "LDR  R1, =0x40091200 \n"
                    "STR  R0, [R1]         \n"
                    "MSR  MSP, %2          \n"//Configure SP register to actual value of stack region
                    "MSR  MSPLIM, %1       \n"
                    :
                    : "r"(g_pfnVectors), "r"(_vStackBase), "r"(_vStackTop)
                    : "r0", "r1", "r2");
```

Figure 4. Enable remap and configure the SP register in startup_mcxa156.c

4.2.2 IAR IDE

To modify the project code to implement the continuous SRAM address in IAR IDE, perform the following steps:

1. Modify the `m_data_end` to include the SRAM X0 Alias region. Also, modify the `DATA_region` to reserve the SRAM X0 region to ensure data security, as shown in Figure 5.

```
define symbol m_interrupts_start              = 0x00000000;
define symbol m_interrupts_end                = 0x000001FF;

define symbol m_text_start                    = 0x00000200;
define symbol m_text_end                      = 0x000FFFFF;

define symbol m_data_start                    = 0x20000000;
define symbol m_data_end                      = 0x2001FFFF;

define symbol m_sramx0_start                  = 0x04000000;
define symbol m_sramx0_end                    = 0x04001FFF;


define memory mem with size = 4G;

define region TEXT_region             = mem:[from m_interrupts_start to m_interrupts_end]
                                      | mem:[from m_text_start to m_text_end];
define region DATA_region = mem:[from m_data_start to m_data_end-__size_cstack__];

define region CSTACK_region = mem:[from m_data_end-__size_cstack__+1 to m_data_end];
```

Figure 5. SRAM settings in linker file

2. Set the initial stack pointer to a valid value for the ROM, as shown in Figure 6.

```
__vector_table
        DCD     sfb(CSTACK)                     ;Set initial stack pointer to a valid value for ROM
        DCD     Reset_Handler

        DCD     NMI_Handler                     ;NMI Handler
        DCD     HardFault_Handler               ;Hard Fault Handler
        DCD     MemManage_Handler               ;MPU Fault Handler
        DCD     BusFault_Handler                ;Bus Fault Handler
        DCD     UsageFault_Handler              ;Usage Fault Handler
```

Figure 6. Set the first word of the vector table in startup_MCXA156.s

3. Enable the corresponding remap bit and set the SP register to the desired value before using the stack, as shown in Figure 7.

```
Reset_Handler
        CPSID   I               ; Mask interrupts
        LDR     R0, =0xE000ED08
        LDR     R1, =__vector_table
        STR     R1, [R0]
        MOVS    R0, #21          ;Enable Remap
        LDR     R1, =0x40091200
        STR     R0, [R1]
        LDR     R0, =sfe(CSTACK) ;Configure SP register to actual value of stack region
        MSR     MSP, R0
        LDR     R0, =sfb(CSTACK)
        MSR     MSPLIM, R0
        CPSIE   I               ; Unmask interrupts
        LDR     R0, =SystemInit
        BLX     R0
        LDR     R0, =__iar_program_start
        BX      R0
```

Figure 7. Enable remap and configure SP register in startup_MCXA156.s

4.2.3 Keil IDE

To modify the project code to implement the continuous SRAM address in the Keil IDE, perform the following steps:

1. Modify the `m_data_size` to include the SRAM X0 Alias region, as shown in Figure 8.

```
#define  m_interrupts_start          0x00000000
#define  m_interrupts_size           0x00000200

#define  m_text_start                0x00000200
#define  m_text_size                 0x000FFE00

#define  m_data_start                0x20000000
#define  m_data_size                 0x00020000
```

Figure 8. SRAM settings in linker file

2. Set the initial stack pointer to a valid value for the ROM, as shown in Figure 9.

```
__Vectors:
    .long   Image$$ARM_LIB_STACK$$ZI$$Base          /* Set initial stack pointer to a valid value for ROM */
    .long   Reset_Handler                           /* Reset Handler */
    .long   NMI_Handler                             /* NMI Handler*/
    .long   HardFault_Handler                       /* Hard Fault Handler*/
```

Figure 9. Set the first word of the vector table in startup_MCXA156.S

3. Enable the corresponding remap bit and set the SP register to the desired value before using the stack, as shown in Figure 10.

```
Reset_Handler:
    cpsid   i               /* Mask interrupts */
    .equ    VTOR, 0xE000ED08
    ldr     r0, =VTOR
    ldr     r1, =__Vectors
    str     r1, [r0]
    movs    r0, #21          /* Enable Remap */
    ldr     r1, =0x40091200
    str     r0, [r1]
    ldr     r2, =Image$$ARM_LIB_STACK$$ZI$$Limit /* Configure SP register to actual value of stack region */
    msr     msp, r2
    ldr     r0, =Image$$ARM_LIB_STACK$$ZI$$Base
    msr     msplim, r0
    ldr     r0,=SystemInit
    blx     r0
    cpsie   i               /* Unmask interrupts */
    ldr     r0,=__main
    bx      r0
```

Figure 10. Enable Remap and configure SP register in startup_MCXA156.S

5 Demo validation

This section provides a demo for validating SP register, read and write for boundary unaligned address, and DMA access to continuous SRAM address.

5.1 Hardware and software requirements

Table 2 describes the hardware and software requirements.

Table 2. Hardware and software details

Category	Description
Hardware	• Board: FRDM-MCXA156 • Cable: One Type-C USB
Software	IDEs supported: • MCUXpresso 11.9.0 or later • IAR embedded Workbench 9.50.1 or later • Keil MDK 5.38 or later

5.2 Set up

To set up this demo, perform the following steps:

1. Use a Type-C USB cable to connect J21 of the FRDM-MCXA156 board and the USB port of the PC.
2. Download the project here: https://github.com/nxp-appcodehub/an-continuous-sram-address-mcxa15x or refer to the software attached to this application note.
3. Build and download the project to the FRDM-MCXA156 board.

5.3 SP register validation

The first word in the vector table, the initial stack pointer, has a value of 0x2001f000, which is a valid value for ROM, as shown in Figure 11.

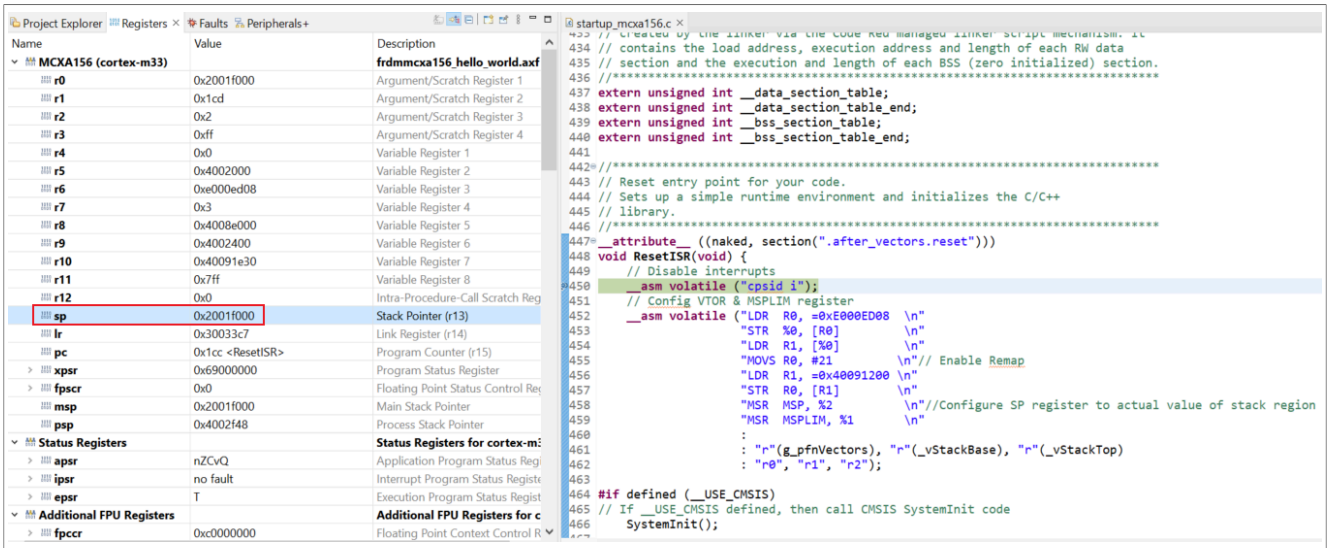

Figure 11. Initial stack pointer value

The corresponding remap bits are enabled, and the value of the SP register is the end address of SRAM X0 Alias plus one, as shown in Figure 12. The above operation is completed before using the stack.

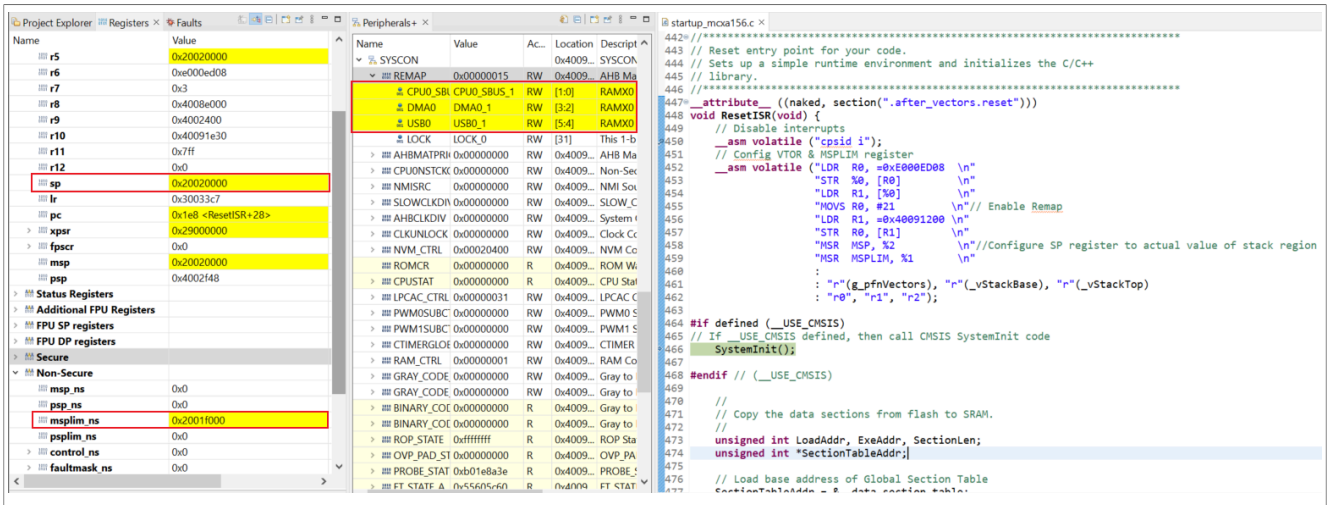

Figure 12. Enable remap and configure the SP register

5.4 Read and write test for boundary unaligned address

Figure 13 shows the read and write test for boundary unaligned address as follows:

- The 32-bit `*p_test` point address is `0x2001DFFF` (the last byte of SRAM B2) and the initial value of `*p_test` is `0x0`.
- Then, write `0xFFFFFFFF` to `*p_test`.
- Finally, the value of `*p_test` is read as `0xffffffff`.

The 32-bit `*p_test` address is not 4 bytes aligned and spans SRAM B2 and SRAM X0 Alias, the read and write works fine.

```
********Read and write test for contiguous SRAM address boundary unaligned address********
32bit *p_test point address : 0x2001DFFF
Initial *p_test = 0x0
Write 0xFFFFFFFF to *p_test
Updated *p_test = 0xffffffff
```

Figure 13. Access to boundary unaligned address

5.5 DMA access to continuous SRAM address test

This demo also tests the DMA access to this continuous SRAM address in Active and Low power mode, and it works fine.

The destination buffer spans between RAM B2 and RAM X0 Alias, and DMA can transport normally as shown from the log information in Figure 14. In Power Down mode, the SYSCON register is in retention state, and can still use DMA partial wakeup to transport data on continuous SRAM address.

```
**********************DMA access to contiguous SRAM address test**********************

******************Active Mode******************

Source Buffer 1:
1       2       3       4
Source Buffer 2:
0       0       0       0
Source Buffer 3:
0       0       0       0
Destination Buffer:
0       0       0       0
Destination Buffer Address:
0x2001dff8      0x2001dffc      0x2001e000      0x2001e004

EDMA Source Buffer 1 to Destination Buffer.
Destination Buffer:
1       2       3       4

EDMA Destination Buffer to Source Buffer 2.
Source Buffer 2:
1       2       3       4

******************Power Down Mode******************
EDMA will transport Destination Buffer to Source Buffer 3.
Input any key to enter Power Down.
Entering Power Down mode...
Please Press SW2(WAKE UP) button to wake MCU and check Source Buffer 3.

MCU woken up
Source Buffer 3:
1       2       3       4
```

Figure 14. DMA access to continuous SRAM address

Figure 15 shows that the MCU is in Power Down mode and uses DMA partial wakeup to transport data on continuous SRAM address.

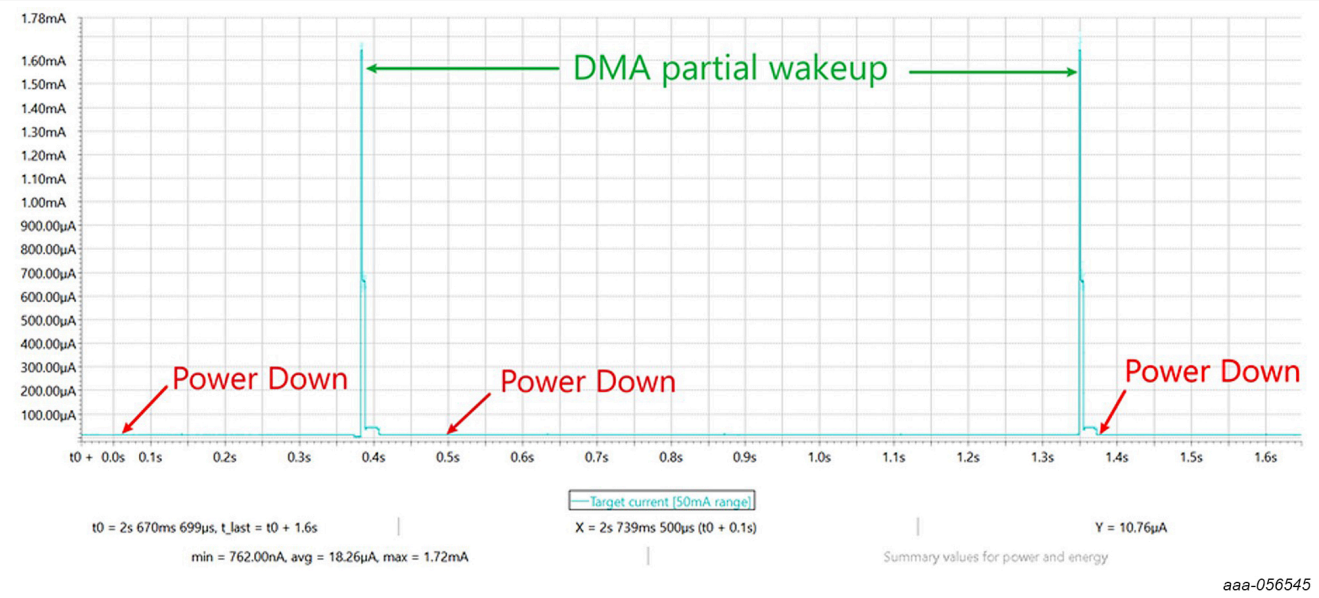

Figure 15. Power consumption measurement

6 Summary

This application note introduces how to configure and use the SRAM X0 Alias to form a continuous SRAM address, and provides a demo to validate the feasibility of continuous SRAM address.

7 Note about the source code in the document

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

Copyright 2024 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.

8 Revision history

Table 3 summarizes the revisions to this document.

Table 3. Revision history

Document ID	Release date	Description
AN14377 v.1	26 July 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