AN14417

Using the SEMC SRAM WAIT feature on i.MX RT1180

Rev. 2.0 — 10 March 2025 Application note

Document information

Information	Content
Keywords	AN14417, SEMC SRAM wait feature, RT1180, i.MX RT1180, smart external memory controller, SEMC, SRAM, WAIT
Abstract	This application note introduces the i.MX RT1180 SEMC SRAM WAIT feature, the background for necessity of the WAIT feature and how it works on i.MX RT1180.

1 Overview

This application note describes the i.MX RT1180 smart external memory controller (SEMC) static random-access memory (SRAM) WAIT feature. The document also provides the background for the necessity of the WAIT feature and how it works on i.MX RT1180. An i.MX RT1170 EVK board helps simulate a device and generate the WAIT signal for evaluation.

2 Importance of the WAIT signal

For some device, each access cycle is not constant and you must not access it by a fixed configuration. In such case, use the WAIT/BUSY/RDY signal to get the maximum bandwidth. For example, a device sometimes needs 80 ns to end the access cycle. Whereas sometimes a device needs 200 ns to end the access cycle. By fixed configuration, one must set the wait time as the maximum access time, which is 200 ns access cycle. In such a case, 120 ns gets wasted for a short period access cycle of 80 ns.

Note: *In different scenario, WAIT is referred to as BUSY or READY/RDY.*

The picture below is an example that the negating WAIT pin terminates the access cycle.

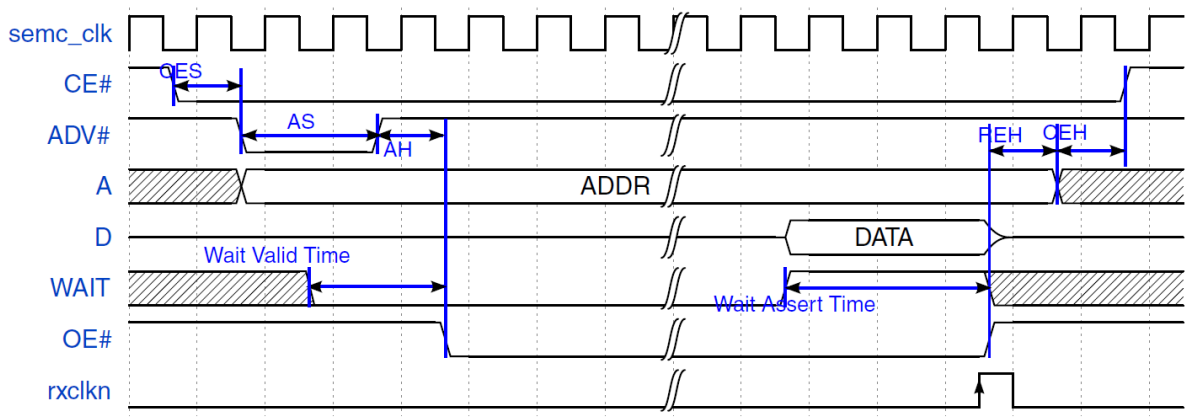

Figure 1. Example of the WAIT pin use case

3 Design of the WAIT feature on i.MX RT1180

This section describes the WAIT feature-related bit and the configuration of the modes.

3.1 WAIT feature-related register bit

The WAIT feature-related register bit includes:

- WAITEN: Bit 2 in SRAMC0/4
- WAITSP: Bit 3 in SRAMC0/4
- WAIT_TIME: Bit 16 in SRAMC0/4

Note:

- *Device 0/CSX 0 uses SRAMC0*
- *Device 1-3/CSX 1-3 uses SRAMC4*

3.2 Configuration for two different modes

The following table lists the two modes used in the application.

Table 1. Configuration of modes

	WAITEN	WAITSP	WAIT_TIME	Comment
Mode 1	1	1	0	If there is WAIT asserted in every access cycle, use this configuration mode.
Mode 2	1	1	1	If the device sometimes asserts WAIT, and sometimes does not assert WAIT, use this configuration mode.

- For Mode 1: SEMC checks the WAIT rising edge. When WAIT rising edge appears, SEMC immediately terminates the current access cycle.
- For Mode 2: SEMC at first counts the REL/WEL down. When a REL/WEL timeout happens, SEMC checks the WAIT line status. If WAIT asserts at this moment, SEMC waits until WAIT negates. If WAIT does not assert at the moment, SEMC terminates the current access cycle.

4 Evaluate SEMC WAIT feature on i.MX RT1180 EVK

This section describes the steps to route signals out from i.MX RT1180 and simulate device behavior. The section also describes the test cases and result for mode 1 and mode 2.

4.1 How to route signals out from i.MX RT1180 EVK

Due to hardware limitation, this application note shows only the basic WAIT feature behavior, instead of a fully and whole SRAM access with WAIT feature. At least four signals route out to evaluate the SEMC WAIT feature. The signals are CS, WAIT, OE, and WE.

The table below guides how to route the signals out from i.MX RT1180 EVK.

Table 2. Route signals out from i.MX RT1180 EVK

Signal	Mapping to SEMC	Watchpoint on i.MX RT1180 EVK
CS	SEMC_CSX0	R188
WAIT	SEMC_BA0	R168
OE	SEMC_A12	Pin36 of U16
WE	SEMC_A11	R167

Figure 2. Route signals out from i.MX RT1180 EVK

4.2 How to simulate device behavior

To evaluate the wait feature in different conditions, we need a device for checking CS. Once CS is low, after `Delay_1` it asserts WAIT. After `Delay_2`, it negates WAIT.

The command in the console enables the programming and specification of `Delay_1` and `Delay_2`.

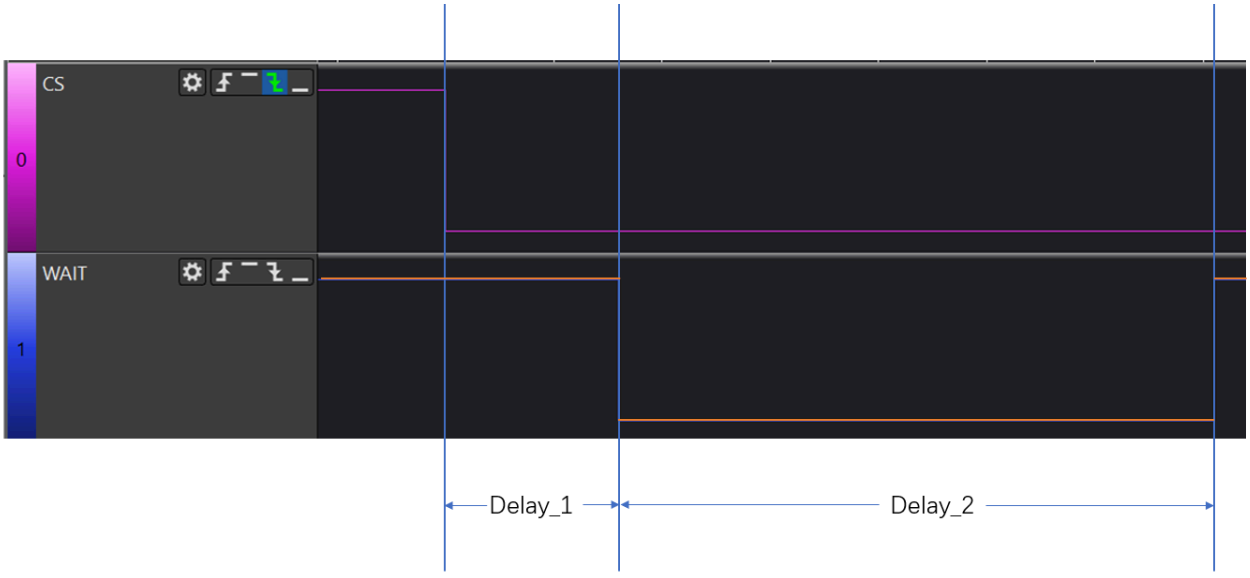

Figure 3. Wait signal generation from device

An i.MX RT1170 EVK board was used for the device test. The figure below shows the connection between the i.MX RT1180 EVK and i.MX RT1170 EVK boards.

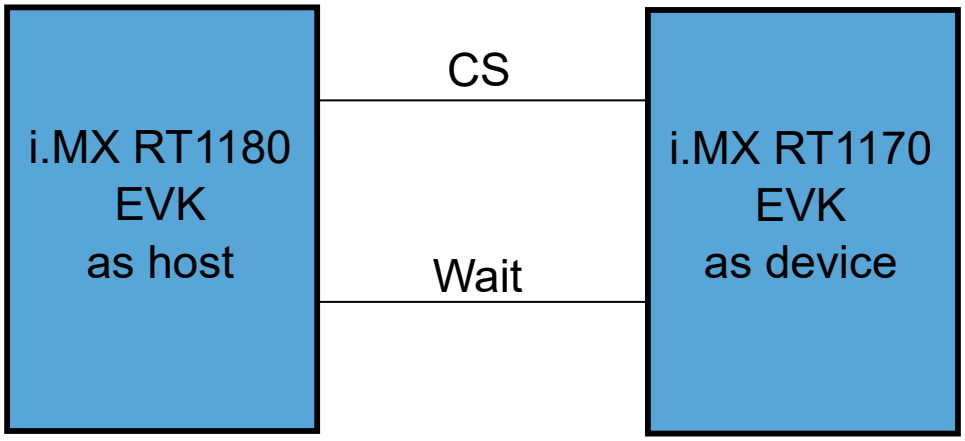

Figure 4. Connection between i.MX RT1180 EVK and i.MX RT1170 EVK

For i.MX RT1170 pin assignment, see the table below.

Table 3. i.MX RT1170 pin assignment

Signal	Pin assignment	Location on board
CS	GPIO_AD_11	J26 pin 4
WAIT	GPIO_AD_10	J26 pin 2
GND		J26 pin 1

For the location of the signals on the board, see the image below.

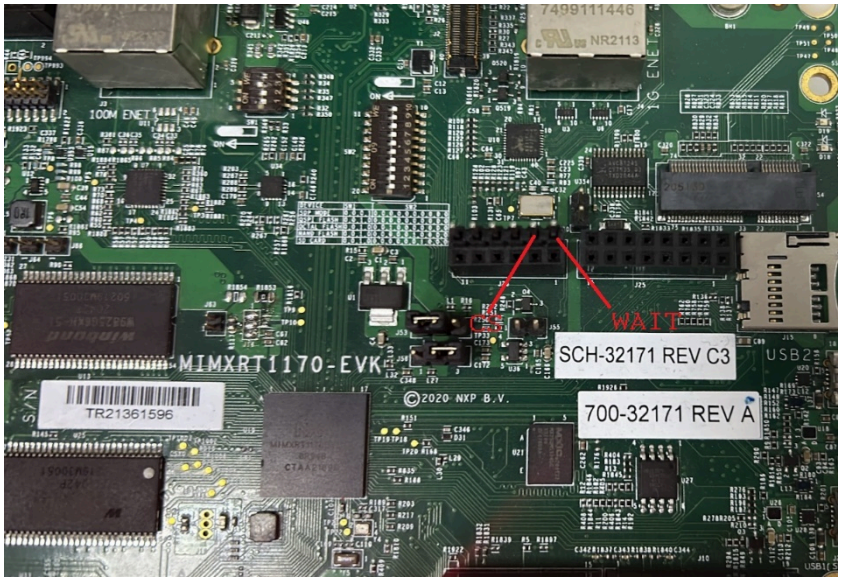

Figure 5. Location of CS/WAIT signal on board

4.3 Test cases and result

This section introduces some test cases to check the WAIT feature on the i.MX RT1180 EVK board.

The `delay` and `wait_low` value mentioned below is the parameter for `void wait_simulate(int delay, int wait_low)` and it is used to generate `Delay_1` and `Delay_2` mentioned in Section 4.2. For more information on code, see Section 5.2.

4.3.1 Test cases for Mode 1

Test case 1: Read, WAIT for 16 ns (`delay=0, wait_low=0`).

Result:

Figure 6. Mode 1, Read, WAIT for 16 ns

Test case 2: Read, WAIT for 104 ns (`delay=0, wait_low=10`). The following image illustrates that the WAIT signal extends the access cycle.

Result:

Figure 7. Mode 1, Read, WAIT for 104 ns

Test case 3: Write, WAIT for 16 ns (`delay=0, wait_low=0`).

Result:

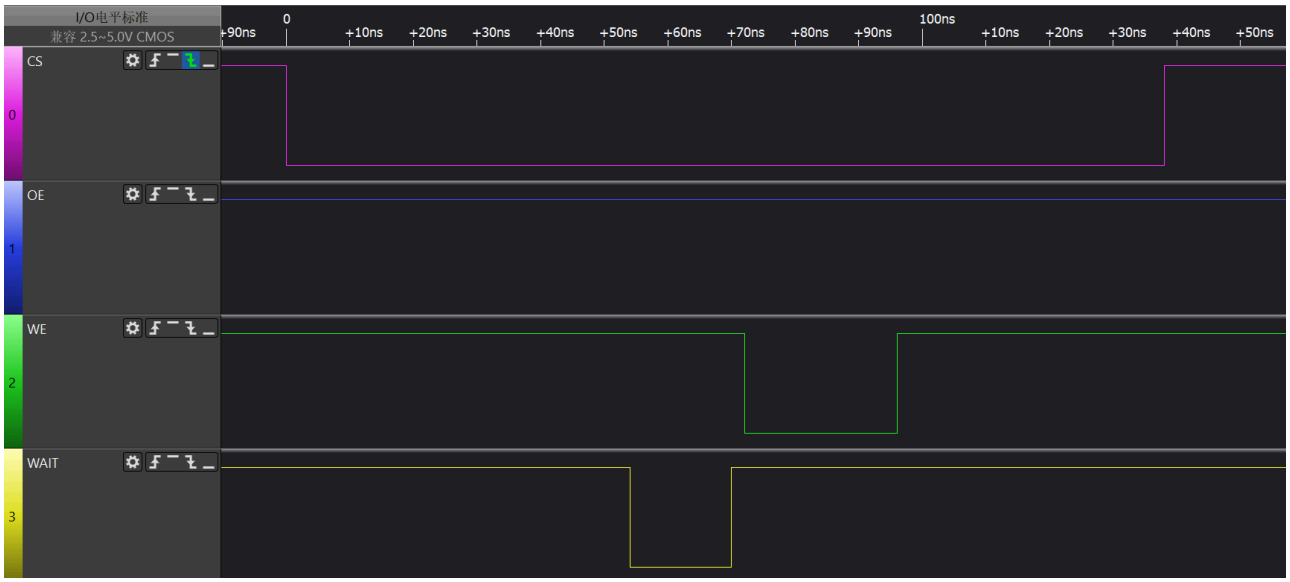

Figure 8. Mode 1, Write, WAIT for 16 ns

Test case 4: Write, WAIT for 104 ns (`delay=0, wait_low=10`). The following image illustrates that the WAIT signal extends the access cycle.

Result:

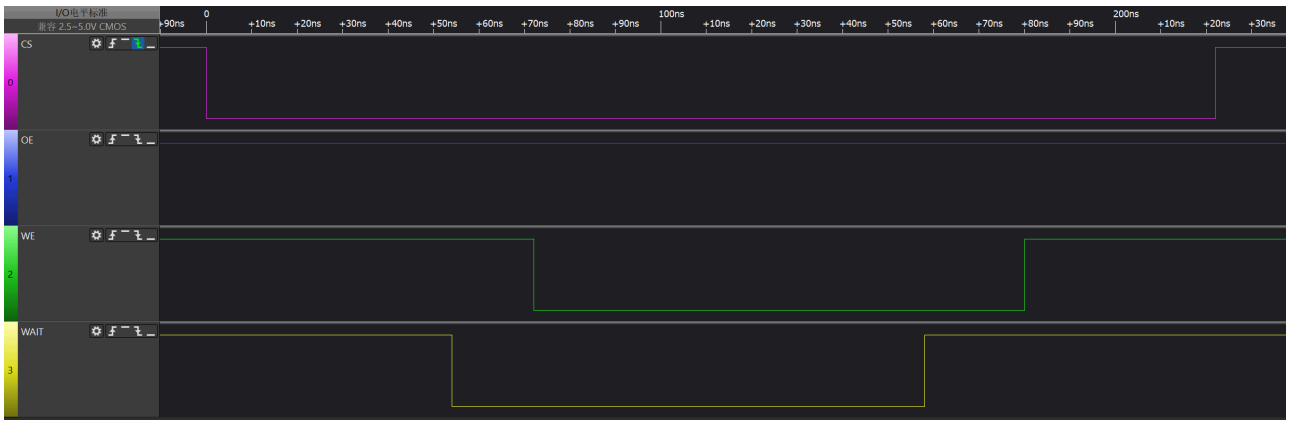

Figure 9. Mode 1, Write, WAIT for 104 ns

4.3.2 Test cases for Mode 2

Test case 1: Read, no WAIT (`delay=20, wait_low=0`). When REL timeout, WAIT is not asserted, then the SEMC begins to terminate the current access cycle.

Result:

Figure 10. Read, no WAIT

Test case 2: Read, with WAIT (`delay=0, wait_low=10`). When REL timeout, WAIT is asserted, then the SEMC terminates the current access cycle after WAIT negates.

Result:

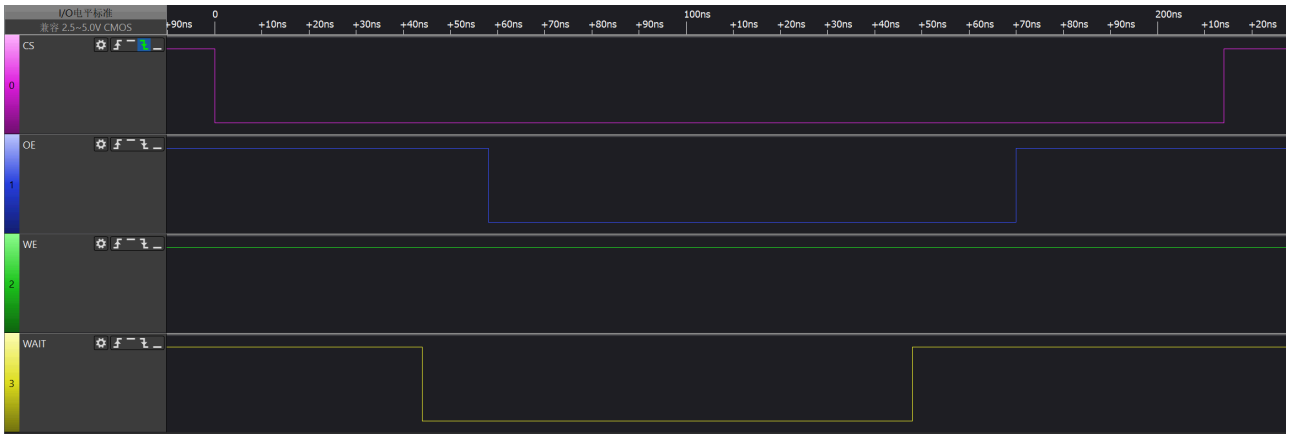

Figure 11. Read, with WAIT

Test case 3: Write, no WAIT (`delay=20, wait_low=0`). When WEL timeout, WAIT is not asserted, then the SEMC begins to terminate the current access cycle.

Result:

Figure 12. Write, no WAIT

Test case 4: Write, with WAIT (`delay=0, wait_low=10`). When WEL timeout, WAIT is asserted, then the SEMC terminates the current access cycle after WAIT negates.

Result:

Figure 13. Write, with WAIT

5 Key code

This section lists the key codes on the i.MX RT1180 EVK and i.MX RT1170 EVK boards.

5.1 Key code on i.MX RT1180

The test code is based on the SEMC driver example in the SDK.

Make sure to configure `GPIO_EMC_B1_41` for `SEMC_CSX00`.

Also, `BOARD_InitPins()` must have the following code added to it.

```
IOMUXC_SetPinMux(IOMUXC_GPIO_EMC_B1_41_SEMC_CSX00, 0U);
IOMUXC_SetPinConfig(IOMUXC_GPIO_EMC_B1_41_SEMC_CSX00, 0x08U);
```

`Semc_sdram.c` must have the following code applied to it.

```
/*
 * Copyright 2017-2020 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */
#include "fsl_debug_console.h"
#include "fsl_device_registers.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "fsl_semc.h"
#include "fsl_cache.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define EXAMPLE_SEMC               SEMC
#define EXAMPLE_SEMC_START_ADDRESS (0x80000000U)
#define EXAMPLE_SEMC_CLK_FREQ      CLOCK_GetRootClockFreq(kCLOCK_Root_Semc)


/*******************************************************************************
 * Code
 ******************************************************************************/
void APP_ConfigMPU(void)
{
    /* Disable code & system cache */
    XCACHE_DisableCache(XCACHE_PC);
    XCACHE_DisableCache(XCACHE_PS);

    /* Disable MPU */
    ARM_MPU_Disable();

    /* Region 9 setting: Memory with Normal type, not shareable, outer/inner
 write through */
    ARM_MPU_SetRegion(2U, ARM_MPU_RBAR(EXAMPLE_SEMC_START_ADDRESS,
 ARM_MPU_SH_OUTER, 0U, 1U, 0U), ARM_MPU_RLAR(0xDFFFFFFF, 1U));

    /* Enable MPU */
    ARM_MPU_Enable(MPU_CTRL_PRIVDEFENA_Msk);

    /* Enable code & system cache */
    XCACHE_EnableCache(XCACHE_PS);
    XCACHE_EnableCache(XCACHE_PC);
}
status_t BOARD_InitSEMC_SRAM(void)
{
    status_t status;
    semc_config_t config;
    semc_sram_config_t sram_config;
    uint32_t clockFrq = EXAMPLE_SEMC_CLK_FREQ;

    memset(&config, 0, sizeof(semc_config_t));
    memset(&sram_config, 0, sizeof(sram_config));

    /* Initialize SEMC. */
    SEMC_GetDefaultConfig(&config);
```

```
    config.dqsMode = kSEMC_Loopbackdqspad; /* For more accurate timing. */
    SEMC_Init(SEMC, &config);


    //SRAM config.
    sram_config.cePinMux = kSEMC_MUXCSX0;
    sram_config.addr27   = kSEMC_MORA27_NONE;
    sram_config.address  = EXAMPLE_SEMC_START_ADDRESS;
    sram_config.memsize_kbytes = 16;
    sram_config.addrPortWidth  = 8;

    sram_config.addrMode = kSEMC_AddrDataNonMux;
    sram_config.burstLen = kSEMC_Nor_BurstLen1;
    sram_config.portSize = kSEMC_PortSize8Bit;
    sram_config.syncMode = kSEMC_AsyncMode;

    sram_config.waitEnable = true;
    sram_config.waitSample = true;

    sram_config.advLevelCtrl = kSEMC_AdvHigh;

    #define NS_SET 20
    sram_config.tCeSetup_Ns       = NS_SET;
    sram_config.tCeHold_Ns        = NS_SET;
    sram_config.tAddrSetup_Ns     = NS_SET;
    sram_config.tAddrHold_Ns      = NS_SET;
    sram_config.tWeLow_Ns         = NS_SET + 20;
    sram_config.tWeHigh_Ns        = NS_SET;
    sram_config.tReLow_Ns         = NS_SET + 20;
    sram_config.tReHigh_Ns        = NS_SET;
    sram_config.tCeInterval_Ns    = NS_SET;

    // For async mode
    sram_config.tTurnAround_Ns     = NS_SET;
    sram_config.tAddr2WriteHold_Ns = NS_SET;


    PRINTF("semc clock = %d\r\n", clockFrq);

    status = SEMC_ConfigureSRAM(SEMC, &sram_config, clockFrq);

    // Enable Mode 2, mask this line to get Mode 1.
    SEMC->SRAMCR0 |= 0x10000;


    PRINTF("SEMC->SRAMCR0 = %x\r\n", SEMC->SRAMCR0);
    PRINTF("SEMC->SRAMCR1 = %x\r\n", SEMC->SRAMCR1);
    PRINTF("SEMC->SRAMCR2 = %x\r\n", SEMC->SRAMCR2);
    PRINTF("SEMC->SRAMCR3 = %x\r\n", SEMC->SRAMCR3);

    if(SEMC->SRAMCR0 & 0x10000)
    {
        PRINTF("wait mode: mode 2.\r\n");
    }
    else
    {
        PRINTF("wait mode: mode 1.\r\n");
    }

    return status;
```

```
}
void test_semc_sram(void)
{
    char c;
    volatile uint8_t  * p_8  = (volatile uint8_t *) EXAMPLE_SEMC_START_ADDRESS;
    volatile uint8_t  r_8;


    while (1)
    {
        PRINTF("Press r for read and w for write. \r\n");
        c = GETCHAR();

        if(c == 'r')
        {
            PRINTF("read. \r\n");
            r_8 = *p_8;
        }

        if(c == 'w')
        {
            PRINTF("write. \r\n");
            *p_8 = r_8;
        }
    }
}


int main(void)
{
    BOARD_ConfigMPU();
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();
    APP_ConfigMPU();

    PRINTF("\r\n SEMC SDRAM Example Start!\r\n");
    if (BOARD_InitSEMC_SRAM() != kStatus_Success)
    {
        PRINTF("\r\n SEMC SDRAM Init Failed\r\n");
    }

    while (1)
    {
        test_semc_sram();
    }
}
```

5.2 Key code on i.MX RT1170

Apply the following code in the hello_world.c file of the hello world example in the SDK.

```
/*
 *  * Copyright 2016-2017 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */
```

```
#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "fsl_iomuxc.h"
#include <cr_section_macros.h>


/*******************************************************************************
 * Code
 ******************************************************************************/
#define WAIT_HIGH CM7_GPIO3->DR_SET = 0x200
#define WAIT_LOW  CM7_GPIO3->DR_CLEAR = 0x200



/*
 delay: Control the time delay to assert WAIT after CS falling edge.
 low:   Control the WAIT low period length.

 Typically used value:
 Short wait period:                 delay = 0,  wait_low = 0,  16ns,  16ns
 Long wait period :                 delay = 0;  wait_low = 10; 0ns,   104 ns
 No wait period during access cycle: delay = 10, wait_low = 0;  104ns, 0ns,
*/
__RAMFUNC(SRAM_ITC_cm7) void wait_simulate(int delay, int wait_low)
{
    while(CM7_GPIO3->DR & 0x400)
        ;

    while(delay)
       delay--;
    WAIT_LOW;

    while(wait_low)
       wait_low--;
    WAIT_HIGH;
}
void set_gpio3_fast(int pin_index)
{
    if(pin_index <= 16)
    {
        IOMUXC_GPR->GPR42 |= (1UL << pin_index);
    }
    else
    {
        IOMUXC_GPR->GPR43 |= (1UL << (pin_index-16));
    }
}
void rgpio_init(void)
{
    gpio_pin_config_t gpio_config_output = {kGPIO_DigitalOutput, 0,
 kGPIO_NoIntmode};
    gpio_pin_config_t gpio_config_input  = {kGPIO_DigitalInput,  0,
 kGPIO_NoIntmode};

    CLOCK_EnableClock(kCLOCK_Iomuxc);

    // WAIT, GPIO_AD_10 --> RGPIO 3, 9,  RGPIO OUTPUT --> J26 pin 2
```

```
    IOMUXC_SetPinMux(IOMUXC_GPIO_AD_10_GPIO_MUX3_IO09,    0U);
    IOMUXC_SetPinConfig(IOMUXC_GPIO_AD_10_GPIO_MUX3_IO09, 0U);

    // CS, GPIO_AD_11 --> GPIO 9, 10, RGPIO INPUT --> J26 pin 4
    IOMUXC_SetPinMux(IOMUXC_GPIO_AD_11_GPIO_MUX3_IO10,    0U);
    IOMUXC_SetPinConfig(IOMUXC_GPIO_AD_11_GPIO_MUX3_IO10, 0U);


    GPIO_PinInit(CM7_GPIO3, 9,  &gpio_config_output);
    GPIO_PinInit(CM7_GPIO3, 10, &gpio_config_input);

    set_gpio3_fast(9);
    set_gpio3_fast(10);

    // Set wait high.
    WAIT_HIGH;
}


int main(void)
{
    int  delay;
    int  low;

    /* Init board hardware. */
    BOARD_ConfigMPU();
    BOARD_InitPins();
    BOARD_BootClockRUN();
    BOARD_InitDebugConsole();

    PRINTF("hello world.\r\n");

    rgpio_init();

    while (1)
    {
       PRINTF("input delay and low: \r\n");
       SCANF("%d %d", &delay, &low);
       PRINTF("delay = %d, low = %d\r\n", delay, low);
       wait_simulate(delay, low);
    }
}
```

6 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 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.

7 Revision history

Table 4 summarizes revisions to this document.

Table 4. Revision history

Document ID	Release date	Description
AN14417 v.2.0	10 March 2025	Replaced "SMEC" with "SEMC"
AN14417 v.1.0	18 February 2025	Initial public revision

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