NUC472 EBI SRAM HEAP

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Heap Space on EBI SRAM
BSP Version	NUC472/NUC442 Series BSP CMSIS v3.03.000
Hardware	Nuvoton NUC472 Demo Board V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

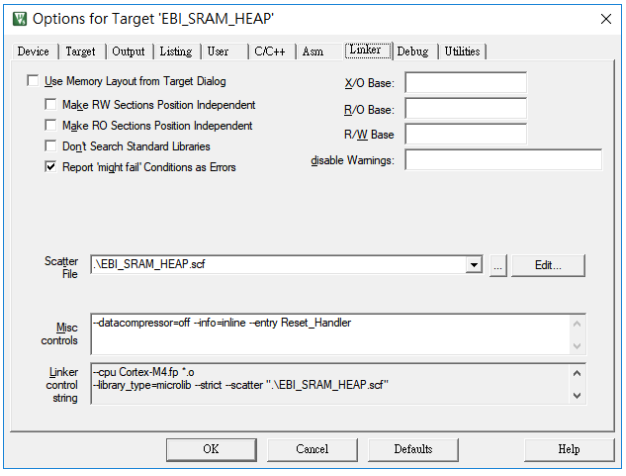
1.1 Introduction

NuMicro® NUC472/NUC442 series is equipped with an external bus interface (EBI) for connecting external devices. NUC472/NUC442 EBI facilitates connecting the device of which control interface is composed of address, data, chip select, and read/write strobe, for example NOR flash, NAND flash, and SRAM. In example EBI_SRAM_HEAP, NUC472/NUC442 employs EBI to connect a SRAM device and make this SRAM as extended memory. The Keil project configures the heap space of EBI_SRAM_HEAP to locate on address of EBI SRAM. This example uses malloc() to allocate memory from heap, that is, EBI SRAM.

1.2 Principle

NuMicro® NUC472/NUC442 series EBI supports four banks, which can physically connect to four different instance of external devices. Each EBI bank has the hardware fixed physical address, which is unchangeable. The physical address for EBI bank 0~4 are 0x60000000, 0x64000000, 0x68000000, and 0x6C000000 respectively. In this example code, EBI_SRAM_HEAP uses EBI bank 0 to connect the BS616LV4017EIG70 SRAM device. This gives EBI_SRAM_HEAP 512K bytes extra memory presented on memory space 0x60000000.

In EBI_SRAM_HEAP Keil project, the scatter file “EBI_SRAM_HEAP.scf” configures heap space be located on 0x60000000, as the following figure:



The content of “EBI_SRAM_HEAP.scf”:

```
LR_IROM1 0x0000000 0x00080000
{
```

```
  ER_IROM1 0x0000000 0x00080000
        {
    *.o (RESET, +First)
    *(InRoot$$Sections)
    .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x0005000
  {
    .ANY (+RW +ZI)
  }

  RW_IRAM2 0x60000000 UNINIT 0x00020000
  {
  startup_nuc472_442.o(HEAP)
  }

}
```

1.3 Demo Result

```
+--------------------------------------------------+
|    EBI_SRAM_HEAP using BS616LV4017EIG70 SRAM     |
+--------------------------------------------------+
Allocate 1024 bytes success, memory address = 0x60000008.
Allocate 5120 bytes success, memory address = 0x60000008.
Allocate 9216 bytes success, memory address = 0x60000008.
Allocate 13312 bytes success, memory address = 0x60000008.
Allocate 256 bytes success, memory address = 0x60000008.
Allocate 512 bytes success, memory address = 0x60000110.
Allocate 768 bytes success, memory address = 0x60000318.
Allocate 1024 bytes success, memory address = 0x60000620.
Allocate 1280 bytes success, memory address = 0x60000a28.
Allocate 1536 bytes success, memory address = 0x60000f30.
Allocate 1792 bytes success, memory address = 0x60001538.
Allocate 2048 bytes success, memory address = 0x60001c40.
0x60000000 --> 0x60010000 <-- 0x60020000
Butterfly test 1... Pass
Butterfly test 2... Pass
```

```
Butterfly test 3... Pass
Butterfly test 4... Pass

Byte Test (Bank-0) ... PASS
Short Test (Bank-0) ... PASS
Word Test (Bank-0) ... PASS

All test passed.
```

2 Code Description

Initialize EBI controller and multiple function pins:

```
    /* Enable EBI engine clock */
    CLK_EnableModuleClock(EBI_MODULE);

    /* Configure EBI multi-function pins */
        SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA10MFP_Msk) ) |
                        SYS_GPA_MFPH_PA10MFP_EBI_A20;     /* A20. =   PA10 */
        SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA9MFP_Msk) ) |
                        SYS_GPA_MFPH_PA9MFP_EBI_A19;      /* A19. =   PA9 */
        SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA8MFP_Msk) ) |
                        SYS_GPA_MFPH_PA8MFP_EBI_A18;      /* A18. =   PA8 */
        SYS->GPA_MFPL = (SYS->GPA_MFPL & (~SYS_GPA_MFPL_PA7MFP_Msk) ) |
                        SYS_GPA_MFPL_PA7MFP_EBI_A17;      /* A17. =   PA7 */
        SYS->GPA_MFPL = (SYS->GPA_MFPL & (~SYS_GPA_MFPL_PA6MFP_Msk) ) |
                        SYS_GPA_MFPL_PA6MFP_EBI_A16;      /* A16. =   PA6 */
        SYS->GPB_MFPH = (SYS->GPB_MFPH & (~SYS_GPB_MFPH_PB13MFP_Msk) ) |
                        SYS_GPB_MFPH_PB13MFP_EBI_AD15;    /* AD15 =   PB13 */
        SYS->GPB_MFPH = (SYS->GPB_MFPH & (~SYS_GPB_MFPH_PB12MFP_Msk) ) |
                        SYS_GPB_MFPH_PB12MFP_EBI_AD14;    /* AD14 =   PB12 */
        SYS->GPB_MFPH = (SYS->GPB_MFPH & (~SYS_GPB_MFPH_PB11MFP_Msk) ) |
                        SYS_GPB_MFPH_PB11MFP_EBI_AD13;    /* AD13 =   PB11 */
        SYS->GPB_MFPH = (SYS->GPB_MFPH & (~SYS_GPB_MFPH_PB10MFP_Msk) ) |
                        SYS_GPB_MFPH_PB10MFP_EBI_AD12;    /* AD12 =   PB10 */
        SYS->GPB_MFPH = (SYS->GPB_MFPH & (~SYS_GPB_MFPH_PB9MFP_Msk) ) |
                        SYS_GPB_MFPH_PB9MFP_EBI_AD11;     /* AD11 =   PB9 */
        SYS->GPB_MFPH = (SYS->GPB_MFPH & (~SYS_GPB_MFPH_PB8MFP_Msk) ) |
                        SYS_GPB_MFPH_PB8MFP_EBI_AD10;     /* AD10 =   PB8 */
        SYS->GPB_MFPL = (SYS->GPB_MFPL & (~SYS_GPB_MFPL_PB7MFP_Msk) ) |
                        SYS_GPB_MFPL_PB7MFP_EBI_AD9;      /* AD9 =    PB7 */
        SYS->GPB_MFPL = (SYS->GPB_MFPL & (~SYS_GPB_MFPL_PB6MFP_Msk) ) |
                        SYS_GPB_MFPL_PB6MFP_EBI_AD8;      /* AD8 =    PB6 */
        SYS->GPB_MFPL = (SYS->GPB_MFPL & (~SYS_GPB_MFPL_PB5MFP_Msk) ) |
                        SYS_GPB_MFPL_PB5MFP_EBI_AD7;      /* AD7 =    PB5 */
        SYS->GPB_MFPL = (SYS->GPB_MFPL & (~SYS_GPB_MFPL_PB4MFP_Msk) ) |
                        SYS_GPB_MFPL_PB4MFP_EBI_AD6;      /* AD6 =    PB4 */
        SYS->GPB_MFPL = (SYS->GPB_MFPL & (~SYS_GPB_MFPL_PB3MFP_Msk) ) |
                        SYS_GPB_MFPL_PB3MFP_EBI_AD5;      /* AD5 =    PB3 */
        SYS->GPB_MFPL = (SYS->GPB_MFPL & (~SYS_GPB_MFPL_PB2MFP_Msk) ) |
```

```
                    SYS_GPB_MFPL_PB2MFP_EBI_AD4;      /* AD4 =    PB2 */
    SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA14MFP_Msk) ) |
                    SYS_GPA_MFPH_PA14MFP_EBI_AD3;     /* AD3. =   PA14 */
    SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA13MFP_Msk) ) |
                    SYS_GPA_MFPH_PA13MFP_EBI_AD2;     /* AD2. =   PA13 */
    SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA12MFP_Msk) ) |
                    SYS_GPA_MFPH_PA12MFP_EBI_AD1;     /* AD1. =   PA12 */
    SYS->GPA_MFPH = (SYS->GPA_MFPH & (~SYS_GPA_MFPH_PA11MFP_Msk) ) |
                    SYS_GPA_MFPH_PA11MFP_EBI_AD0;     /* AD0. =   PA11 */
    SYS->GPE_MFPL = (SYS->GPE_MFPL & (~SYS_GPE_MFPL_PE6MFP_Msk) ) |
                    SYS_GPE_MFPL_PE6MFP_EBI_nWR;      /* PE.6 =   nWR */
    SYS->GPE_MFPL = (SYS->GPE_MFPL & (~SYS_GPE_MFPL_PE7MFP_Msk) ) |
                    SYS_GPE_MFPL_PE7MFP_EBI_nRD;      /* PE.7 =   nRD */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE8MFP_Msk) ) |
                    SYS_GPE_MFPH_PE8MFP_EBI_ALE;      /* PE.8 =   ALE */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE9MFP_Msk) ) |
                    SYS_GPE_MFPH_PE9MFP_EBI_nWRH;     /* PE.9 =   WRH */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE10MFP_Msk) ) |
                    SYS_GPE_MFPH_PE10MFP_EBI_nWRL;    /* PE.10 =  WRL */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE11MFP_Msk) ) |
                    SYS_GPE_MFPH_PE11MFP_EBI_nCS0;    /* PE.11 = nCS0 */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE12MFP_Msk) ) |
                    SYS_GPE_MFPH_PE12MFP_EBI_nCS1;    /* PE.12 = nCS1 */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE13MFP_Msk) ) |
                    SYS_GPE_MFPH_PE13MFP_EBI_nCS2;    /* PE.13 = nCS2 */
    SYS->GPE_MFPH = (SYS->GPE_MFPH & (~SYS_GPE_MFPH_PE14MFP_Msk) ) |
                    SYS_GPE_MFPH_PE14MFP_EBI_nCS3;    /* PE.14 = nCS3 */
```

Initialize EBI banks:

```
void EBI_Init(void)
{
    const uint32_t u32Timing = 0x21C;

    /* Open EBI interface */
    EBI_Open(EBI_BANK0, EBI_BUSWIDTH_16BIT, EBI_TIMING_NORMAL,
             EBI_SEPARATEMODE_DISABLE, EBI_CS_ACTIVE_LOW);
    EBI_Open(EBI_BANK1, EBI_BUSWIDTH_16BIT, EBI_TIMING_NORMAL,
             EBI_SEPARATEMODE_DISABLE, EBI_CS_ACTIVE_LOW);
    EBI_Open(EBI_BANK2, EBI_BUSWIDTH_16BIT, EBI_TIMING_NORMAL,
             EBI_SEPARATEMODE_DISABLE, EBI_CS_ACTIVE_LOW);
```

```
    EBI_Open(EBI_BANK3, EBI_BUSWIDTH_16BIT, EBI_TIMING_NORMAL,
             EBI_SEPARATEMODE_DISABLE, EBI_CS_ACTIVE_LOW);

    /* Configure EBI timing */
    EBI_SetBusTiming(EBI_BANK0, u32Timing, EBI_MCLKDIV_2);
    EBI_SetBusTiming(EBI_BANK1, u32Timing, EBI_MCLKDIV_2);
    EBI_SetBusTiming(EBI_BANK2, u32Timing, EBI_MCLKDIV_2);
    EBI_SetBusTiming(EBI_BANK3, u32Timing, EBI_MCLKDIV_2);
}
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC472/NUC442 Series BSP CMSIS v3.03.000
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - Nuvoton NUC472 Demo Board V1.0
  - SRAM BS616LV4017EIG70 x 2
- Diagram

PA.11	↔	AD.0
PA.12	↔	AD.1
PA.13	↔	AD.2
PA.14	↔	AD.3
PB.2	↔	AD.4
PB.3	↔	AD.5
PB.4	↔	AD.6
PB.5	↔	AD.7
PB.6	↔	AD.8
PB.7	↔	AD.9
PB.8	↔	AD.10
PB.9	↔	AD.11
PB.10	↔	AD.12

NUC472/NUC442		BS616LV4017EIG70
PB.11	↔	AD.13
PB.12	↔	AD.14
PB.13	↔	AD.15
PA.6	→	A16
PA.7	→	A17
PA.8	→	A18
PA.9	→	A19
PA.10	→	A20
PE.6	→	nWR
PE.7	→	nRD
PE.8	→	ALE
PE.9	→	WRH
PE.10	→	WRL
PE.11	→	nCS0
PE.12	→	nCS1
PE.13	→	nCS2
PE.14	→	nCS3

4 Directory Information

- EC_NUC472_EBI_SRAM_HEAP_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. Browsing into sample code folder by 4.Directory Information and double click EBI_SRAM_HEAP.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
June 27, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.