Lab Assignment 4

Jose Monroy Villalobos

San José State University
Department of Computer Engineering
CMPE 146-01, Real-Time Embedded System Co-Design, Fall 2024

Exercise 1.1 String Modifications:

Debug Console Outputs:

```
[CORTEX_M4_0] string1: 123123

string2: 123123

string1: 123123

string2: 023123

string1 Address: 0x00002D10

string2 Address: 0x200009E4
```

Explanation:

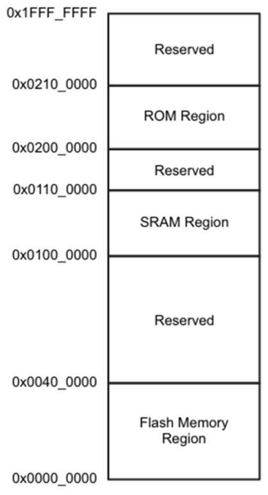

Figure 6-2. Code Zone Memory Map

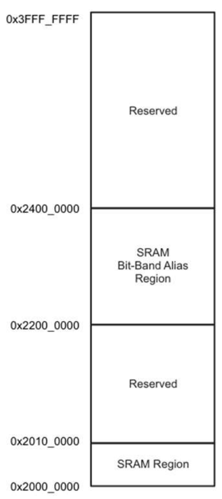

Figure 6-3. SRAM Zone Memory Map

The behavior observed was that string 1 was not written to while string2 was written to. I modified the code to print the address and used these memory map in the data sheet identify where the strings are stored. String 1 address, 0x00002D10, indicates it is in the Flash Main Memory, while string 2 Address, 0x200009E4, indicates it is in SRAM Region. This is the reason as to the behavior observed where string 1 did not actually change. To change string 1 successfully we would need to use a different method such as using driverlib functions. Quoting from 1.2 exercise, "By default, the entire flash memory on the MCU is write protected".

Exercise 1.2 Program flash memory:

1. Function to convert a memory location to bank number and sector mask:

```
bool get_flash_bank_sector(uint32_t mem_address, uint32_t* bank_number, uint32_t*
sector_mask){

    if (mem_address < 0x0001FFFF) {

        *bank_number = FLASH_MAIN_MEMORY_SPACE_BANK0;  // Bank 0
        *sector_mask = 1 << (mem_address / 0x1000);  // 4 KB sectors
        return true;

        }
    else if (mem_address >= 0x00020000 && mem_address < 0x0003FFFF) {
        *bank_number = FLASH_MAIN_MEMORY_SPACE_BANK1;  // Bank 1
        *sector_mask = 1 << ((mem_address - 0x00020000) / 0x1000);  // 4 KB sectors
        return true;
        }
        return false;

}
```

2. Code to unprotect and program the memory:

```
//unprotect
    if (get_flash_bank_sector((uint32_t)string1, &bank_number, &sector_mask)) {
        FlashCtl_unprotectSector(bank_number, sector_mask);
        printf("bank_number: 0x%08X\n", bank_number);
        printf("sector_mask: 0x%08X\n\n", sector_mask);
    }

    printf("string3 Address: 0x%08lX\n", (unsigned long)string3);
    ;

    if (get_flash_bank_sector((uint32_t)string3, &bank_number, &sector_mask)) {
        FlashCtl_unprotectSector(bank_number, sector_mask);
        printf("bank_number: 0x%08X\n", bank_number);
        printf("sector_mask: 0x%08X\n\n", sector_mask);
    }
    //program memory
    ROM_FlashCtl_programMemory(string2, string1, 2)
    ROM_FlashCtl_programMemory(string2, string3, 2);
```

Debug Console Outputs:

```
[CORTEX_M4_0]
string1: 123123
string2: 123123
string3: 123123

string1 Address: 0x000281E8
bank_number: 0x00000002
sector_mask: 0x00000100

string3 Address: 0x000000E4
bank_number: 0x00000001
sector_mask: 0x00000001

string1: 003123
string2: 043123
string3: 003123
```

Explanation:

Flash allows us to write a 1 to a 0 but not the other way without some additional code. The result for string 1 and 3 from being 123123 to 003123 is because we wrote the first two bytes of string 2 which are 04. If we look at the bits it becomes clear.

1	2	0	4
0001	0010	0000	0100

We wrote bytes that contained 0s where the 1s where so it resulted in the first two bytes of string 1 and 3 to become 00.

Exercise 2 Non-volatile power-on-reset counter:

```
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

#include <stdint.h>

#include <stdbool.h>
```

```
#include <stdio.h>



const uint32_t myFlashConst = 0x11100000;

bool get_flash_bank_sector(uint32_t mem_address, uint32_t* bank_number, uint32_t*
sector_mask);



void main(void)

{

  MAP_WDT_A_holdTimer();




  uint32_t bank_number, sector_mask;

  uint32_t decrement = 0;

  //unlock

  printf("myFlashConst: 0x%08lX\n", myFlashConst);

  printf("myFlashConst Address: 0x%08lX\n", (unsigned long)&myFlashConst);

  if (get_flash_bank_sector((uint32_t)&myFlashConst, &bank_number, &sector_mask)) {

      FlashCtl_unprotectSector(bank_number, sector_mask);

      printf("bank_number: 0x%08X\n", bank_number);
```

```
    printf("sector_mask: 0x%08X\n\n", sector_mask);

  }




//detect count and decrement

   if(myFlashConst==0x11100000){

          printf("myFlashConst: 0x%08lX\n", myFlashConst);

   ROM_FlashCtl_programMemory(&decrement, (void*)&myFlashConst, 1);

   }

   else if(myFlashConst==0x01100000){

              printf("myFlashConst: 0x%08lX\n", myFlashConst);

      ROM_FlashCtl_programMemory(&decrement, (void*)&myFlashConst, 2);

      }

   else if(myFlashConst==0x00100000){

              printf("myFlashConst: 0x%08lX\n", myFlashConst);

      ROM_FlashCtl_programMemory(&decrement, (void*)&myFlashConst, 3);

      }
```

Attempted to get this one and failed. The goal was to in each power on delete one of the the leading hex digit and since it started with 111 it would last 3 cycles. There is no way to debug and see what is going on when I unplug and plug back in and so I was stuck and unable to see what was going wrong. If I had a display on the board then I could debug.

Exercise 3 SRAM function:

Spent too much time trying to figure out exercise 2 that I did not get to this exercise.

Appendix:

Exercise 1.1 String Modifications:

```
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>

char* string1  = "123123";
char string2[] = "123123";

void main(void)
{
    MAP_WDT_A_holdTimer();
    printf("string1: %s\n", string1);
    printf("string2: %s\n", string2);
    string1[0] = '0';
    string2[0] = '0';
    printf("string1: %s\n", string1);
    printf("string2: %s\n", string2);

    //Allows us to find where the strings are stored
    printf("string1 Address: 0x%08lX\n", (unsigned long)string1);
    printf("string2 Address: 0x%08lX\n", (unsigned long)string2);
}
```

Exercise 1.2 Program flash memory:

```
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>

char* string1  = "123123";
char string2[] = "123123";
const char string3[1024 * 160] = {"123123"};

bool get_flash_bank_sector(uint32_t mem_address, uint32_t* bank_number, uint32_t*
sector_mask);

void main(void)
{
    MAP_WDT_A_holdTimer();
    printf("\nstring1: %s\n", string1);
    printf("string2: %s\n", string2);
    printf("string3: %s\n\n", string3);

    string1[0] = '0';

    string2[0] = '0';
    string2[1] = '4';

    /*  Unprotect the sector:
     *  bool FlashCtl_unprotectSector(uint_fast8_t memorySpace, uint32_t sectorMask);
     *
     *  To program the memory, use the following DriverLib function:
     *  bool FlashCtl_programMemory(void* src, void* dest, uint32_t length);
     *
     *  In order to modify the other two strings, string1 and string3, we will need
to use the FlashCtl_xxx DriverLib
     *  functions.  Duplicate the first two bytes of string2 onto string1 and string3
using the following calls.
     *  ROM_FlashCtl_programMemory(string2, string1, 2);
     *  ROM_FlashCtl_programMemory(string2, string3, 2);
    */
    uint32_t bank_number, sector_mask;

    //unprotect
        printf("string1 Address: 0x%08lX\n", (unsigned long)string1);
    if (get_flash_bank_sector((uint32_t)string1, &bank_number, &sector_mask)) {
        FlashCtl_unprotectSector(bank_number, sector_mask);
        printf("bank_number: 0x%08X\n", bank_number);
        printf("sector_mask: 0x%08X\n\n", sector_mask);
    }

    printf("string3 Address: 0x%08lX\n", (unsigned long)string3);
    ;

    if (get_flash_bank_sector((uint32_t)string3, &bank_number, &sector_mask)) {
        FlashCtl_unprotectSector(bank_number, sector_mask);
        printf("bank_number: 0x%08X\n", bank_number);
        printf("sector_mask: 0x%08X\n\n", sector_mask);
    }
    //program memory
    ROM_FlashCtl_programMemory(string2, string1, 2)
    ROM_FlashCtl_programMemory(string2, string3, 2);
    printf("string1: %s\n", string1);
    printf("string2: %s\n", string2);
    printf("string3: %s\n", string3);



    //printf("string2 Address: 0x%08lX\n", (unsigned long)string2);

}

bool get_flash_bank_sector(uint32_t mem_address, uint32_t* bank_number, uint32_t*
sector_mask){

    if (mem_address < 0x0001FFFF) {

        *bank_number = FLASH_MAIN_MEMORY_SPACE_BANK0;  // Bank 0
        *sector_mask = 1 << (mem_address / 0x1000);  // 4 KB sectors
        return true;

        }
    else if (mem_address >= 0x00020000 && mem_address < 0x0003FFFF) {
        *bank_number = FLASH_MAIN_MEMORY_SPACE_BANK1;  // Bank 1
        *sector_mask = 1 << ((mem_address - 0x00020000) / 0x1000);  // 4 KB sectors
        return true;
        }
        return false;

}
```

Exercise 2 Non-volatile power-on-reset counter:

```
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

#include <stdint.h>

#include <stdbool.h>

#include <stdio.h>



const uint32_t myFlashConst = 0x11100000;

bool get_flash_bank_sector(uint32_t mem_address, uint32_t* bank_number, uint32_t*

sector_mask);



void main(void)

{

  MAP_WDT_A_holdTimer();





  uint32_t bank_number, sector_mask;

  uint32_t decrement = 0;

  //unlock

  printf("myFlashConst: 0x%08lX\n", myFlashConst);
```

```
    printf("myFlashConst Address: 0x%08lX\n", (unsigned long)&myFlashConst);

    if (get_flash_bank_sector((uint32_t)&myFlashConst, &bank_number, &sector_mask)) {

        FlashCtl_unprotectSector(bank_number, sector_mask);

        printf("bank_number: 0x%08X\n", bank_number);

        printf("sector_mask: 0x%08X\n\n", sector_mask);

     }




//detect count and decrement

      if(myFlashConst==0x11100000){

             printf("myFlashConst: 0x%08lX\n", myFlashConst);

      ROM_FlashCtl_programMemory(&decrement, (void*)&myFlashConst, 1);

      }

      else if(myFlashConst==0x01100000){

                 printf("myFlashConst: 0x%08lX\n", myFlashConst);

          ROM_FlashCtl_programMemory(&decrement, (void*)&myFlashConst, 2);

          }

      else if(myFlashConst==0x00100000){

                 printf("myFlashConst: 0x%08lX\n", myFlashConst);
          ROM_FlashCtl_programMemory(&decrement, (void*)&myFlashConst, 3);

          }





    MAP_GPIO_setAsOutputPin(GPIO_PORT_P2, GPIO_PIN0);

    MAP_GPIO_setAsOutputPin(GPIO_PORT_P2, GPIO_PIN1);

    MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN0);

    MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN1);


    if(myFlashConst>0){

      while (1){


      __delay_cycles(1500000); //3mhz / 2


      MAP_GPIO_toggleOutputOnPin(GPIO_PORT_P2, GPIO_PIN1);

      }


    }
```

```
  else if(myFlashConst==0){

    MAP_GPIO_toggleOutputOnPin(GPIO_PORT_P2, GPIO_PIN1);

    while(1){



    }



  }



}



bool get_flash_bank_sector(uint32_t mem_address, uint32_t* bank_number, uint32_t*
sector_mask){



  if (mem_address < 0x0001FFFF) {



    *bank_number = FLASH_MAIN_MEMORY_SPACE_BANK0;  // Bank 0

    *sector_mask = 1 << (mem_address / 0x1000);  // 4 KB sectors

    return true;
    }

  else if (mem_address >= 0x00020000 && mem_address < 0x0003FFFF) {

    *bank_number = FLASH_MAIN_MEMORY_SPACE_BANK1;  // Bank 1

    *sector_mask = 1 << ((mem_address - 0x00020000) / 0x1000);  // 4 KB sectors

    return true;

    }

    return false;



}
```

Exercise 3 SRAM function: