

**EE**

**L**

**47**

**42**

**C**

**:**

**Embedded Systems**

**Homework**

**1**

# QUESTION 1. (10 points)

*For the questions below, write the code using the masks that are pre-defined in the header file. (E.g:*

*BIT0:0000 0001, BIT1:0000 0010, …, BIT7:1000 0000).*

Perform the operations below on the 8-bit variable (uint\_8t data).

Part a) Write code that performs the three operations below. Perform each operation independently of the others.

* Set bit 2. data |= BIT2;
* Clear bit 2. data &= ~BIT2;
* Invert bit 2. data ^= BIT2;

Part b) Write code that performs the three operations below. Perform each operation independently of the others.

* Set bits 4 and 5. data |= BIT4 | BIT5;
* Clear bits 4 and 5. data &= ~(BIT4 | BIT5)
* Invert bits 4 and 5. data ^= BIT4 | BIT5;
* Set bit 4 and clear bit 5. data = (data | BIT4) & ~BIT5;

Part c) Write an if-condition line for each of the cases below. Perform each operation independently of the others.

* Check if bit 3 is 1. if (data & BIT3) {}
* Check if bit 3 is 0. if (~data & BIT3) {}
* Check if bits 3,4 are 1,1. if (data & (BIT3 | BIT4)) {}
* Check if bit 3 is 0 and bit 4 is 1. if (~data & BIT3 && data & BIT4) {}
* Check if bits 3, 4 are 0,0. if (~data & (BIT3 | BIT4)) {}

# QUESTION 2. (10 points)

A module on the microcontroller is configured using a control register called CTL that has the format shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| SLP | CLK | CAP | IE |

2- 3- 2- 1-

* SLP: selects sleep mode; value between 0 and 3
* CLK: selects clock speed; value between 0 and 7
* CAP: selects built-in capacitor value; choice between 0 and 3
* IE: interrupt enable bit (1: enable/ 0: disable)

To support programming the device, the environment has declared the symbolic constants:

SLP\_3: 1100 0000

SLP\_2: 1000 0000

SLP\_1: 0100 0000

SLP\_0: 0000 0000

CLK\_7: 0011 1000

CLK\_6: 0011 0000 …

CLK\_0: 0000 0000

CAP\_3: 0000 0110 …

CAP\_0: 0000 0000

IE: 0000 0001

*Perform all the operations below using the masks defined above.*

Part a) Write a line of code that configures the module as the following:

(Sleep mode 1)(Clock speed 5)(Capacitor value 2)(Interrupts enabled) CTL = SLP\_1 | CLK\_5 | CAP\_2 | IE;

Part b) For the operation above, show the masks used and the final value of CTL in binary.   
Mask: 0110 1101, since I’m using assignment operator, the CTL contains the same value as the mask.

Part c) Write a piece of code that changes SLP to 2. The current value of SLP is unknown.

CTL &= ~ SLP\_3;

CTL |= SLP\_2;

Part d) Write a piece of code that changes CLK to 6. The current value of CLK is unknown.

CTL &= ~CLK\_7;

CTL |= CLK\_6;

Part e) Write an if-condition line that checks if SLP=2.

if ((CTL & SLP\_3) == SLP\_2) {}

Part f) Write an if-condition line that checks if CLK=6.

if ((CTL & CLK\_7) == CLK\_6) {}

Part g) Write an if-condition that checks if the current value of CLK is either of (0, 2, 4, 6).

switch(CTL & CLK\_7) {

case CLK\_6:

case CLK\_4:

case CLK\_2:

case CLK\_0: break;

}

# QUESTION 3. (10 points)

Part a) A memory is byte addressable and has a 18-bit address. All the addresses are valid. What is the total size of the memory?

Part b) A memory is byte addressable and has a total size of 17,408 bytes (17 KB). What is the smallest address size that can be used for this memory? since we cannot have partial bits, the minimum address size is 15 bits or 32768.

# QUESTION 4. (10 points)

Part a) A computer maintains memory alignment. At what addresses can we store a byte variable? What about a 16-bit variable? For a byte it can be stored at any address, whereas, 16-bit datatype must be stored on a memory boundary that is divisible by 2.

Part b) A computer maintains memory alignment. Show how the variables below are stored in the memory if they have to be stored in the order they are declared (x, f, y, g, z), starting at address 400. Show the value at each address (including empty spots).

unsigned char x; // 8-bit variable short int f; // 16-bit variable

unsigned char y; short int g; unsigned char z;

400: x  
401: padding – unused  
402: lower byte of f  
403: upper byte of f  
404: y  
405: padding – unused  
406: lower byte of g  
407: upper byte of g  
408: z

Part c) Repeat the question above knowing that the memory should be aligned but the variables can be stored in any order.

(assuming little endian memory)  
400: lower byte of f  
401: upper byte of f  
402: lower byte of g  
403: upper byte of g  
404: x  
405: y  
406: z

# QUESTION 5. (10 points)

Part a) Explain the Big Endian and the Little Endian configurations. In big endian, the most significant byte is stored first (lower memory address) for example, the integer (16-bit) 0xAACC would be stored in RAM in the following layout:  
100: 0xAA

101: 0xCC

Little endian is opposite of this behavior.

Part b) Show how the data (0x45CD) is stored at address 400 in either of the configurations.   
Big Endian

400: 0x45

401: 0xCD

Little Endian

400: 0xCD

401: 0x45

Part c) Which configuration is used in the MSP430? Little Endian

# QUESTION 6. (10 points)

Part a) A microcontroller’s memory map allocates the FLASH code space to the address range [0x0500 to 0x0CFF]. What is the code size, in bytes, that is supported by this microcontroller? 51967 bytes

Part b) The vector table contains memory addresses (a vector is a memory address). In a certain MSP430 device, the vector table is in the range [0xFFC0 to 0xFFFF]. The memory address is 16-bit. How many vectors does this vector table support? 31 vectors