

**EE**

**L**

**47**

**42**

**C**

**:**

**Embedded Systems**

**Homework**

**2**

# QUESTION 1. (10 points)

1. Timer\_A is using a 300 KHz (300,000 Hz) clock signal. What is the timer’s period (in seconds) if the continuous mode is used? Give the answer for all values of ID (Input Divider).
2. Timer\_A is using a 300 KHz (300,000 Hz) clock signal. We’re aiming at a timer period of 0.5 seconds using the up mode. Find suitable values of TACCR0 and ID (Input Divider). Give all the possible solutions.   
   Only and are possible, are not possible because the timer register will overflow.   
   For   
   For

# QUESTION 2. (10 points)

1. Timer\_A is using ACLK configured to a 16 KHz (16,384 Hz) crystal. What is the timer period if the continuous mode is used? Give the answer for all the values of ID (Input Divider).
2. Timer\_A is using an 5 MHz (5,000,000 Hz) clock signal. Can we configure the timer to (directly) generate a delay of 0.5 seconds? Show your analysis.   
   No, if we directly configure Timer\_A for a 0.5 seconds delay it is not possible because the timer period is smaller than the delay and therefore the TAR register will overflow too early. The following is the (longest) timer period,

# QUESTION 3. (10 points)

Part a) Two clock technologies used in microcontrollers are the crystal oscillator and the RC oscillator.

For each of them, comment on:

* Startup speed (after the clock has been disabled for a while)
* Accuracy
* Stability with respect to temperature and voltage variations
* Price

|  |  |  |
| --- | --- | --- |
|  | Crystal Oscillator | RC Oscillator |
| Startup speed | Long startup time | Short startup time |
| Accuracy | More accurate | Less accurate |
| Stability | More stable | Less stable |
| Price | $ | $ |

# QUESTION 4. (10 points)

Part a) Why does embedded programming use ‘extensions’ to C? To easily expose unique functionality for the specific microcontroller.

Part b) What is the downside of extensions regarding code portability? The extensions are specific the microcontroller and to manufacture, porting to other MCU will require code modifications.

Part c) What is an intrinsic function? A compiler directive to insert an assembly instruction in C-code, such as invoking an interrupt.

Part d) In MSP430, interrupts are enabled/disabled by writing to the bit called GIE in the status register (SR: R2). Why do we use functions like below? Can’t we write the code directly in C?

\_enable\_interrupts();

\_disable\_interrupts();

We use the compiler functions because they’re safer and prone to less errors, to avoid mixing C-code with assembly, and for better code -interoperability with different MCUs.

# QUESTION 5. (10 points)

Part a) How many bits is an integer (int) in the C language? The bit-width of int is compiler dependent, the C standard states that the int data-type must have a minimum bit-width of 16 bits but it can be as big as 32-bit. For the MSP430 compiler, the int datatype is 16-bit wide.

Part b) Why does embedded programming use data type like (uint\_8t, int\_8t, uint\_16t, int\_16t, …) ? Part c) A programmer wrote a software delay loop that counts the variable (unsigned int counter) from 0 up to 45,000 to create a small delay. If the user wishes to double the delay, can they simply increase the upperbound to 90,000?

The reason to use standard integer declared types is to improve the code interoperability with other compilers and other MCU, ensure the data has the desired bit width, and for optimization reasons when using the uintX\_fast\_t/uintX\_least\_t types.

The “unsigned int” datatype is 16-bit wide in MSP430 compiler, therefore, an increase from 45,000 to 90,000 would cause an overflow causing a smaller delay value than expected. The correct approach is using the “uint32\_t” datatype defined in “stdint.h” header file; this is a 32-bit wide integer which could contain the desired delay value.

Part d) If the code contains a delay loop and we noticed that no delay is being created at run-time. What should we suspect during debugging?  
1. Ensure the delay loop has not been removed due to compiler optimization.

2. Verify the loop conditions (it may be reversed).

3. Check the initial conditions of the loop variable.

4. Check if the function containing the delay-loop is called.

# Practice Questions

Do not submit; these questions were solved in the class.

**PRACTICE 1.**

## a) *Using the continuous mode…*

Write a C code that uses Timer\_A in the continuous mode. Use ACLK based on the 12 KHz (12,000) VLO. Within the timer, adjust the clock signal so that is becomes 6,000 Hz. Each time TAR rolls back to zero, toggle the red LED, which is mapped to Port 1.0 and is active high. What is the timer’s period? **Use polling, i.e., poll the timer flag to find out when the duration has elapsed.**

Before writing the code, fill the configuration below by looking at the help sheet.

## Configuration

TASSEL =

ID =

Mode = TACLR =

### b) Using the up mode…

Write a C code that uses Timer\_A in the up mode. Use ACLK based on the 12 KHz VLO. Within the timer, adjust the clock signal so that is becomes 3,000 Hz. Our goal is to generate a delay of 0.5 seconds. How many cycles does this correspond to? When the duration of the timer elapses, toggle the red LED, which is mapped to Port 1.0 and is active high. **Use polling, i.e., poll the timer flag to find out when the duration has elapsed.**

Before writing the code, fill the configuration below by looking at the help sheet.

## Configuration

TASSEL =

ID =

Mode =

TACLR =

TACCR0 =

If you run this code on the basic LaunchPad (G2553), use these lines to divert ACLK to the 12 KHz VLO.

// Code that sets ACLK to VLO @ 12 KHz

// Write this at the top of the main

BCSCTL1 &= ~XTS; // Set XTS=0

BCSCTL3 &= ~LFXT1S\_3; // Clear LFXT1S

BCSCTL3 |= LFXT1S\_2; // Set LFXT1S=2 (VLO)

**PRACTICE 2.**

Write a C code that uses Timer\_A based on SMCLK (set to ~1 MHz by default). Our goal is to generate a 4-second delay. The problem is that this clock frequency is very high and a 4-second delay is a huge number of cycles that can’t be measured with a 16-bit register. First, divide the clock by 8 to slow it down. Then, use the ‘up mode’ to generate a delay of 0.4 seconds. Then, write a code that counts ten such delays to get a 4-second delay. Upon the 4-second interval, flash the red LED, which is mapped to P1.0 (active high). **Use polling, i.e., poll the timer flag to find out when the duration has elapsed.**

Start by writing the timer’s configuration.

## Configuration

TASSEL =

ID =

Mode =

TACLR =

TACCR0 =