UNIVERSITY OF LIMERICK

**OLLSCOIL LUIMNIGH**

**FACULTY OF SCIENCE AND ENGINEERING**

**DEPARTMENT OF ELECTRONIC & COMPUTER**

**ENGINEERING**

**MODULE CODE: ED5502 (Specialist Diploma in Embedded Systems)**

**MODULE TITLE: Digital Systems 4 (Sample Exam)**

**SEMESTER: Spring 2019**

**DURATION OF EXAM: 2 Hours**

**LECTURER: Dr Ciaran MacNamee**

**INSTRUCTIONS TO CANDIDATES:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* **This exam is worth 30% of the Assessment for the module**
* **It consists of two parts**
* **Part 1 is worth 20% of the module Assessment**
* **Part 2 is worth 10% of the module Assessment**
* **Answer four questions (of six) from Part 1**
* **Answer one question (of two) from Part 2**

**Part 1**

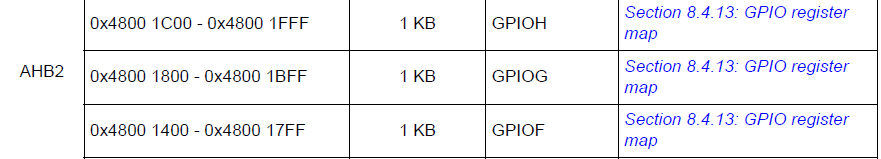
Marks

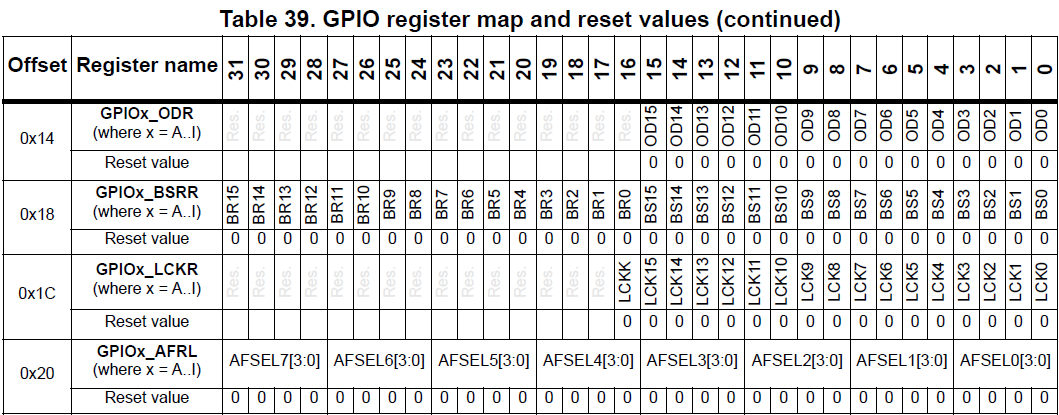
**Q1**

(5)

Use the information given below to find the absolute physical address of the registers GPIOG\_ODR and GPIOF\_BSRR







**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

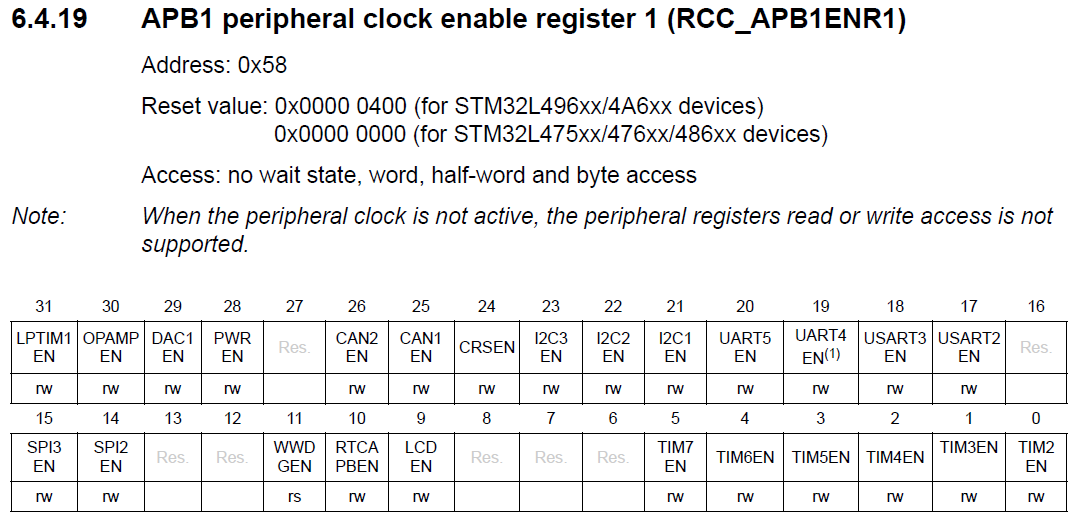
(5)

**Q2**

(5)

Marks

Write a line of C code to enable the Clock used for UART5, LCD and also for TIM7. The register format for the relevant enable register is given below:



(5)

**Q3**

Timer/Counter TIM2 is to be used in its Input Capture mode. Given the following information:

TIM2 is a 32-bit counter

TIM2 clock source is 100MHz

TIM2 Prescale Register has been set to 24 (actual prescale value = 25)

TIM2 Auto Reload Register = 4294967295 (its maximum possible value)

1. What is the shortest time period that can be measured using TIM2’s input capture registers if they have been initialised in this way?
2. What is longest time period that can be measured using TIM2’s input capture registers, if they have been initialised in this way, without counting timer overflows?

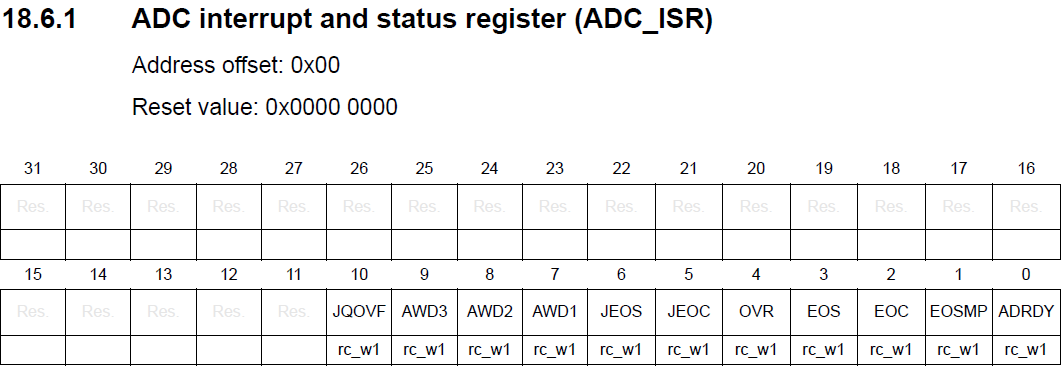
**Q4**

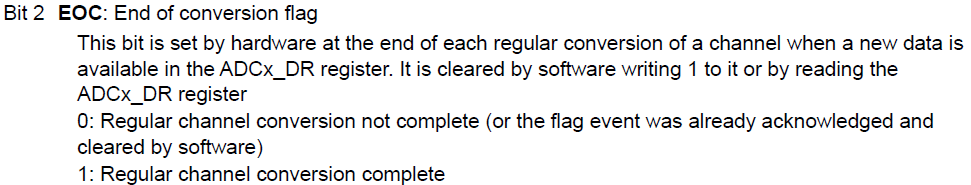
Refer to the ADC Interrupt and Status Register and the ADC Data Register to write a fragment of C code using ADC2 that:

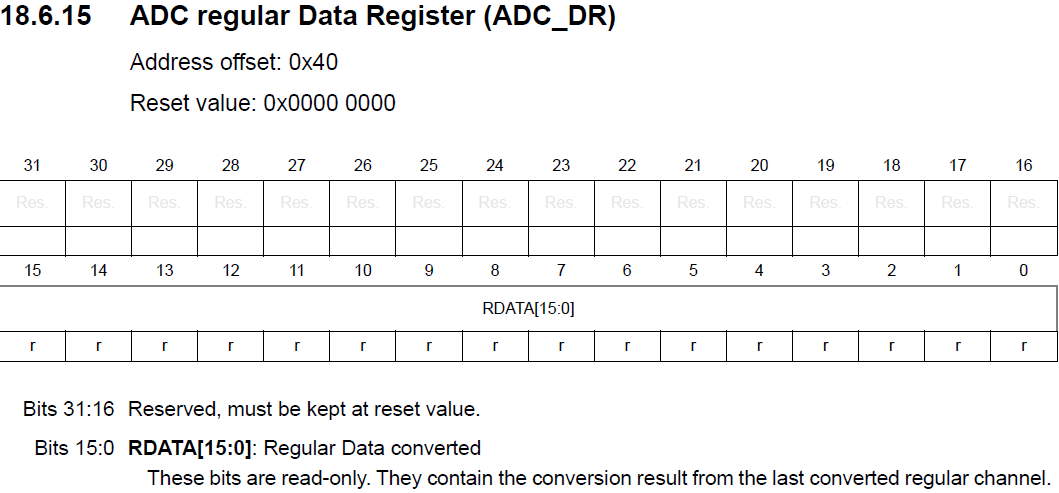
Polls the End of Conversion Flag bit (waits until it indicates that a conversion has been completed)

then reads the conversion result into a uint32\_t variable called ‘adcResult’

then uses this variable to calculate the input voltage at the ADC2 input, given that the ADC2 has been set for 12-bit conversion data and uses a reference voltage of 3.0V. The voltage is stored in a float variable called ‘voltageInput’.

****

****

****

**Q5**

(5)

Marks

Assume that all bits of port GPIOA have been set as outputs, and that all bits of GPIOB have been set as inputs.

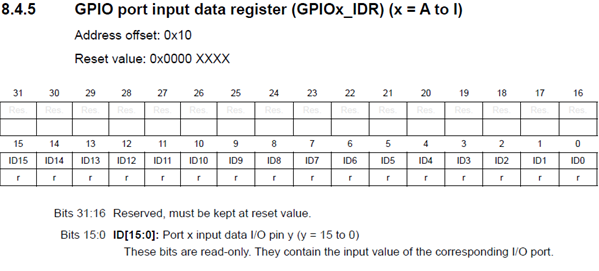
Write a fragment of C code to test bit 11 of GPIOB

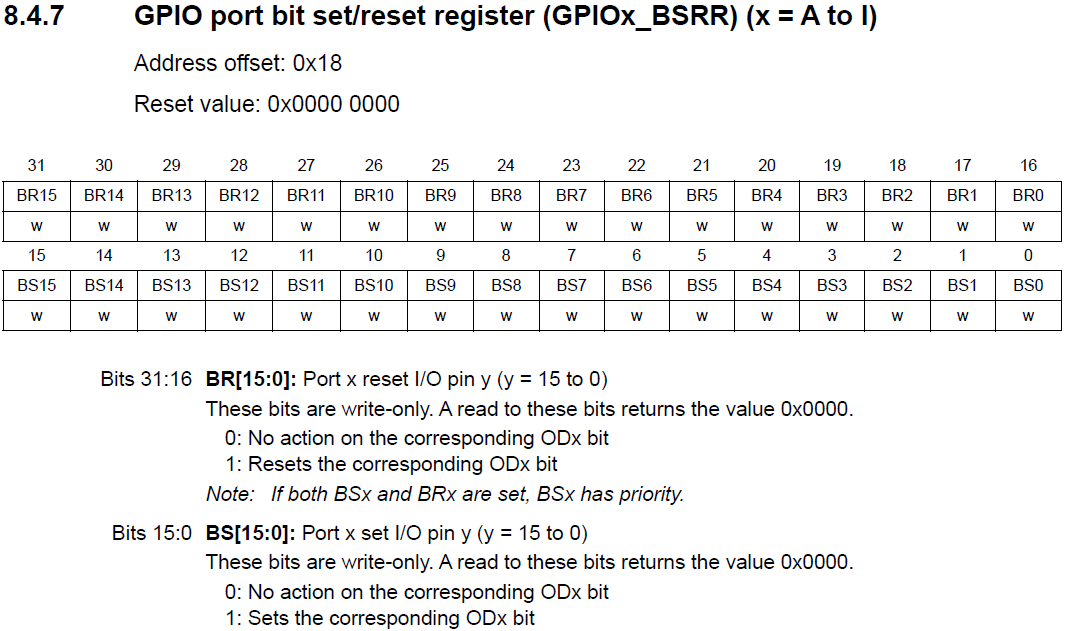
If GPIOB bit 10 is high, set bits 13, 7 and 2 of GPIOA to 1 leaving all other bits unchanged.

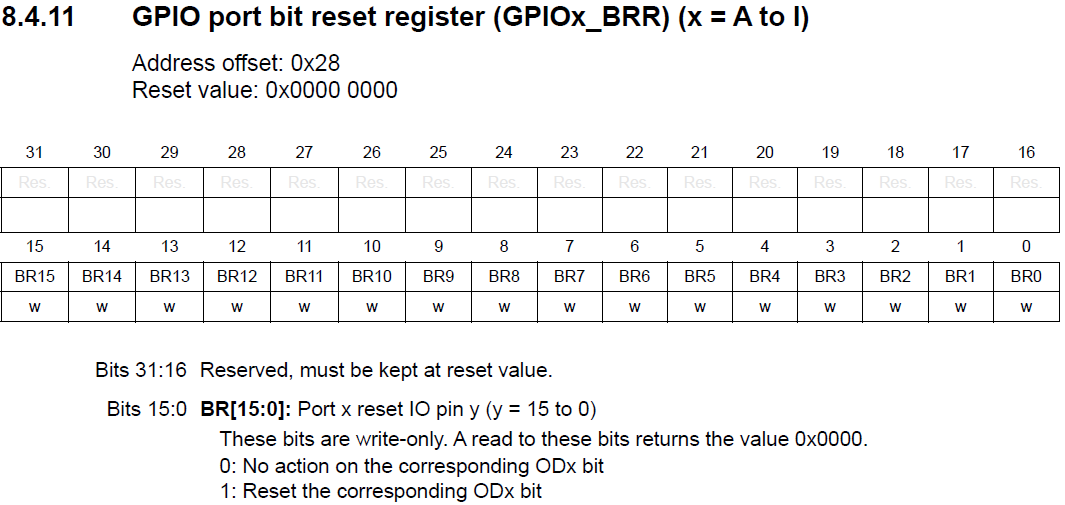
If GPIOB bit 10 is low, clear bits 10 and 5 of GPIOA to 0 leaving all other bits unchanged.

What is meant by atomic setting of these register bits?

In this question you must use the BSRR or BRR registers for atomic access to set or clear these port bits.







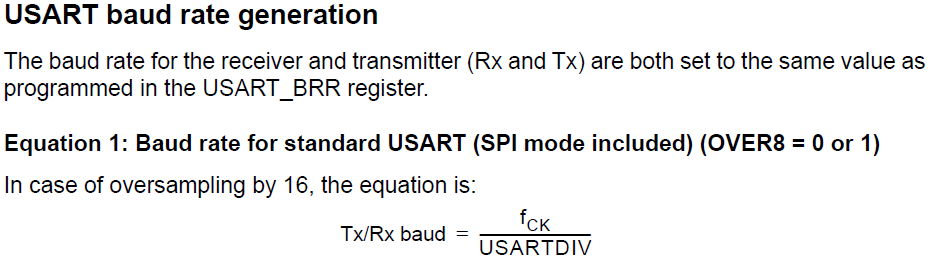
**Q6**

(5)

Marks

1. Referring to the Figure below, calculate the value of the USARTDIV field to set a Baud Rate of 115200 Baud, assuming a clock frequency of 16 MHz and that 16 times oversampling has been enabled.
2. Calculate the % error in the generated Baud Rate ie

((Calculated Baud Rate – Ideal Baud Rate)/Ideal Baud Rate) \* 100



(Note: there is no need to consider the fractional part of USARTDIV here).

**Part 2**

(10)

Marks

**Q1**

Write a program to run on an STM32L476RG microcontroller that completes an initialisation sequence then carries out the following sequence of operations in an infinite loop:

The program polls the state of the TIM3 Timer Status register, for an edge detected on TIM3 Channel 1. You may assume that the 16-bit TIM3 Timer has been initialised for Input Capture operation using Channel 1, based on the rising edge.

The program waits until the first edge has been detected storing the captured counter contents from CCR1 register.

The program then waits for a second input capture edge detection and uses the newly captured CCR1 to calculate the number of clock cycles that have elapsed between rising edges. The program should calculate the time period between successive edges based on the fact that the TIM3 clock input frequency is 50 MHz, with a prescale setting of 25. The calculated time period should be stored in a variable called ‘time\_period’.

The program calculates the input frequency corresponding to time\_period in kHz, storing the result in a float variable called ‘inputFrequency’ and transmits the result to a user on USART2, formatted as “Input Frequency = %6.4f”.

**Assumptions:**

You may assume that the sprintf function is available to allow formatting of your data.

int sprintf ( char \* str, const char \* format, ... );

You may assume that all initialisations have been completed in a function called InitialiseAll(); You should place this function at the correct position in your program.

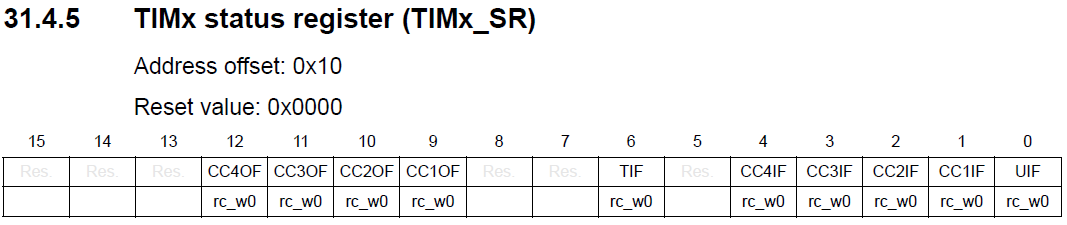
void InitialiseAll (void);

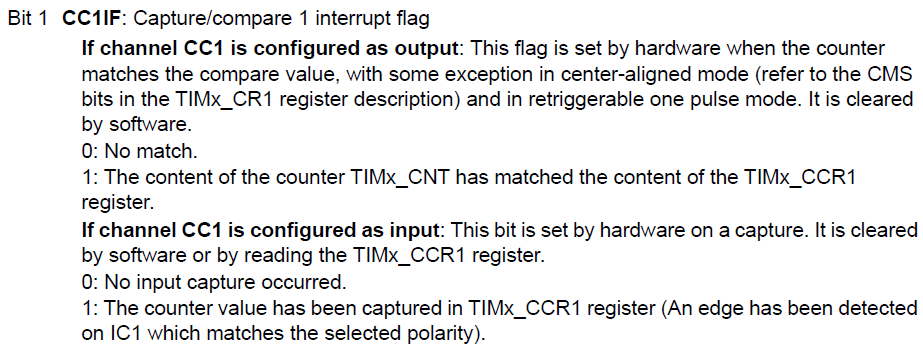
You may assume the existence of a function called USART2\_Write(char \*message); which sends a string to USART2. You don’t need to write this, just call it.

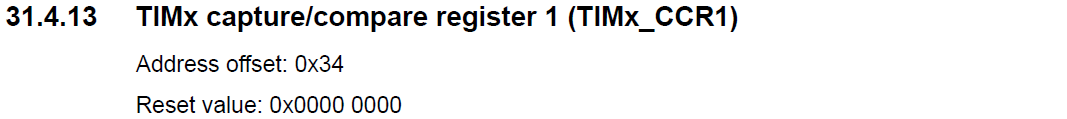
void USART2\_Write (char \*message);

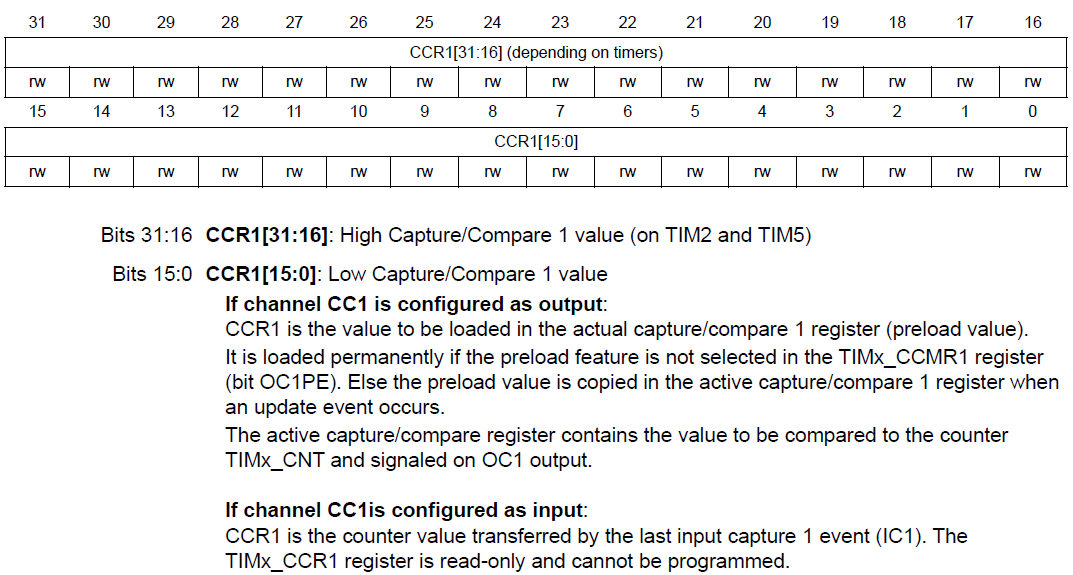
The TIM3 has been initialised for Direct Input Capture on Channel 1, with an effective prescale value of 25. The clock input to the precaler unit is 50MHz.

**Relevant Registers:**









(You do not need to initialise or enable the TIM3, assume that this was done in the initialisation function).

**Q2**

(10)

Marks

Write a program to run on an STM32L476RG microcontroller that completes an initialisation sequence then carries out the following sequence of operations in an infinite loop:

The program polls the state of the TIM3 Timer Status register, for an edge detected on TIM3 Channel 1. You may assume that the 16-bit TIM3 Timer has been initialised for Input Capture operation using Channel 1, based on the rising edge.

The program waits until the first edge has been detected storing the captured counter contents from CCR1 register.

The program then waits for a second input capture edge detection and uses the newly captured CCR1 to calculate the number of clock cycles that have elapsed between rising edges. The program should calculate the time period between successive edges based on the fact that the TIM3 clock input frequency is 50 MHz, with a prescale setting of 25. The calculated time period should be stored in a variable called ‘time\_period’.

The program calculates the input frequency corresponding to time\_period in kHz, storing the result in a float variable called ‘inputFrequency’ and transmits the result to a user on USART2, formatted as “Input Frequency = %6.4f”.

**Assumptions:**

You may assume that the sprintf function is available to allow formatting of your data.

int sprintf ( char \* str, const char \* format, ... );

You may assume that all initialisations have been completed in a function called InitialiseAll(); You should place this function at the correct position in your program.

void InitialiseAll (void);

You may assume the existence of a function called USART2\_Write(char \*message); which sends a string to USART2. You don’t need to write this, just call it.

void USART2\_Write (char \*message);

The TIM3 has been initialised for Direct Input Capture on Channel 1, with an effective prescale value of 25. The clock input to the precaler unit is 50MHz.

TIM\_HandleTypeDef htim3; has been declared so the you can identify TIM3 as htim3

**Relevant HAL Macros and Libraries:**

\_\_HAL\_TIM\_GET\_FLAG(HANDLE, FLAG)

This macro returns the state (SET or RESET) of the specified bit of the Status Register of the Timer HANDLE. Can be used to get the state of htim3’s CCR1 Flag

uint32\_t HAL\_TIM\_ReadCapturedValue(TIM\_HandleTypeDef \*htim, uint32\_t Channel);

This function returns the converted Input Capture Register Captured Value, for a given timer and Channel Number.