EV Embedded Control Systems Task

Name: Adham Mohamed Elhelly

E-mail: adhamacc1@gmail.com

Question 1

Github Link: https://github.com/adhammo/PWM

Question 2

a. When to use DMA (direct memory access)?

We use DMA when the CPU cannot keep up with rate of data transfer or when the CPU needs to preform work while waiting for a relatively slow I/O data transfer. [Wikipedia]

Basically, DMA lets peripherals access memory directly bypassing the CPU. Which frees the CPU to do other stuff while data is transferred from or to the memory from a certain peripheral.

b. The ADC data is wrong. Why?

Because the signal has high output impedance (as not to change much if output impedance changed by a relatively small amount), and because RA2 is not relatively small with respect to signal output impedance, it drops some of the signal voltage and hence affecting the data of the ADC.

c. “volatile”, “static” and “extern”?

volatile:

- It is used with variables and it tells the compiler not to optimize that variable as it will be changed (wrote to) inside an Interrupt Service Routine and its data can be changed intermediate a CPU operation involving its value.
- It is used when we want to **write** to a variable from **inside** an Interrupt Service Routine.

Useful when we want to set a flag whenever an interrupt has occurred so we can handle that flag in the main loop after that we clear the flag.

static:

It is used when each object source file (.c) (object) must have its own value of a static variable and/or its own implementation of a static function.

All the source files have **different values** (independent variables) of a static variable.

Each source file must have the **declaration and implementation** of a static function.

Useful when we do not want to share a function implementation as to encapsulate the function inside a specific source file (it can use only static resources inside that source file) and we may add an extern function to access that static function from outside that source file.

extern:

It is used when more than one source file shares a certain variable and/or share a certain function.

All the source files share the **same value** (dependent variables) of an extern variable.

Each source file must have **only the declaration** of the extern function and **at least one** of the source files must have an implementation of the function.

Useful when we want to make initialization or global functions (extern by default) as it only has one implementation and can be accessed from any source file.

d. UART

Baud Rate: 2bits / 2ms = 1000 bits/sec

Data: 0b01001101 = 0x4D

e. What do you know about?

Vector table:

A table which links between Interrupt Requests and Interrupt Service Routines so when the CPU receives an Interrupt Request it fetches for the ISR associated with that Interrupt Request.

Start-up code:

It is a piece of code which runs the first thing after booting as to setup some components that are essential for the execution of the main program (i.e. setting up the interrupt vector table, setting up the stack, setting up external memory, etc.).

Bootloader:

It is a software which connects the hardware (controlled by the firmware) of a microcontroller and the kernel (on top of which runs the main program).

When the microcontroller initially starts up, it loads the firmware which looks for and loads a bootloader from the flash memory, then the bootloader initializes (loads to memory, decompress, etc.) the kernel and gives it control over the CPU.

Hardware abstraction layer

It is a layer of programming which allows the kernel (operating system) to communicate with the hardware abstracting all the complicated details required for that communication.

Bare metal programming

It is act of writing a firmware which controls the hardware directly without the need of a kernel (operating system) and thus no need for a hardware abstraction layer, generally it is used with low-end microcontrollers which don't have the specifications to run an operating system.

Polling vs interrupt

Polling: The act of retrieving data synchronously whenever needed irrespective of its change. (i.e. sampling every one second, etc.).

Interrupt: The act of retrieving data asynchronously when an interrupt request is initiated. (i.e. a timer overflow, value of pin has changed in an input interrupt, etc.).

Prescaler

A value used to divide the clock of the microcontroller as to slow down a timer counting process, mostly used in timing when the timer cannot count for a certain duration without overflowing.

ADC conversion time

It is the time taken (measured by clock cycles) by an ADC to sample an analog value and convert it to a digital value, usually it refers to serial ADCs as they use successive approximation.

Sampling frequency

It is the number of voltage samples taken from a signal in one second as to approximate the signal (convert it from continuous time to discrete values).

Nyquist theorem states that the minimum sampling frequency used to approximate a signal is double the frequency of that signal.

Code refactoring

It is the process of changing some parts of a code without affecting its output and functionality (i.e. formatting, renaming, optimizing, etc.) for computability reasons.

Unit testing

It is a software test for each unit in the program separately as to ensure that every unit is working as it should with minimum interfering with other units.

C programming build process

It is the process of converting source code written in C (High level) to machine language instructions specific to the targeted CPU architecture, first is the preprocessing stage, then compiling and finally linking.

Waterfall and scrum frameworks

It is a way (steps) to organize work on a certain project which enables you to do twice the work with half the effort (work more efficiently).

It starts by defining the requirements after that comes design where you lay the basic foundations of the software then implementation where you write the code to preform certain functions and achieve requirements then you test and verify your work and finally maintain your final product.

Software Development Life Cycle

It is a process used to develop software and it consists of the following repeating steps:

1. Planning
2. Analysis
3. Design
4. Implementation
5. Testing and Integration
6. Maintenance