Name: ________________________________

Student ID: __________________________

You can use a single 8.5x11” “cheat sheet” (one side).
The sheet must be a standard thickness with no staples, tape, etc.

For the multiple choice questions, please fill in the circle for the correct answer. For the free-response questions, please write clearly and ensure that your answer fits in the box provided. Good luck!
1. (2 points) A microcontroller’s clock tree is configured to provide a 16 MHz clock to your timer peripheral. The timer peripheral is configured with a prescaler (divider) of 8, and its operating mode is to interrupt when the COUNT is equal to the OVF value. What should you set the OVF to so the interrupt fires 10 times a second?
   - 10
   - 10,000
   - 20,000
   - 100,000
   - 200,000

2. (2 points) Implementing a software driver to configure a hardware peripheral in a microcontroller primarily involves:
   - Executing peripheral-specific input and output instructions
   - Reading and writing specific memory addresses
   - Reading and writing CPU registers (e.g., the stack pointer)
   - None of the above

3. (2 points) A General Purpose Input and Output (GPIO) peripheral provides the following functionality in a microcontroller:
   - An input for receiving interrupts from external hardware
   - An output to control Light Emitting Diodes (LEDs)
   - An output for debugging if an interrupt handler is being executed
   - All of the above

4. (2 points) Polling is more efficient (requires fewer CPU cycles) than interrupts for monitoring the state of a peripheral when:
   - The peripheral’s state only changes once
   - The peripheral’s state changes frequently (within a few clock cycles)
   - The peripheral’s state is read only
   - All of the above

5. (2 points) The \( \text{I}^2\text{C} \) bus operates slower (typically 400 kbps or less) than other serial busses (e.g., UART and SPI) because:
   - It acknowledges every message (byte) sent on the bus
   - It addresses every message (byte) sent on the bus
   - It uses pullup resistors on the clock and data lines and the devices operate as open collectors (attached to GND or disconnected).
   - All of the above

6. (2 points) Accelerometers can determine the orientation of an embedded systems because they measure:
   - Air pressure
   - Gravity
   - The earth’s magnetic field
   - The light coming from the sun
   - None of the above
7. (2 points) How much time has passed when a 12 MHz oscillator has cycled 24,000 times?
   - 50 microseconds
   - 2 milliseconds
   - 1 millisecond
   - 1 second

8. (2 points) A serial bus transfers more than one bit per clock cycle.
   - True
   - False

9. (2 points) The header files included with an embedded system such as ARM’s Cortex Microcontroller Software Interface Standard (CMSIS) make software development easier because:
   - They define C structures and enumerations that capture the register layout of each peripheral.
   - They define portable macros that abstract away the memory map of the specific processor.
   - They include convenience functions to generate hardware configuration parameters.
   - All of the above

10. (2 points) There is a computationally expensive task that you want to run at most once a second, but it is ok if you miss running the task every once in a while, as long as the developer knows how often the task is missed. The task is implemented outside of the interrupt handler, and the logic to trigger it is implemented inside the interrupt handler. Where do you implement the logic to count how many executions of the task have been missed?
   - Inside the interrupt handler
   - Outside of the interrupt handler

11. (8 points) You want to count the number of milliseconds that have passed on your microcontroller that is running off a 10 MHz clock. As a well-trained embedded systems engineer, you decide that you should use a timer peripheral because it is an energy efficient way to track time. You properly configure your timer TC1 so it fires an interrupt once every millisecond, and you configure the interrupt controller to enable interrupts from TC1. You implemented the interrupt handler in the following way:

```c
uint32_t msecs_passed;

void TC1_Handler(void) {
    msecs_passed++;
}
```

Have you implemented the interrupt handler properly? If not, what is incorrect? What will the value of `msecs_passed` be after the timer has run for 7 seconds? Explain how you arrived at your answer.
12. (4 points) An engineer wants to write a program that toggles an LED 60 times a second because they want to test the timing of a 120 fps video camera. They use a timer overflow interrupt to control the GPIO that is connected to the LED. Unfortunately, after flashing the program to the MCU, the LED stays dark and is not blinking. Describe four bugs (software and/or hardware) that could be preventing the LED from blinking (at least one sentence each). Then indicate the priority you would assign to looking at each of these bugs in the debugging process and explain why you chose that ordering.

13. (4 points) Another engineer is assigned the same task. They also decide to use a timer interrupt to control the LED. After downloading the program, unfortunately, the LED is not blinking either. However, the LED is bright (different from the previous case). Describe two bugs that could be preventing the LED from blinking (at least one sentence each). Then indicate the priority you would assign to each bug in the debugging process and explain why you chose that ordering.
14. (4 points) You are tasked with building a new feature for a smart doorbell: delivery detection. This feature detects if a package is being delivered to your home by looking for a brown colored box in the video frame. The microcontroller in the doorbell is connected to a camera. The camera captures an image once every 10 seconds and immediately transfers the image (matrix of pixels) to the microcontroller for processing.

We don’t want to drop any images because we are capturing them only once every 10 seconds and we don’t want to miss the delivery person. Unfortunately, the microcontroller in the doorbell was not selected for having many unused CPU cycles, and it’s onboard serial hardware is slow: transferring the image takes longer than it takes to capture it. When an image is done transferring to the CPU, the next image starts transferring immediately after. Is it possible to implement delivery detection on this embedded system without dropping any images? If not, explain why. If so, provide a high level overview of how you would use the functionality in the embedded system (e.g., interrupts) to implement this.