SIT329 Advanced Embedded Systems

Task 4.0P: Configuring the hardware timer on the ARM Cortex

Introduction

In event driven embedded programming, timing is a critical factor. For those that have used Arduino, we would be familiar with the use of the *delay(ms)* function which takes arguments as number of milliseconds. The *delay()* function directly accesses a hardware timer through the timer setup and timer control registers. What if we need a different timing resolution, for example 10.5 milliseconds? Can this be achieved? In this task, we look at how timers on the ARM Cortex are configured to achieve the required functions.

Task:

In this task, you will write your own C-functions to directly access, configure and control the timer registers of the SAMD21G18A MCU. You will need to refer to the SAMD21G18A MCU datasheet. The task is as follows:

1. An application requires a timer to keep track of events that occur within a range of 0-60 second intervals with a timing resolution of 0.01s. We select to use timer TCx on the SAMD21G18A MCU. Design the timer for this application by answering the following:
   a) What types of timers are available on your MCU chipset and how are they referred to? e.g. TC0 means Timer/Counter 0.
   b) Select an appropriate TCx for use. Determine the appropriate GCLK input clock frequency for TCx and the TCx timer width to use.
   c) Select the clock generator and prescalar values for the corresponding registers and calculate the values to load into the necessary registers. Show your calculations.
   d) If only the 8MHz clock source is available for use, what is the smallest error achievable in the timer tick time with your selection in b)?

2. Write a *setup_TCx()* function to configure the timer with the parameters in 1) using the CMSIS API methods. For example, if you selected TC0 to use, then your function should be *setup_TC0()*.

3. Write a *wait_TCx(T)* function that returns only after T ms of the timer TCx has elapsed. The wait function works similarly as the delay function, except that the input argument T represents time of 0.01s not 1 millisecond. For example, if you selected TC0 to use, then your function should be *wait_TC0(T)* .

4. **Design Task**: An embedded application for measuring temperature requires the following:

   a) A DHT22 sensor to measure temperature every 30 seconds.
   b) LED1 turns on and stays on if temperature is greater than 20 °C.

c) LED2 blinks with a frequency of 2Hz if the temperature is greater than 25 °C.

Develop the embedded firmware using event driven programming techniques. CMSIS commands are required for the MCU, while DHT22 can be interfaced using the Arduino library for this task.

Submission:

1. A pdf file that shows clearly your answers to sections 1-4 of the task above. You can section this as a report format.
2. For section 1, explain how you arrived at the parameters selected to setup the timer. For example, you may have found that certain timers could not be used due to them being allocated on the Arduino Nano 33 IoT.
3. For section 2-3, include the full code listing including comments.
4. For section 4, design task include the event table, software flowchart and full code listing including comments. A link to a short video showing the following process – show your code, compile your code and demonstrate that the embedded system works to the specification.
5. Include any references used in completing this task.

Submit your work:

When you are ready, login to OnTrack and submit your pdf which consolidates all the items mentio ned in the submission detail section above. Remember to save and backup your work.

Complete your work:

After your submission, your OnTrack reviewer (tutor) will review your submission and give you feedback in about 5 business days. Your reviewer may further ask you some questions on the weekly topics and/or about your submissions. You are required to address your OnTrack reviewer's questions as a form of task discussions. Please frequently login to OnTrack for the task Discuss/Demonstrate or Resubmit equivalent to fix your work (if needed) based on the feedback to get your task signed as Complete.