CSSE3010 Stage 2

Alive Timer and Joystick Input Control

(Closed Shoes MUST BE Worn in the labs)

---

1 Assessment

- Git due 4pm Monday in week 5 .
- There is no demo. Do not attend a lab session. All lab sessions are cancelled
- No worksheet or workbook is due.
- Course Marks: 10%
- Electronic Course Profile Pass Hurdle: Must be submitted.

2 Resources

- Nucleo-F429ZI platform
- LED Light Bar
- 74HC08 AND Gate
- Logic Analyser (LA)
- Xenon Monitor

3 Structure

Your final stage code must be titled `main.c` and saved in your `stage2` folder.
PATH: `csse3010/stage/stage2`

Listed below are the required mylib libraries, that you will need to develop for this stage. They should be saved in your `mylib/mylib_library_folder` folder.
PATH: `csse3010/mylib/mylib_library_folder`.

- Alive Timer HAL Library
- Joystick HAL Library
- PWM HAL Library
- ISS HAL Library
- ITA1000G HAL Library

4 Introduction

This stage will introduce the Timer Hardware Abstract Layer (HAL) libraries of the Nucleo platform. The Nucleo platform provides various timer functions such as overflow interrupts, input capture and Pulse Width Modulation generation. You will be using the LED Bar and Joystick.

5 PART A Tasks - Alive Timer Control

5.1 Workbook Tasks - Not Assessed

The workbook tasks are optional and are not assessed. It is common and good practice to keep a workbook, with a schematic and firmware notes, for future reference.

5.2 Worksheet Tasks - Not Assessed

The worksheet contains helpful questions related to this stage. Worksheet 3 is not assessed and is not compulsory for you to attempt. The worksheet task is designed to help you complete stage 2 .

5.3 Design Tasks - Do Not Attend Labs

The code required for the design tasks must be uploaded to your git repository by the specified due date in week 5 . The lab sessions are cancelled. Alternate online sessions will be organised on BlackBoard.

5.3.1 Git Requirements

You MUST have your stage and mylib code in git, in the correct folders and it must be able to be compiled by the markers, in order to be assessed for this stage. If your code only compiles on your computer and cannot be compiled by the markers, then it is not acceptable for marking. Please ensure that you do the following:

- Folder names - Your stage and mylib code MUST be in the correct folder, as specified. Folder name must be alphanumeric characters and contain NO spaces e.g. stage1 NOT stage 1 or stage-1.
- Header file includes - MUST NOT have a file path. e.g. `include "sxxxxxxx_hal_joystick.h"` is correct BUT NOT:
  `include "/home/csse3010/mylib/joystick/sxxxxxxx_hal_joystick.h"`. To include a file path for header files, modify the CFLAGS in filelist.mk.
- filelist.mk - your code MUST compile with a filelist.mk file. Do not rename filelist.mk.
- Code not compiled or not used - do not push to git, any files containing code that is not used or that will not compile. If you want to push code that is unfinished, then use compiler directive `#IFNDEF _DONOTBUILD`, to not compile unused code.

- Gitlab - Check your code is correctly in your git repository by logging into gitlab: `https://csse3010.uqcloud.net/csse3010/`. Check the stages and mylib folders. If your code is not in the correct place, then it will not be compiled correctly.
- .gitignore - **you MUST HAVE A .gitignore file** in the stages, mylib and project folders. Do not put a .gitignore file into a subfolder e.g. stage1 - See `https://csse3010.uqcloud.net/csse3010/docs/setup/git/git/#task_5_creating_a_gitignore_file`
- gitg - use gitg (type gitg in terminal) to check which files are staged and can be pushed to git.

For more git details, see the user guide - `https://csse3010.uqcloud.net/csse3010/docs/setup/git/git/`

5.3.2 mylib Setup

You MUST FOLLOW the Template Code given in the `sourcelib/examples/templates/mylib` folder. Your mylib code must meet the guidelines specified in the mylib and platform build guides, on Blackboard. You MUST create the right file structure in the `mylib` git folder.

You will create mylib HAL library files for controlling the alive timer (atimer). Refer to the alive timer HAL specifications, on Blackboard.

5.3.3 Design Task 1A: Alive Timer Control using Console

The atimer should automatically start with a default clock speed of 25kHz and a period of 2ms. The atimer must use HAL Timer 3. The atimer pin used should be the D10 pin. Control the atimer using the serial console. You must implement the commands in Table 5.3.4. Each command is presented by a character, received over the serial USB connection.

Command Char	Description
'f'	Pause or freeze the atimer
'r'	Resume the atimer
'z'	Restart or zero the atimer
'c'	Print the current value of the atimer (in ticks)
't'	Print the current value of the atimer (in ms)

Refer to `getting_started/console` and `timer/timer_interrupt` examples.

5.3.4 Design Task 2A: Alive Timer Clock Speed and Period Control

Control the clock speed and period of the atimer. The output pin of the atimer must clearly toggle at a frequency of $0.5 * (1/new\_atimer\_period)$. The output pin must be viewed, using the LA. The number of clock cycles per atimer output pin toggle should be recalculated, whenever the atimer's clock speed is changed, so that the atimer output pin toggle frequency is constant.

Command Char	Description
'+'	Increase the output pin period by 10ms
'-'	Decrease the output pin period by 10ms
'i'	Increase the HAL clock speed by 1kHz
'd'	Decrease the HAL clock speed by 1kHz

6 PART B Tasks - Joystick Input Control

6.1 PWM (Pulse Width Modulation)

The basic idea behind PWM control is that, in order to transfer the same amount of power as a continuous analog signal, we can switch our binary output at a specified frequency between VCC and GND and the proportion of time spent at VCC will correspond to the amount of power we send.

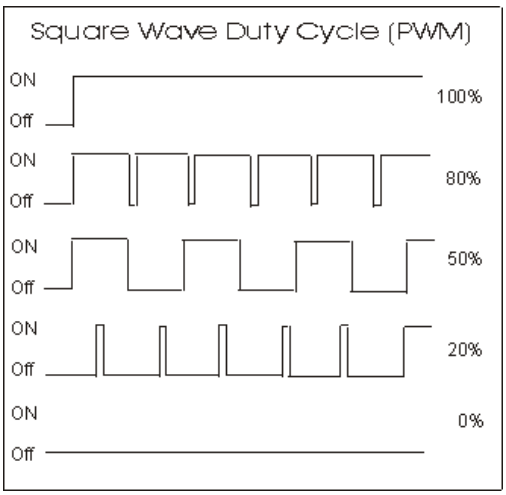

Figure 1: PWM duty cycles

The **duty cycle** of PWM signal is defined as the ratio of HIGH pulse duration to the period of the signal. PWM is used so often in modern digital devices that almost all micro-controllers have built-in hardware that will produce a PWM signal.

6.2 Worksheet Tasks - Not Assessed

The worksheet contains helpful questions related to this stage. Worksheet 3 is not assessed and is not compulsory for you to attempt. The worksheet task is designed to help you complete stage 2 .

6.3 Design Tasks - Do Not Attend Labs

The code required for the design tasks must be uploaded to your git repository by the specified due date in week 5 . The lab sessions are cancelled. Alternate online sessions will be organised on BlackBoard.

6.3.1 Git Requirements

You MUST have your stage and mylib code in git, in the correct folders and it must be able to be compiled by the markers, in order to be assessed for this stage. If your code only compiles on your computer and cannot be compiled by the markers, then it is not acceptable for marking. Please ensure that you follow the previously given git requirements.

6.3.2 mylib Setup

You MUST FOLLOW the Template Code given in the `sourcelib/examples/templates/mylib` folder. Your mylib code must meet the guidelines specified in the mylib and platform build guides, on Blackboard. You MUST create the right file structure in the `mylib` git folder.

You will create mylib HAL library files for interfacing the the Joystick. Refer to the Joystick HAL specifications, on Blackboard.

6.3.3 Design Task 1B: Joystick Input Control

The joystick is a device that consists of two rotary potentiometers and a switch. The potentiometers are used to measure the joystick's movement in the X (left/right) and Y (forward/backward) directions. The switch is used to detect if the joystick is pressed.
Implement the Joystick mylib HAL driver functions. When the joystick is in the up right position, the rotary potentiometers will be at their mid point positions. Implement the Joystick mylib HAL driver functions. The Joystick Z signal should be synchronised by being connected to the ISS HAL mylib driver as source 3.

Connect the Joystick using a 5 pin Female to Female to the Joystick Connector and connect to the Nucleo pins using a 3 pin Female to Male wire. Table 1 shows the Nucleo pin connections for the Joystick.

Table 1: Nucleo Board Pin Connections

Nucleo Pin	**Connection**
A1	VRx (X)
A2	VRy (Y)
A3	SW (Z)
3V3	+5V
GND	GND

Refer to `adc` example and the CSSE3010 Docs Wiring and Hardware Guide, on BlackBoard.

6.3.4 Design Task 2B: LED Bar Meter Display and Brightness Control

Implement a LED Bar Meter that is controlled by the joystick signal. The LED Bar segments should light up, according to the position of the joystick Y signal. Full deflection should

light up all LED segments, upright (mid-point) should only light up half of the LED segments.

Use the joystick X signal to control the brightness of the lower 4 LED segments. Generate a single PWM signal (500ms period), according to the joystick X signal and use it to control the brightness of the lower 4 LED segments (Hint: 74HC08). Implement the PWM HAL mylib functions. Connect the LED brightness control signal to Nucleo D6. You must choose a suitable timer and timer clock frequency.

Use the joystick Z signal to clear the LED Bar Meter by turning of all LED segments. The ISS mylib HAL event count function must be used, to determine when the LED Bar Meter must be cleared.

Refer to the `pwm` examples.

6.3.5 Challenge Task: Metronome - Optional and Not Assessed

When the character 'm' is received over the serial USB connection, switch your system to metronome mode. In metronome, the LED segments should light up and mimic the movement of a metronome `https://www.youtube.com/watch?v=gsJEMH_emBM`.

Use the Joystick X signal to control the speed of the metronome and the joystick Y signal should control the brightness of the LED segments in 4 bands, as shown in Table 2.

Table 2: Metronome Bands

Band	**LED Segments**
0	2,1,0
1	4,3
2	6,5
3	9,8,7

Pressing the Joystick Z, should select the band to be controlled (assume starting from 0). The band selection, should wrap back to 0 from band 3. The brightness of the Band does not have to be maintained, when the Joystick Z is pressed. Use the same PWM parameters, from previous task. Hint: You may need to use multiple PWM signal and expand your PWM HAL mylib.

7 Criterion

The stage demonstrations are marked according to the criterion outlined in the table below. If you fail to demonstrate sufficient understanding and functionality in the specified marking time you will not be allowed to repeat the stage. You must pass the pre-demo checks before you are allowed to demo. **All code assessed for the stage must be your own work.**

7.0.1 Pre Demo Checks

The following criteria **must** be met **before** you are allowed to demo.

Check	**P/F**
Your latest stage and mylib code **must** be in git.	
Your git repository must be up to date with the latest version of sourcelib.	
Your stage code must build without errors.	
Your mylib and top comments are correctly filled out.	

Failure to meet pre-demo checks will mean that you are not allowed to demo the stage.

7.0.2 Demo and Implementation Criterion

You must be able to combine all design tasks, into the same file and demo all design tasks, without reprogramming your Nucleo. Note: You may be asked to make minor modifications to your code by the assessor, during your demo, which must be passed.

Design Task 1A: Atimer Control using Console	
0	No commands implemented/function or atimer does not work correctly.
1	Some commands are implemented and function correctly, and the atimer works correctly.
2	All commands are imeplemented and function correctly, and the atimer works correctly.
Design Task 2A: Atimer Frequency and Period Control	
0	Both Atimer Frequency and Period control does not work, correctly.
1	Either Atimer Frequency or Period control does work correctly (may have some errors).
2	Both Atimer Frequency and Period control works correctly.
Mylib Atimer HAL Implementation	
0	Atimer HAL is not implemented at all or does not function.
1	Atimer HAL is only partially implemented and has partial correct functionality.
2	Atimer HAL is fully implemented and functions correctly.
Mylib Atimer HAL Configuration Implementation	
0	Atimer HAL configuration header file is not implemented or correctly included in the Atimer HAL source files and Makefile.
1	Atimer HAL is implemented and and is included in the Atimer HAL source but cannot be included using the Makefile compiler directive.
2	Atimer HAL is implemented and and is included in the Atimer HAL source using the Makefile compiler directive.

Design Task 1B: Joystick	
0	No values from Joystick X and Y can be read.
1	Values form Joystick X and Y can be read.
Design Task 2B: LED Bar Display	
0	LED Bar display does not light up.
1	LED Bar display is responsive to the Joystick Y signal but the LED Bar output is not correct.
2	LED Bar display is correctly controlled by the Joystick Y signal.
Design Task 3B: LED Bar Display Brightness Control	
0	LED Bar Display Brightness control does not work correctly .
1	LED Bar Display Brightness is controlled by the Joystick X signal but the LED Bar output is not correct.
2	LED Bar Display Brightness is correctly controlled by the Joystick X signal.
Joystick and PWM Mylib HAL Implementation	
0	Joystick and PWM Mylib HAL is not implemented at all or does not function.
1	Joystick and PWM Mylib HAL is only partially implemented and has partial correct functionality.
2	Joystick and PWM Mylib HAL is fully implemented and functions correctly.

Code Style	
0	One or more style errors found in one tutor selected stage file.
1	No style errors found in **one** tutor selected stage file.

Student Name:

Student Number	**Mark (/16)**	**Marker**	**Date**