University of East Anglia

School of Computing Sciences

CMP-5045B – Embedded Systems

Laboratory Sheet:

Touch Control and Buzzer

1 Introduction

This labsheet introduces the capacitive touch sensor and active-buzzer peripherals provided in the Kuman kit. The skills required to implement such devices will be pivotal to the development of your own system as it makes use of GPIO. It is important to make sure that one understands the labsheet and provided code, and can complete the basic tasks as that will help widen the range of peripherals that can be used for the coursework.
Note: from lectures, you should know the low-level architectures of pins, and how they work. Therefore, this labsheet does not explain some bits in detail. Hence, if you do not understand some parts that are not explained here, you are advised to read the lecture slides.

2 Equipment

1. STM32F746-G discovery board
2. Touch Control
3. Active-Buzzer
4. Keil uVision MDK-Lite (v5.18)

[1] https://arduinomodules.info/ky-012-active-buzzer-module/

[2] http://underpop.online.fr/p/produino-jog-type-touch-sensor-capacitive-touch-switch-module-for-arduino-blue/

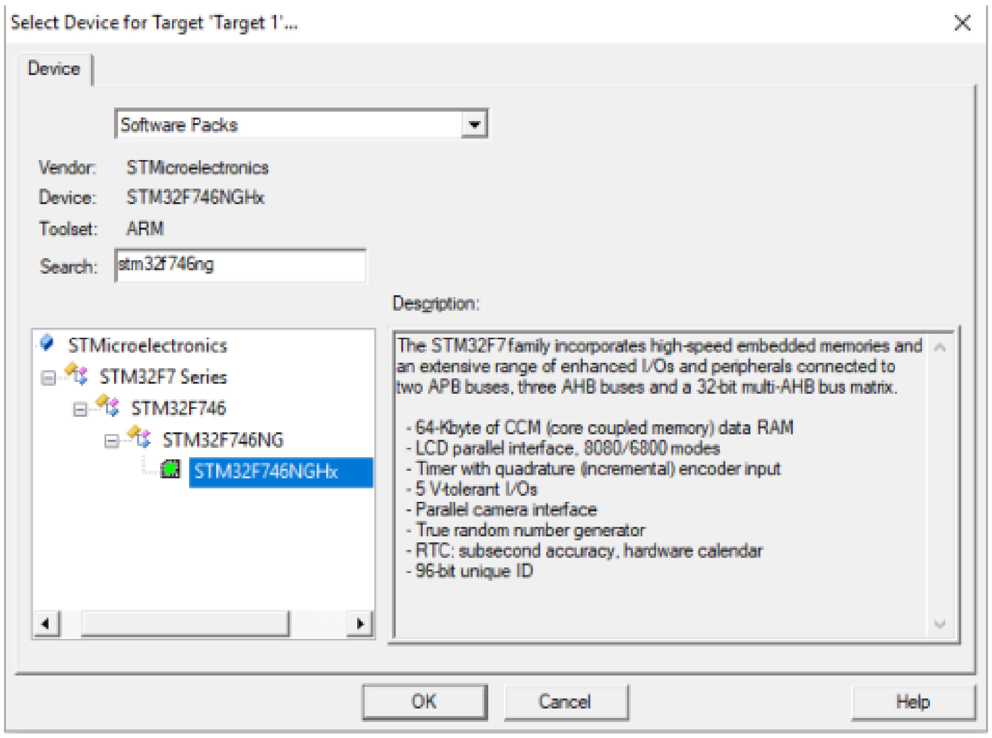
3 Objective

To create a project that makes use of GPIO pins to turn on the buzzer when the touch capacitor control peripheral is pressed.

4 Procedure

1. First, we have to create a new project. You are reminded to work in C:\temp folder or on a USB memory stick. The **STM32F746-G** discovery board comes with the **STM32F746NG** device.

2. The RTE manager is configured as shown below:



Vendor: STMicroelectronics
Device: STM32F746NGHx
Toolset: ARM
Search: stm32f746ng

STMicroelectronics
STM32F7 Series
STM32F746
STM32F746NG
STM32F746NGHx

Description:
The STM32F7 family incorporates high-speed embedded memories and an extensive range of enhanced I/Os and peripherals connected to two APB buses, three AHB buses and a 32-bit multi-AHB bus matrix.

- 64-Kbyte of CCM (core coupled memory) data RAM
- LCD parallel interface, 8080/6800 modes
- Timer with quadrature (incremental) encoder input
- 5 V-tolerant I/Os
- Parallel camera interface
- True random number generator
- RTC: subsecond accuracy, hardware calendar
- 96-bit unique ID

Board Support ⇒ STM32F746G-Discovery;

CMSIS ⇒ Core;

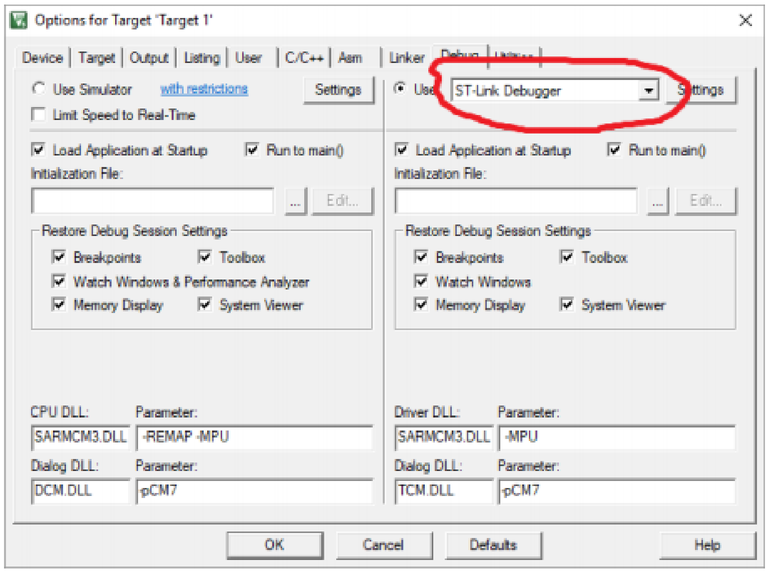
Device ⇒ Startup;

Device ⇒ STM32Cube Framework (API) ⇒ Classic;

Device ⇒ STM32Cube ⇒ HAL Common, Cortex, DMA, Flash, GPIO, I2C, PWR, RCC, TIM.

Click 'Resolve' once you have selected the components, then click 'OK'.

3. It is important to remember that we have to switch the debugger to the **ST-LINK** one instead of the default: **ULINK**, and tick the **'Reset and Run'** box in the debugger settings.

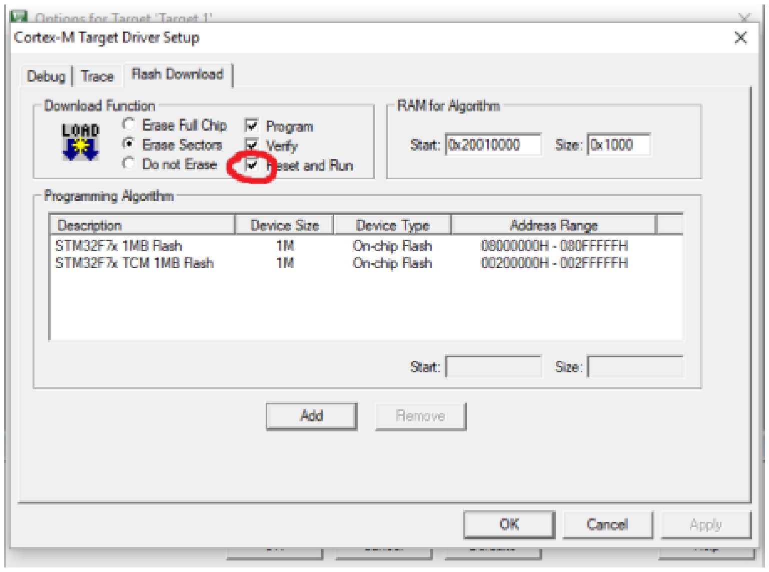


Options for Target 'Target 1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

Left pane:
- ○ Use Simulator with restrictions — Settings
- ☐ Limit Speed to Real-Time
- ✓ Load Application at Startup ✓ Run to main()
- Initialization File:
- Restore Debug Session Settings: ✓ Breakpoints ✓ Toolbox ✓ Watch Windows & Performance Analyzer ✓ Memory Display ✓ System Viewer
- CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU
- Dialog DLL: DCM.DLL Parameter: -pCM7

Right pane:
- ● Use: ST-Link Debugger — Settings
- ✓ Load Application at Startup ✓ Run to main()
- Initialization File:
- Restore Debug Session Settings: ✓ Breakpoints ✓ Toolbox ✓ Watch Windows ✓ Memory Display ✓ System Viewer
- Driver DLL: SARMCM3.DLL Parameter: -MPU
- Dialog DLL: TCM.DLL Parameter: -pCM7



Cortex-M Target Driver Setup

Download Function:
- ○ Erase Full Chip ✓ Program
- ● Erase Sectors ✓ Verify
- ○ Do not Erase ✓ Reset and Run

RAM for Algorithm: Start: 0x20010000 Size: 0x1000

Programming Algorithm

Description	Device Size	Device Type	Address Range
STM32F7x 1MB Flash	1M	On-chip Flash	08000000H - 080FFFFFH
STM32F7x TCM 1MB Flash	1M	On-chip Flash	00200000H - 002FFFFFH

Start: Size:

4. The 3 documents that will be useful for this lab are the page 23 of the manual[3] and the datasheets for both the ky012 active-buzzer[1] and the produino capacitive touch sensor[2].

[1] https://arduinomodules.info/ky-012-active-buzzer-module/

[2] http://underpop.online.fr/p/produino-jog-type-touch-sensor-capacitive-touch-switch-module-for-arduino-blue/

[3] https://www.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf

5. Next, we will write some code to configure a pin and turn it on. Carefully examine the code below and make sure that you understand what each line does. It is a good practice to read source files and documentation to get a better understanding (e.g.: you can right click on GPIO_InitTypeDef⇒'Go To Definition' to see more details about that struct).

```
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_gpio.h"

int main(void){
 GPIO_InitTypeDef gpio;
 //You now need to choose 2 GPIO pins to use for the
 //buzzer and touch sensor. The ones used here are D11
 //& D12 respectfully but it is advised you change them
 __HAL_RCC_GPIOB_CLK_ENABLE();

 //For Buzzer:
 //set mode as output, pulldown
 gpio.Mode = GPIO_MODE_OUTPUT_PP;
 gpio.Pull = GPIO_PULLDOWN;
 gpio.Speed = GPIO_SPEED_HIGH;
 gpio.Pin = GPIO_PIN_14;
 HAL_GPIO_Init(GPIOB, &gpio);

 //For Touch Sensor:
 //set mode as input, PullDown
 gpio.Mode = GPIO_MODE_INPUT;
 gpio.Pull = GPIO_PULLDOWN;
 gpio.Speed = GPIO_SPEED_HIGH;
 gpio.Pin = GPIO_PIN_15;
 HAL_GPIO_Init(GPIOB, &gpio);

 while(1){
  if(HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_15) == 1){
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_SET);
  } else {
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_RESET);
  }
 }
}
```

[1] https://arduinomodules.info/ky-012-active-buzzer-module/

[2] http://underpop.online.fr/p/produino-jog-type-touch-sensor-capacitive-touch-switch-module-for-arduino-blue/

[3] https://www.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf

Copy the code from above, carefully read it and change the ports to get an understanding of how the peripherals work. Collect a breadboard and some wires connecting them to the pins located on the peripherals.

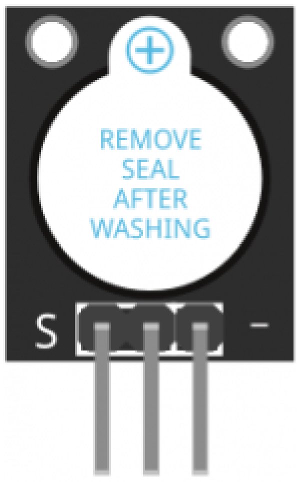


Figure 1: Active Buzzer

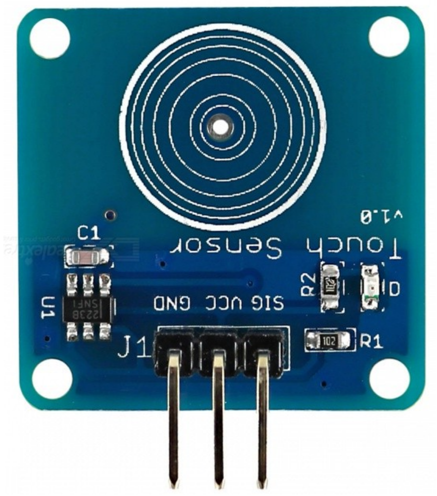


Figure 2: Capacitive Touch Sensor

Pin Position	**Buzzer**	**Touch Sensor**
Left	Signal	GND
Middle	VCC	VCC
Right	GND	Signal

Once you have connected up the peripherals, loaded the code into the main of your project, you should build and download your project into the discovery board's flash.
When you press down on the capacitive touch sensor, the active buzzer should begin to make noise. If not, double check all connections are correct and that the right pins have been enabled in your code.

Tasks:

1) Write a function in the format 'void wait(int time)' and implement it so that the buzzer turning off is delayed until after the touch sensor is released. The numeric value of the delay is in essence arbitrary; however if one intends on using iterative loops to complete this, it is advised to use a number of similar magnitude to 500,000 in order to see clear results.
2) Try changing the initial function to sound the buzzer by default and then use the touch sensor in order to toggle it on and off after a single press (i.e: doesn't require

the user to hold down on the sensor).

3) Extension (This is a much harder task but will make the end of module coursework far easier): Introduce the code from myBlinky.c that you developed in week 1 and look into making it threaded so that the two work independently. (Hint: It is strongly recommended you look into ‘cmsis_os.h’ to implement this - https://www.keil.com/pack/doc/CMSIS/RTOS/html/usingOS.html is a good starting place to understand it.)