Embedded Systems Coursework

Additional Labsheet

Cameron Shipman

Introduction

This labsheet will make use of and demonstrate how to use GPIO pins to run simple peripherals, the examples used are: the buzzer and two-colour LED.

Equipment

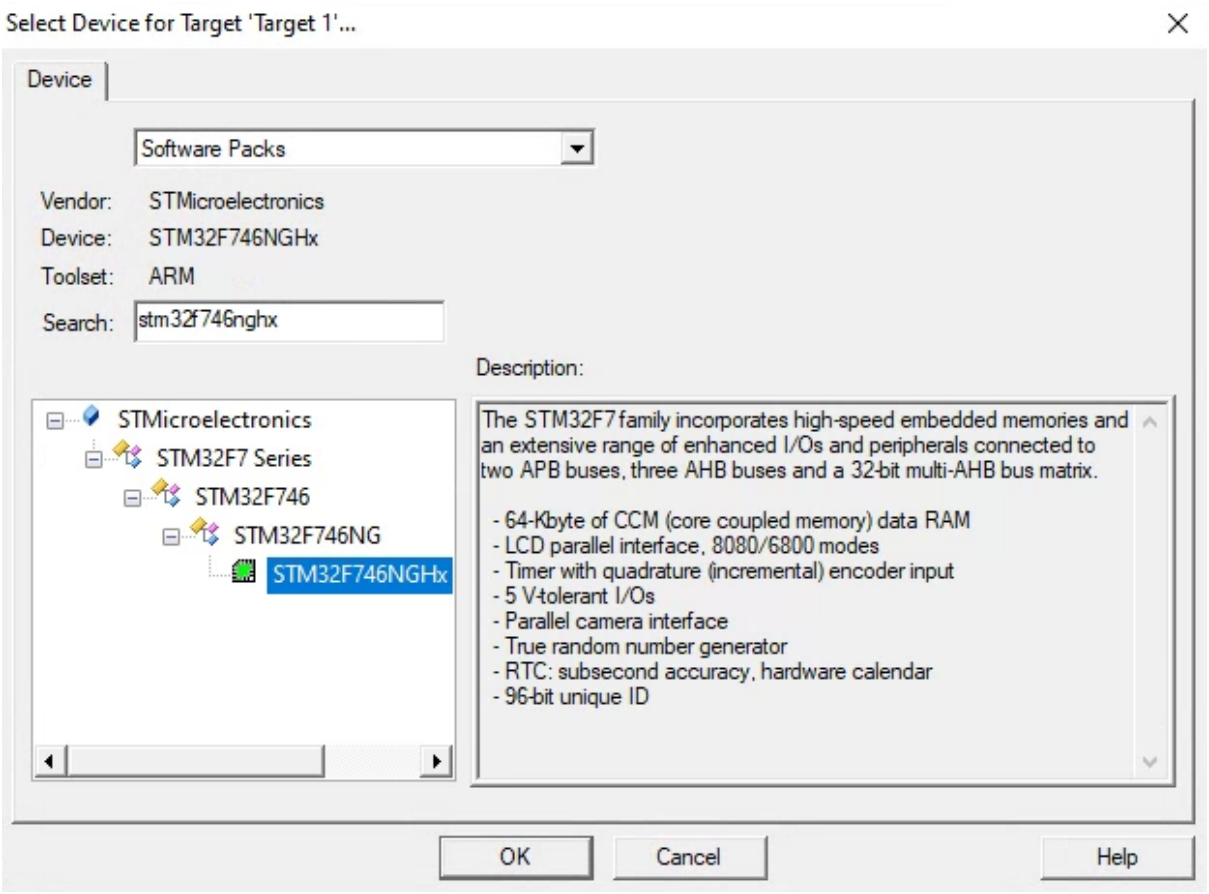
STM32F746-G discovery board
Buzzer
Two-colour LED
Keil uVision MDK-Lite v5

Objective

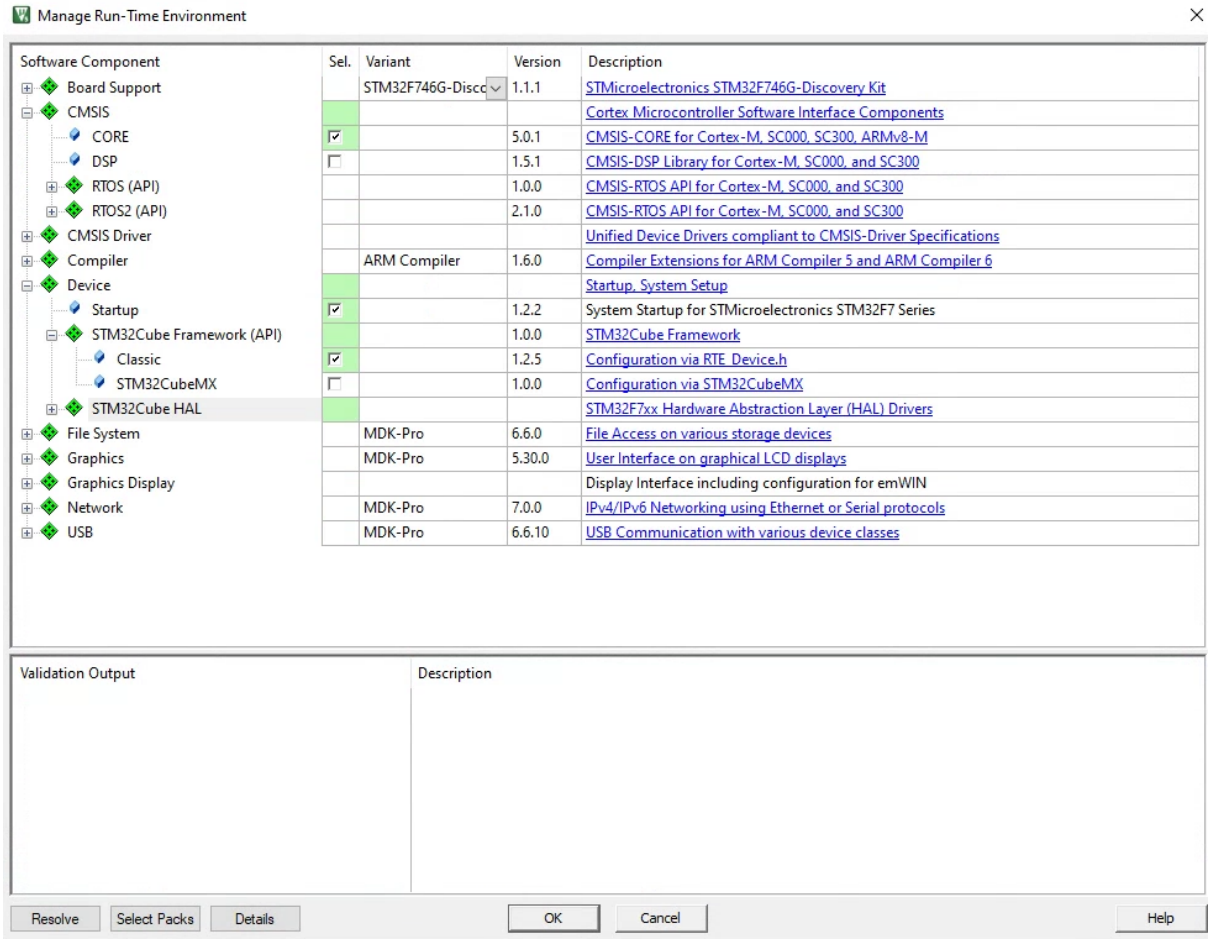
To set up a project that can use GPIOs to power on and off simple peripherals (the buzzer and two-colour LED).

Procedure

1. Create a new project with the STM32F746NGHx device.

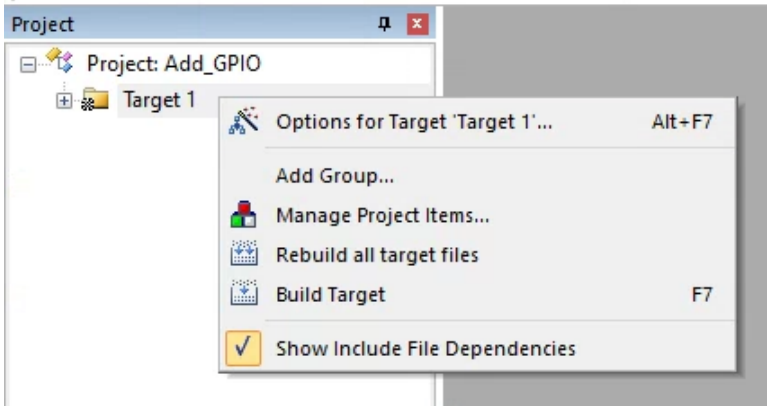


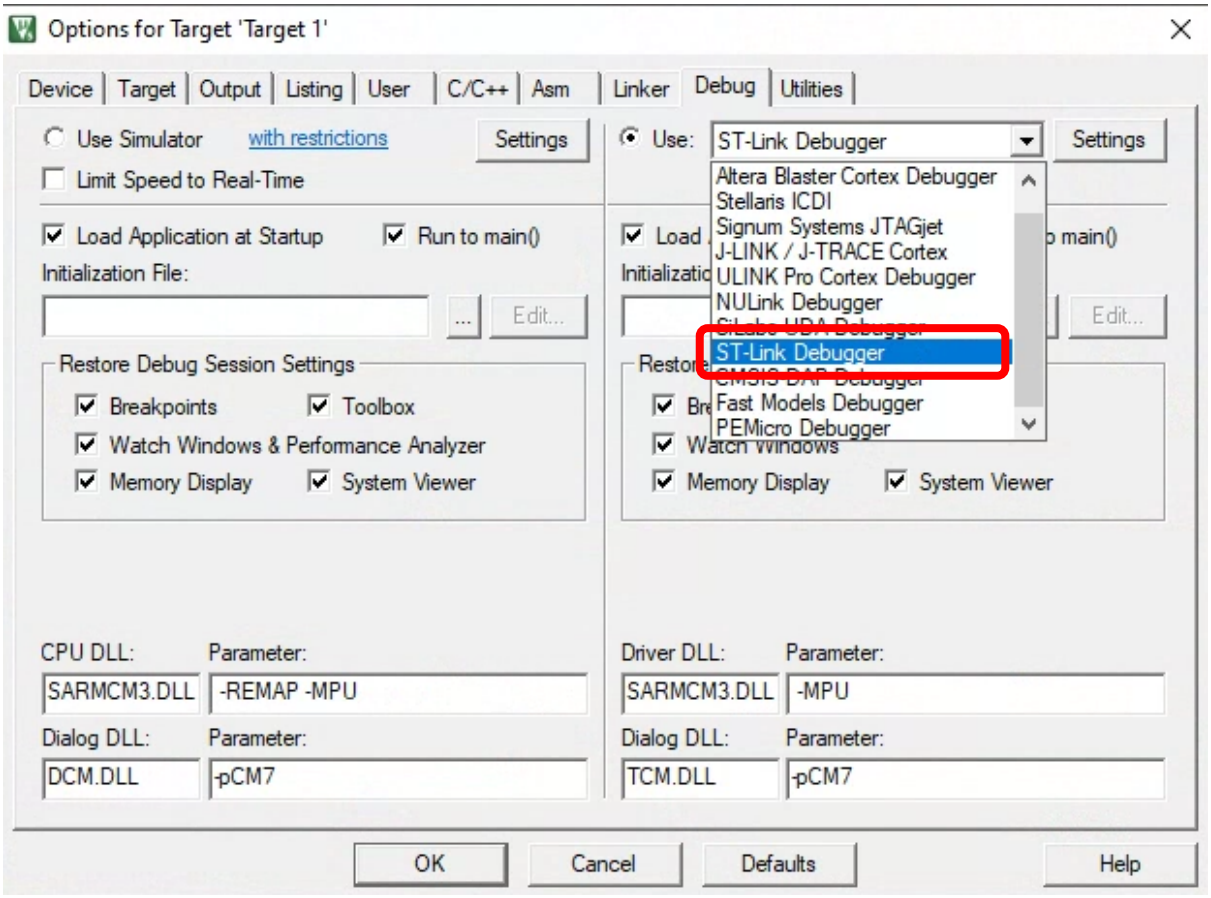
2. Configure the RTE manager as below:
   - Board Support → STM32F746G-Discovery
   - CMSIS → Core
   - Device → Startup
   - Device → STM32Cube Framework (API) → Classic
   - Device → STM32Cube HAL → Common, Cortex, DMA, GPIO, PWR, RCC

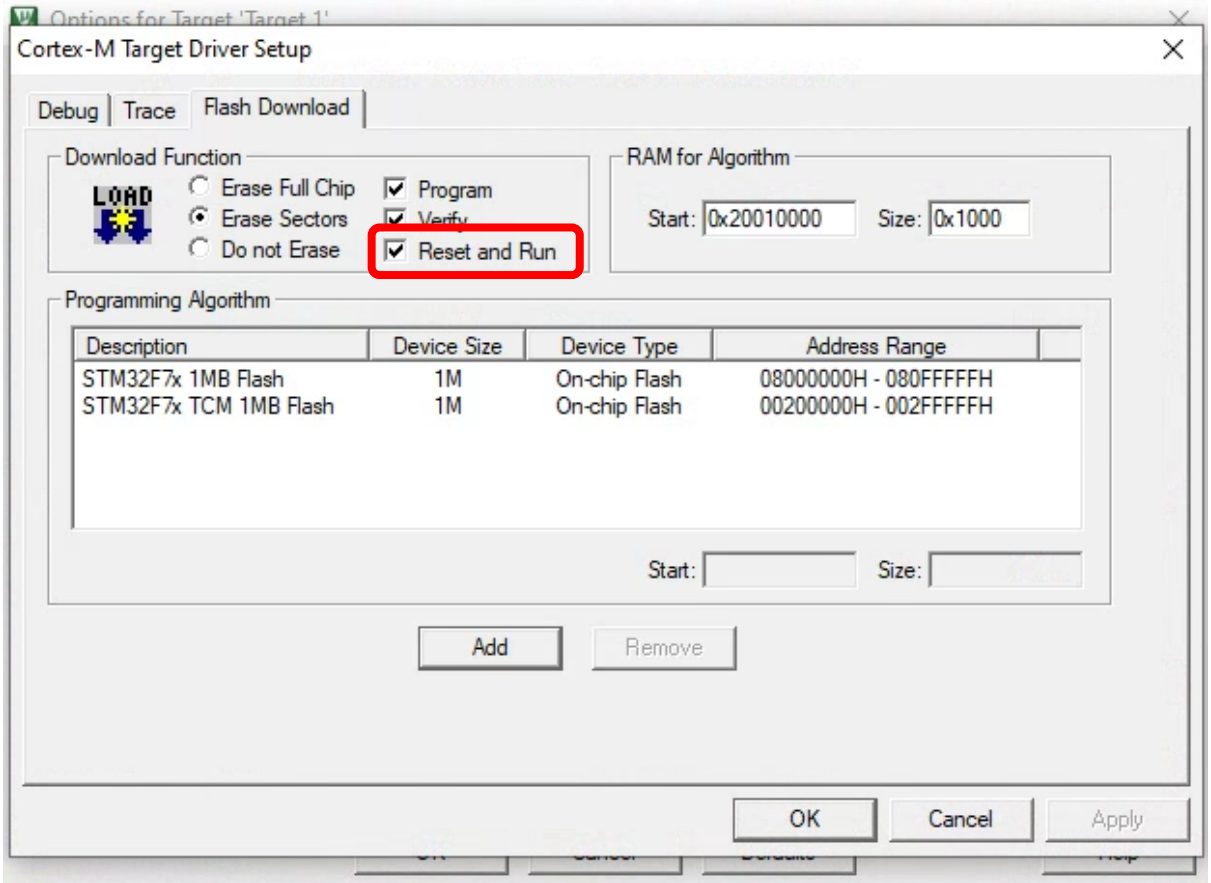


Click Resolve once you have selected these, then click OK.

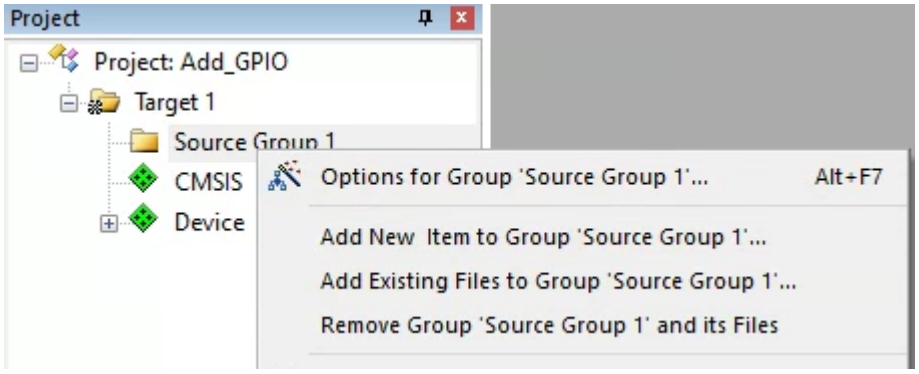
3. In the left Project panel, right click on **Target 1** and select Options. Navigate to the **Debug** tab. Switch the debugger to **ST-LINK** from the default **ULINK**, and enable **Reset and Run** in the debugger settings.



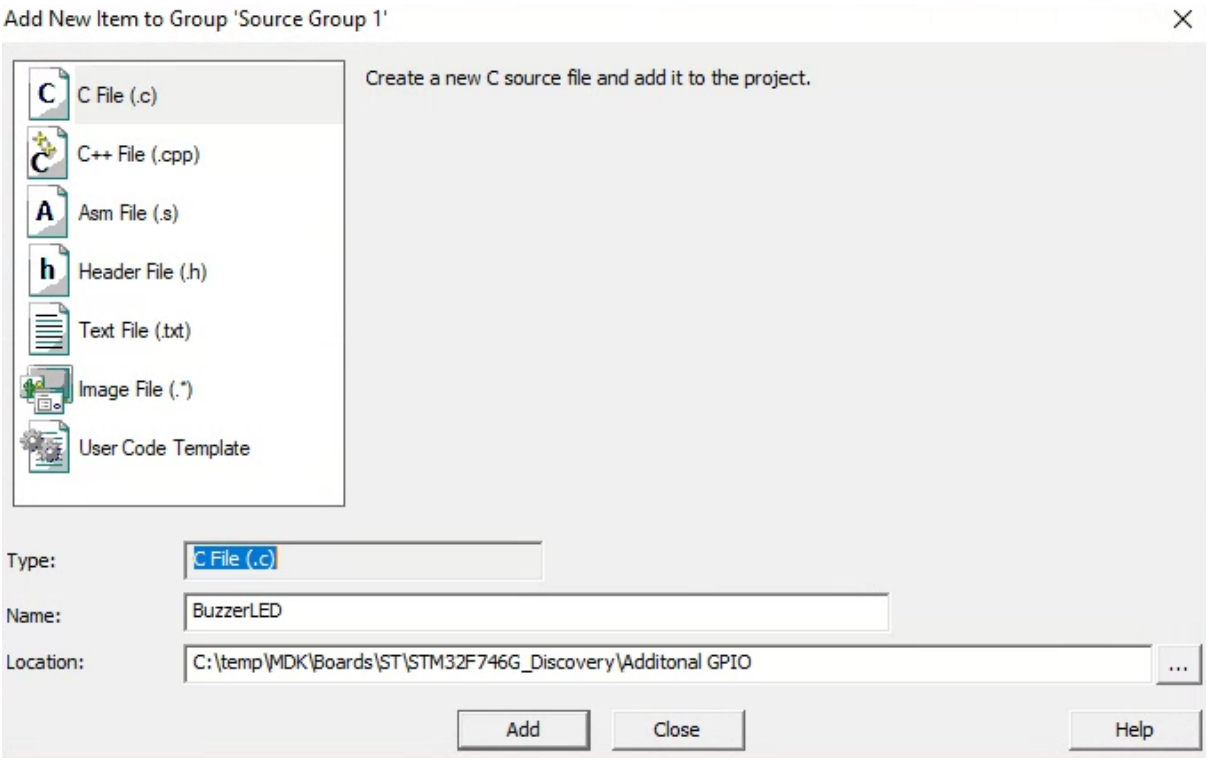




4. To begin, first make a C file in the project. Expand Target 1 in the left panel and right click on Source Group 1 and select Add New Item.



Select C File and enter in a desired name.



5. Just like for a 7-seg, we will use GPIO power and ground for the components. For this example, pins D3 and D4 will be used, however, any pins can be configured for the same purpose.
   Use https://www.st.com/resource/en/user_manual/dm00190424-discovery-kit-for-stm32f7-series-with-stm32f746ng-mcu-stmicroelectronics.pdf If you wish to learn the peripheral base for other pins, this document has the necessary table
6. We will write some code to configure and power these pins. The code below includes the instructions required to initialise and turn them on.

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

int main(void){
    GPIO_InitTypeDef gpio3;
    GPIO_InitTypeDef gpio4;


    //enable clock for B and G base
    __HAL_RCC_GPIOG_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    //set mode as output, no pull
    gpio3.Mode = GPIO_MODE_OUTPUT_PP;
    gpio3.Pull = GPIO_NOPULL;
    gpio3.Speed = GPIO_SPEED_HIGH;
    gpio3.Pin = GPIO_PIN_4;
    HAL_GPIO_Init(GPIOB, &gpio3);

    gpio4.Mode = GPIO_MODE_OUTPUT_PP;
    gpio4.Pull = GPIO_NOPULL;
    gpio4.Speed = GPIO_SPEED_HIGH;
    gpio4.Pin = GPIO_PIN_7;
    HAL_GPIO_Init(GPIOG, &gpio4);

    //set the pins (this will turn them on)
    HAL_GPIO_WritePin(GPIOB,GPIO_PIN_4,GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOG,GPIO_PIN_7,GPIO_PIN_SET);
}
```

Copy this code into your c file. Take the buzzer and LED, both components have similar pinouts. They have 3 pins with the middle being disconnected. The S side is power and – side is for ground. Plug either of your two configured pins (in this case D3 and D4) into the peripherals S side and connect their ground pins to the GND pin on the board. You may need a breadboard for the components to share this connection. It also recommended to add a resistor here to protect the components from burning/damage.

7. When building and flashing this code, it will simply turn on the buzzer and LED indefinitely.

GPIO Buzzer and LED Task

- Modify the code so the buzzer and/or LED can turn off after a certain period of time.
- Try and make this repetitive so that the LED flashes and buzzer beeps. (HAL_delay function will be useful for this feature).