Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 4/26/2021

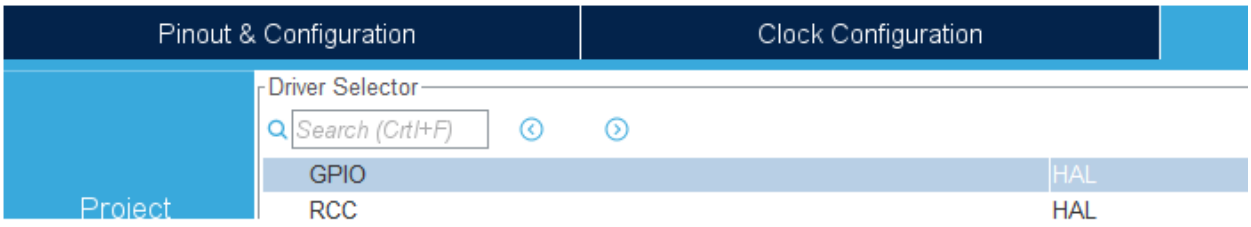
Assignment 3: HAL API Usage

The following will document completion of the third assignment for ECE-40291, with the stated goals of:

1. Use STM32CubeIDE to generate the default code for the STM32 Discovery Board, being sure to include the HAL when generating the code.
2. Add code to blink the LED2 at 250 ms when the Blue Push Button in GPIO_PIN_SET condition
3. Add code to blink the LED2 at 1000 ms when the Blue Push Button is in GPIO_PIN_RESET condition
4. Use HAL to read the state of the Blue Push Button, hence changing the rate of LED flash (as in #2 and #3)

1. Use STM32CubeIDE to generate the default code for the STM32 Discovery Board, being sure to include the HAL when generating the code.

To begin, load the IDE and generate a default project for the IOT Disco board using the same process as seen in the previous assignments. While selecting the configuration options, ensure that the GPIO configuration is set to the "HAL" option as opposed to the "LL" selection made in the previous assignment.



2. Add code to blink the LED2 at 250 ms when the Blue Push Button in GPIO_PIN_SET condition

At this point, we can begin to develop our application code as needed. This serves as an excellent example of what makes the HAL so useful and easy to develop with. To accomplish the blink function, we can leverage the built in HAL_GPIO_ReadPin(), HAL_GPIO_TogglePin() and HAL_Delay() functions, providing the appropriate GPIO information and a time delay in milliseconds, as seen below.

```
133     if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13))
134     {
135         HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
136         HAL_Delay(250);
137     }
```

Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

Date: 4/26/2021

3. Add code to blink the LED2 at 1000 ms when the Blue Push Button is in GPIO_PIN_RESET condition

In the same manner, those functions can be applied for the other state of the button not being pressed.

```
138         else
139         {
140             HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
141             HAL_Delay(1000);
142         }
```

4. Use HAL to read the state of the Blue Push Button, hence changing the rate of LED flash (as in #2 and #3)

With this simple code written, it can all be tied together in the main() while(1) loop.

```
118   /* Infinite loop */
119   /* USER CODE BEGIN WHILE */
120   while (1)
121   {
122     /* USER CODE END WHILE */
123
124     /* USER CODE BEGIN 3 */
125
126       /*
127       * Within loop, poll for status of Blue Button, GPIOC.13 (active low).
128       * If GPIO_PIN_SET (button not pushed): blink LED2, GPIOB.14 (active high)
129       * on a 250mS cadence.
130       * If GPIO_PIN_RESET (button pushed): blink LED2 on a 1000mS cadence
131       */
132
133       if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13))
134       {
135           HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
136           HAL_Delay(250);
137       }
138       else
139       {
140           HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);
141           HAL_Delay(1000);
142       }
143   }
144   /* USER CODE END 3 */
```

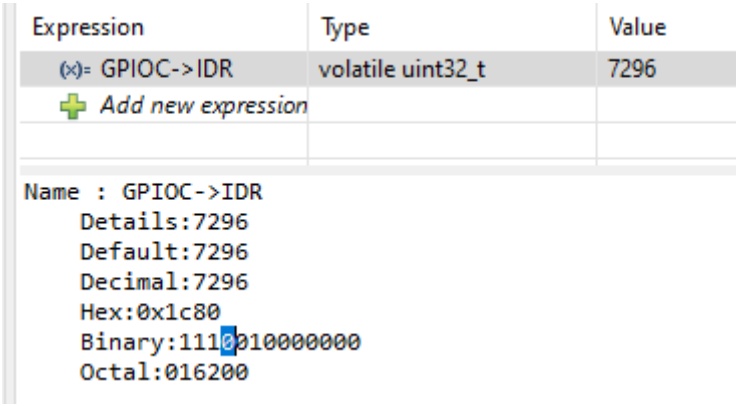
As an example of what makes the HAL APIs so easy to use, one can use the GPIO macros as seen above that call out the ports and pins, or the names assigned by the pin mapping graphical interface could also be used, such as in the below example.

Nathan Bunnell
Embedded Controller Programming with C
ECE-40291
Student ID: U08895857

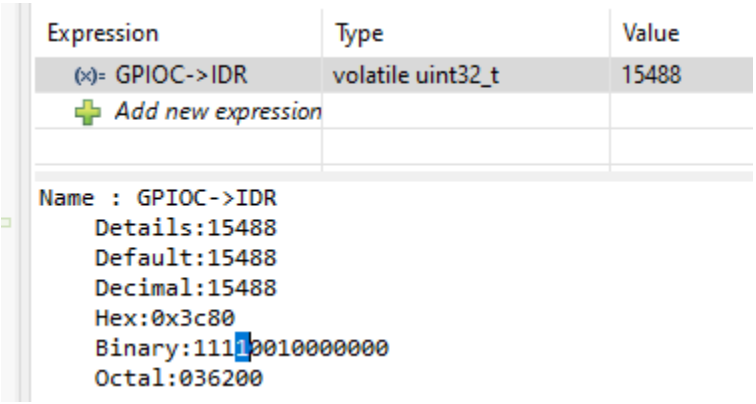
Date: 4/26/2021

```
if(HAL_GPIO_ReadPin(BUTTON_EXTI13_GPIO_Port, BUTTON_EXTI13_Pin))
{
    HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
    HAL_Delay(250);
}
else
{
    HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
    HAL_Delay(1000);
}
```

So, with the application code written and compiling without errors or warnings, we can flash the Disco board and run the program. With the debugger running, we can step through the code and monitor the status of GPIOC's IDR register, seeing when the button is pushed:



Or not pressed:



From her, we can also run the program as normal and monitor the blink rate visually.

In closing, this short introduction to the HAL APIs has displayed how easy it is to quickly develop application code with a minimal amount of extra effort. It also displayed the flexibility built in to the HAL and IDE, allowing the use of either the GPIO or the default & user assigned names via the GUI.