To practice with program-controlled I/O, write a program that creates and displays one of two patterns on the 4 LEDs on the Discovery board. Under the control of the User push button (the blue button).

<table>
<thead>
<tr>
<th>LED</th>
<th>Description</th>
<th>Port Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED3</td>
<td>(orange)</td>
<td>PD13</td>
</tr>
<tr>
<td>LED4</td>
<td>(green)</td>
<td>PD12</td>
</tr>
<tr>
<td>LED5</td>
<td>(red)</td>
<td>PD14</td>
</tr>
<tr>
<td>LED6</td>
<td>(blue)</td>
<td>PD15</td>
</tr>
<tr>
<td>User button</td>
<td>(blue)</td>
<td>PA0</td>
</tr>
</tbody>
</table>

The program is to operate as follows.
1. Initially, all LEDs are off.
2. On the first press of the user button, the LEDs should turn on with the following pattern: **LED3 – LED4 – LED6 – LED5 – ALL OFF** *(each LED remains ON until ALL OFF)*
   - This pattern is to repeat until the next button press. Note that you should see the LEDs turn on in a counter-clockwise circular pattern.
   - Each step of the pattern is to be held for exactly one-half second.
3. On the next press of the user button, the LED pattern is to change to the following: **LED3 – LED5 – LED6 – LED4 – ALL OFF** *(each LED remains ON until ALL OFF)*
   - This pattern is to repeat until the next button press. Note that you should see the LEDs turn on in a clockwise circular pattern.
   - Each step of the pattern is to be held for exactly one-half second.
4. On the next button press, return to step 1 (all LEDs off) and repeat steps 1-4 continuously.
5. You must compensate for any “bounce” in the button, so that there is exactly one action per button press.

The program is to contain the following six modules:
1. An **output handler**, written in ARM assembly language, which writes patterns to the LEDs.
2. An **input handler**, written in ARM assembly language, which tests the user button, and sets a global variable.
3. A **system tick timer interrupt handler**, written in C, which is activated every one-half second. This routine should call the output handler, if the LEDs are to be changed.
4. A **main program**, written in C, which executes in a continuous loop, calling the input handler every time through the loop.
5. The “startup” and “system initialization” files for the STM32F4-Discovery board (inserted if you select “startup” under “device” in the “Manage Run-Time Environment” window when you create the project.)
6. The STM32F4xx microcontroller header file, stm32f4xx.h. (At top of main program, use `#include “stm32F4xx.h”`)

See program testing and submission on the next page.
Testing and submitting the program:

You may test the program in RAM on the board or in flash memory. The final version is to be programmed into the flash memory of the board, so that the program can be demonstrated without being connected to the Keil debugger.

Print and submit the source program, *and also email it to me.*

Instead of printing multiple debug windows, bring your programmed board to my office and demonstrate the program’s operation to me.