What build environment are you using?

STM32CubeIDE version 1.9.0 on STM32L476RG development board.

Can you step through the code to see what each line does?

From the “main.c” file:

```
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : Blue_Btn_Pin */
  GPIO_InitStruct.Pin = Blue_Btn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(Blue_Btn_GPIO_Port, &GPIO_InitStruct),

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

}
```

Clock is enabled for GPIO A, B, C and H

LED2 pin is set to 0V

Blue button pin: input, interrupt on falling edge, no internal pullup (4.7k connected outside the µC)

LED2 pin: output, push-pull mode, no pullup, low speed

External interrupts initialization (#10 to #15) priority (0) and preemption priority (0).

Interrupts #10 to #15 are enabled within the Nested vectored interrupt controller (NVIC)

The interrupt service routine from the “stm32l4xx_it.c” file:

```c
/******************************************************************************/
/* STM32L4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32l4xx.s).                   */
/******************************************************************************/

/**
  * @brief This function handles EXTI line[15:10] interrupts.
  */
void EXTI15_10_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI15_10_IRQn 0 */

  /* USER CODE END EXTI15_10_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(Blue_Btn_Pin);
  /* USER CODE BEGIN EXTI15_10_IRQn 1 */

  /* USER CODE END EXTI15_10_IRQn 1 */
}
```

Interrupt Handler attached to the blue button pin

Push-button debouncing and LED toggling

```c
/* USER CODE BEGIN 4 */

// External Interrupt ISR Handler CallBackFun
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    counterOutside++;
    currentMillis = HAL_GetTick();

    if (GPIO_Pin == Blue_Btn_Pin && (currentMillis - previousMillis > 10))
    /* Check that interrupt source is the button external interrupt
       Debounce delay set to 10ms but adjustable according to button type; any new interrupts during the delay are ignored */
    {
        counterInside++;
        HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin); // Toggle the LED
        previousMillis = currentMillis;
    }
}

/* USER CODE END 4 */
```

The debouncing routine is not mine, it originates from this website. Two counters are put in place, for the sake of the demo: one adds up the interrupt events outside the “if” code while the other takes care of the events inside the “if”. A higher outside counter value indicates that bounces happened which were masked thanks to the 10ms delay.

(x)= Variab | Break | Expre | Modul | Regist | Live E | SFRs

Expression	Type	Value
(x)= counterOutside	uint32_t	10
(x)= counterInside	uint32_t	9
Add new expression		

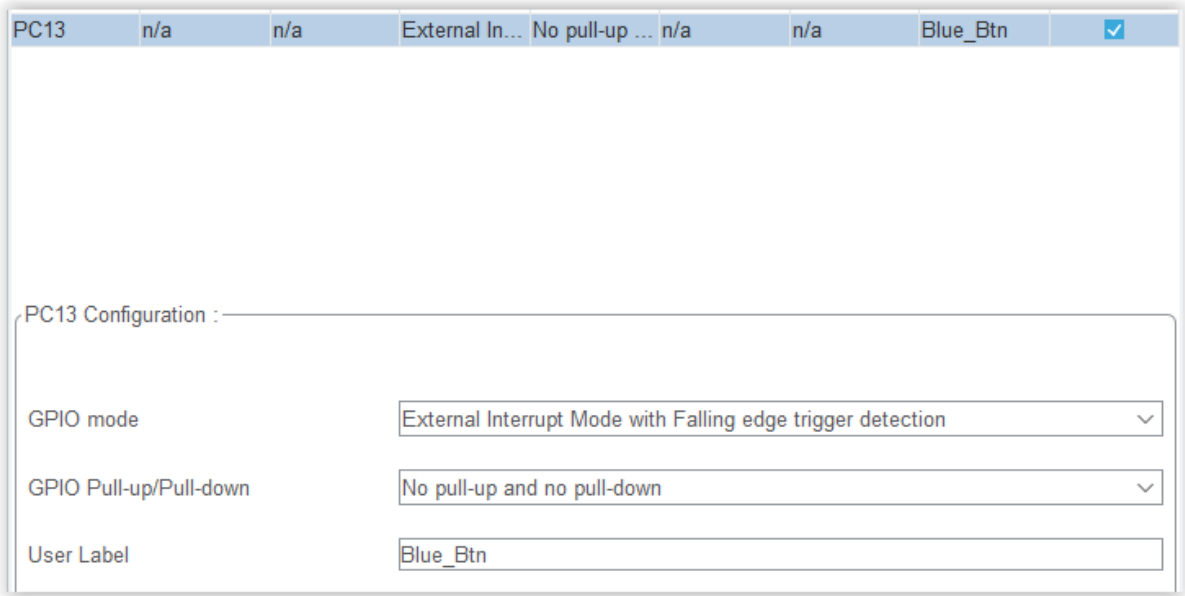
What are the hardware registers that cause the LED to turn on and off?

From the STM32L476RG Reference Manual RM0351, the relevant registers for turning the LED (LED2 on Port A, Pin 5) on and off are:

- The GPIOx_MODER, to set the pin as input, output, alternate function or analog
- The GPIOx_BSRR, which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR)
- The GPIOx_ODR, which stores the data to be output the pin

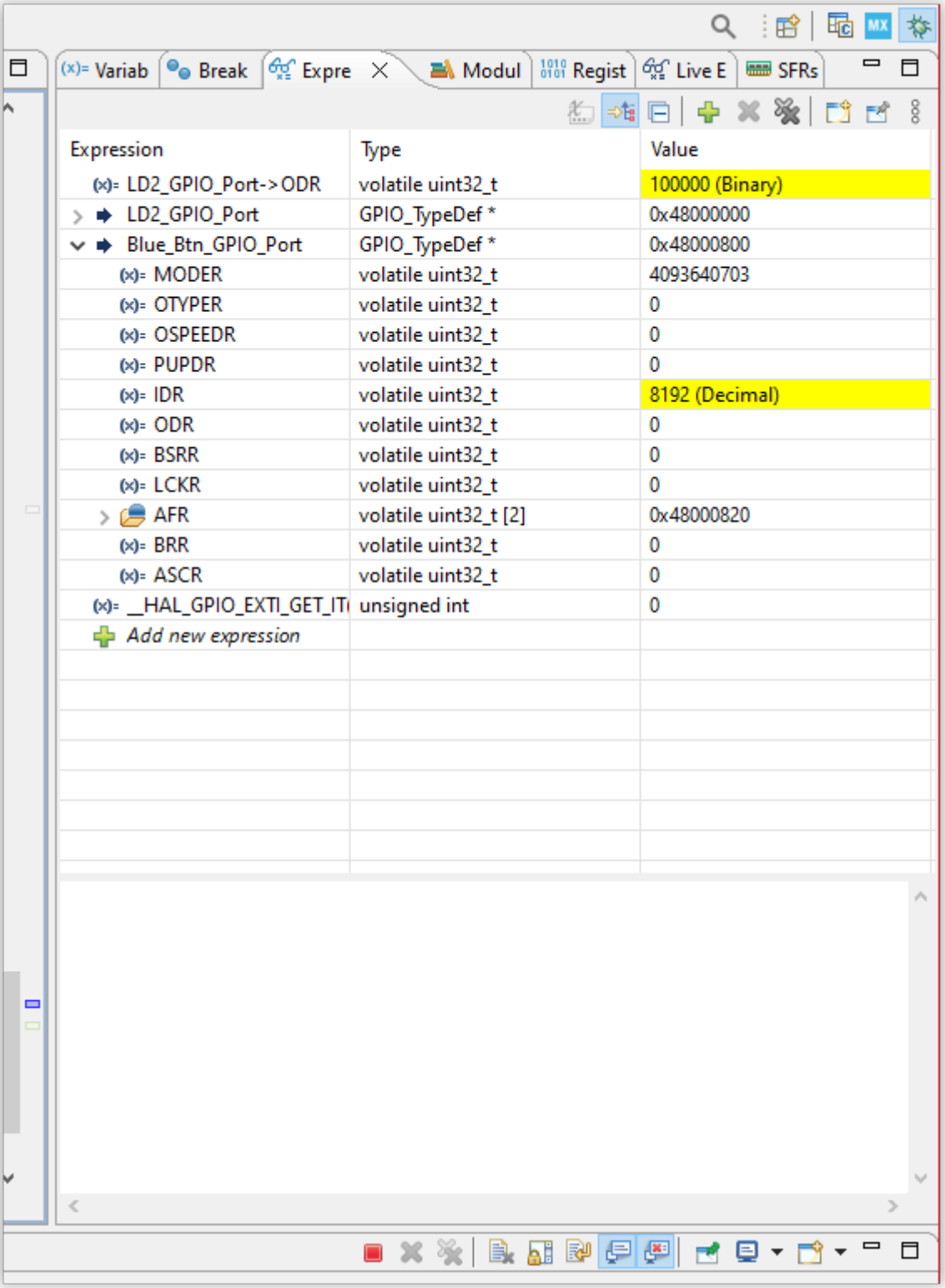
What are the registers that you read in order to find out the state of the button?

The blue button pin is initialized as an input on Port C, Pin 13. The register used to read the pin state is GPIOx_IDR. The pull-up/pull-down register GPIOx_PUPDR is not used due to the presence of an external 4.7kohm resistor on that pin. An external interrupt EXTI is set to trigger on a falling edge.



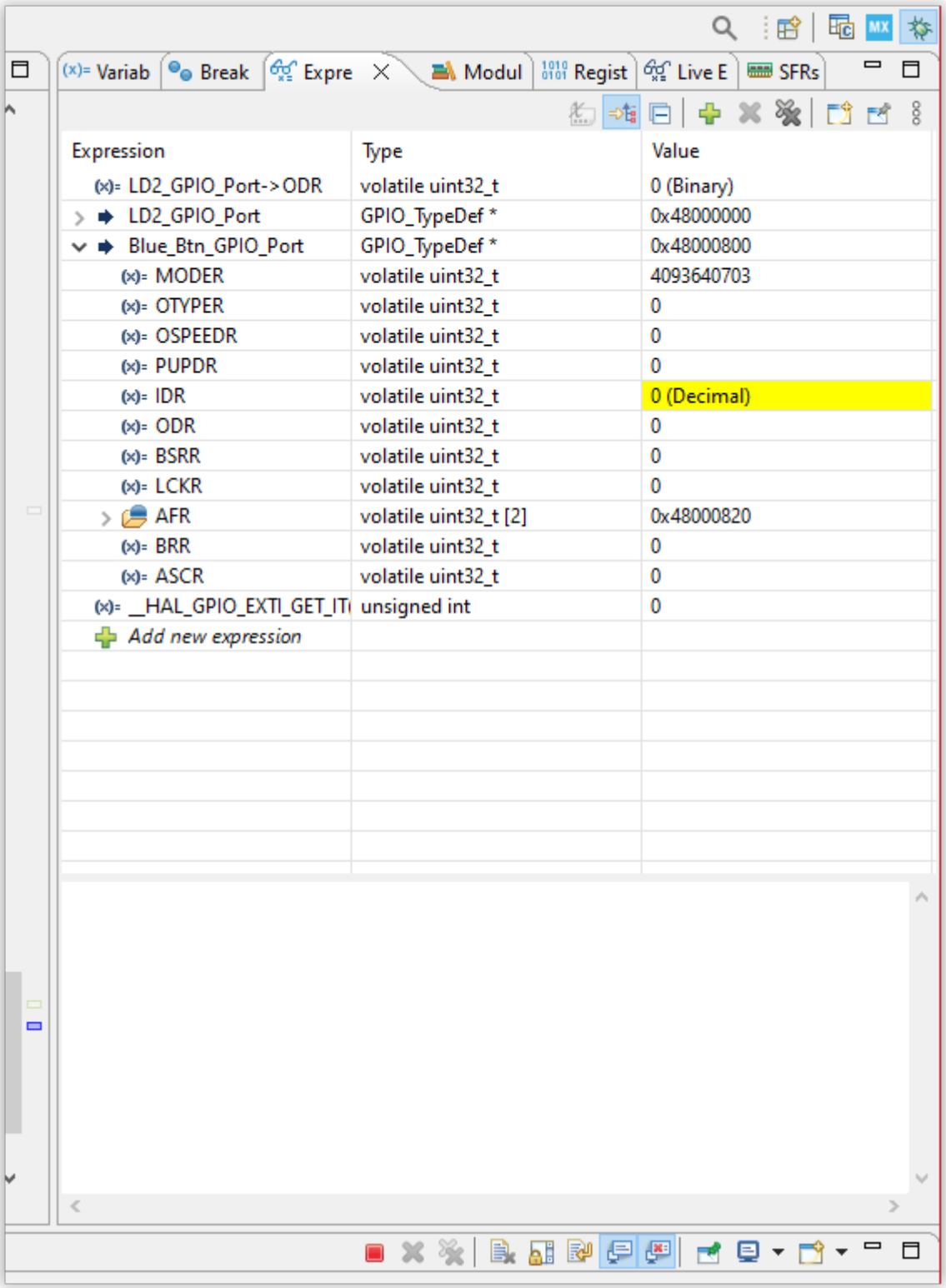
Can you read the register directly and see the button change in a debugger or by printing out the value of the memory at the register's address?

Button released (Idle state – Pin 13 pulled to VCC)



IDR decimal value is 8192 that is 0010 0000 0000 0000 in binary. Therefore, bit 13's value is indeed set to 1.

Button pressed (Pin 13 to GND)



Expression	Type	Value
LD2_GPIO_Port->ODR	volatile uint32_t	0 (Binary)
LD2_GPIO_Port	GPIO_TypeDef *	0x48000000
Blue_Btn_GPIO_Port	GPIO_TypeDef *	0x48000800
MODER	volatile uint32_t	4093640703
OTYPER	volatile uint32_t	0
OSPEEDR	volatile uint32_t	0
PUPDR	volatile uint32_t	0
IDR	volatile uint32_t	0 (Decimal)
ODR	volatile uint32_t	0
BSRR	volatile uint32_t	0
LCKR	volatile uint32_t	0
AFR	volatile uint32_t [2]	0x48000820
BRR	volatile uint32_t	0
ASCR	volatile uint32_t	0
__HAL_GPIO_EXTI_GET_IT	unsigned int	0
Add new expression		

IDR register value is now zero.

See also the practical implementation video file “Blinky_L476.mp4”