EXPERIMENT 6: USART PERIPHERALS

Objectives

The objectives of Experiment 6 is

- to learn how to use Universal Synchrous / Asynchrous Serial Communnications peripherals

Apparatus Required:

- STM32CubeMx
- Keil µVision (MDK ARM)
- STM32 ST-Link Utility
- STM32F4 Microcontroller
- 2 Jumper Cables

Preliminary Work:

1. Study the USART (L08) notes.
2. Write the codes of the experimental work in Keil µVision.

Experimental Work:

1. Create a new project in CubeMx (Figure 1). Select STMF407VGTx, then STMF407G-DISC1 and finally Start Project (Figure 2). First adjust the Pinout&Configuration settings. Close the unnecessary pins (Figure 3).

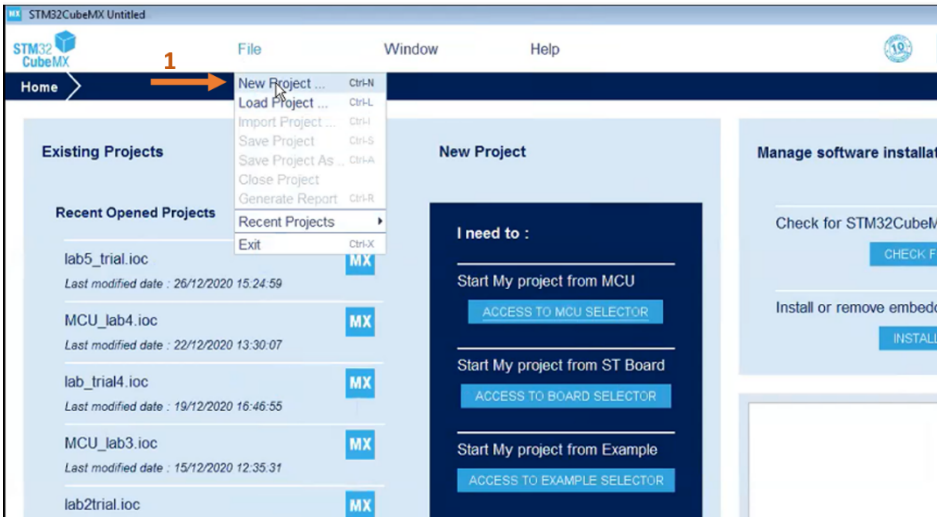

Figure 1

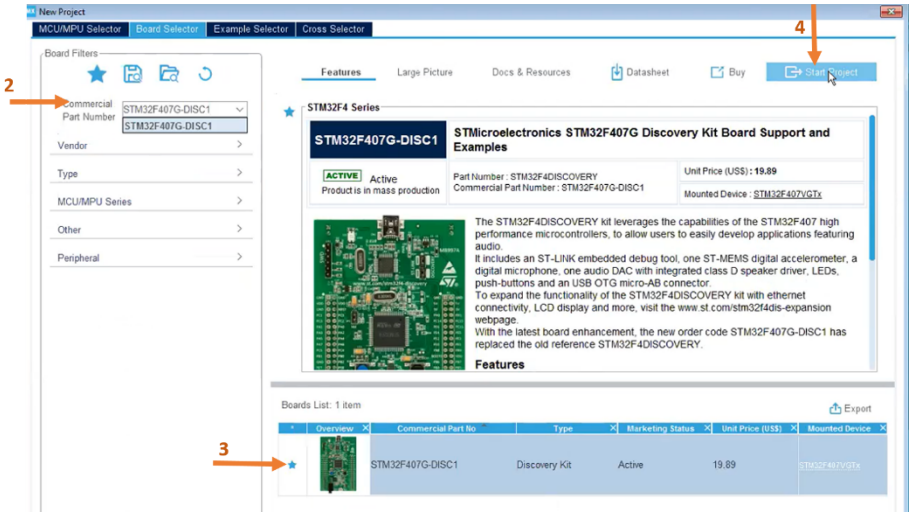

Figure 2

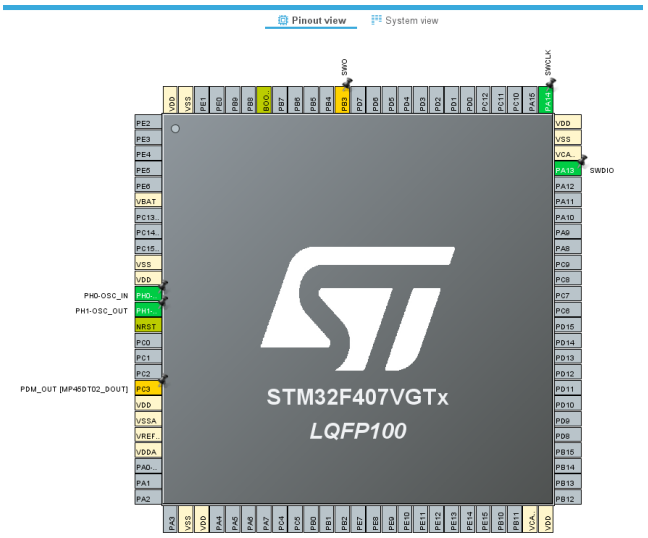

Figure 3

2. We use USART1 and USART2 peripherals in the lecture. Set the configuration for USART1 (Figure 4) and USART 2 (Figure 5). Use USARTs in Asynchronous Mode. BoudRate is 9600 Bits/s, Word Length is 8 Bits, Parity is None, Stop Bit is 1, Data Direction is Receive and Transmit, Over Sampling is 16 Samples for both of the USARTs. Select the PA0 pin as GPIO_EXTI0. Go to GPIO settings and set the GPIO configurations for the PA0 pin as in Figure 6. Set the NVIC configurations as in Figure 7.

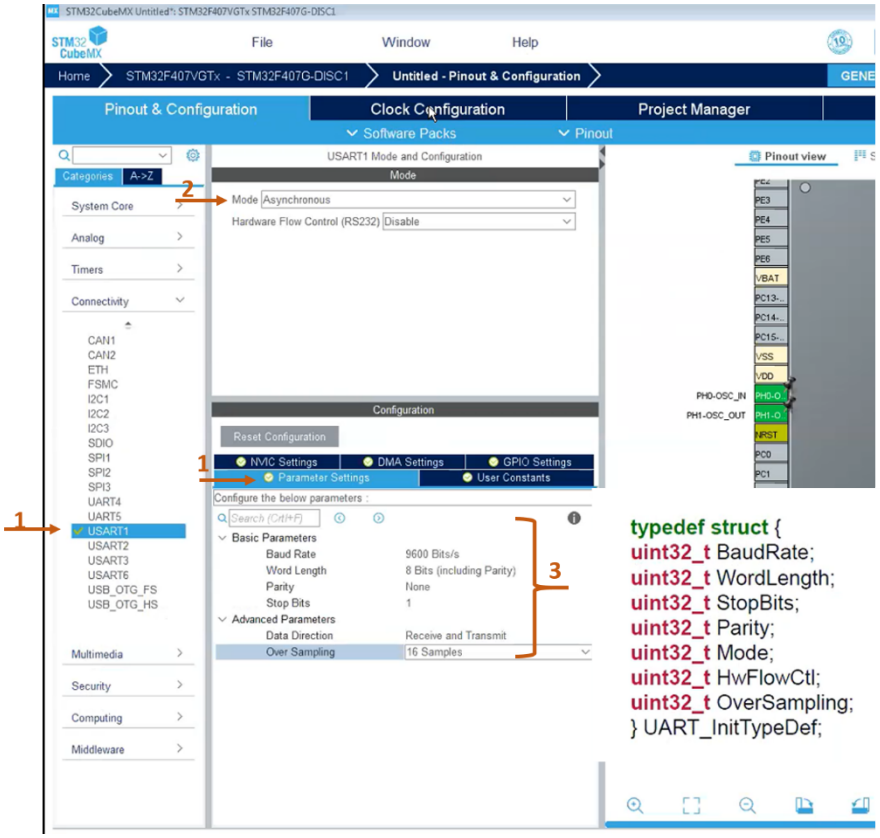

Figure 4

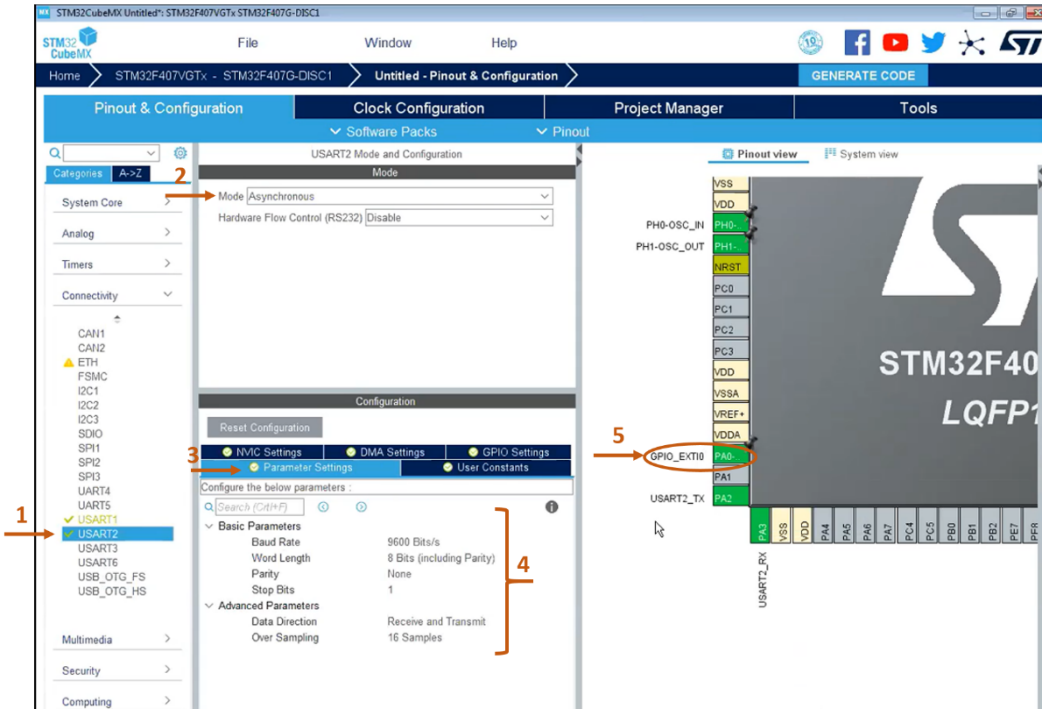

Figure 5

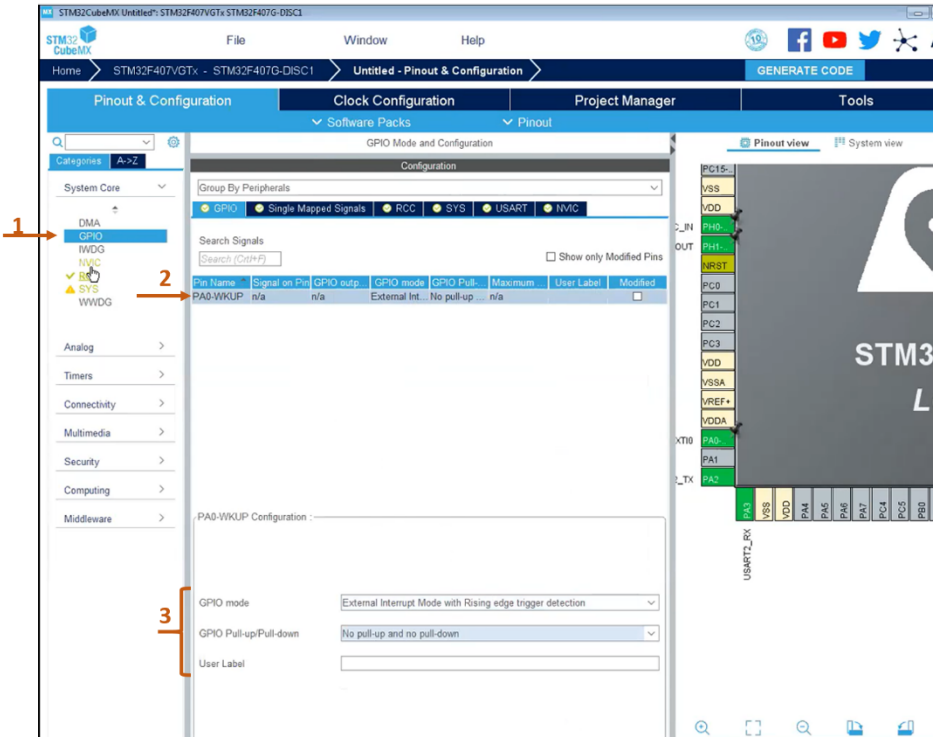

Figure 6

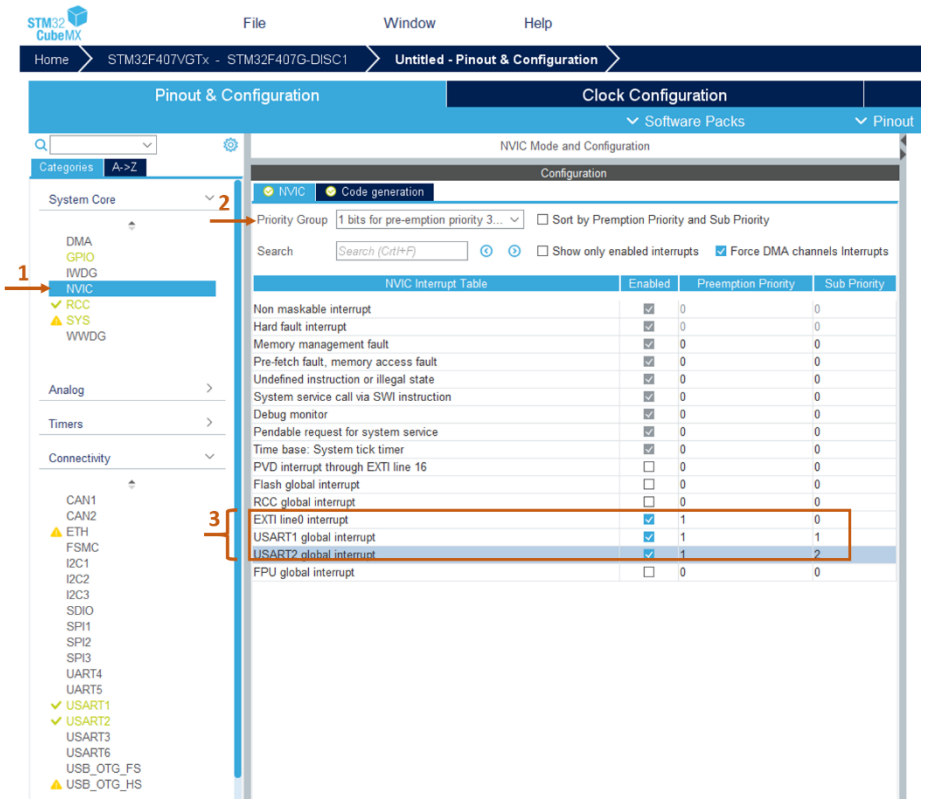

Figure 7

3. There is no hardware connection between the USART. Use the cables to connect the USART peripherals to each other. (Connect PA2 with PA10; Connect PA3 with PA9 using cables). Now,

you can set the project manager configurations as in Figure 8 and go to the Keil µVision by selecting Generate Code.

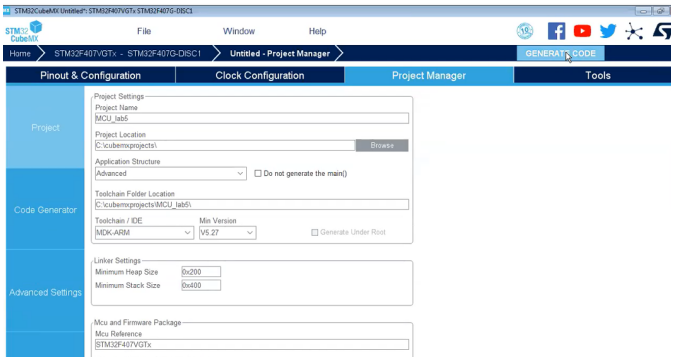

Figure 8

4. Go to main.c file and build the codes. Examine the main.c file to observe the configurations which are already done in CubeMx. We write the codes in interrupt mode. Open the stm32fxx_it.c interrupt file.

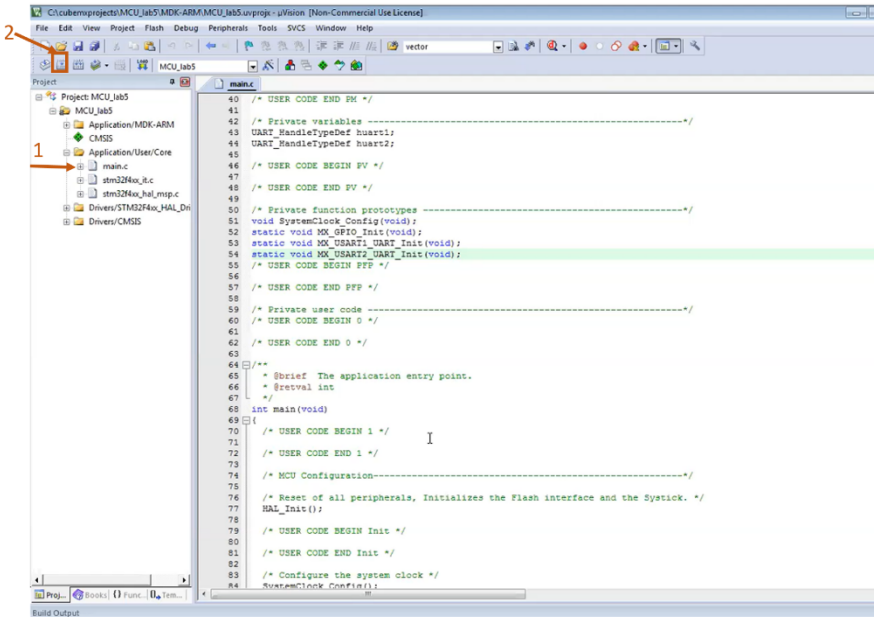

Figure 9

5. When the button is pushed, transmit data from UART2 and write it into the transmit buffer. Then go to UART2 and receive data from UART1. Write the data into the receive buffer.
   We are going to write the codes in interrupt mode. Open the stm32fxx_it.c interrupt file and write the codes in this file (Figure 12). First, define the transmit and receive buffer which will keep the transmitted and received data (Figure 10) . Use HAL_UART_Transmit_IT function in EXTI0_IRQHandler function and HAL_UART_Receive_IT function in USART2_IRQHandler function which is given in Figure 10. Finally, build and Load the codes. Follow the change of variables using the Debug menu (Figure 13).

```
43  /* Private variables ---------------------------------------------
44  /* USER CODE BEGIN PV */
45  char transmit[]="hello";
46  char receive [6]; // 6:sizeof(hello)+1 null bit
47  /* USER CODE END PV */
```

Figure 10

UART Communication in Interrupt Mode

To transmit a sequence of bytes in interrupt mode, the HAL defines the function:
HAL_UART_Transmit_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);

To receive sequence of bytes in interrupt mode, the HAL defines the function:
HAL_UART_Receive_IT(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size);

Figure 11

main.c | stm32f4xx_it.c*

```
205  void EXTI0_IRQHandler(void)
206  {
207    /* USER CODE BEGIN EXTI0_IRQn 0 */
208
209    /* USER CODE END EXTI0_IRQn 0 */
210    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0); //reset the IRQ
211    HAL_UART_Transmit_IT(&huart2, (uint8_t*)transmit, sizeof(transmit)); //Transmit the transmit variable from USART2 when we push the button.
212    /* USER CODE BEGIN EXTI0_IRQn 1 */
213
214    /* USER CODE END EXTI0_IRQn 1 */
215  }
216  /**
217    * @brief This function handles USART1 global interrupt.
218    */
219  void USART1_IRQHandler(void)
220  {
221    /* USER CODE BEGIN USART1_IRQn 0 */
222
223    /* USER CODE END USART1_IRQn 0 */
224    HAL_UART_IRQHandler(&huart1);
225    /* USER CODE BEGIN USART1_IRQn 1 */
226
227    /* USER CODE END USART1_IRQn 1 */
228  }
229
230  /**
231    * @brief This function handles USART2 global interrupt.
232    */
233  void USART2_IRQHandler(void)
234  {
235    /* USER CODE BEGIN USART2_IRQn 0 */
236
237    /* USER CODE END USART2_IRQn 0 */
238    HAL_UART_IRQHandler(&huart2);
239    /* USER CODE BEGIN USART2_IRQn 1 */
240  HAL_UART_Receive_IT(&huart1, (uint8_t*)receive,sizeof(receive));//Receive the data from UART1
241    /* USER CODE END USART2_IRQn 1 */
242  }
```

Figure 12

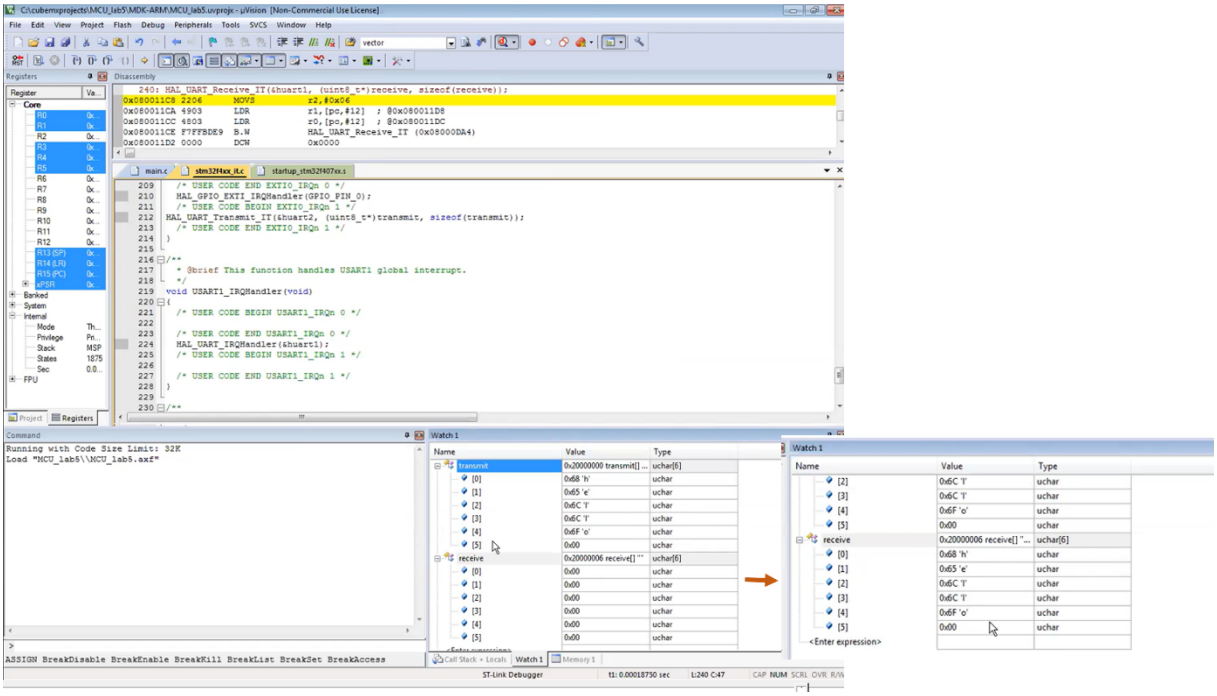

Figure 13

6. You will make the changes within the EXTI0_IRQHandler function. Define tansmit1, transmit2, receive and i variables. Assign “hello” to transmit1 and “world” to transmit2 in char type (Figure 14). Each time the button is pushed, the value of the i variable increases by 1. If it is an even number, transmit1 variable, if odd, transmit 2 variable from USART2 (Figure 15). Let USART2 also receive this data from USART1. Follow the change of variables using the Debug menu (Figure 16).

```
42
43  /* Private variables ----------
44  /* USER CODE BEGIN PV */
45  int i;
46  char transmit1[]="hello";
47  char transmit2[]="world";
48  char receive [6];
49  /* USER CODE END PV */
50
```

Figure 14

```
207 void EXTI0_IRQHandler(void)
208 {
209   /* USER CODE BEGIN EXTI0_IRQn 0 */
210
211   /* USER CODE END EXTI0_IRQn 0 */
212   HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0); //reset the IRQ
213   HAL_Delay(100);
214   i=i+1;
215   if (i%2==0)
216   {
217   HAL_UART_Transmit_IT(&huart2, (uint8_t*)transmit1, sizeof(transmit1)); //Transmit the transmit1 variable from USART2 when we push the button.
218   }
219   else
220   {
221   HAL_UART_Transmit_IT(&huart2, (uint8_t*)transmit2, sizeof(transmit2)); //Transmit the transmit2 variable from USART2 when we push the button.
222   }
223   /* USER CODE BEGIN EXTI0_IRQn 1 */
```

Figure 15

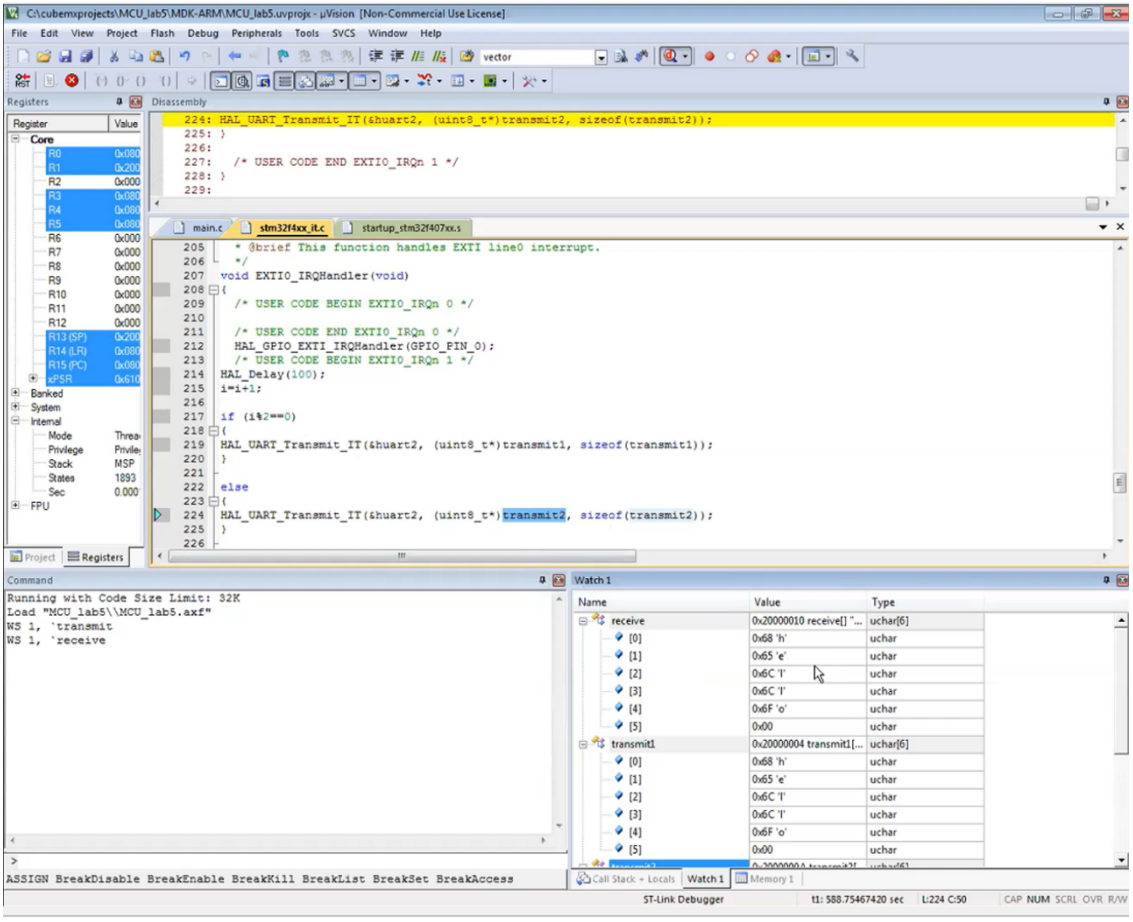
C:\cubemxprojects\MCU_lab5\MDK-ARM\MCU_lab5.uvprojx - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Register
Value
Core
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)
xPSR
Banked
System
Internal
Mode
Privilege
Stack
States
Sec
FPU
Project
Registers
Disassembly
224: HAL_UART_Transmit_IT(&huart2, (uint8_t*)transmit2, sizeof(transmit2));
225: }
226:
227: /* USER CODE END EXTI0_IRQn 1 */
228: }
229:
main.c
stm32f4xx_it.c
startup_stm32f407xx.s
205 * @brief This function handles EXTI line0 interrupt.
206 */
207 void EXTI0_IRQHandler(void)
208 {
209 /* USER CODE BEGIN EXTI0_IRQn 0 */
210
211 /* USER CODE END EXTI0_IRQn 0 */
212 HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
213 /* USER CODE BEGIN EXTI0_IRQn 1 */
214 HAL_Delay(100);
215 i=i+1;
216
217 if (i%2==0)
218 {
219 HAL_UART_Transmit_IT(&huart2, (uint8_t*)transmit1, sizeof(transmit1));
220 }
221
222 else
223 {
224 HAL_UART_Transmit_IT(&huart2, (uint8_t*)transmit2, sizeof(transmit2));
225 }
226
Command
Running with Code Size Limit: 32K
Load "MCU_lab5\\MCU_lab5.axf"
WS 1, `transmit
WS 1, `receive
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
Watch 1
Name
Value
Type
receive
0x20000010 receive[] "...
uchar[6]
[0]
0x68 'h'
uchar
[1]
0x65 'e'
uchar
[2]
0x6C 'l'
uchar
[3]
0x6C 'l'
uchar
[4]
0x6F 'o'
uchar
[5]
0x00
uchar
transmit1
0x20000004 transmit1[...
uchar[6]
[0]
0x68 'h'
uchar
[1]
0x65 'e'
uchar
[2]
0x6C 'l'
uchar
[3]
0x6C 'l'
uchar
[4]
0x6F 'o'
uchar
[5]
0x00
uchar
Call Stack + Locals
Watch 1
Memory 1
ST-Link Debugger
t1: 588.75467420 sec
L:224 C:50
CAP NUM SCRL OVR R/W

Figure 16