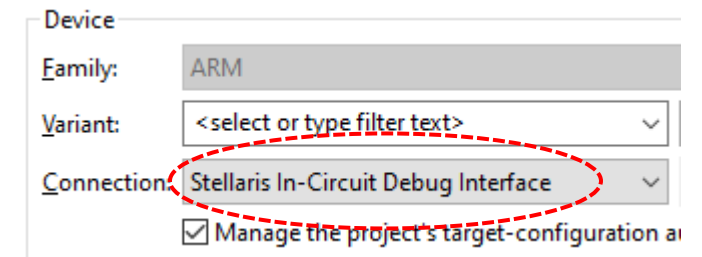


Debugging

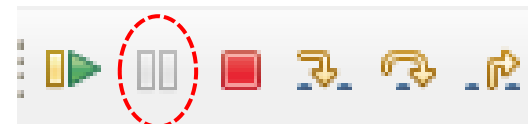
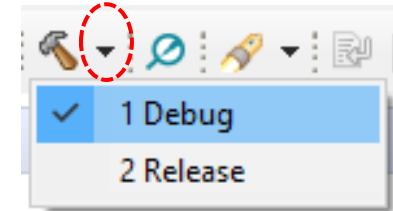
Debugging

- Debugging while the code is running on the actual device
- Stellaris ICDI – in-circuit debug interface
- Make sure Stellaris ICDI is selected for your project
 - Project -> Properties -> CCS General



Debugging (continued)

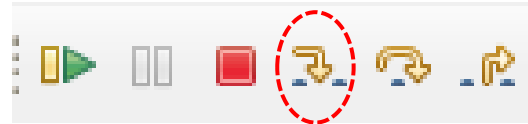
- Build Debug version
- Start the Debugger (enter into debug mode)
- Start running the code in the debugger
- Pause (only possible when running)
- Stop



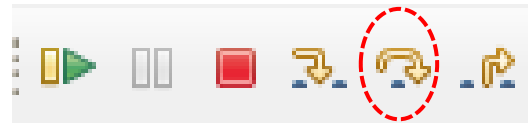
Debugging (continued)

- Step by step debugging

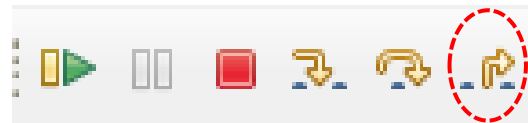
- Step into (function)



- Step over line



- Step out of (function)



- Stop

Debugging (continued)

- Switch between Editor and Debug modes (upper right corner)



Debugging (continued)

workspace_v8 - Lab2/main_lab2.c - Code Composer Studio

File Edit View Project Tools Run Scripts Window Help

Debug

Lab2 [Code Composer Studio - Device Debugging]
Assignment01 [Code Composer Studio - Device Debugging]
Lab2 [Code Composer Studio - Device Debugging]
Stellaris In-Circuit Debug Interface/CORTEX_M4_0 (Suspended)
main() at main_lab2.c:20 0x0000033C
ResetISR() at tm4c123gh6pm_startup_ccs_gcc.c:296 0x00000432
0x00000000 (no symbols are defined)

Variables

Name	Type	Value	Location
dummy	int	0	0x2000062C
mask	unsigned short	1	0x20000636
var_task_A	unsigned short	65535	0x20000634
var_task_B	unsigned short	65535	0x20000632
var_task_C	unsigned short	43690	0x20000630

Expressions Registers

main_lab2.c main.c systick.c gpio.c countled.c counter.c button.c main_a1_g7.c main_a1_g21.c main.c

```
5 /*
6
7 #include <stdint.h>
8 #include "tm4c123gh6pm.h"
9 #include "emp_type.h"
10
11
12 int main(void)
13 {
14     INT16U var_task_A = 0xFFFF;
15     INT16U var_task_B = 0xFFFF;
16     INT16U var_task_C = 0xAAAA;
17     INT16U mask = 0x0001;
18
19     // Task A
20     var_task_A &= ~(0x60);
21
22     // Task B
23     var_task_B = ((var_task_B & ~(0x07E0)) | 0x03C0);
```

Code execution is currently here

Console

Lab2
CORTEX_M4_0: GEL Output:
Memory Map Initialization Complete

Writable Smart Insert 10:1 Launching Assignment01: (61%) LE

Debugging (continued)

Click on arrow to get register values

Click to get registers

Code execution is currently here

The screenshot shows the Code Composer Studio interface during a debug session. The main window displays the source code of `main_lab2.c`. The current execution point is at line 45, which is highlighted in green. The right-hand pane shows the 'Registers' window, which lists several GPIO registers and their values. The bottom pane shows the console output, which includes the text 'Lab2' and 'CORTEX_M4_0: GEL Output: Memory Map Initialization Complete'.

Name	Value	Description
GPIO_PORTF		GPIO register offsets
GPIO_DATA	0x00000016	GPIO Data [Memory Mapped]
GPIO_DIR	0x0000000E	GPIO Direction [Memory Mapped]
GPIO_IS	0x00000000	GPIO Interrupt Sense [Memory Mapped]
GPIO_IBE	0x00000000	GPIO Interrupt Both Edges [Memory Mapped]

```
35     if(GPIO_PORTF_DATA_R & 0x10)           // switch not pressed
36     {
37         GPIO_PORTF_DATA_R &= ~(0x02);      // red LED off
38         GPIO_PORTF_DATA_R |= 0x04;         // blue LED on
39     }
40     else                                     // switch pressed
41     {
42         if(var_task_C & mask)               // is the masked bit a 1?
43         {
44             GPIO_PORTF_DATA_R |= 0x02;      // red LED on
45             GPIO_PORTF_DATA_R &= ~(0x04);   // blue LED off
46         }
47         mask=mask<<1;                       // shift masking bit by 1
48         while(!(GPIO_PORTF_DATA_R & 0x10)) // wait here until the switch is released
49         {
50         }
51     }
52 }
53 return 0;
```

Lab2
CORTEX_M4_0: GEL Output:
Memory Map Initialization Complete

Launching Assignment01: (61%)

Debugging - breakpoints


Double click here to create a breakpoint

```
35     if(GPIO_PORTF_DATA_R & 0x10)           // switch not pressed
36     {
37         GPIO_PORTF_DATA_R &= ~(0x02);      // red LED off
38         GPIO_PORTF_DATA_R |= 0x04;         // blue LED on
39     }
40     else                                     // switch pressed
41     {
42         if(var_task_C & mask)               // is the masked bit a 1?
43         {
44             GPIO_PORTF_DATA_R |= 0x02;      // red LED on
45             GPIO_PORTF_DATA_R &= ~(0x04);   // blue LED off
46         }
47         mask=mask<<1;                      // shift masking bit by 1
48         while(!(GPIO_PORTF_DATA_R & 0x10)) // wait here until the switch is released
49         {
50             //
51         }
52     }
53     return 0;
```

Breakpoint created

```
35     if(GPIO_PORTF_DATA_R & 0x10)           // switch not pressed
36     {
37         GPIO_PORTF_DATA_R &= ~(0x02);      // red LED off
38         GPIO_PORTF_DATA_R |= 0x04;         // blue LED on
39     }
40     else                                     // switch pressed
41     {
42         if(var_task_C & mask)               // is the masked bit a 1?
43         {
44             GPIO_PORTF_DATA_R |= 0x02;      // red LED on
45             GPIO_PORTF_DATA_R &= ~(0x04);   // blue LED off
46         }
47         mask=mask<<1;                      // shift masking bit by 1
48         while(!(GPIO_PORTF_DATA_R & 0x10)) // wait here until the switch is released
49         {
50             //
51         }
52     }
53     return 0;
```


Debugging - breakpoints

- Now when you resume debug execution  , it will automatically stop once it hits the defined breakpoint
- Try it out