

Testing

REGISTERS AND PERIPHERALS

Objectives

At the end of this section, you will be able to

- Use register contents as part of expected pass/fail testing outcomes

testIDEA

Testing

REGISTERS AND PERIPHERALS

1	Testing the MCU's peripherals	3-4
2	Testing peripherals	5-7
	Navigate to register	5
	Watch register's memory	6
	Test cases	7
3	Summary	8-9

testIDEA

1 TESTING THE MCU'S PERIPHERALS

In this unit we will examine how to use the contents of the microcontroller's Special Function Registers (SFRs) as the *Expected* value of a testIDEA test case.

Such tests could be considered to digress towards Integration Testing rather than pure Unit Testing. However, in the world of embedded development and working close to the hardware, such testing is required, often when testing the functionality of peripheral drivers.

Here we will test that the function *digitalWrite()* works correctly, the function we have been using to turn a specified digital GPIO pin on or off.

```
void loop() {  
  
    digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)  
    delay(500);                // wait for a second  
    digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW  
    delay(500);                // wait for a second  
}
```

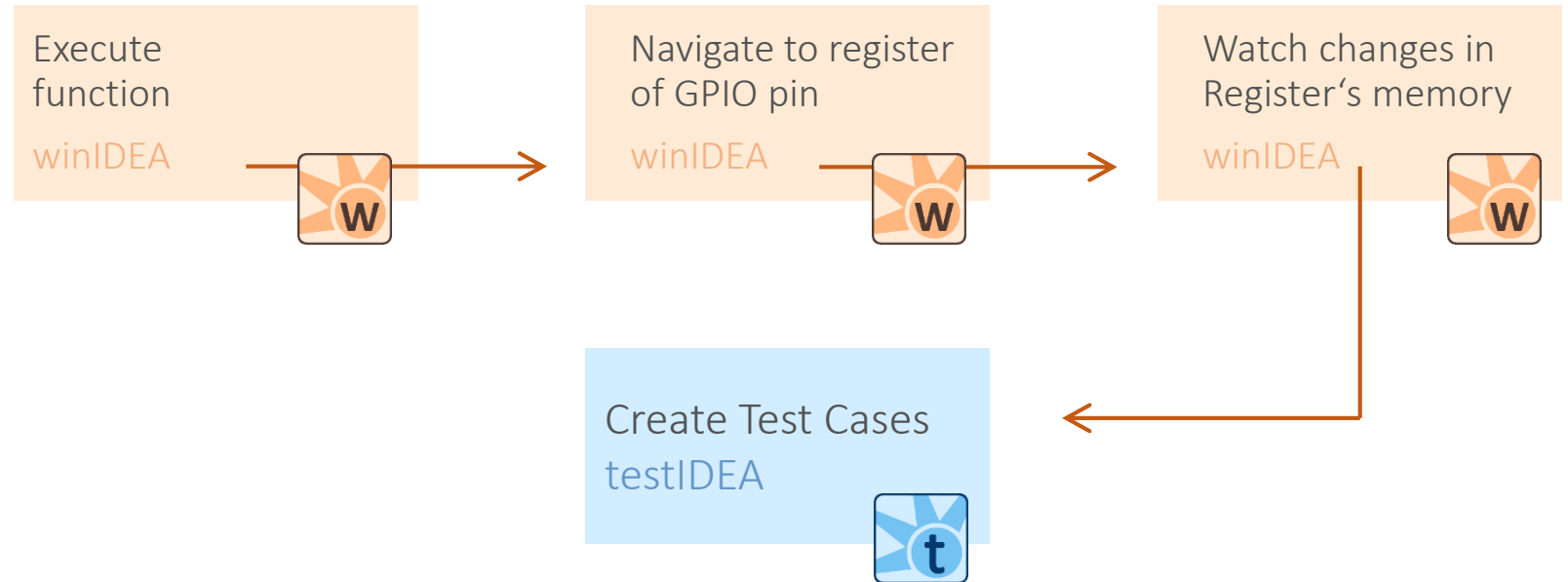
1 TESTING THE MCU'S PERIPHERALS



One potential method would be to create a hardware-in-the-loop (HIL) platform and check that the associated GPIO pin actually acquires the expected value. This would, however, require the GPIO to be connected to an IOM input of the BlueBox™.

Instead we will execute the function and then check that the register used to control the GPIO pin has been changed appropriately.

Concept



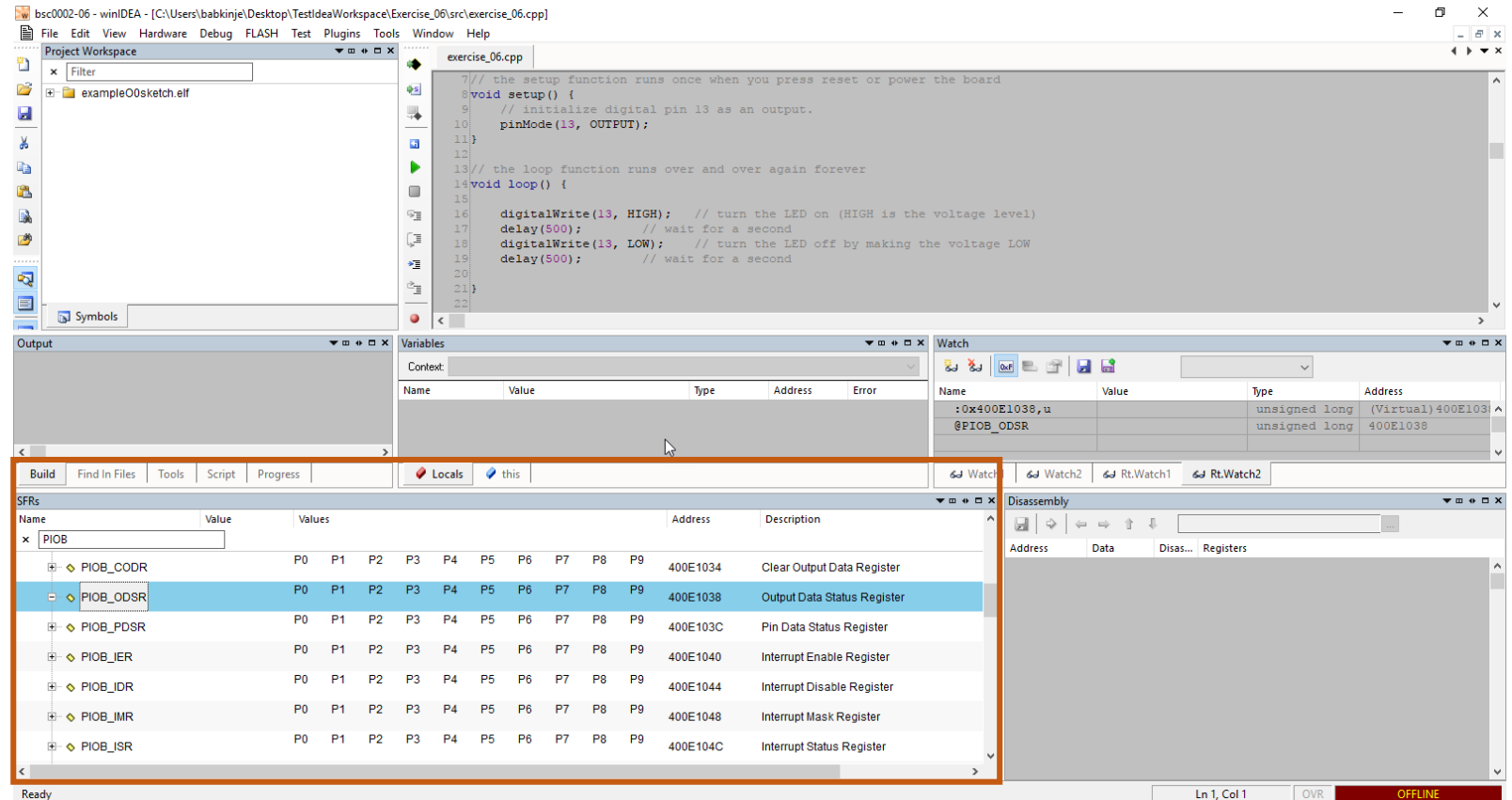
2 TESTING PERIPHERALS - NAVIGATE TO REGISTER



In the winIDEA workspace, one way to examine the status of the GPIO pin that controls the LED is using the Special Function Registers (SFR) window.

The LED is attached to a pin under the control of register PIOB_ODSR, as shown in the screenshot opposite.

From the SFR window it is possible to determine the address of this register in memory, namely 0x400E1038.



2 TESTING PERIPHERALS - WATCH REGISTER'S MEMORY



The address of the register can be added to the Real-time Watch window. You will see the value updating as the code executes. The register can be accessed in two ways:

`:0x400E1038,u` – the address and data type (updates in “real time”)

`@PIOB_ODSR` – the name of the register associated with the pin being controller (doesn't update in real time by default – see following slide)



We see that when the LED is on, `PIOB_ODSR == 0x08000000`, and when off it is `0x00000000`.

Name	Value	Type	Address
:0x400E1038,u	0x08000000	unsigned long	(Virtual) 400E1038
@PIOB_ODSR	0x08000000	unsigned long	400E1038

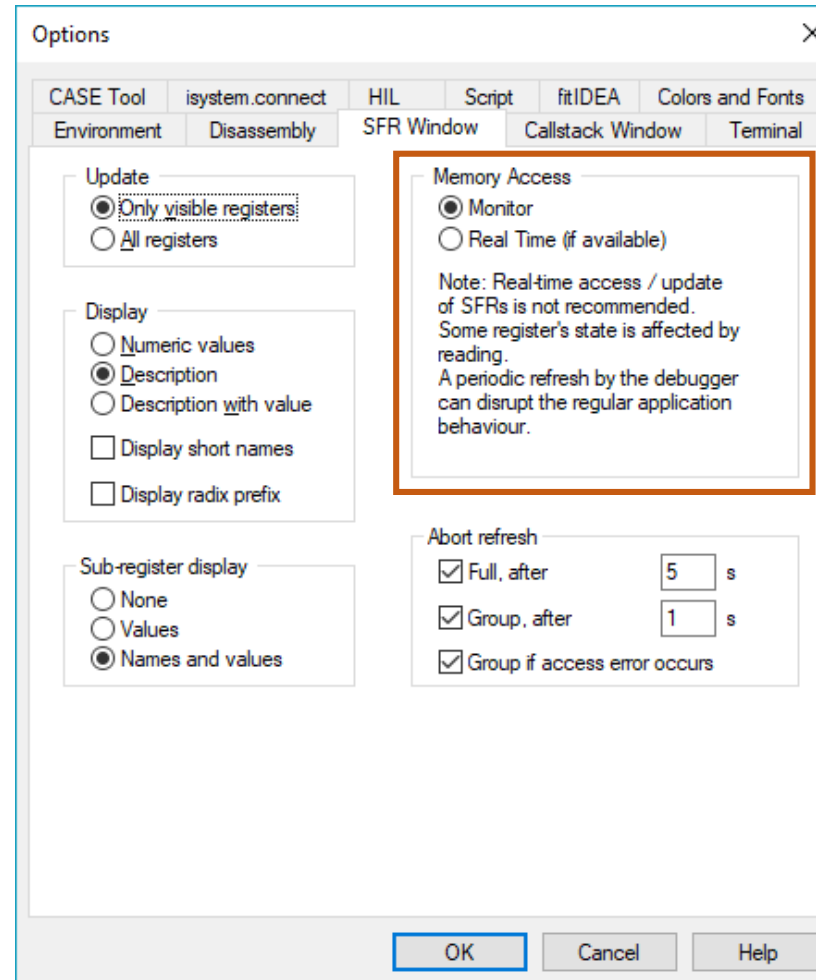


Be careful when adding registers in the “Real-time Watch” window as sometimes the mere act of reading/writing the register can cause its contents to change, such as a register that clears its bits after having been read.

2 TESTING PERIPHERALS - WATCH REGISTER'S MEMORY



This default functionality regarding real time updates for SFRs can be changed if required. Simply open the Tools -> Options dialog and change the *Memory Access* option to *Real Time*.



2 TESTING PERIPHERALS - TEST CASES



Testing the *digitalWrite()* function in testIDEA is quite straightforward. Simply create a test that calls the function with the required parameters (pin number 13 and '0' or '1' for the logic level desired).

In the “*Expected*” field we simply mask for the desired bit in the target register and compare the outcome with the expected outcome for the test to be considered to have passed.

This looks as shown opposite (testIDEA view is in *Table* mode).

?	func					assert			
	func	params				expressions			
		0	1			0			
0	<i>i</i> digitalWrite	<i>i</i> 13	<i>i</i> 1	<i>i</i>	<i>i</i>	<i>i</i>	<i>i</i>	(@PIOB_ODSR & 0x08000000) == 0x08000000	
1	<i>i</i> digitalWrite	<i>i</i> 13	<i>i</i> 0	<i>i</i>	<i>i</i>	<i>i</i>	<i>i</i>	(@PIOB_ODSR & 0x08000000) == 0	
	+								

SUMMARY

testIDEA

- testIDEA can also use the contents of registers, rather than variables, for its *Expected* values.
- Tests pass or fail dependent on the results found in the peripheral registers.
- Approach can be used to develop tests for microcontroller peripheral drivers.

