Lab 7 Timer Generated ISRs

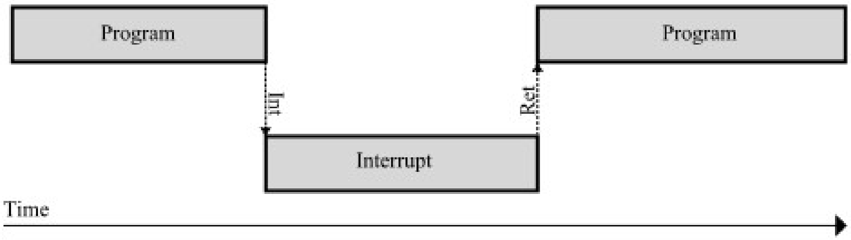


Image from wikibooks.org

Section 1 Lab Objectives

When this lab exercise is completed, the student should be able to use microprocessor resources to:

1. Create a timer that triggers an interrupt.
2. Create an interrupt service routine (ISR) to stop the timer.
3. Create two tasks to control independent LEDs.

Section 2 PreLaboratory Preparation

Prior to your scheduled laboratory meeting time the following items need to be completed.

On Line Learning

1. Watch *Getting Started with STM32 and Nucleo Part 6: Timers and Timer Interrupts* `https://www.youtube.com/watch?v=VfbW6nfG4kw`
2. Watch *STM32 Guide #4: Generated Code, HAL, and Bare Metal* `https://www.youtube.com/watch?v=txnViYePocg`
3. Watch *Lecture 5: Memory Mapped I/O* `https://www.youtube.com/watch?v=aT5XMOrid7Y`

Preparation for the Prelab Quiz

The quiz will be available in lab for the first ten (10) minutes of your laboratory session. You may use any of your prelab preparation as a reference while taking the quiz.

Section 3 Timer Control

Create code for your Nucleo-L476RG that:

1. Setups up timer 16 to generate an interrupt at 10Hz.
2. Starts timer 16 (TIM16) in time base interrupt mode.
3. Toggles an LED each time the interrupt is generated.

   > The LED will flash at 5Hz because it toggles every time the interrupt is generated.

4. Stops timer 16 (TIM16) ten seconds later.

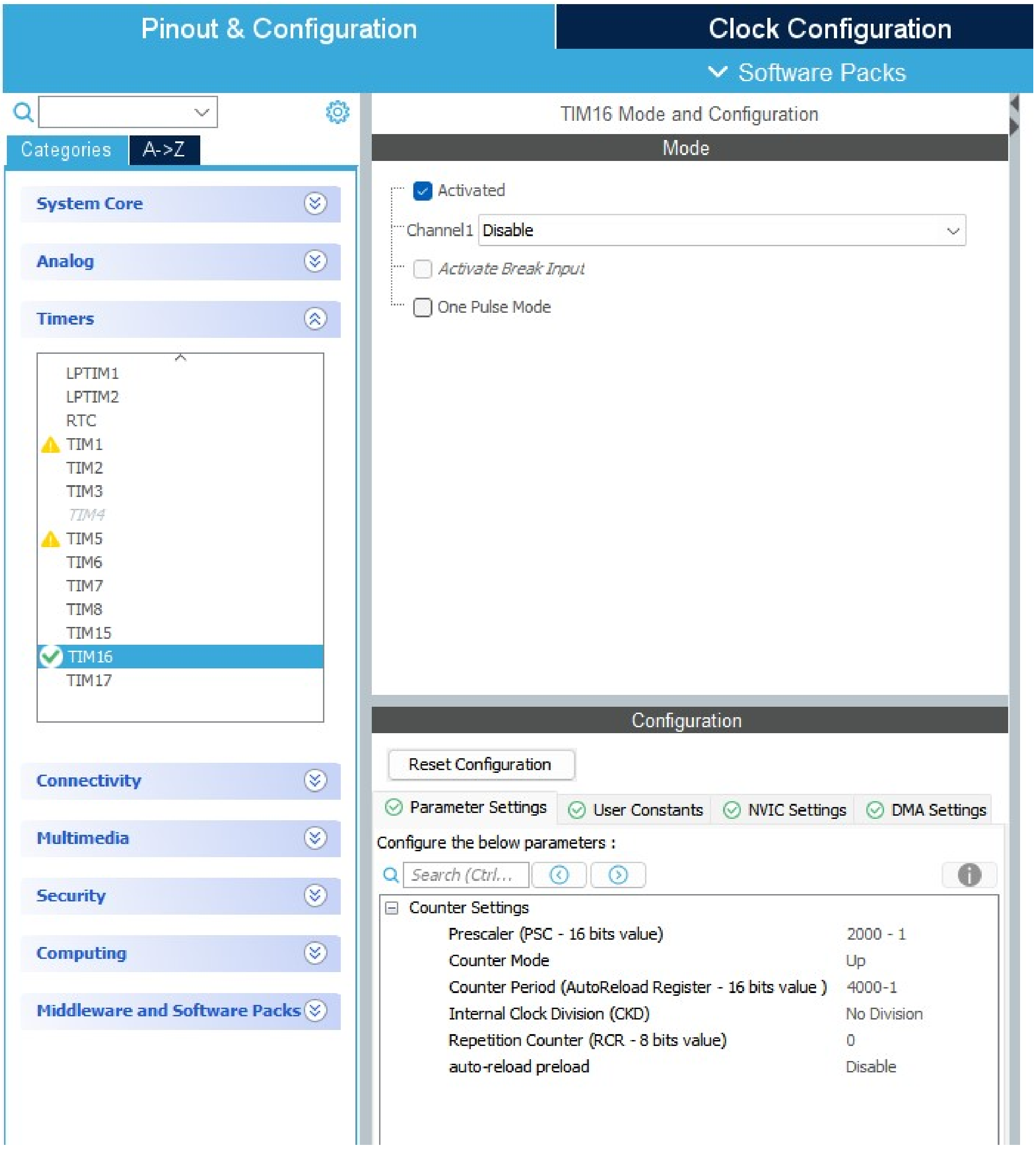
> When using RTOS remember to change the (HAL) timebase source to one of the unused timers. The system will create a callback function for this timer. You can add code to this callback function that responds to TIM16.

> The compiler is unaware of variables being updated in callback functions due to interrupts. To prevent them from being optimized out, put the keyword volatile before the variable type.

5. Uses a task to flash another LED at 1Hz.
6. Uses a task to flash another LED at 0.5Hz.
7. Get a sign-off by showing the three blinking LEDs.

Section 4 CubeMX, HAL and Bare-Metal

This section will take a closer look at the code that was written in Section 3.



☑ Activated

Channel1 Disable

□ *Activate Break Input*

□ One Pulse Mode

Configure the below parameters :

Counter Settings	
Prescaler (PSC - 16 bits value)	2000 - 1
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	4000-1
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Disable

Figure 1: Sample TIM16 IOC Configuration

Section 4.1 STM32CubeMX

1. In main.c find the function call that initializes timer 16 called MX_TIM16_Init().

2. Select the function call and press F3. The IDE will locate and display the function.

3. The Prescaler (PSC - 16 bit value) setting in Figure 1 is loaded into the Prescaler member of htim16's embedded struct Init. What member is used to assign the "Counter Period (AutoReload Register - 16 bit value)" in Figure 1?

   htim16.Init.________________

4. In main.c find the function call that initializes the general purpose inputs and outputs called MX_GPIO_Init().

5. What member is used to assign the built in LED (LD2_Pin)?

   GPIO_InitStruct.________________

STM32CubeMX creates the graphical environment that generates the IOC file.

Section 4.2 HAL and Bare-Metal

1. Find RM0351, the reference manual for the STM32L4xx series.

2. Search RM0351 for Memory Organization and use section 2.2s memory map diagram to determine the bottom address for the Peripherals. It is best to search and find the diagram yourself, but if you are confused, the memory map is provided in Figure 2.

   0x_______________

3. Search RM0351 for GPIOx_ODR and GPIOx_BSRR. It is best to search and find the register descriptions yourself, but if you are confused, the descriptions are provided in Figure 3.

4. Go to the HAL_GPIO_TogglePin command in task 1. Right-click on the line to Toggle a break point.

5. In main.c, go to the interrupt callback function and comment out all calls to HAL_GPIO_Toggle() or HAL_GPIO_Write(). The interrupt will override the task and make it difficult to analyze.

6. Run in debug mode. When the debugger stops at your task, step into the HAL_GPIO_TogglePin() command.

7. Continue to step, reading the ODR. Hover over each variable to get its value.

8. Use the space below and these values to predict the write value to BSRR.

Fill in values:

ODR = 0x_____________________________________(binary and decimal)

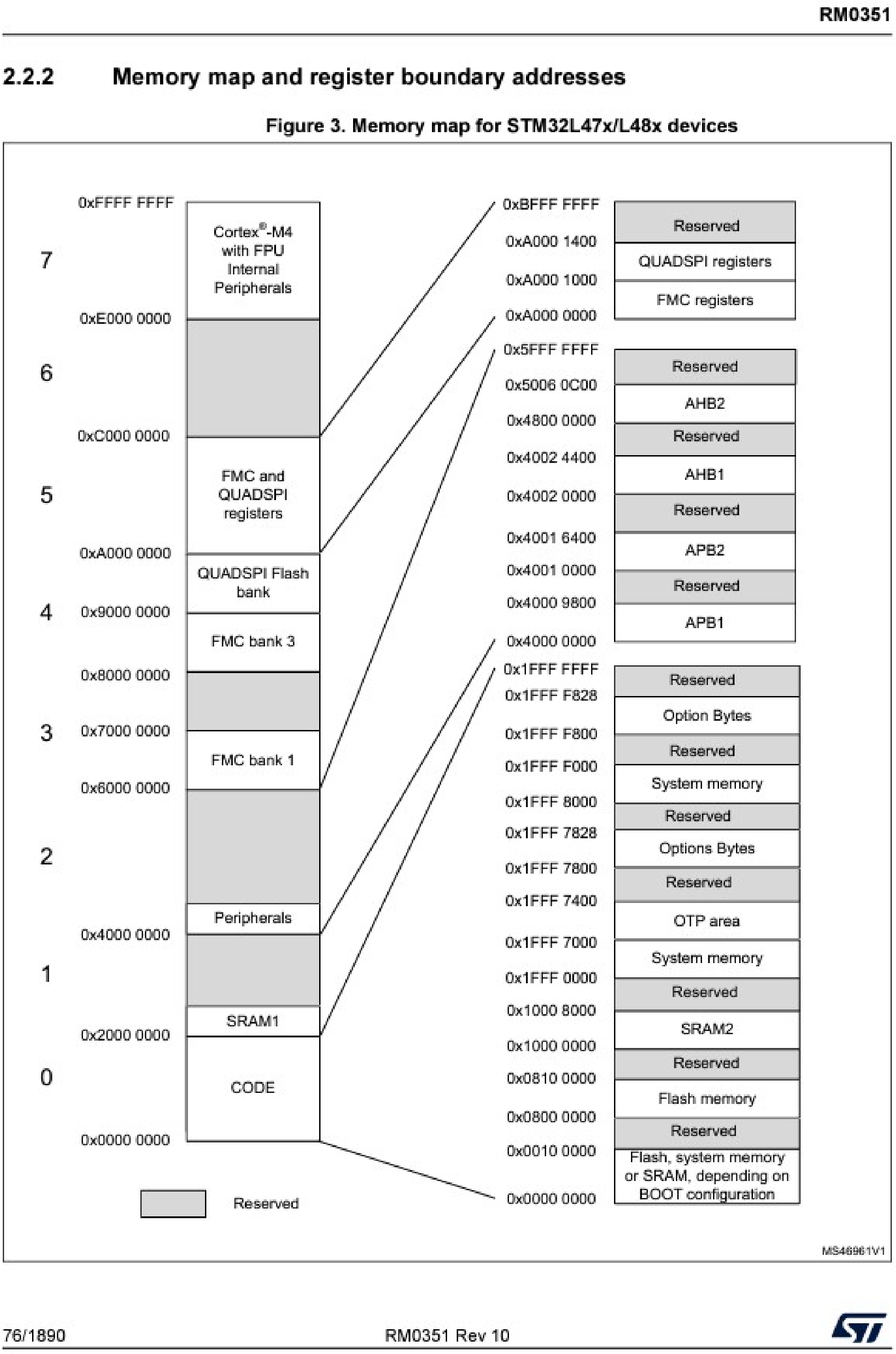
Using the ODR, which outputs are high? For example, a value of 32 would mean OD5 is on (0000 0000 0010 0000 = 32). In port A this is PA5, the on-board LED.

GPIO_Pin = _____________________________________ (binary and decimal)

Using the ODR, which pin is this?

GPIO_NUMBER = ______________________________ (binary and decimal)

32 bit (binary) GPIOx-¿BSRR = ______________________________



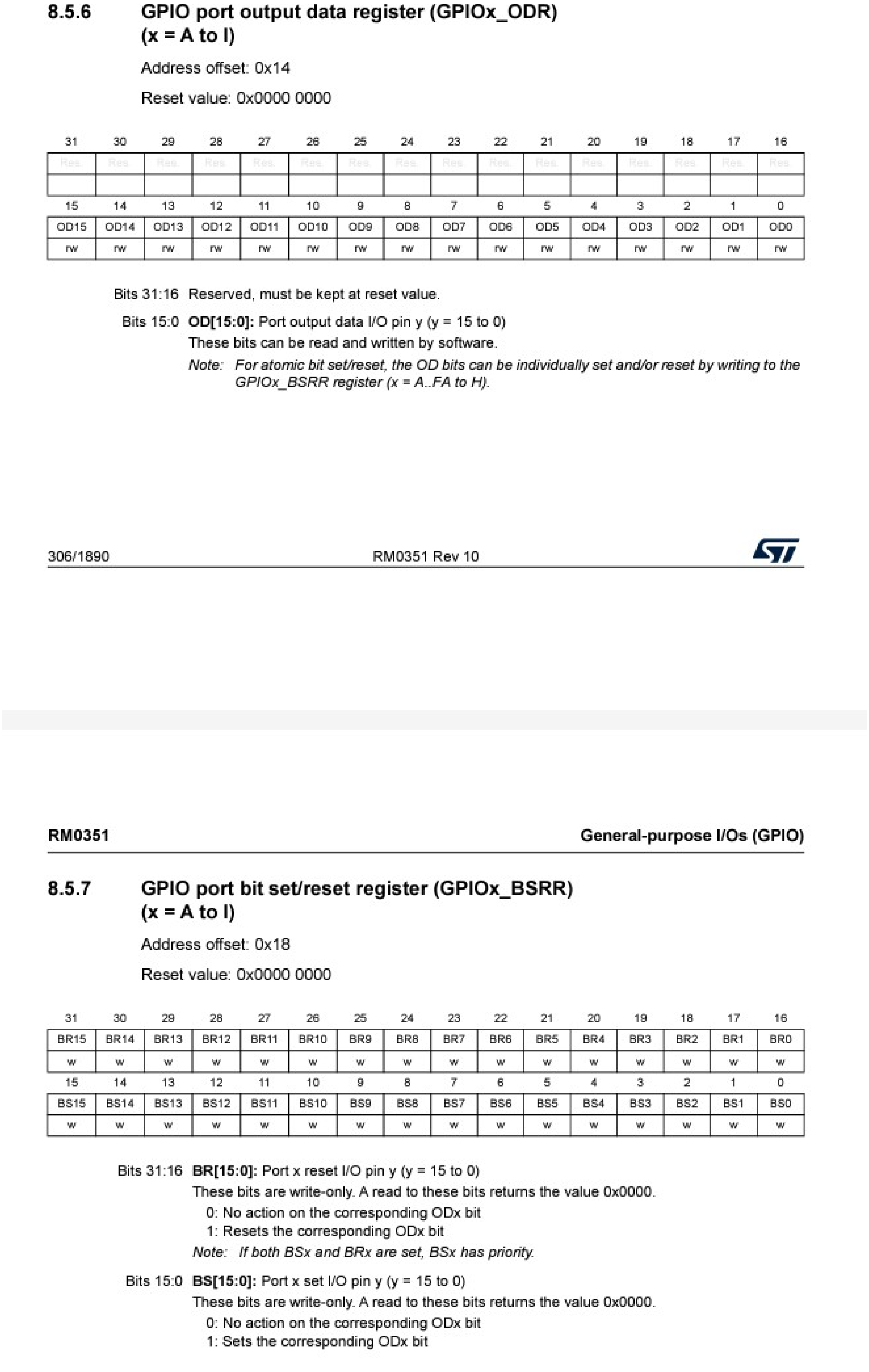
RM0351

2.2.2 Memory map and register boundary addresses

Figure 3. Memory map for STM32L47x/L48x devices

76/1890 RM0351 Rev 10

Figure 2: Memory Map



8.5.6 GPIO port output data register (GPIOx_ODR) (x = A to I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OD[15:0]:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..FA to H).

306/1890 RM0351 Rev 10

RM0351 General-purpose I/Os (GPIO)

8.5.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit
1: Resets the corresponding ODx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODx bit
1: Sets the corresponding ODx bit

Figure 3: GPIO Registers ODR and BSRR

Section 5 Sign-offs

Name:__________________________________

____ ____________________

Section 3: 3 LEDs. 1 interrupt, 2 RTOS Task controlled. Date .

____ ____________________

Section 4: List of members, peripheral address and GPIO example. Date .