

Lab 2: Timer and PWM

Quang Hai Nguyen

Revision 0.99, 07.01.2020

SAMD21 workshop

©2017 by ARROW

All rights reserved. No part of this manual shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, desktop publishing, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this document, the publisher and author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein. All terms mentioned in this manual that are known to be trademarks or service marks are listed below. In addition, terms suspected of being trademarks or service marks have been appropriately capitalized. ARROW cannot attest to the accuracy of this information. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Revision History

Revision, Date	Editor	Subject (major changes)
Revision 0.99, 07.01.2020	Quang Hai Nguyen	Preliminary

Table of Contents





Revision History	3
Table of Contents	4
List of Figures.....	5
List of Icon Identifiers.....	6
Prerequisite	7
Hardware	7
Software	7
Labs description	7
Description.....	7
Assignment.....	7
Lab procedure	8
Setup the timer and the PWM with Atmel Start	8
Writing application.....	11
Contact information	14

List of Figures

No table of figures entries found.

List of Icon Identifiers

Table 1: Icon Identifiers List

-  Extra information about a topic
-  Task needs to be done
-  Important information or a warning
-  Result expected to see

Prerequisite

Hardware

- Laptop or PC with Windows 7 or Windows 10
- A SAMD21-Xplained-Pro evaluation board (provided during the workshop)
- Extension boards, LEDs, Potential meter, buttons and jumper wires (provided during the workshop)
- A micro USB cable (provided during the workshop)
- Breadboard (provided during the workshop)
- **Internet access without proxy or firewall**

Software

- Terminal program (e.g. TeraTerm)
- Atmel Studio 7 ([link](#))

Labs description

Description

This lab will show you how to configure the timer and the PWM with Atmel Start. We will program the PWM to create a heartbeat LED and the timer to print a message every second on the terminal.

Assignment

1. Setup the timer and the PWM with Atmel Start
2. Writing application



Some of the basic configuration steps are skipped in this lab so make sure you have already started lab 1 to learn the basic steps before continuing with this lab.

Lab procedure

Setup the timer and the PWM with Atmel Start

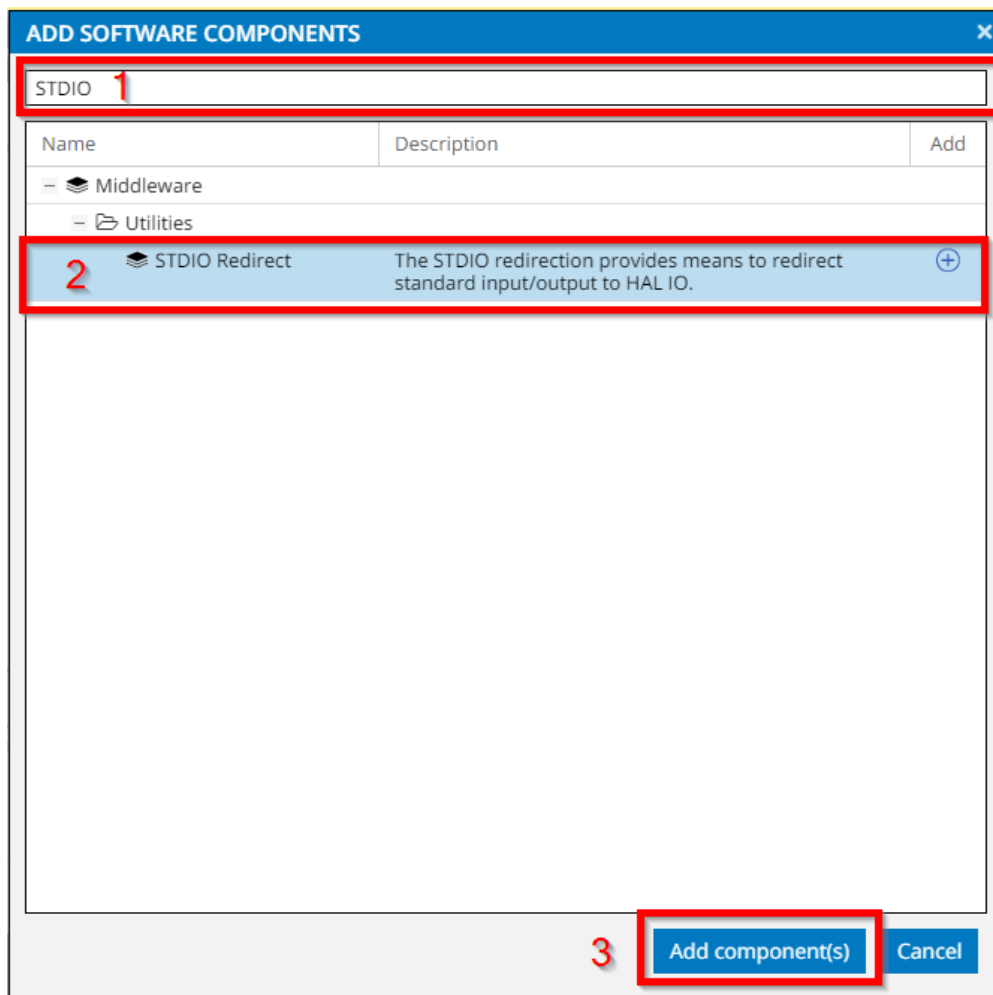
In Atmel Studio, create a new Atmel Start project.

When it is prompted, choose the correct device for your project. In this case, it is SAM D21 Xplained Pro.

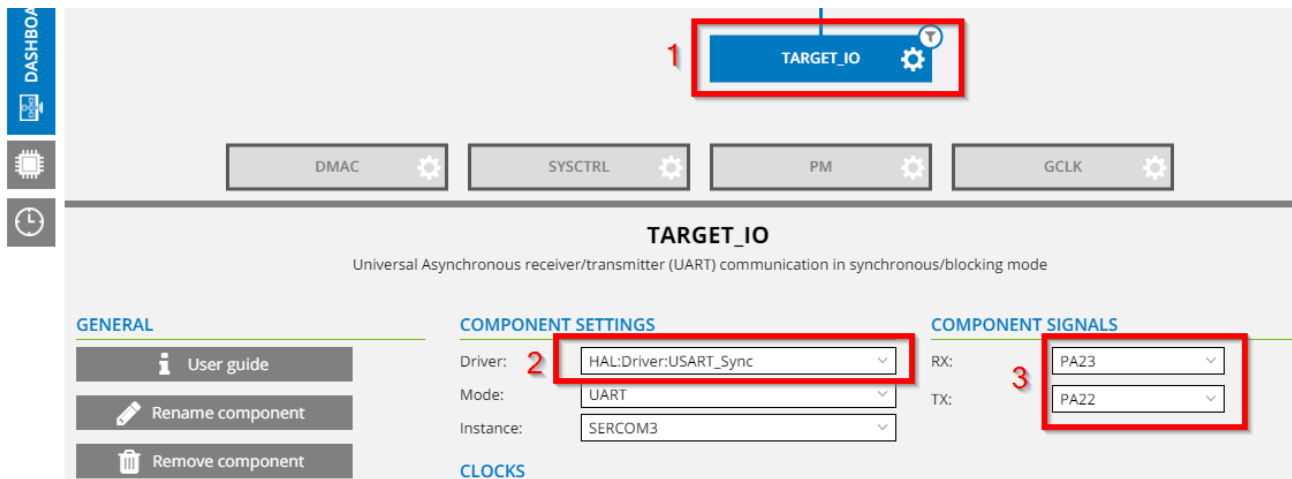
Give your project a name. For my case, I name it SAMD21_Lab3.

For this project, we need three components: a timer, a PWM and STDIO to send the debug messages to the terminal. We will start setting up the STDIO.

Click “Add software component”. In the prompted window, enter “STDIO” into the filter text box and select “STDIO Redirect”

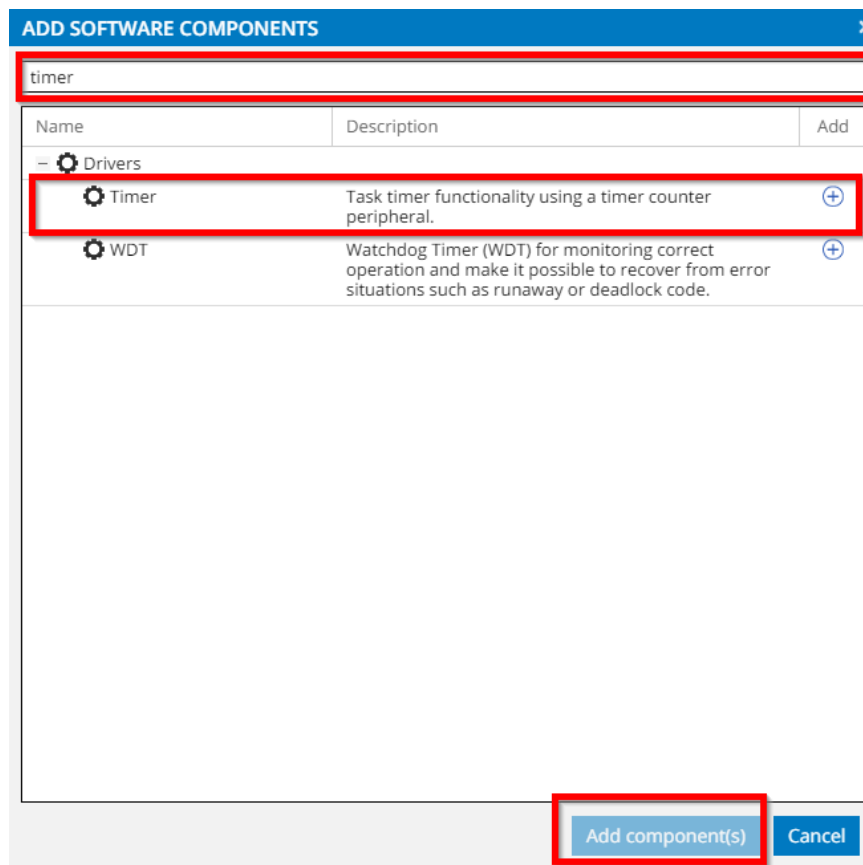


Click “TARGET_IO” and set the correct pin for the UART bus. From the “Board user guide”, we know that pin PA22 and PA23 are routed to output pins of USB Virtual Com. Therefore, they are configured as follows:



We will keep other configurations as they are.

Next, we will add a timer component. Click “Add software component”, enter timer in the filter text box, choose “Timer” under the Drivers category.



For the timer component, we want to use timer instance TC3 and the length of one-timer tick is 1ms. Therefore, we set it as follows:

TIMER_0
Task timer functionality using a TC peripheral

GENERAL

User guide

Rename component

Remove component

COMPONENT SETTINGS

Driver: HAL:Driver:Timer

Instance: TC3

CLOCKS

TC: Generic clock generator 0 (1 MHz)

COMPONENT SIGNALS

WO/0: ----

WO/1: ----

HAL:DRIVER:TIMER (TIMER) CONFIGURATION ON TC3

BASIC CONFIGURATION

Prescaler: Divide by 8

Length of one timer tick in uS: 1000

EVENT CONTROL

Enable: ☐

ADVANCED CONFIGURATION

Enable: ☐

After that, we add a PWM component to the project with the same procedure.

ADD SOFTWARE COMPONENTS

PWM 1

Name	Description	Add
+ Middleware		
- Drivers		
PWM 2	Pulse-width modulation (PWM) to create an analog behavior digitally by controlling the amount of power transferred to the connected peripheral.	+

3 Add component(s) Cancel

For the PWM, we want to use the time instance TCC0 and output the signal on PB30, which is connected to LED0

PWM_0
PWM functionality using a TCC peripheral

GENERAL	COMPONENT SETTINGS	COMPONENT SIGNALS
<div>User guide</div> <div>Rename component</div> <div>Remove component</div>	Driver: HAL:Driver:PWM Instance: TCC0 CLOCK: Generic clock generator 0 (1 MHz)	WO/0: PB30 WO/1: ---- WO/2: ---- WO/3: ---- WO/4: ---- WO/5: ---- WO/6: ---- WO/7: ----

We also want to have the PWM working as single slope, using Channel 0 for capture and compare:

HAL:DRIVER:PWM (PWM) CONFIGURATION ON TCC0

BASIC SETTINGS	PWM WAVEFORM OUTPUT SETTINGS
TCC0 Prescaler: Divide by 8	TCC0 Waveform Period Value (uS): 0x2710 hex v
	TCC0 Waveform Duty Value (0.1%): 0x1f4 hex v
ADVANCED SETTINGS Run in standby: <input type="checkbox"/> TCC0 Prescaler and Counter Synchronization Selection: Reload or reset counter on next GCLK TCC0 Waveform Generation Selection: Single-slope PWM TCC0 Auto Lock: <input type="checkbox"/> TCC0 Capture Channel 0 Enable: <input type="checkbox"/> TCC0 Capture Channel 1 Enable: <input type="checkbox"/> TCC0 Capture Channel 2 Enable: <input type="checkbox"/> TCC0 Capture Channel 3 Enable: <input type="checkbox"/> TCC0 Lock update: <input checked="" type="checkbox"/> TCC0 Debug Running Mode: <input type="checkbox"/>	TCC0 Waveform Channel Select: 0x0 hex v

The duty cycle and period are not important because we will set them later in the application.

Generate the code.

Writing application

Compile to check the code generation process causes no errors.

To understand how the timer and the PWM works, we have a look at the two files “hal/include/hal_pwm.h” and “hal/include/hal_timer.h”

To make the PWM working, we need two mandatory functions, which are `pwm_set_parameters()` and `pwm_enable()`. Optionally, if we need an interrupt for the application, we will call the function `pwm_register_callback()`.

The `pwm_set_parameters()` function allows us to change the duty cycle and the period of the PWM signal.

The `pwm_enable()` function enables the output signal to the selected pin.

If an interrupt is applied, we apply the function `pwm_register_callback()` which registers a call back function to an event, e.g. error or period-complete.

For the timer, it is required two functions to operate, the `timer_add_task()`, which creates a new timer task, and `timer_start()` function, which starts the timer.

The `timer_add_task()` requires a `timer_task` structure, which specifies the task's interval, the callback function, and the timer mode, to operate.

With the above information, let's go back to the `main.c` file to implement them. Below the include, we declare the variables as follows:

```
uint32_t duty_cycle = 0;
bool fade_in = true;
uint32_t timer_count = 0;
struct timer_task TIMER_0_task1;
```

Next, we add two callback functions, one for the timer to print a message on the terminal and one for the PWM to create heartbeat signals.

```
static void TIMER_0_task1_cb(const struct timer_task *const timer_task)
{
    timer_count++;
    printf("timer count: %d\n", timer_count);
}

static void PWM0_Period_callback(const struct pwm_descriptor *const descr)
{
    if(fade_in){
        duty_cycle = duty_cycle + 10;
        if(duty_cycle == 1000){
            fade_in = false;
        }
    }
    else{
        duty_cycle = duty_cycle - 10;
        if(duty_cycle == 0){
            fade_in = true;
        }
    }

    pwm_set_parameters(&PWM_0, 1000, duty_cycle);
}
```

Finally, we initiate and start the timer and the PWM. In the `main()` function, add the following snippet

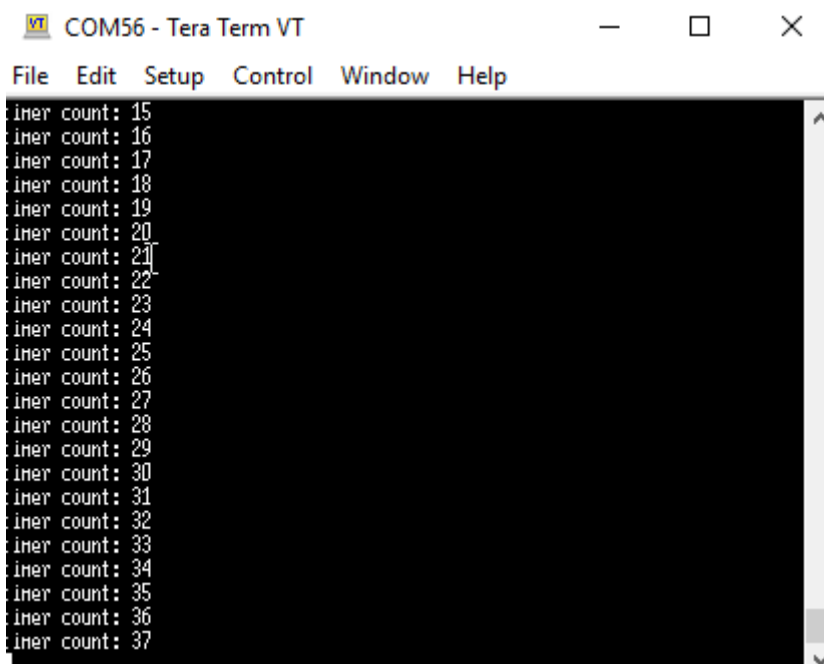
```
printf("SAMD21_Lab2\r\n");

pwm_register_callback(&PWM_0, PWM_PERIOD_CB, PWM0_Period_callback);
pwm_set_parameters(&PWM_0, 1000, duty_cycle);
pwm_enable(&PWM_0);

TIMER_0_task1.interval = 1000;
TIMER_0_task1.cb      = TIMER_0_task1_cb;
TIMER_0_task1.mode    = TIMER_TASK_REPEAT;

timer_add_task(&TIMER_0, &TIMER_0_task1);
timer_start(&TIMER_0);
```

Compile and run the application. On the terminal, you will see the heartbeat signal on LED0 and message appearing on the terminal every one second.



The screenshot shows a terminal window with the title 'COM56 - Tera Term VT'. The menu bar includes 'File', 'Edit', 'Setup', 'Control', 'Window', and 'Help'. The terminal output displays a series of messages: 'iner count: 15', 'iner count: 16', 'iner count: 17', 'iner count: 18', 'iner count: 19', 'iner count: 20', 'iner count: 21', 'iner count: 22', 'iner count: 23', 'iner count: 24', 'iner count: 25', 'iner count: 26', 'iner count: 27', 'iner count: 28', 'iner count: 29', 'iner count: 30', 'iner count: 31', 'iner count: 32', 'iner count: 33', 'iner count: 34', 'iner count: 35', 'iner count: 36', and 'iner count: 37'. A vertical scrollbar is visible on the right side of the terminal window.

Congratulation! You have finished lab 4.

Contact information

Quang Hai Nguyen, B. Eng.
Field Application Engineer

P +49 6102 50308228
M +49 1511 6242003
quanghai.nguyen@arrow.com

Arrow Central Europe GmbH
Frankfurter Straße, 211
63263 Neu-Isenburg

THE END