| **VIDEO P6-2-2a-PWM Version 2** | | |
| --- | --- | --- |
| Cell | **Visuals** | **Audio** |
| 1 | TEXT ON SCREEN:  TCPWM:  **T**imer  **C**ounter  **PWM** | Welcome back to Cypress Academy, PSoC 6 101. In this video, I will show you how to setup the PSoC PWMs - which we call TCPWMs - to control the servos in the robotic arm for our BLE-controlled robotic arm project. TCPWM stands for Timer, Counter, PWM because the blocks can be used for any of those functions. In this video, we'll just use them as PWMs. Let’s get started. |
| 2 | GRAPHIC:  Show Graphic1 from below. Maybe we want to find a way to animate this? Perhaps have the pulse width vary over time? | First, if you have never used a PWM let me tell you a little bit about them. They are actually really simple little devices. PWMs have in input clock and one digital output signal. Inside of them there is a counter and two registers. When you clock the PWM the counter counts up. When the value in the counter reaches the value in the first register – often called the compare register - the output state toggles… then the counter keeps counting until the value reaches the value in the 2nd register (which is called the period) at which point the output toggles again.  There are a billion little variants of the scheme above… counters can count down… multiple outputs… output starts high or low… dead bands, etc… but that is the basic idea. |
| 3 |  | Inside of the PSoC 6 there are two banks of PWMs. The first bank has 24 PWMs that have 16-bit counters. The second bank has 8 PWMs with 32-bit counters. You can read more about this in chapter 27 of the PSoC 6 technical reference manual. |
| 4 |  | Allrighty… Let’s make a new project to understand how the TCPWM component works. |
| 5 | SCREEN CAPTURE:  P6-2-2a-PWM\_capture1.trec | Make a new project. Let’s call it “BasicTCPWM”. This project will vary the brightness of an LED with a very simple delay scheme. |
| 6 |  | If you blink an LED really fast - like about 30Hz or more - your eye will not see that it is blinking at all. If you vary the amount of time that it is on versus off the LED will appear brighter or dimmer based on the on/off ratio. If it is on all the time, that will be the brightest it will ever get… if it is on only 10% of the time then it will appear very dim. This on/off ratio is called the duty cycle.  PWMs are perfect for controlling duty cycle. The ratio of the compare value to the period is the duty cycle of the PWM. If I set a period to be 100 then the compare value can be between 0 and 100… which is also known as percent. This is what I am going to do in the next several PWM examples. |
| 7 | SCREEN CAPTURE:  P6-2-2a-PWM\_capture2.trec | Start by adding the TCPWM component by dragging and dropping it in to the schematic. Let’s rename it to PWM and set the period to 100 and compare to 0.  I will now drag in a pin component. Again, we’ll rename it to RED.  Now for the clock, drag and drop that component and we can keep the default configuration of the clock.  Make sure you wire the clock to the PWM and the pin to the PWM as I did when I dragged the components into the design.  Now to assign the pin component to the physical pin on the PSoC 6 device. In this case a red LED is connected to P0[3].  Time to generate the application. |
| 8 | SCREEN CAPTURE:  P6-2-2a-PWM\_capture3.trec | Now let’s go the CM4 main application file.  I will add a local variable to the main function called compareValue and set the default value to 50.  Start the PWM using the simple start API command.  In the infinite for loop, I will change the compare value of the PWM using the “CY\_TCPWM\_PWM\_SetCompare0” API call and the PWM\_HW macro. Each PWM hardware block has multiple PWM counters, so the SetCompare function needs to know which one I want to change. In this case, I’ll use the macro PWM\_CNT\_NUM. Now the last argument is used to set the compare value, I’ll set it to the local variable we created, compareValue.  Now I want to increment the local variable, compareValue, by 1 and mod it by 100 so it counts up to 100, the maximum period we set before, and resets to 0.  Finally, we need a delay so we can actually see the change, so let’s use the CyDelay API and 20 milliseconds. This will make the LED slowly dim and then go back to maximum brightness. |
| 9 | SCREEN CAPTURE:  P6-2-2a-PWM\_capture4.trec | Now, build, program and test. |
| 10 | VIDEO:  Show kit LED brightness varying | Awesome! I love blinky LEDs! |
| 11 | TEXT ON SCREEN:  Cypress Developer Community  community.cypress.com  VIDEO:  Show video of ARH email and twitter windows. | You can post your comments and questions in our PSoC 6 community or as always you are welcome to email me at alan\_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions. |

**Graphic1**

