Type A Board Dev Guide IV

Wilson 2020.8.4

https://github.com/TAMU-Robomasters/Tutorial

Roadmap:

1. STM32CubeMX, Keil uvision
2. LED, GPIO
3. Timer
4. PWM, passive buzzer, servo
5. Buttons
6. USB
7. Flash
8. I2C, IST8310 (magnetic sensor)
9. OLED
10. BMI088 (gyroscope)
11. Motor control with CAN
12. freeRTOS
13. IMU
14. Chassis tasks
15. Gimbal control
16. BIG PICTURE

Recall last time:

1. Know how to use CubeMX to configure FreeRTOS and FreeRTOS task queue
2. Passive buzzer
3. USB communication
4. Monitor serial communication on computer with putty

Proposed hardware list:

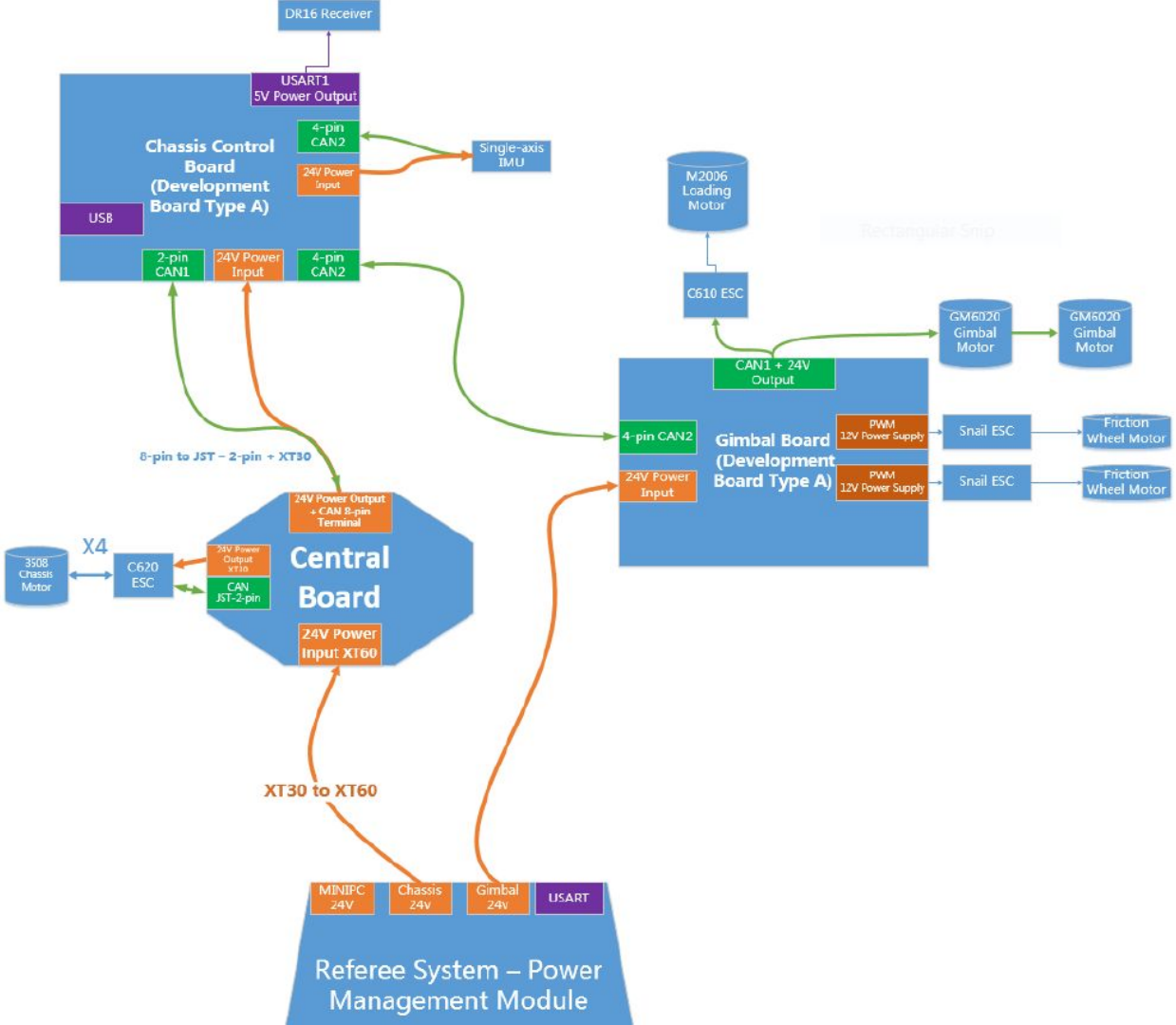
https://rm-static.djicdn.com/tem/17348/ICRA%202019%20DJI%20RoboMaster%20AI%20Challenge%20AI%20Robot%20User%20Manual%20V1.0.pdf

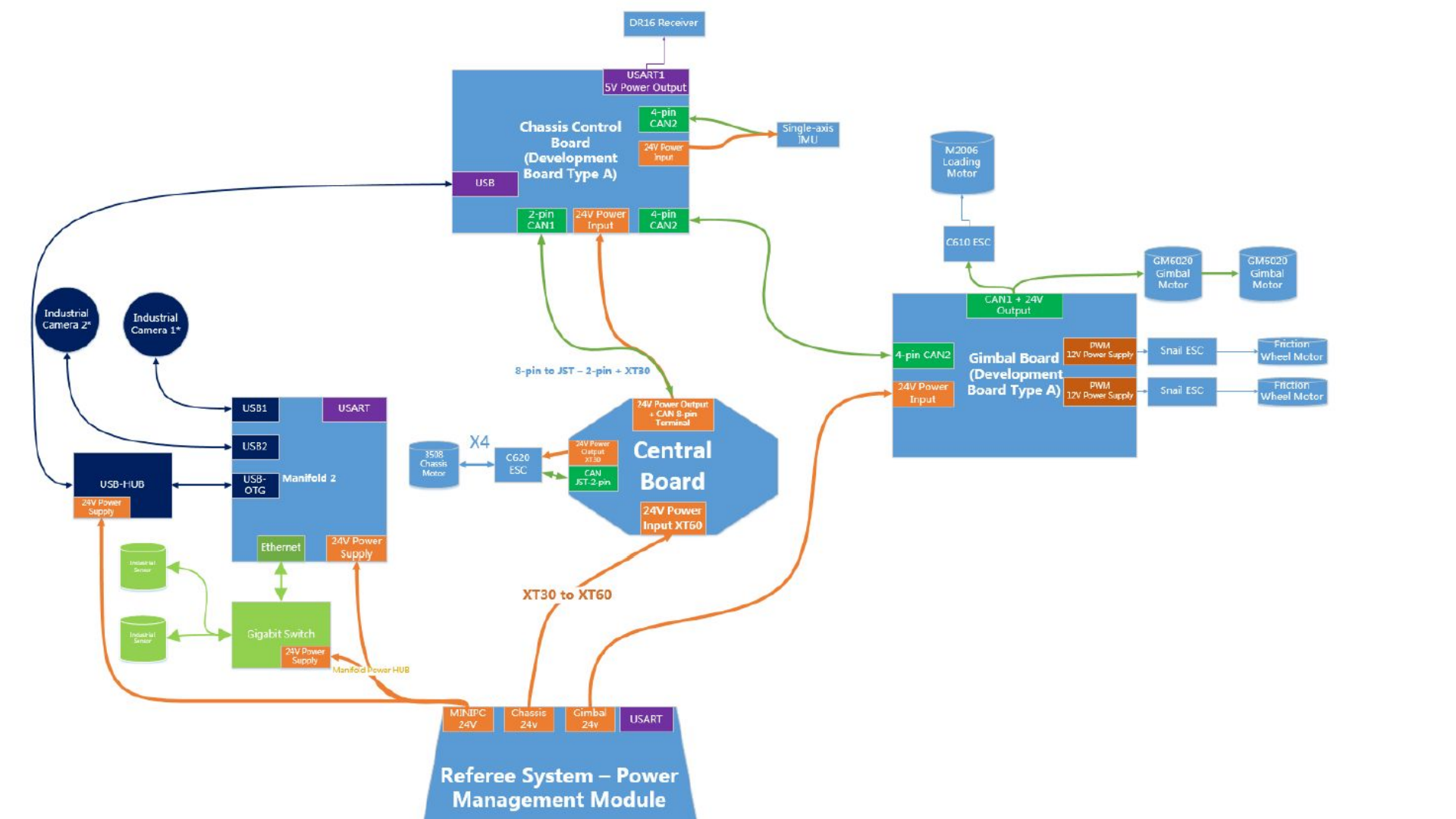
https://www.robomaster.com/en-US/products/components/detail/1843

Subject to changes!

M3508 + C620 ESC	x4	Chassis
M2006 + C610 ESC	x1	Loading ammo
GM6020	x2	Gimbal
Snail 2305 + Snail 430 R ESC	x2	Firing ammo
Type A dev board	x1	Chassis, gimbal
Battery	x1	Power source
Remote control + receiver	x1	Commanding
Referee system	x1	Monitoring

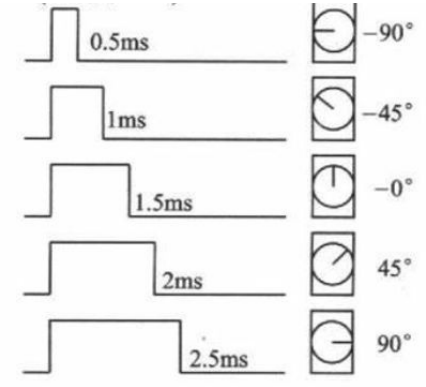
Have a look at the links. They contain graphs and explanations about the big picture!


DR16 Receiver
USART1
5V Power Output
Chassis Control Board (Development Board Type A)
4-pin CAN2
24V Power Input
Single-axis IMU
USB
2-pin CAN1
24V Power Input
4-pin CAN2
M2006 Loading Motor
C610 ESC
GM6020 Gimbal Motor
GM6020 Gimbal Motor
CAN1 + 24V Output
4-pin CAN2
24V Power Input
Gimbal Board (Development Board Type A)
PWM 12V Power Supply
PWM 12V Power Supply
Snail ESC
Snail ESC
Friction Wheel Motor
Friction Wheel Motor
8-pin to JST - 2-pin + XT30
24V Power Output + CAN 8-pin Terminal
Central Board
24V Power Output XT30
CAN JST-2-pin
24V Power Input XT60
X4
3508 Chassis Motor
C620 ESC
XT30 to XT60
MINIPC 24V
Chassis 24v
Gimbal 24v
USART
Referee System – Power Management Module


DR16 Receiver
USART1
5V Power Output
Chassis Control Board (Development Board Type A)
4-pin CAN2
24V Power Input
Single-axis IMU
USB
2-pin CAN1
24V Power Input
4-pin CAN2
M2006 Loading Motor
C610 ESC
GM6020 Gimbal Motor
GM6020 Gimbal Motor
CAN1 + 24V Output
Industrial Camera 2*
Industrial Camera 1*
8-pin to JST – 2-pin + XT30
4-pin CAN2
24V Power Input
Gimbal Board (Development Board Type A)
PWM 12V Power Supply
Snail ESC
Friction Wheel Motor
PWM 12V Power Supply
Snail ESC
Friction Wheel Motor
USB1
USART
USB2
USB-HUB
USB-OTG
Manifold 2
24V Power Supply
Ethernet
24V Power Supply
24V Power Output + CAN 8-pin Terminal
X4
3508 Chassis Motor
C620 ESC
24V Power Output XT30
CAN JST-2-pin
Central Board
24V Power Input XT60
XT30 to XT60
Industrial Sensor
Industrial Sensor
Gigabit Switch
24V Power Supply
Manifold Power HUB
MINIPC 24V
Chassis 24v
Gimbal 24v
USART
Referee System – Power Management Module

Servo control:

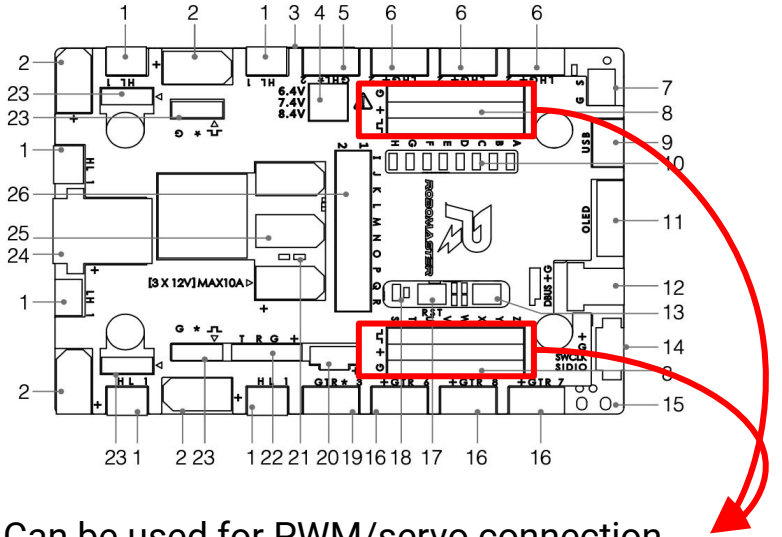
Servo used on our robots is driven by fixed-frequency PWM wave. To control the angle the servo positions at, change the duty cycle of the PWM signal.



Example: PWM at 50Hz, signal period 20ms

Servo wiring:

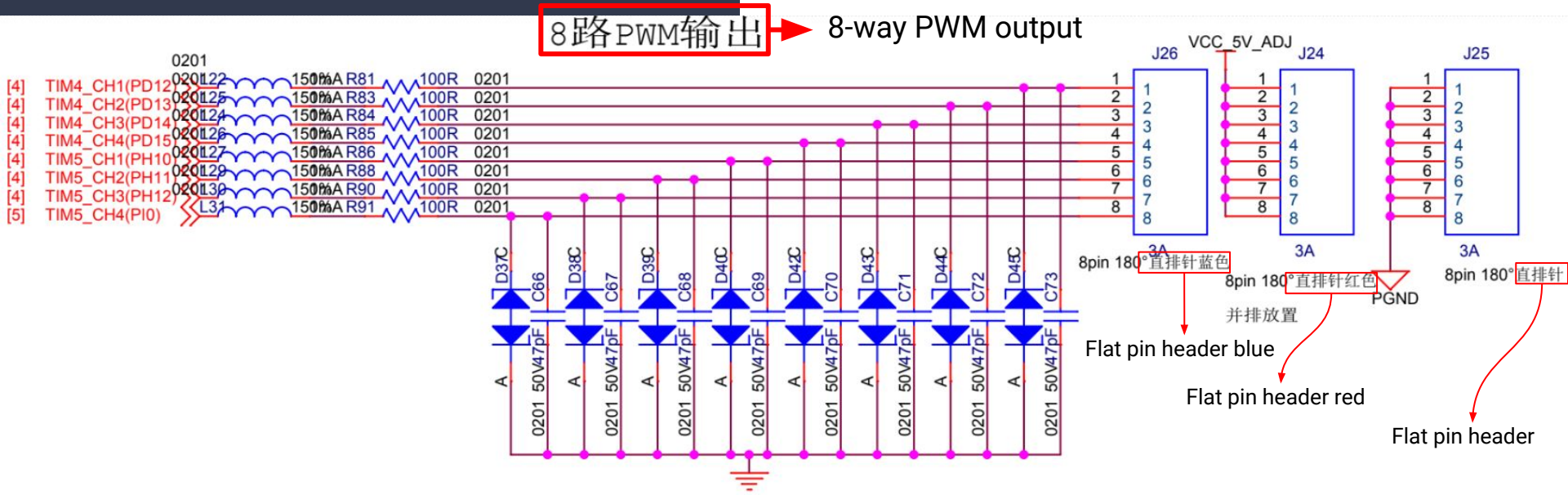
3 connections per servo: GND, VCC, PWM



Can be used for PWM/servo connection.

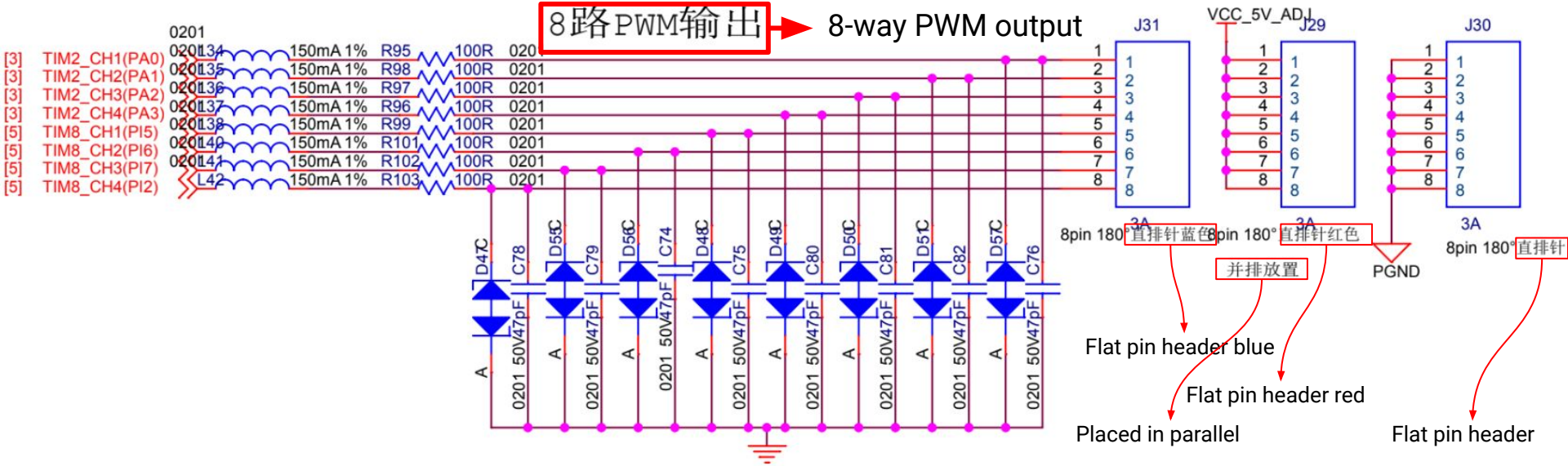
What timer(s) to look at: 1/2

Let's use TIM5_CH4(PI0) and TIM5_CH3(PH12) to connect the two servos!



What timer(s) to look at: 2/2

Remember on type A dev board, two rows of PWM output pins are available!



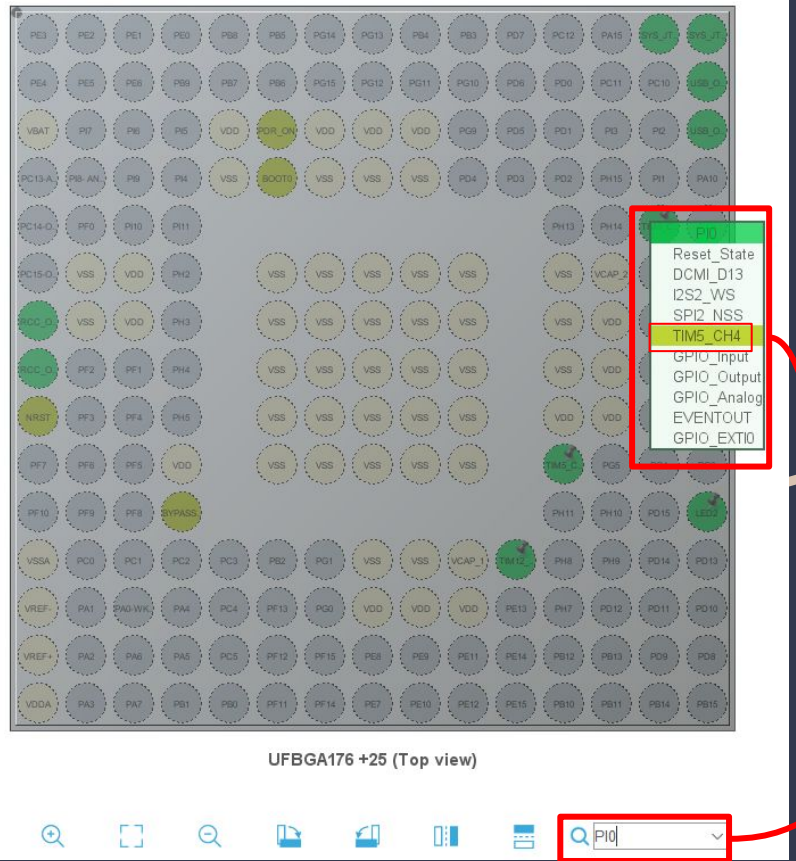
Servo specs:

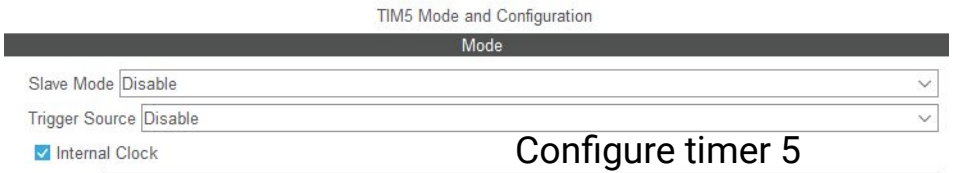
Screenshot from Amazon

Model	RDS5160
Operating Voltage	6-8.4V DC
Operating Angle	270 degrees
Torque	(58~70)kg*cm
Pulse Width Range	0.5~2.5 ms

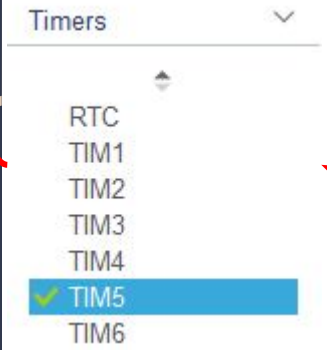
Usually servos are driven by PWM of 50 Hz, i.e. period=20ms, to operate the servo. By changing the duty cycle, servos can turn to different angles.

CubeMX Settings:





Configure timer 5



Search for PI0 and PH12. Set to TIM5_CH4 and TIM5_CH3 correspondingly

More on Timer 5:

Where to find information?
Keil -> Books

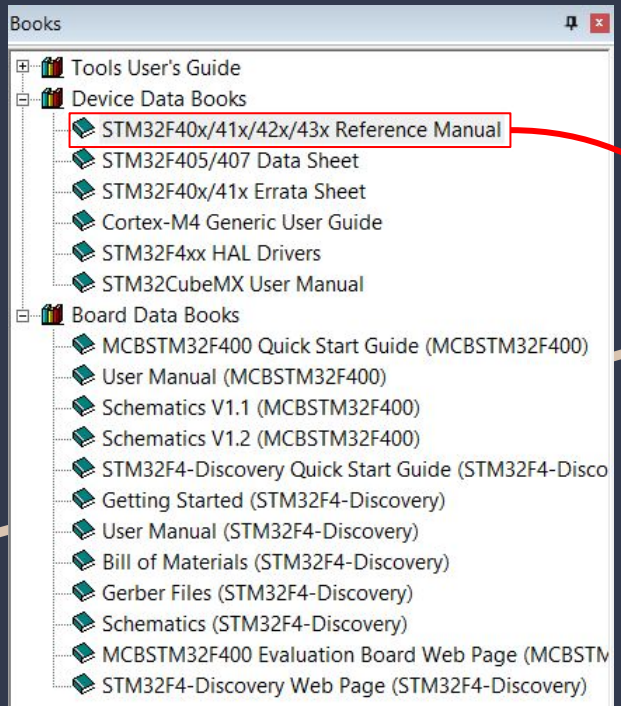
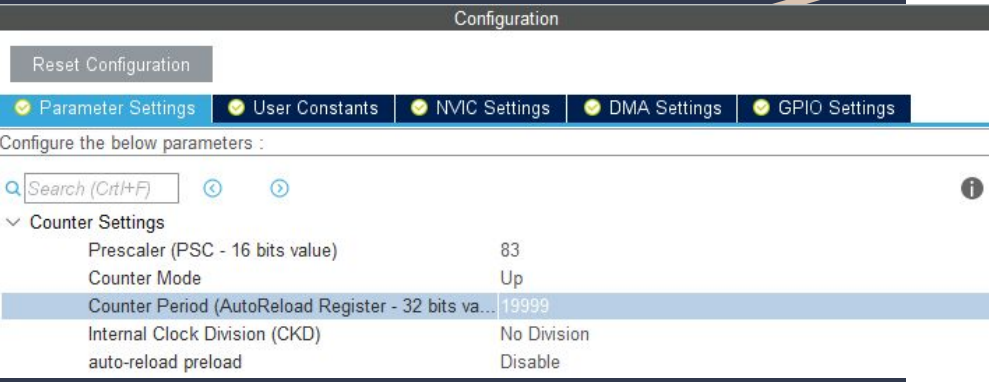


Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4000 7400 - 0x4000 77FF	DAC	APB1	Section 14.5.15: DAC register map on page 453
0x4000 7000 - 0x4000 73FF	PWR		Section 5.6: PWR register map on page 149
0x4000 6800 - 0x4000 6BFF	CAN2		Section 32.9.5: bxCAN register map on page 1120
0x4000 6400 - 0x4000 67FF	CAN1		
0x4000 5C00 - 0x4000 5FFF	I2C3		Section 27.6.11: I2C register map on page 875
0x4000 5800 - 0x4000 5BFF	I2C2		
0x4000 5400 - 0x4000 57FF	I2C1		
0x4000 5000 - 0x4000 53FF	UART5		Section 30.6.8: USART register map on page 1021
0x4000 4C00 - 0x4000 4FFF	UART4		
0x4000 4800 - 0x4000 4BFF	USART3		
0x4000 4400 - 0x4000 47FF	USART2		
0x4000 4000 - 0x4000 43FF	I2S3ext		Section 28.5.10: SPI register map on page 928
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3		
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2		
0x4000 3400 - 0x4000 37FF	I2S2ext		
0x4000 3000 - 0x4000 33FF	IWDG		Section 21.4.5: IWDG register map on page 713
0x4000 2C00 - 0x4000 2FFF	WWDG		Section 22.6.4: WWDG register map on page 720
0x4000 2800 - 0x4000 2BFF	RTC & BKP Registers		Section 26.6.21: RTC register map on page 837
0x4000 2000 - 0x4000 23FF	TIM14		Section 19.5.12: TIM10/11/13/14 register map on page 694
0x4000 1C00 - 0x4000 1FFF	TIM13		
0x4000 1800 - 0x4000 1BFF	TIM12		Section 19.4.13: TIM9/12 register map on page 684
0x4000 1400 - 0x4000 17FF	TIM7		Section 20.4.9: TIM6&TIM7 register map on page 708
0x4000 1000 - 0x4000 13FF	TIM6		
0x4000 0C00 - 0x4000 0FFF	TIM5		Section 18.4.21: TIMx register map on page 646
0x4000 0800 - 0x4000 0BFF	TIM4		
0x4000 0400 - 0x4000 07FF	TIM3		
0x4000 0000 - 0x4000 03FF	TIM2		

Go back to CubeMX:

Time duration is 2ms for high level. So will stay at 45 degrees at booting time.



As we see from the previous slide, timer 5 is connected to the APB1 bus. In CubeMX clock configuration (we are re-using the clock configuration from the previous tutorial), APB1 timer clock is running at 84 MHz. To have a PWM wave running at 50 Hz, we need to scale down the clock frequency. We need to scale down a factor of 84M/50 = 1680000.

First, we scale down 84 MHz by a factor of 84, so set the pre-scalar to 84 - 1 = 83. -> 1 MHz

Second, we further scale down by a factor of 20000. So set counter period to 20000- 1 = 19999. -> 50 Hz

For each PWM channel, set the pulse to 2000. So at booting time, the duty cycle is 2000/20000 = 10%

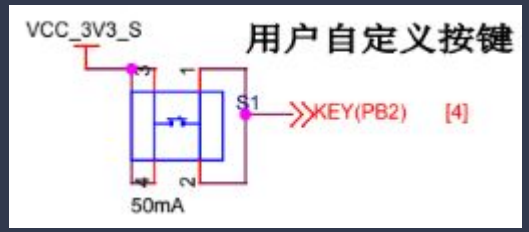
Then add a task to FreeRTOS that moves servos back and forth at a period of 5 seconds.

Coding in Keil

```
// in main() from main.c
HAL_TIM_Base_Start_IT(&htim5);
HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_3);
HAL_TIM_PWM_Start(&htim5, TIM_CHANNEL_4);

// in main.c, outside main()
void Servotask_entry(void *argument){
      while(1){
      __HAL_TIM_SetCompare(&htim5, TIM_CHANNEL_3, 1000);
      __HAL_TIM_SetCompare(&htim5, TIM_CHANNEL_4, 1000);
      osDelay(5000);
      __HAL_TIM_SetCompare(&htim5, TIM_CHANNEL_3, 2000);
      __HAL_TIM_SetCompare(&htim5, TIM_CHANNEL_4, 2000);
      osDelay(5000);
      }
}
```

Button:



The black button on type A dev board is for reset.

The white button however can be used for other purposes.

Enable PB2 in CubeMX and set its mode to GPIO_EXTI2.

Go to System Core -> GPIO, find the settings for PB2 and change to the following:

PB2 Configuration :

GPIO mode	External Interrupt Mode with Rising/Falling edge trigger detection
GPIO Pull-up/Pull-down	Pull-up
User Label	Custom_Button

Also enable PG3 (LED) to GPIO output

FreeRTOS setting:

Add a task called Buttontask to the FreeRTOS task queue in CubeMX.

Generate the code.

Keil:

```
void Buttontask_entry(void *argument){
  while(1){
    if (HAL_GPIO_ReadPin(GPIOB, Custom_Button_Pin) == GPIO_PIN_SET){
      HAL_GPIO_WritePin(Button_LED_GPIO_Port, Button_LED_Pin, GPIO_PIN_RESET);
    }
    else{
      HAL_GPIO_WritePin(Button_LED_GPIO_Port, Button_LED_Pin, GPIO_PIN_SET);
    }
  }
}
```

And every time the white button is pressed, you should see the green LED on/off.

Reference:

https://www.amazon.com/ANNIMOS-Digital-Voltage-Stainless-Waterproof/dp/B07KTSCN4J