Type A Board Dev Guide III

Wilson 2020.7.28

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 timer
2. Interrupt priority groups and how to set them
3. Registers used in timers
4. Calculate timer interrupt period
5. Function calls to initialize, start and use timer interrupts

FreeRTOS 1/2:

"FreeRTOS is a real-time operating system kernel for embedded devices that has been ported to 35 microcontroller platforms. It is distributed under the MIT License."

"FreeRTOS provides methods for multiple threads or tasks, mutexes, semaphores and software timers."

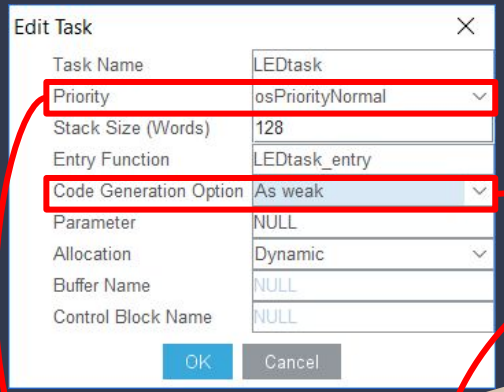
FreeRTOS 2/2:

Default: generate a common task function

As weak: generate a task function decorated with __weak

As external: generate a task function used externally

More information can be found in `freertos.c`



Priority setting

```c
/* USER CODE BEGIN Header_LEDtask_entry */
/**
  * @brief  Function implementing the LEDtask thread.
  * @param  argument: Not used
  * @retval None
  */
/* USER CODE END Header_LEDtask_entry */
__weak void LEDtask_entry(void *argument)
{
  /* USER CODE BEGIN LEDtask_entry */
  /* Infinite loop */
  for(;;)
  {
    osDelay(1);
  }
  /* USER CODE END LEDtask_entry */
}
```

main.c

```c
  /* Call init function for freertos objects (in freertos.c) */
  MX_FREERTOS_Init();
```

FreeRTOS Summary:

How to start writing FreeRTOS tasks:

1. Enable FreeRTOS middleware in CubeMX.
2. Add and configure tasks under the FreeRTOS middleware tab.
3. In the generated template code, implement those tasks.

How FreeRTOS tasks are called:

1. Peripheral initialization, i.e. `MX_GPIO_Init()`
2. `MX_FREERTOS_Init()`
3. Create handler based on the task definition
4. Start scheduler, `osKernelStart()`

PWM:

PWM = Pulse Width Modulation

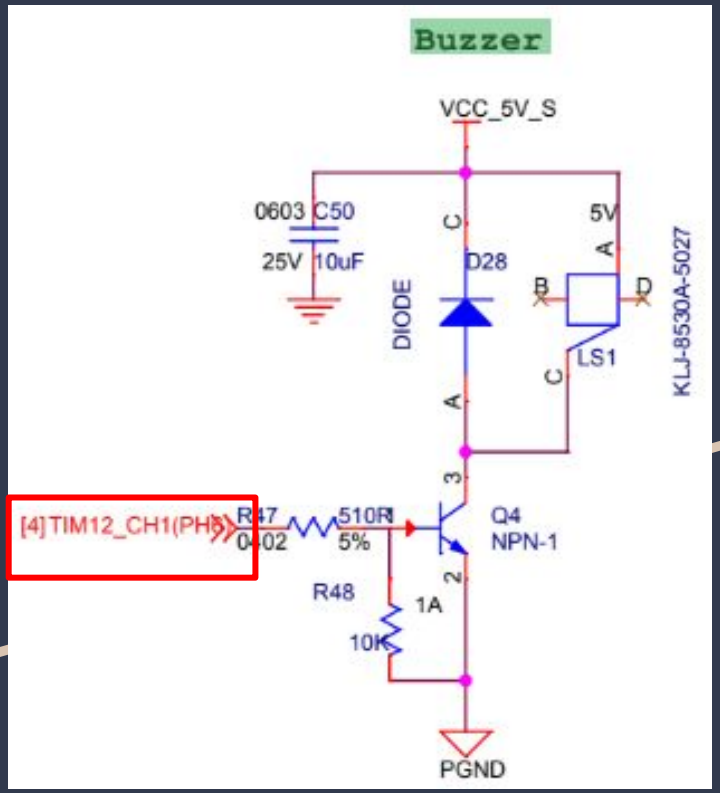
Use 1 and 0 with different durations to approximate an analogue value.

-> Turn 1 and 0 into some value between 1 and 0

Duty cycle: the percentage of the duration of a 1 value within a cycle

Correspondingly, the higher the frequency, the better the approximation is, i.e. the combination of 1s and 0s looks more like an analogue value instead of a digital value.

Passive Buzzer:



Type A board passive buzzer is hard-wired to Timer 12 Channel 1, with pin identifier PH6.

Enable TIM12 in CubeMX, set channel 1 to PWM Generation CH1.

According to the user manual, the passive buzzer found on type A dev board can be driven using PWM at 2700HZ.

Timer 12 is connected to APB1 bus, based on our clock configuration, AP1 bus runs at 84MHz.

Functions for changing timer parameters:

```
__HAL_TIM_PRESCALER(&<timer>, <uint>);

// for changing the frequency

__HAL_TIM_SetCompare(&<timer>,<channel>,<uint>);

// for setting the duty cycle
```

After trial and error, setting the duty cycle with the number 5000 seems to be a good choice.

In CubeMX, the counter period can be set to be 9999 for easy frequency calculation.

Counter Settings	
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits val...	9999
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Wrap the PWM functions up:

```
// family mart tune
buzzer_on(24, 5000);
HAL_Delay(250);
buzzer_on(31, 5000);
HAL_Delay(250);
buzzer_on(42, 5000);
HAL_Delay(250);
buzzer_on(31, 5000);
HAL_Delay(250);
buzzer_on(27, 5000);
HAL_Delay(250);
buzzer_on(20, 5000);
HAL_Delay(500);
buzzer_on(27, 5000);
HAL_Delay(250);
buzzer_on(24, 5000);
HAL_Delay(250);
buzzer_on(27, 5000);
HAL_Delay(250);
buzzer_on(42, 5000);
HAL_Delay(250);
buzzer_on(31, 5000);
HAL_Delay(1000);
buzzer_off();
```

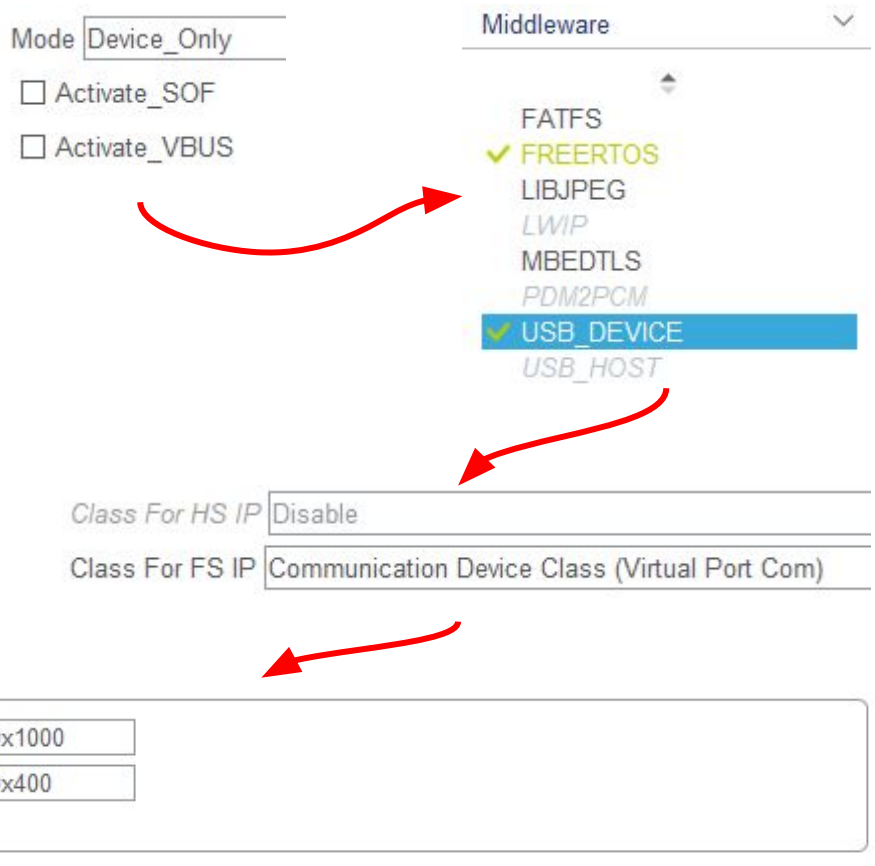
```
void buzzer_on(uint16_t psc, uint16_t pwm){
    __HAL_TIM_PRESCALER(&htim12, psc);
    __HAL_TIM_SetCompare(&htim12, TIM_CHANNEL_1, pwm);
}

void buzzer_off(void)
{
    __HAL_TIM_SetCompare(&htim12, TIM_CHANNEL_1, 0);
}
```

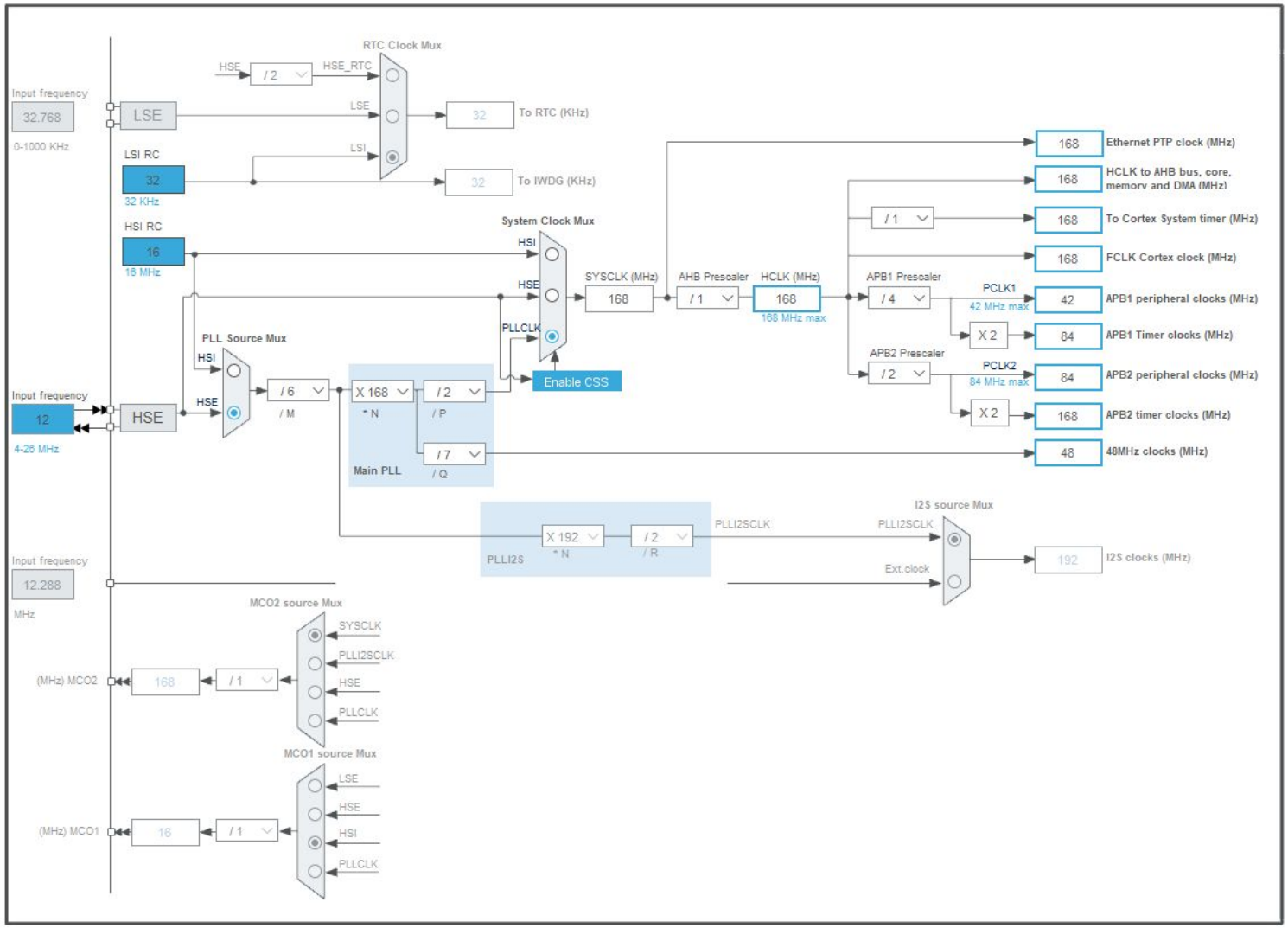
```
HAL_TIM_Base_Start_IT(&htim12);
HAL_TIM_PWM_Start(&htim12, TIM_CHANNEL_1);
```

You need to manually write those two lines! Otherwise the timer is initialized but not started. And PWM will not be working! CubeMX will generate code that initializes the timer but not the one that starts the timer!

Use USB on type A dev board:



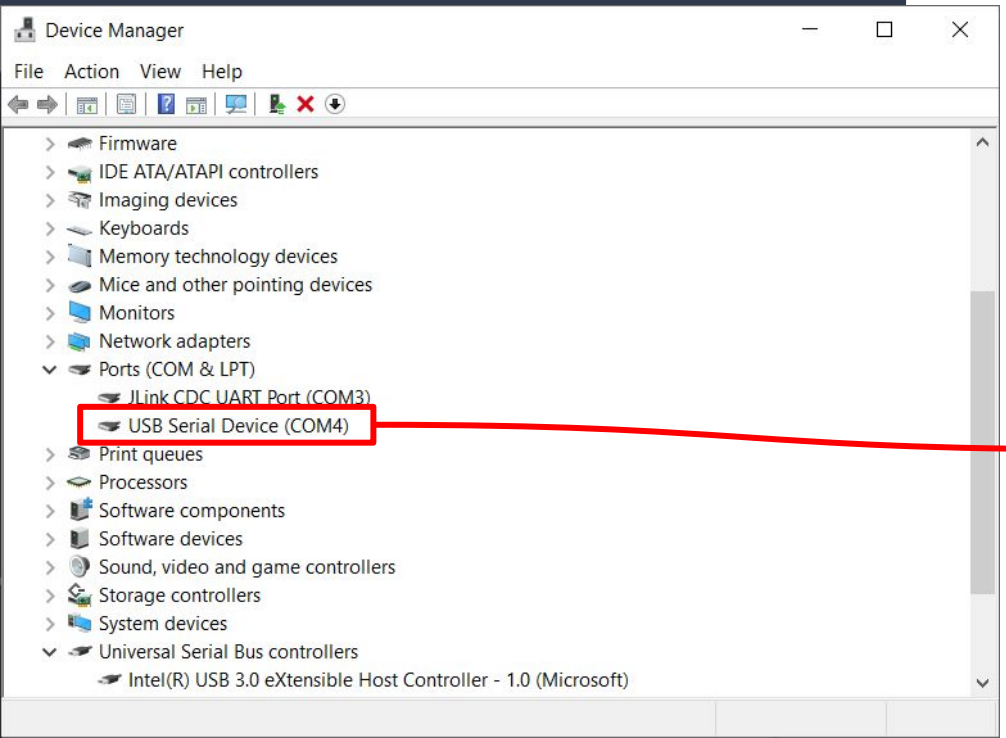
Change in project manager -> Project



Adjust the clock tree after enabling USB in CubeMX.

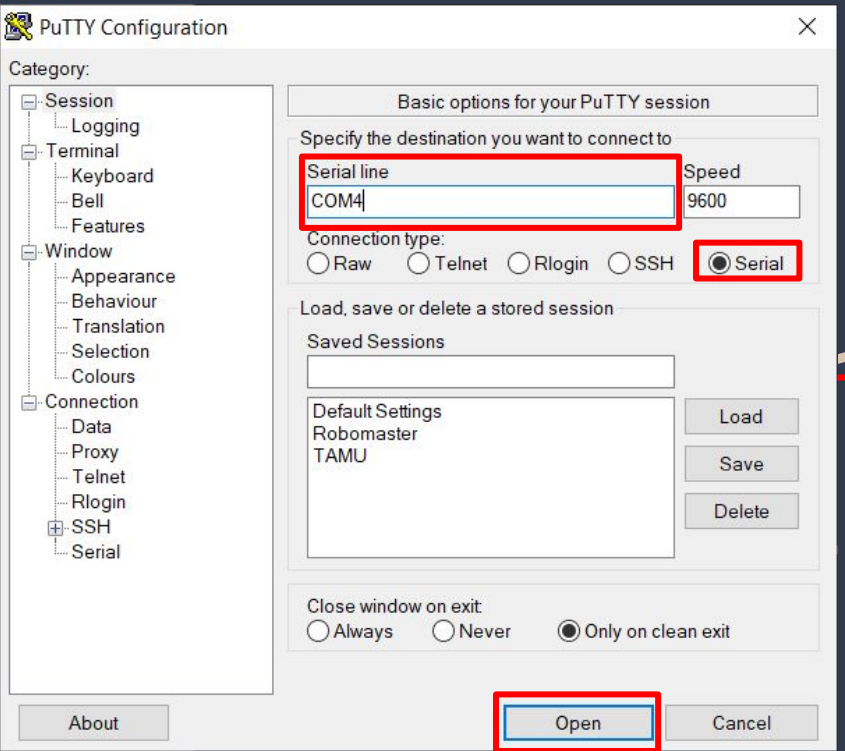
Send data from USB:

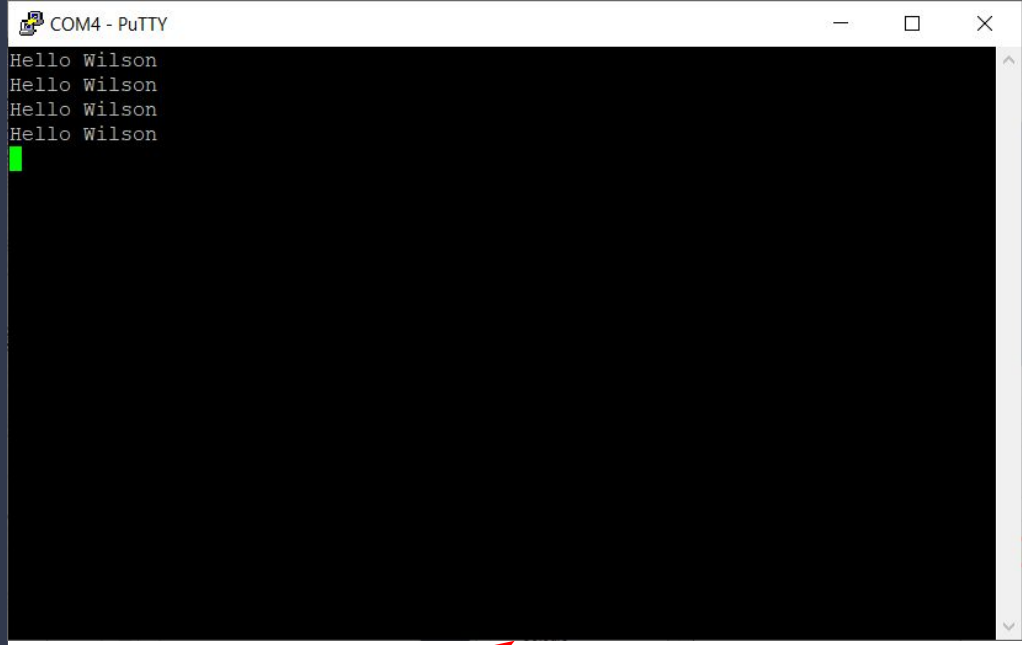
1. Call `MX_USB_DEVICE_Init()` to initialize USB device support libraries
2. Use `CDC_Transmit_FS(buffer, sizeof(buffer))` to send information. `Buffer` is a char array
3. Use the Windows device manager to check for the port number of the type A dev board.
4. Use putty to monitor serial port information



So on my computer, the USB port on the type A dev board is recognized as “COM4” port under the “Ports (COM & LPT)” category in Windows device manager. The port number is used when communicating via a serial monitor like putty.

Putty:


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Serial line
COM4
Speed
9600
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
Robomaster
TAMU
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel


COM4 - PuTTY
Hello Wilson
Hello Wilson
Hello Wilson
Hello Wilson

References:

https://en.wikipedia.org/wiki/FreeRTOS