

## Chapter 2:

### **CH32V003 PROGRAMMING: HOW TO USE GPIO AS OUTPUT**

We will see how we can set any GPIO Pin as output and set the Pin state as high or low. We saw this in GPIO Toggle example code also but here in this article we will go in a little deeper.

#### **4.1: GPIO Ports:**

There are three port groups available in the MCU, Port **A**, **C** and **D** and in each group, number of pins are there.

Each Port pin can be configured as GPIO output and can be used in the application.

#### **4.2 Uses of GPIO:**

The GPIO port can be configured for multiple input or output modes, with built-in pull-up or pull-down resistors that can be turned off, and can be configured for push-pull or open-drain functions. the GPIO port can also be multiplexed for other functions.

GPIOs are mainly used for connecting LEDs, Relay control, providing status and control signals when connecting other devices or could be used for driving LCDs like OLED in SPI mode where SPI is emulated by GPIO bit banging, etc.

##### **4.2.1 Main Features**

- i. Floating input
- ii. Pull-up input
- iii. Dropdown input
- iv. Analog input
- v. Open drain output
- vi. Push-pull output
- vii. Multiplexing the inputs and outputs of functions

Many pins have multiplexing capabilities, and many other peripherals map their output and input channels to these pins. The specific usage of these multiplexed pins needs to be referred to the individual peripherals, and the content of whether these pins are multiplexed and remapped is explained in this chapter.

### **4.3 Pinout Configuration:**

#### **CH32V003F4P6**

1	PD4/A7/UCK/T2CH1ETR/OPO/T1CH4ETR_	PD3/A4/T2CH2/AETR/UCTS/T1CH4_	20
2	PD5/A5/UTX/T2CH4_/URX_	PD2/A3/T1CH1/T2CH3_/T1CH2N_	19
3	PD6/A6/URX/T2CH3_/UTX_	PD1/SWIO/AETR2/T1CH3N/SCL_/URX_	18
4	PD7/NRST/T2CH4/OPP1/UCK_	PC7/MISO/T1CH2_/T2CH2_/URTS_	17
5	PA1/OSCI/A1/T1CH2/OPN0	PC6/MOSI/T1CH1CH3N_/UCTS_/SDA_	16
6	PA2/OSCO/A0/T1CH2N/OPP0/AETR2_	PC5/SCK/T1ETR/T2CH1ETR_/SCL_/UCK_/T1CH3_	15
7	VSS	PC4/A2/T1CH4/MCO/T1CH1CH2N_	14
8	PD0/T1CH1N/OPN1/SDA_/UTX_	PC3/T1CH3/T1CH1N_/UCTS_	13
9	VDD	PC2/SCL/URTS/T1BKIN/AETR_/T2CH2_/T1ETR_	12
10	PC0/T2CH3/UTX_/NSS_/T1CH3_	PC1/SDA/NSS/T2CH4_/T2CH1ETR_/T1BKIN_/URX_	11

*CH32V003 TSSOP-20 Package Pinout*

***It is recommended to read the datasheet and go through the electrical characteristics section to know more about capability and limitation of GPIO port/pins. Like output current sourcing and sinking capability, etc.***

**Please note PD7 is by default is configured as MCU reset pin, you need to configure it as GPIO by configuring it with WCH Link Programmer Utility or in the code.**

Two options to change

1.

```
FLASH_Unlock(); FLASH_EraseOptionBytes();  
FLASH_UserOptionByteConfig(OB_STOP_NoRST, OB_STDBY_NoRST, OB_RST_NoEN,  
OB_PowerON_Start_Mode_USER);  
FLASH_Unlock();
```

2: If you have WCH-Link Utility in your computer. You also can use it to change PD7 mode.

WCH-LinkUtility V1.80

File Target View Help



MCU Core: RISC-V Series: CH32V00X Address: 0x08000000

☒ Erase All ☒ Program ☒ Verify ☒ Reset and Run

☐ Enable MCU Code Read-Protect ☒ Disable MCU Code Read-Protect

CLK Speed: Low

Name	Value
MCU UID	
Flash Size	
Read Protect	
Link Version	
Write Protect	

Disable Two-Line Interface

☒ Disable Standby-Mode RST ☒ Enable Soft-Ctrl IWDG

DATA0: 0x FF

DATA1: 0x FF

RF 2.4G AccessAddr: 0x CREATE GET SET

Enable mul-func, ignored pin status within 12ms

Enable mul-func, ignored pin status within 128us

Enable mul-func, ignored pin status within 1ms

Enable mul-func, ignored pin status within 12ms

Disable mul-func, PD7 is used for IO function

WRP0: 0x FF ☒ 0 ☒ 1 ☒ 2 ☒ 3

WRP1: 0x FF ☒ 8 ☒ 9 ☒ 10 ☒ 11 ☒ 12 ☒ 13 ☒ 14 ☒ 15

WRP2: 0x FF ☐ 16 ☐ 17 ☐ 18 ☐ 19 ☐ 20 ☐ 21 ☐ 22 ☐ 23

WRP3: 0x FF ☐ 24 ☐ 25 ☐ 26 ☐ 27 ☐ 28 ☐ 29 ☐ 30 ☐ 31

Firmware: (Users\OWNER\Desktop\CH641\CH641EVT\_V1.1\EVT\EXAM\FLASH\FLASH\_Program\obj\FLASH\_Test641.hex

☐ Auto download when WCH-Link was linked

Detection Interval(S): 5

Chip Flash Addr: 0x 8000000 Size: 0x 4000 Data Width: 16bytes ☐ Show ASCII Clear

Connected WCH-Link List: RISC-V Link [#1]

Refresh

Active WCH-Link Mode:

Get

Set

Operation Result:



Result Collect:

Succ:0 | Total:0

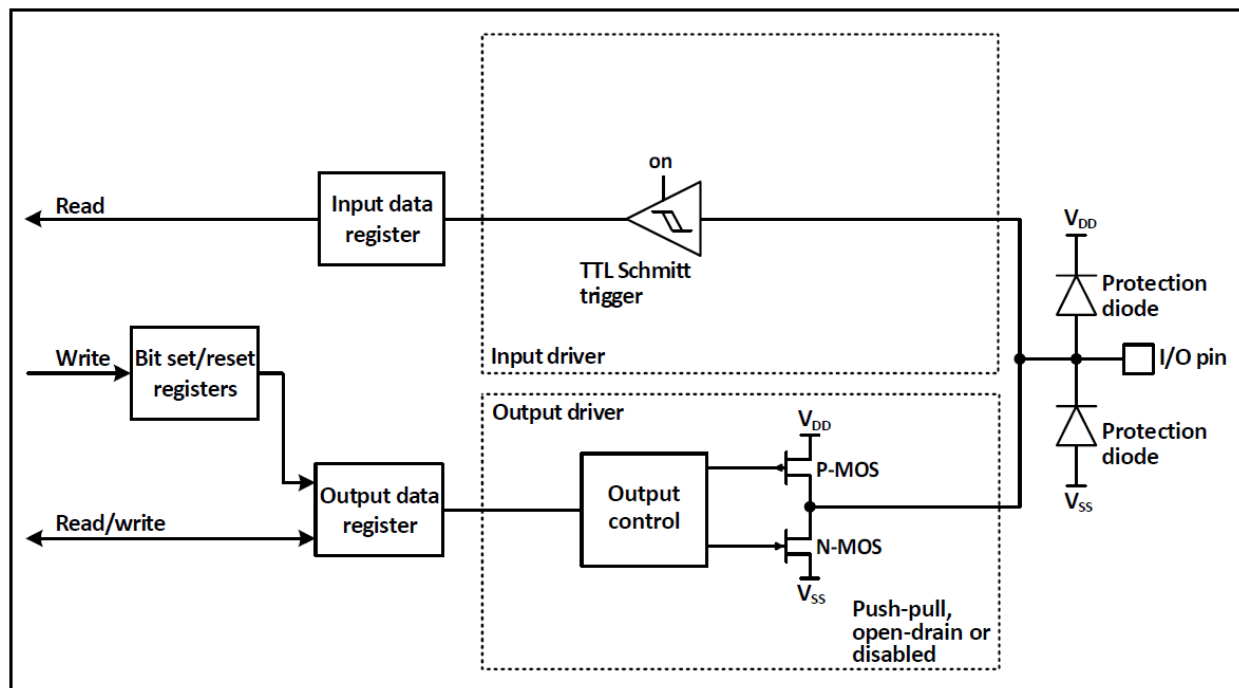
Clear

11:07:36:307>> Connected RISC-V mode WCH-Link Cnt:2

## 4.4 Output Configuration of GPIO:

When a GPIO is configured as an Output pin the Output Driver can be configured in Push-Pull or Open-Drain without using multiplexing function. The resistors on Input Driver are disabled therefore, the Voltage level (GPIO Status) appearing on GPIO set through output driver are stored in input data register at each clock cycle with TTL Schmitt trigger turning on.

In Push-Pull output mode status of GPIO can be obtained through reading the Output data register.



## 4.5 GPIO Initialization

After reset GPIOs are they run in their initial states and most of them run in floating states. We need to initialize the GPIOs according to the desired functionality. We need to define these parameters for initializing the GPIO pin that will be described in the below example.

In order to configure any pin as GPIO output, you need to use the following code, let us call that in a function “GPIOConfig” as shown below, this will be the initialization code and need to be called before using the GPIO.

- 1) For any configuration, clock for that GPIO port needs to be enabled first.
- 2) Three Parameters are there for any GPIO when configuring as output
  - **GPIO\_Pin:** this is to define which Pin, for example for D0 it will be GPIO\_Pin\_0
  - **GPIO\_Mode:** to configure if the output will be an open drain (GPIO\_Mode\_Out\_OD) or push-pull (GPIO\_Mode\_Out\_PP)

- **GPIO\_Speed:** to configure how fast you want control the GPIO. There are three options: GPIO\_Speed\_10MHz, GPIO\_Speed\_2MHz, GPIO\_Speed\_50MHz. This basically configures the drive strength of the GPIO internally.

## 4.6 GPIO as Output Example Code for CH32V003:

```
void GPIO_Config (void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0}; //structure variable used for the
    GPIO configuration
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE); // to Enable the clock
    for Port D
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0; // Defines which Pin to configure
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // Defines Output Type
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // Defines speed
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

If **multiple pins** of same port need to be configured with similar settings, you can write as shown below (GPIO D0 and D1):

```
void GPIO_Config(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0}; //structure variable used for the
    GPIO configuration
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE); // to Enable the clock
    for Port D
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1; // Defines which Pin
    to configure
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // Defines Output Type
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // Defines speed
    GPIO_Init(GPIOD, &GPIO_InitStructure); }
```

but, if you have **different settings** for different GPIO you can do like this (GPIO D0 and D1):

And, now suppose if you want to configure different pins of different ports, you can write code like this: (GPIO D0 and C1).

```
void GPIO_Config(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0}; //structure variable used for the
    GPIO configuration
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE); // to Enable the clock
    for Port D
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0; // Defines which Pin to configure
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // Defines Output Type
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; // Defines speed

    GPIO_Init(GPIOD, &GPIO_InitStructure);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1; // Defines which Pin to configure
```

```

GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_OD; // Defines Output Type
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; // Defines speed

GPIO_Init(GPIOD, &GPIO_InitStructure);
}

```

Now, let us see which all functions are available for controlling the GPIO pins or port.

if you go through ch32v00x\_gpio.h header file, you can see there following function which you will be using for GPIO output operations

Function names are self-explanatory.

```

void      GPIO_SetBits(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
void      GPIO_ResetBits(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
void      GPIO_WriteBit(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, BitAction
BitVal);
void      GPIO_Write(GPIO_TypeDef *GPIOx, uint16_t PortVal);
void      GPIO_PinLockConfig(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);

```

## 4.7 Toggle GPIO as Output (Example):

**Main Code:**

```

#include "debug.h"

/*****
 * @fn      GPIO_Toggle_INIT
 *
 * @brief   Initializes GPIOA.0
 *
 * @return  none
 */
void GPIO_Toggle_INIT(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}

/*****
 * @fn      main
 *
 * @brief   Main program.
 *
 * @return  none
 */

```

```

int main(void)
{
    u8 i = 0;

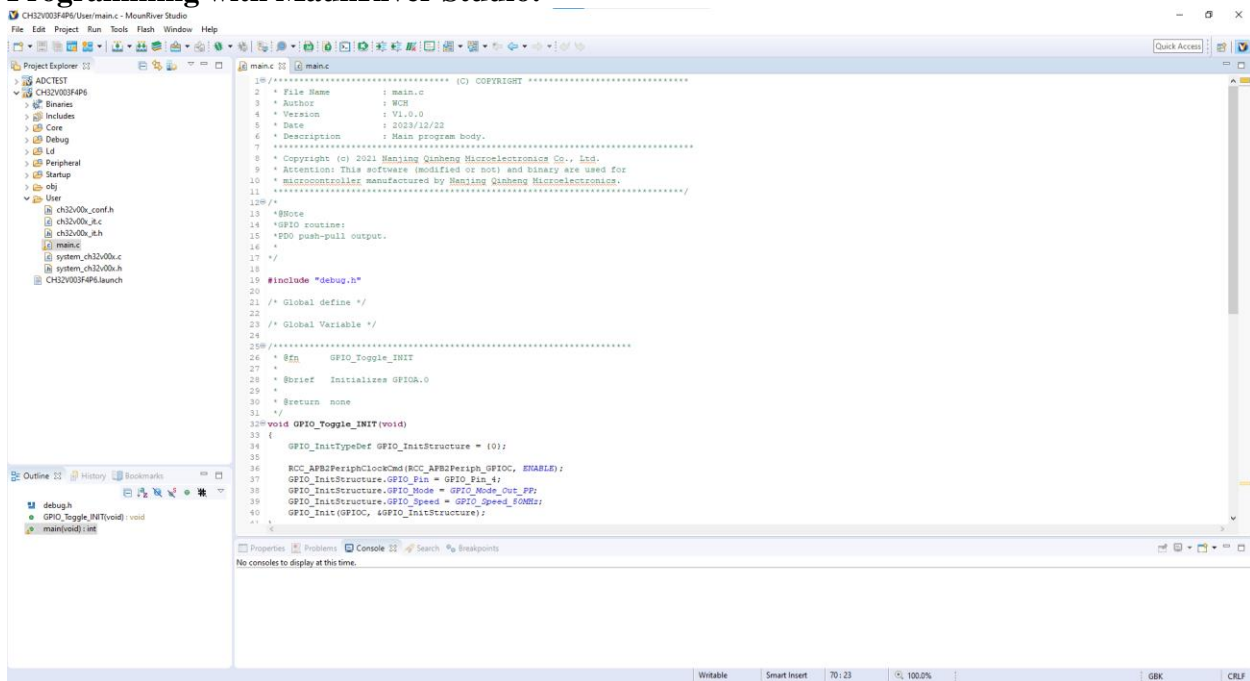
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
    SystemCoreClockUpdate();
    Delay_Init();
    #if (SDI_PRINT == SDI_PR_OPEN)
        SDI_Printf_Enable();
    #else
        USART_Printf_Init(115200);
    #endif
    printf("SystemClk:%d\r\n", SystemCoreClock);
    printf("ChipID:%08x\r\n", DBGMCU_GetCHIPID());
    printf("GPIO Toggle TEST\r\n");

    GPIO_Toggle_INIT();

    while(1)
    {
        Delay_Ms(250); //SET THE DELAY VALUE HERE
        GPIO_WriteBit(GPIOD, GPIO_Pin_0, (i == 0) ? (i = Bit_SET) : (i =
Bit_RESET));
    }
}

```

## Programming with MaunRiver Studio:



## Build the code ...

CDT Build Console [CH32V003F4P6]

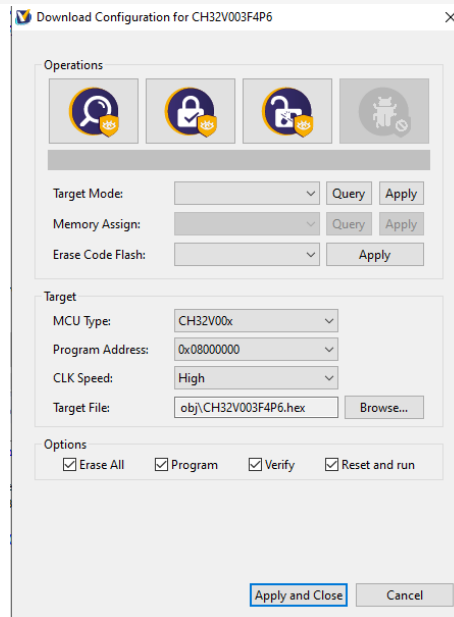
16:54:23 \*\*\*\* Incremental Build of configuration obj for project CH32V003F4P6 \*\*\*\*

make -j4 all

text	data	bss	dec	hex	filename
3588	56	264	3908	f44	CH32V003F4P6.elf

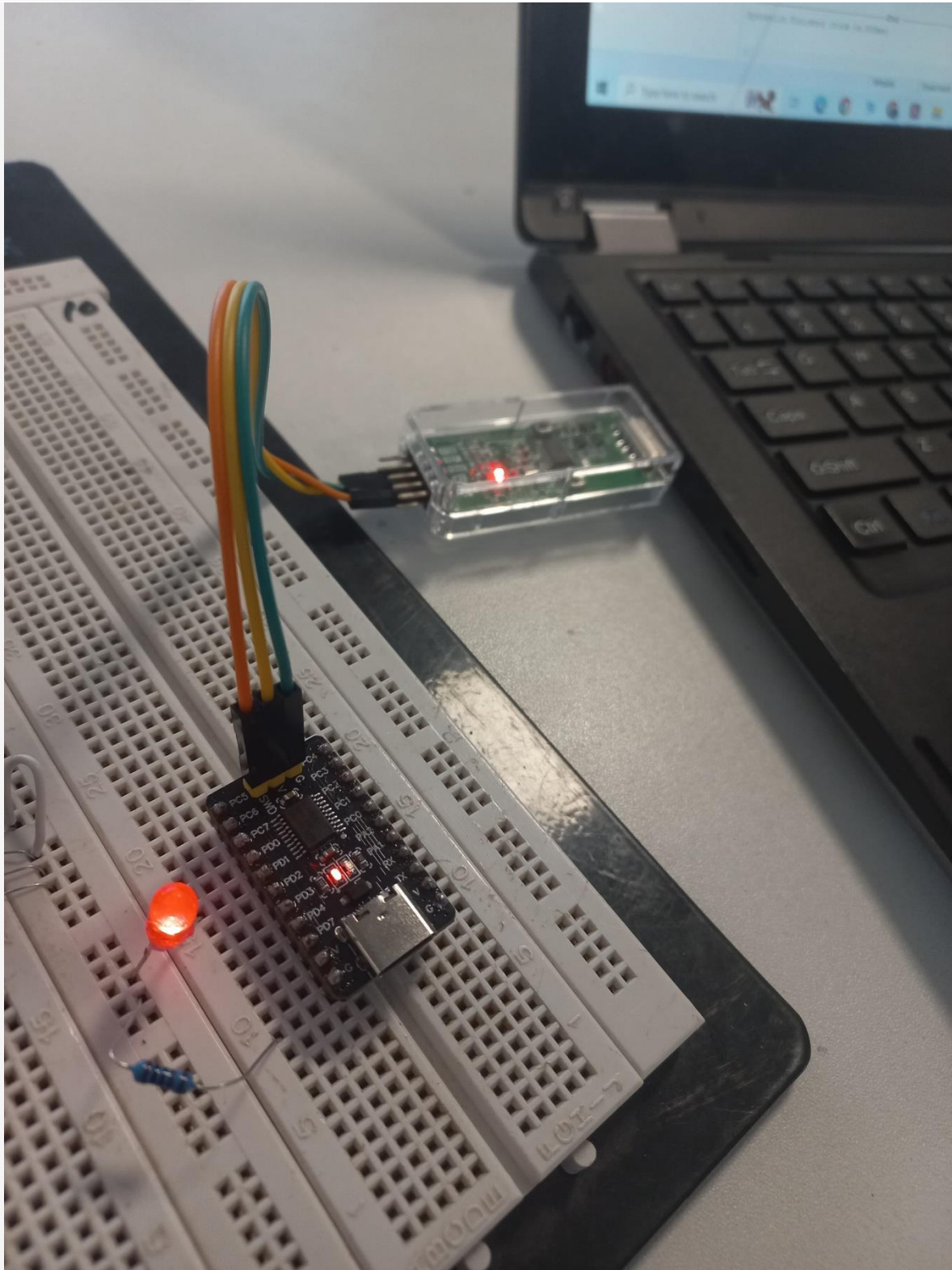
16:54:23 Build Finished. 0 errors, 0 warnings. (took 612ms)

Target Mode is set to RV-Link mode for RISC V and DAP for the ARM Processors.





LED Blink ON State:



LED Blink OFF State:

