

Chapter 3:

CH32V003 PROGRAMMING: HOW TO USE GPIO AS INPUT

We have seen how to use GPIO as Output in the previous chapter. Now we will use GPIO as an input. There are total **18** Input/output pins available on the MCU board.

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 input/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.

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

Not all the pins are 5V volt tolerant (FT is 5V tolerant). See the pinout table in the datasheet as shown in the image below. Only PC1, PC2, PC5 and PC6 are 5V tolerant. (You should verify from the data sheet of the MCU).

16	10	7	-	PC0	I/O	PC0	T2CH3	NSS_1/UTX_3/T2CH3_2 /T1CH3_1
1	11	8	5	PC1	I/O/FT	PC1	SDA/NSS	T1BKIN_1/T2CH4_1 T2CH1ETR ⁽¹⁾ _2/URX_3 /T2CH1ETR ⁽¹⁾ _3/T1BKIN_3
2	12	9	6	PC2	I/O/FT	PC2	SCL/URTS/T1BKIN	AETR_1/T2CH2_1 /T1ETR_3/URTS_1 /T1BKIN_2
3	13	10	-	PC3	I/O	PC3	T1CH3	T1CH1N_1/UCTS_1 /T1CH3_2/T1CH1N_3
4	14	11	7	PC4	I/O/A	PC4	T1CH4/MCO/A2	T1CH2N_1/T1CH4_2 /T1CH1_3
-	15	12	-	PC5	I/O/FT	PC5	SCK/T1ETR	T2CH1ETR ⁽¹⁾ _1/SCL_2 /SCL_3/UCK_3/T1ETR_1 /T1CH3_3/SCK_1
5	16	13	-	PC6	I/O/FT	PC6	MOSI	T1CH1_1/UCTS_2/SDA_2 /SDA_3/UCTS_3/T1CH3N_ 3 /MOSI_1
6	17	14	-	PC7	I/O	PC7	MISO	T1CH2_1/URTS_2 /T2CH2_3/URTS_3 /T1CH2_3/MISO_1
7	18	15	8	PD1	I/O/A	PD1	SWIO/T1CH3N/AETR2	SCL_1/URX_1/T1CH3N_1 /T1CH3N_2
-	19	16	-	PD2	I/O/A	PD2	T1CH1/A3	T2CH3_1/T1CH2N_3 /T1CH1_2
-	20	17	-	PD3	I/O/A	PD3	A4/T2CH2/AETR/UCTS	T2CH2_2/T1CH4_1

Note: 1. TIM2_CH1, TIM2_ETR;

2. The value after the underline of the remapping function indicates the configuration value of the corresponding bit in the AFIO register. For example: T1CH4_3 indicates that the corresponding bit of the AFIO register is configured as 11b;

3. Explanation of table abbreviations:

I = TTL/CMOS level Schmitt input.

O = CMOS level tri-state output.

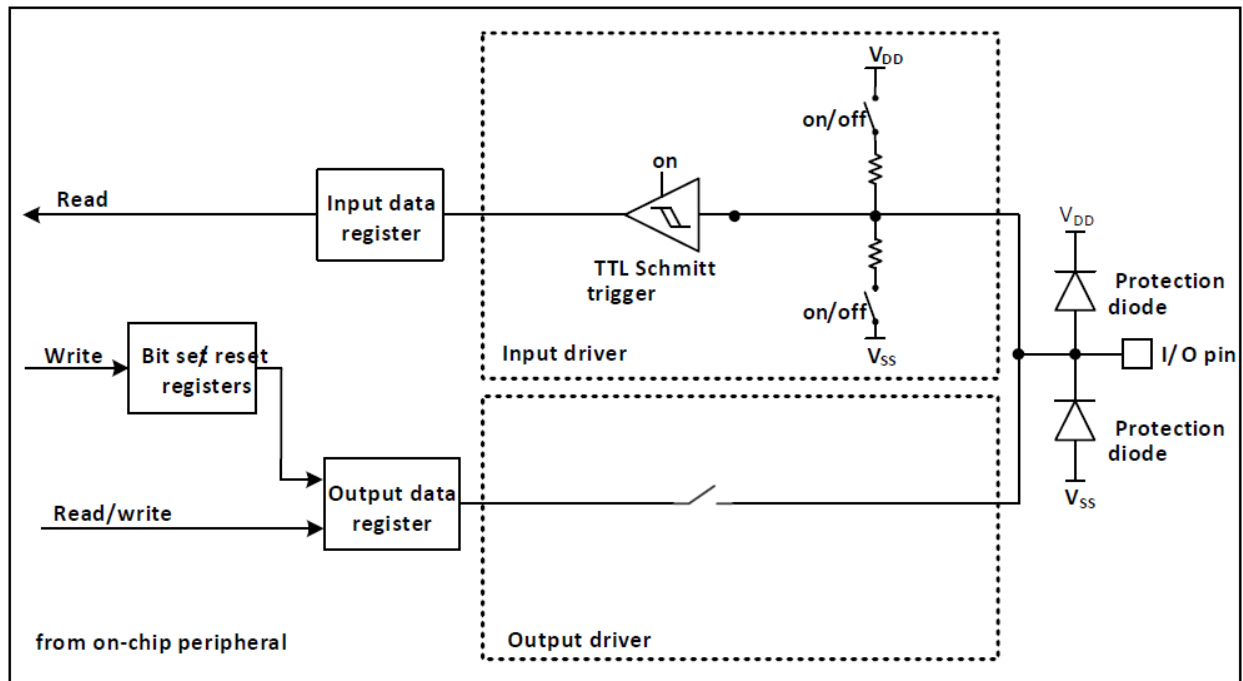
P = power supply.

FT = 5V tolerant.

A = Analog signal input or output.

NOTE: 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.

4.3 Input Configuration:



GPIO module input configuration structure block diagram

When the IO port is configured in input mode, the output driver is **disconnected**, the input pull-up and pulldown are selectable, and no multiplexed functions or analog inputs are connected. The data on each IO port is sampled into the input data register at each AHB clock, and the level status of the corresponding pin is obtained by reading the corresponding bit of the input data register.

4.4 GPIO as Input Example Code for CH32V003

In order to configure any pin as GPIO as input, 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.

For any configuration, clock for that GPIO port needs to be enabled first.

Two parameters are there for any GPIO when configuring as output

- **GPIO_Pin**: this is to define which Pin, for example for D3 it will be GPIO_Pin_3
- **GPIO_Mode**: to configure if the Input will be with internal **pull-up resistance** (**GPIO_Mode_IPU**) or **pull-down resistance** (**GPIO_Mode_IPD**) or input which is floating (**GPIO_Mode_IN_FLOATING**).

```
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_3; // Defines which Pin to configure
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // Defines Output Type
    GPIO_Init(GPIOD, &GPIO_InitStructure);
}
```

In order to read the GPIO input there are two functions

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin);
uint16_t GPIO_ReadInputData(GPIO_TypeDef *GPIOx);
```

- **GPIO_ReadInputDataBit** is used to get the status of single bit/pin whether it is high or low.
- **GPIO_ReadInputData** function is used to read the full port, example if D0-7 data port need to be read, this function could be used.

4.5 GPIO Input in Polling Mode:

Let's now see how we can write the code to check the status GPIO Input pin and then glow the LED as per the status. So, we will see if GPIO Input pin is Low we will glow the LED and if it is High LED will be OFF.

D3 is the input and D4 is output for LED.

```

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_3; // Defines which Pin to
configure
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // Defines Output Type
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4; // 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);

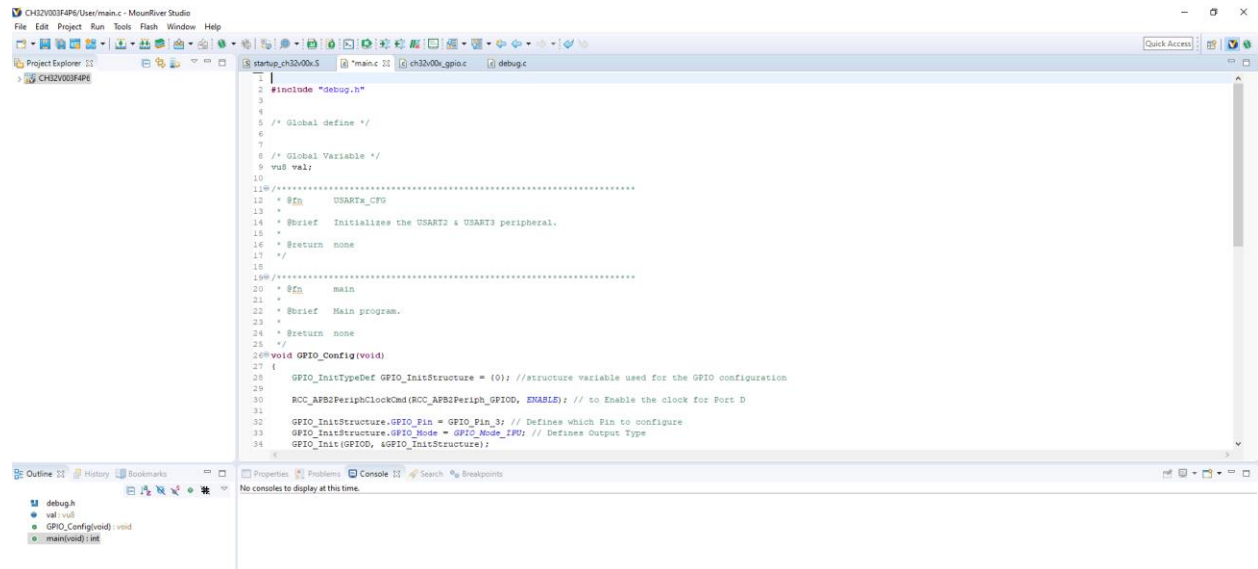
}

/*****
 * @fn      main
 *
 * @brief    Main program.
 *
 * @return    none
 */
int main(void)
{
    uint8_t GPIOInputStatus = 0;
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    SystemCoreClockUpdate();
    Delay_Init();
    GPIO_Config();

    while(1)
    {
        GPIOInputStatus = GPIO_ReadInputDataBit(GPIOD,GPIO_Pin_3);
        if(GPIOInputStatus == 0)
        {
            GPIO_WriteBit(GPIOD, GPIO_Pin_4, RESET);
        }
        else
        {
            GPIO_WriteBit(GPIOD, GPIO_Pin_4, SET);
        }
        Delay_Ms(100);
    }
}

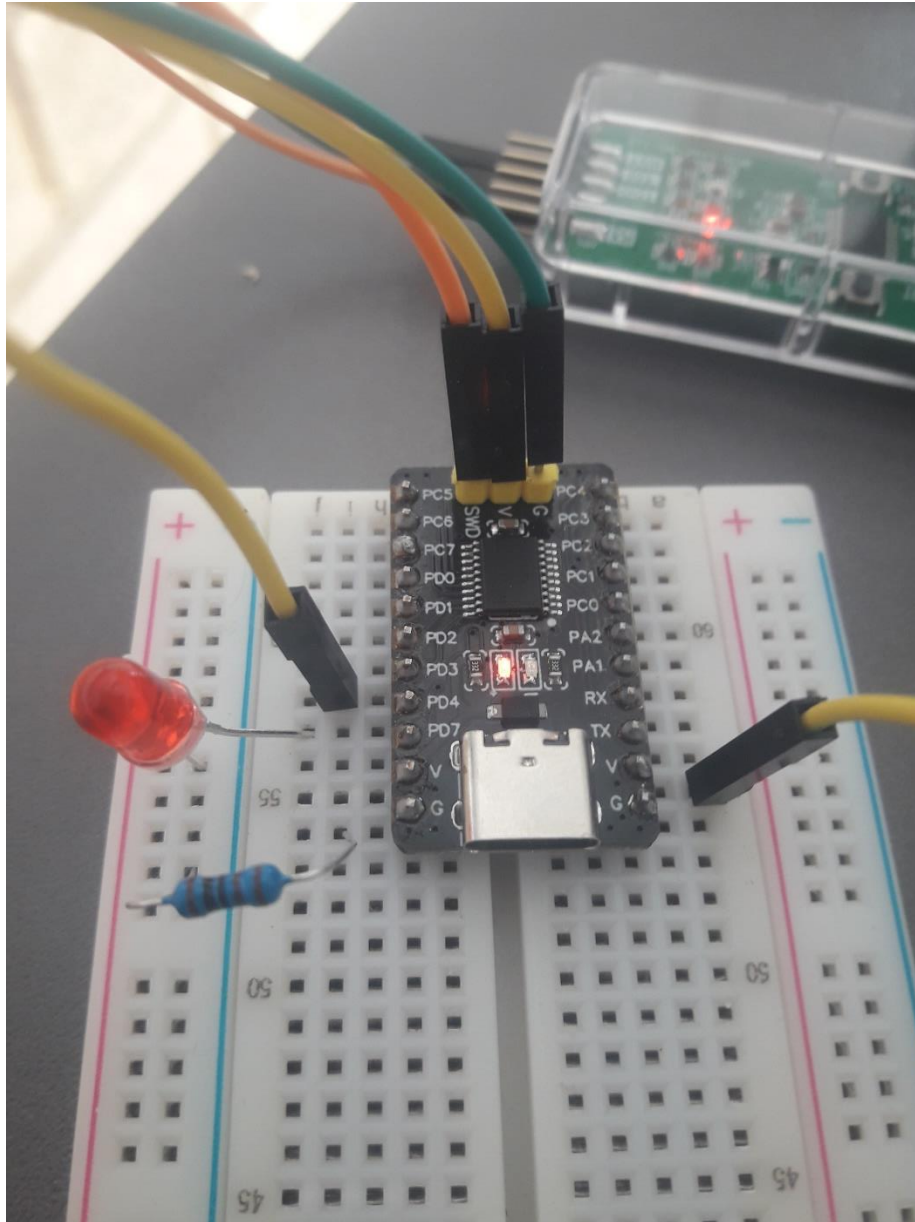
```

sMounRiver Studio

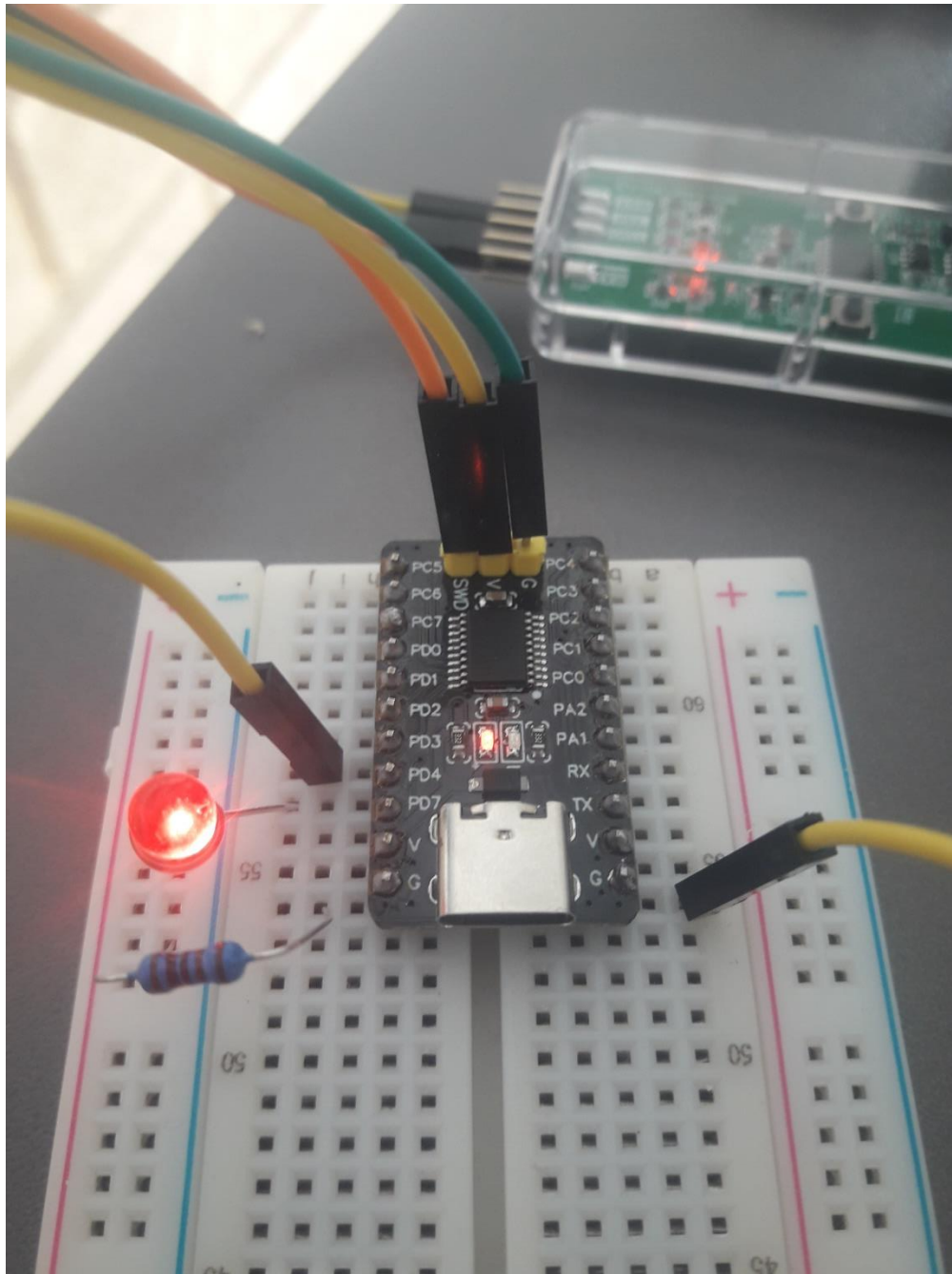


4.6 Hardware DEMO:

Here the Pin PD3 is connected to Vcc (3.3 V), therefore when we read pin PD3 its in HIGH-state and the MCU turns PD4 into LOW-state.



Here the Pin PD3 is connected to GND (0 V), therefore when we read pin PD3 its in LOW-state and the MCU turns PD4 into HIGH-state.



4.7 GPIO as External Interrupt Input

```
/****** (C) COPYRIGHT *****  
*****  
* File Name      : main.c  
* Author         : WCH  
* Version        : V1.0.0  
* Date          : 2023/12/22  
* Description    : Main program body.  
  
*****  
****  
* Copyright (c) 2021 Nanjing Qinheng Microelectronics Co., Ltd.  
* Attention: This software (modified or not) and binary are used for  
* microcontroller manufactured by Nanjing Qinheng Microelectronics.  
  
*****  
**/  
  
/*  
* @Note  
* External lines trigger ADC conversion routine:  
* ADC channel 2 (PC4) - rule group channel, external trigger pin (PD3) high  
level  
* triggers EXTI line 3 event, In this mode, an ADC conversion is triggered by  
an  
* event on the EXTI line 3, and an EOC interrupt is generated after the  
conversion  
* is completed.  
*  
*/  
  
#include "debug.h"  
  
/* Global Variable */  
  
/******  
* @fn          ADC_Function_Init  
*  
* @brief       Initializes ADC collection.  
*  
* @return      none  
*/  
void ADC_Function_Init(void)  
{  
    ADC_InitTypeDef  ADC_InitStructure = {0};  
    GPIO_InitTypeDef GPIO_InitStructure = {0};  
    NVIC_InitTypeDef NVIC_InitStructure = {0};  
  
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);  
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);  
    RCC_ADCCLKConfig(RCC_PCLK2_Div8);  
  
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;  
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;  
    GPIO_Init(GPIOC, &GPIO_InitStructure);  
}
```

```

    ADC_DeInit(ADC1);
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
    ADC_InitStructure.ADC_ExternalTrigConv =
ADC_ExternalTrigConv_Ext_PD3_PC2;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &ADC_InitStructure);

    ADC_RegularChannelConfig(ADC1, ADC_Channel_2, 1,
ADC_SampleTime_241Cycles);
    ADC_ExternalTrigConvCmd(ADC1, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = ADC_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);

    ADC_ExternalTrig_DLY(ADC1, ADC_ExternalTrigInjec_DLY, 0x10); /* external
trigger sources delay time */
    ADC_Calibration_Vol(ADC1, ADC_CALVOL_50PERCENT);
    ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);
    ADC_Cmd(ADC1, ENABLE);

    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));
}

/*****
* @fn      EXTI_Event_Init
*
* @brief   Initializes EXTI.
*
* @return  none
*/
void EXTI_Event_Init(void)
{
    EXTI_InitTypeDef EXTI_InitStructure = {0};
    GPIO_InitTypeDef GPIO_InitStructure = {0};

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);

    GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource3);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    EXTI_InitStructure.EXTI_Line = EXTI_Line3;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Event;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;

```

```

EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);
}

/*****
 * @fn      main
 *
 * @brief    Main program.
 *
 * @return   none
 */
int main(void)
{
    SystemCoreClockUpdate();
    Delay_Init();
#ifdef 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());

    EXTI_Event_Init();
    ADC_Function_Init();

    while(1);
}

void ADC1_IRQHandler(void) __attribute__((interrupt("WCH-Interrupt-fast")));

/*****
 * @fn      ADC1_IRQHandler
 *
 * @brief    ADC1 Interrupt Service Function.
 *
 * @return   none
 */
void ADC1_IRQHandler(void)
{
    u16 ADC_val;

    if(ADC_GetITStatus(ADC1, ADC_IT_EOC))
    {
        printf("ADC Extline trigger conversion...\r\n");
        ADC_val = ADC_GetConversionValue(ADC1);
        printf("ADC-%04d\r\n", ADC_val);
    }

    ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
}

```

OUTPUT

For reading the serial port we used Hercules Setup Utility. For reading serial port the Baudrate is set same at which USART “printf” is initialized. Then click on Open to open the port serial communication.

