图片包含 游戏机, 夜空

描述已自动生成 徽标

描述已自动生成

# 嵌入式系统技术 课程实验报告

## 实验名称 输入捕获开发实验

#### 软件与通信学院 20通信工程1班 班

成 绩

姓名 孟凡钧学号 20434020109 同作者 实验日期 年 月 日

实验过程心得体会：

指导教师签字：

#### 实验报告六

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 班级 | 20通信工程1班 | | 姓名 | 孟凡钧 | 学号 | 20434020109 |
| 日期 |  | | 教室 | 407 | 成绩 |  |
| 实验名称 | |  | | | | |
| 实验目的 | |  | | | | |
| 设备与材料 | |  | | | | |
| 实验内容 | | 一、列出所有通用定时器的输入捕获引脚。    二、说明如何测量输入信号的电平持续时间与信号频率。  当有效的脉冲变化被捕获那么计数器的值就会自动被捕获/比较寄存器捕获“，如果我们想要测量高电平的持续时间，我们可以当上升沿出现时我们将计数器的初始计数值置0，下降沿出现时我们将此时的计数器的值加载到捕获/比较计数器中进行捕获读取。最后”（捕获的值-计数器初始值）\*计数器的单位计数时长=高电平持续时间“。  三、详细说明程序中各项初始化过程。 | | | | |

四、程序代码及运行结果图片（可附页）

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2022 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "tim.h"

#include "usart.h"

#include "gpio.h"

#include <stdio.h>

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

uint32\_t IC\_TIMES; // 捕获次数，单位1ms

uint8\_t IC\_START\_FLAG; // 捕获开始标志，1：已捕获到高电平；0：还没有捕获到高电平

uint8\_t IC\_DONE\_FLAG; // 捕获完成标志，1：已完成一次高电平捕获

uint16\_t IC\_VALUE; // 输入捕获的捕获值

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

/\* USER CODE BEGIN PFP \*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

uint32\_t time = 0;

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_TIM2\_Init();

MX\_USART1\_UART\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_TIM\_IC\_Start\_IT(&htim2,TIM\_CHANNEL\_1); //开启TIM2的捕获通道1

\_\_HAL\_TIM\_ENABLE\_IT(&htim2,TIM\_IT\_UPDATE); //使能更新中断

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

HAL\_Delay(10);

if(IC\_DONE\_FLAG == 1) // 如果完成一次高电平捕获

{

IC\_DONE\_FLAG = 0; // 标志清零

time = IC\_TIMES \* 1000; // 脉冲时间为捕获次数 \* 1000us

time += IC\_VALUE; // 加上捕获时间（小于1ms的部分）

printf("High level: %d us\n", time);

}

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)

{

Error\_Handler();

}

}

/\* USER CODE BEGIN 4 \*/

// 定时器计数溢出中断处理回调函数

void HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim)

{

if(IC\_DONE\_FLAG == 0) // 未完成捕获

{

if(IC\_START\_FLAG == 1) // 已经捕获到了高电平

{

IC\_TIMES++; // 捕获次数加一

}

}

}

//定时器输入捕获中断处理回调函数

void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim)// 捕获中断发生时执行

{

if(IC\_DONE\_FLAG == 0) // 未完成捕获

{

if(IC\_START\_FLAG == 1) // 原来是高电平，现在捕获到一个下降沿

{

IC\_VALUE = HAL\_TIM\_ReadCapturedValue(htim, TIM\_CHANNEL\_1); // 获取捕获值

TIM\_RESET\_CAPTUREPOLARITY(htim,TIM\_CHANNEL\_1); // 先清除原来的设置

TIM\_SET\_CAPTUREPOLARITY(htim,TIM\_CHANNEL\_1,TIM\_ICPOLARITY\_RISING);// 配置为上升沿捕获

IC\_START\_FLAG = 0; // 标志复位

IC\_DONE\_FLAG = 1; // 完成一次高电平捕获

}

else // 捕获还未开始，第一次捕获到上升沿

{

IC\_TIMES = 0; // 捕获次数清零

IC\_VALUE = 0; // 捕获值清零

IC\_START\_FLAG = 1; // 设置捕获到了上边沿的标志

TIM\_RESET\_CAPTUREPOLARITY(htim,TIM\_CHANNEL\_1); // 先清除原来的设置

TIM\_SET\_CAPTUREPOLARITY(htim,TIM\_CHANNEL\_1,TIM\_ICPOLARITY\_FALLING);// 配置为下降沿捕获

}

\_\_HAL\_TIM\_SET\_COUNTER(htim,0); // 定时器计数值清零

}

}

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

\_\_disable\_irq();

while (1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/