## 杭州电子科技大学创新实践实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学院 | 计算机学院 | 班级 |  | 学号 |  |
| 姓名 |  | 日期 | 2024.05.13 | 成绩 |  |
| 实验题目 | 电机驱动实现3:Timer产生PWM | | | | |
|  | | | | | |
| 实验目的 | 1、学习网站上模块13的视频部分，下载“讲解”“实验”pdf，  2、完成maze代码中的Lab13\_timer  3、补全函数Motor\_Init，Motor\_Forward，Motor\_Backward，Motor\_Left和Motor\_Right函数，需要用Timer实现PWM | | | | |
| 硬件原理 | MSP432板卡上有TimerA定时器模块，它的管脚具有输入捕获和输出比较功能。以用Timer-A实现两个PWM输出上下计数模式为例，定时器初始计数值为0，当计数值达到7时，P2.4=1发生toggle反转，P2.4=0；当计数值达到10（最高点）时，进行Reset操作，计数值开始下降；当计数值再次达到7时，P2.4再次发生toggle翻转，P2.4=0；最后计数值回到0，定时器的一个周期结束。 | | | | |
| 关键代码及注释 | **#include** <stdint.h>  **#include** "msp.h"  **#include** "../inc/CortexM.h"  **#include** "../inc/PWM.h"  // \*\*\*\*\*\*\*Lab 13 solution\*\*\*\*\*\*\*  // ------------Motor\_Init------------  // Initialize GPIO pins for output, which will be  // used to control the direction of the motors and  // to enable or disable the drivers.  // The motors are initially stopped, the drivers  // are initially powered down, and the PWM speed  // control is uninitialized.  // Input: none  // Output: none  **void** **Motor\_Init**(**void**){  // write this as part of Lab 13  P1->SEL0 &= ~(BIT6+BIT7); // P1.6 & 1.7 control direction  P1->SEL1 &= ~(BIT6+BIT7); // P1.6 & 1.7 as GPIO  P1->DIR |= BIT6+BIT7; // P1.6 & 1.7 as output  P1->OUT &= ~(BIT6+BIT7); // P1.6 & 1.7 output low  P3->SEL0 &= ~(BIT6+BIT7); // nSLEEP P3.6 & 3.7  P3->SEL1 &= ~(BIT6+BIT7); // P3.6 & 3.7 as GPIO  P3->DIR |= BIT6+BIT7; // P3.6 & 3.7 as output  P3->OUT &= ~(BIT6+BIT7); // P3.6 & 3.7 output high, put drivers to sleep  }  // ------------Motor\_Stop------------  // Stop the motors, power down the drivers, and  // set the PWM speed control to 0% duty cycle.  // Input: none  // Output: none  **void** **Motor\_Stop**(**void**){  // write this as part of Lab 13  P3->OUT &= ~(BIT6+BIT7); // Power down the drivers  PWM\_Duty3(0); // Set PWM duty cycle to 0  PWM\_Duty4(0);  }  // ------------Motor\_Forward------------  // Drive the robot forward by running left and  // right wheels forward with the given duty  // cycles.  // Input: leftDuty duty cycle of left wheel (0 to 14,998)  // rightDuty duty cycle of right wheel (0 to 14,998)  // Output: none  // Assumes: Motor\_Init() has been called  **void** **Motor\_Forward**(uint16\_t leftDuty, uint16\_t rightDuty){  // write this as part of Lab 13  P1->OUT &= ~(BIT6+BIT7); // Both wheels moving forward  PWM\_Init34(15000,rightDuty,leftDuty); // Set duty cycle  P3->OUT |= BIT6+BIT7; // Enable the drivers  }  // ------------Motor\_Right------------  // Turn the robot to the right by running the  // left wheel forward and the right wheel  // backward with the given duty cycles.  // Input: leftDuty duty cycle of left wheel (0 to 14,998)  // rightDuty duty cycle of right wheel (0 to 14,998)  // Output: none  // Assumes: Motor\_Init() has been called  **void** **Motor\_Right**(uint16\_t leftDuty, uint16\_t rightDuty){  // write this as part of Lab 13  P1->OUT &= ~BIT7; // Left wheel moving forward  P1->OUT |= BIT6; // Right wheel moving backward  PWM\_Init34(15000,rightDuty,leftDuty); // Set duty cycle  P3->OUT |= BIT6+BIT7; // Enable the drivers  }  // ------------Motor\_Left------------  // Turn the robot to the left by running the  // left wheel backward and the right wheel  // forward with the given duty cycles.  // Input: leftDuty duty cycle of left wheel (0 to 14,998)  // rightDuty duty cycle of right wheel (0 to 14,998)  // Output: none  // Assumes: Motor\_Init() has been called  **void** **Motor\_Left**(uint16\_t leftDuty, uint16\_t rightDuty){  // write this as part of Lab 13  P1->OUT |= BIT7; // Left wheel moving backward  P1->OUT &= ~BIT6; // Right wheel moving forward  PWM\_Init34(15000,rightDuty,leftDuty); // Set duty cycle  P3->OUT |= BIT6+BIT7; // Enable the drivers  }  // ------------Motor\_Backward------------  // Drive the robot backward by running left and  // right wheels backward with the given duty  // cycles.  // Input: leftDuty duty cycle of left wheel (0 to 14,998)  // rightDuty duty cycle of right wheel (0 to 14,998)  // Output: none  // Assumes: Motor\_Init() has been called  **void** **Motor\_Backward**(uint16\_t leftDuty, uint16\_t rightDuty){  // write this as part of Lab 13  P1->OUT |= BIT6+BIT7; // Both wheels moving backward  PWM\_Init34(15000,rightDuty,leftDuty); // Set duty cycle  P3->OUT |= BIT6+BIT7; // Enable the drivers  }  **#include** "msp.h"  **#include** "..\inc\bump.h"  **#include** "..\inc\Clock.h"  **#include** "..\inc\SysTick.h"  **#include** "..\inc\CortexM.h"  **#include** "..\inc\LaunchPad.h"  **#include** "..\inc\Motor.h"  **#include** "..\inc\TimerA1.h"  **#include** "..\inc\TExaS.h"  // Driver test  **void** **TimedPause**(uint32\_t time){  Clock\_Delay1ms(time); // run for a while and stop  Motor\_Stop();  **while**(LaunchPad\_Input()==0); // wait for touch  **while**(LaunchPad\_Input()); // wait for release  }  **int** **Program13\_1**(**void**){  Clock\_Init48MHz();  LaunchPad\_Init(); // built-in switches and LEDs  Bump\_Init(); // bump switches  Motor\_Init(); // your function  **while**(1){  TimedPause(4000);  Motor\_Forward(7500,7500); // your function  TimedPause(2000);  Motor\_Backward(7500,7500); // your function  TimedPause(3000);  Motor\_Left(5000,5000); // your function  TimedPause(3000);  Motor\_Right(5000,5000); // your function  }  }  // Test of Periodic interrupt  **#define** REDLED (\*((**volatile** uint8\_t \*)(0x42098060)))  **#define** BLUELED (\*((**volatile** uint8\_t \*)(0x42098068)))  uint32\_t Time;  **void** **Task**(**void**){  REDLED ^= 0x01; // toggle P2.0  REDLED ^= 0x01; // toggle P2.0  Time = Time + 1;  REDLED ^= 0x01; // toggle P2.0  }  **int** **Program13\_2**(**void**){  Clock\_Init48MHz();  LaunchPad\_Init(); // built-in switches and LEDs  TimerA1\_Init(&Task,50000); // 10 Hz  EnableInterrupts();  **while**(1){  BLUELED ^= 0x01; // toggle P2.1  }  }  **int** **main**(**void**){  // write a main program that uses PWM to move the robot  // like Program13\_1, but uses TimerA1 to periodically  // check the bump switches, stopping the robot on a collision  Motor\_Forward(5000,5000);    **while**(1){    }  } | | | | |
| 实验步骤 | 补全函数Motor\_Init，Motor\_Forward，Motor\_Backward，Motor\_Left和Motor\_Right函数，并设置P2.6和P2.7管脚为定时器模块，用于控制小车轮子转动频率，将代码烧录到小车上，运行小车。 | | | | |
| 实验结果 | 小车前进，后退，只让左轮转动和只让右轮转动功能都能实现，定时器正常工作，轮子运行状态稳定 | | | | |
| 思考与反馈 | 周期计算是一个简单重复的事情，和上一节用模拟PWM的方法让CPU进行周期计算相比，使用定时器完成这项工作，可以让CPU空出来做其他更有意义的事情，提高系统的工作效率。 | | | | |