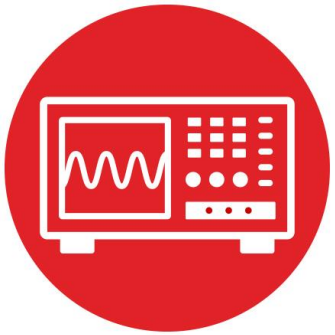

模块 3

实验：PWM 控制电机

实验：PWM 控制电机

3.0 目标

本实验的目的是开发转动电机所需的软件。该软件可以使用**脉冲宽度调制**（PWM）改变传送到每个电机的**电功率**。 在这个模块中，

1. 您将学习 MSP432 Timer_A 模块。
2. 您将配置 Timer A0 以创建两个 PWM 输出。
3. 您将配置 Timer A1 以创建一个额外的周期性中断。
4. 您将开发用于移动的低层机器人命令。

小知识：PWM 是微控制器影响其世界的有效的手段。它是有效的，因为将定时器重载值设置为 10000 将创建一个基本上为 14 位精度的输出。这是有效的，因为定时器和数字开关电路（DRV8838）的成本远低于等效的模拟放大器。

3.1 入门

3.1.1 从下面的软件工程起步

浏览以下 3 个项目：
PWMSine (使用 PWM 和定时器来创建正弦波输出)
PeriodicTimerA0Ints (使用 Timer_A0 创建一个周期性中断)
PWM_Motors（视频中 PWM 控制电机对应的程序）
Lab_Motors (本实验开始项目)

注：您将无法在机器人上运行 PeriodicTimerA0Ints 项目，因为该项目使用 Timer_A0，您需要将 Timer_A0 用于机器人的两个 PWM 输出。

3.1.2 参考资料

MSP432P4xx Technical Reference Manual, Timer_A (SLAU356)
MSP432P401R Datasheet, msp432p401m.pdf (SLAS826)
JN_BUS_2RSLKB01_V0.3.pdf Data sheet for power board
drv8838.pdf Data sheet for the H-bridge driver

3.1.3 阅读材料

Volume 1 Sections 8.7, and 9.7
Embedded Systems: Introduction to the MSP432 Microcontroller",
Volume 2 Sections 6.2, 6.3, and 6.5
Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller"

3.1.4 本实验所需的组件

数量	组件描述	制造商	型号
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	Al Alloy Chassis & Structure	JiangNiu	
1	**总线驱动板**	JiangNiu	JN_BUS_2RSLKB01_V0.3
2	ϕ65mm Rubber wheel	JiangNiu	
2	Encoder+DC Motor+Gear reducer	JiangNiu	
1	Battery Holder	JiangNiu	
2	Li Battery 3.7V 600mAh	JiangNiu	

实验：PWM 控制电机

3.1.5 所需 实验设备

示波器 (至少 10 kHz 采样的 1 个或 2 个通道)
逻辑分析仪 (至少 10 kHz 采样的 4 个通道)

3.2 系统设计要求

本实验的第一个目标是编写可以调整两个电机的应用功率的软件。您将在 P2.6 和 P2.7 引脚上创建 PWM 输出，这些引脚连接到电机驱动板的 PWML 和 PWMR（EN 输入到 DRV8838）。两个输出的周期应固定为 7 ms（70 Hz）。但是，软件应该能够独立地将 EN 引脚的占空比设置为 0 到 14,998（0 到 99.99%）。在 70 Hz 时，电机不会响应个别的高点和低点；相反，电机将响应平均水平。 更具体地说，传递的功率将是

$$P = V * I * duty/10000; \quad 0 \le duty \le 14,998$$

其中 **V** 是电压，**I** 是电流，如实验 12 中先前所测量的。

本实验的第二个目标是使用 Timer_ A1 创建一个额外的周期性中断。高级别主程序将在运行时使用函数指针初始化此周期性中断，从而提供抽象和代码重用。

本实验的结果是一个直线行驶的系统，直到其中一个碰撞传感器检测到碰撞。 但是，此解决方案将需要非常少的软件开销。

3.3 实验准备

请参阅 DRV8838 的数据手册，了解这六个信号的软件输出值如何影响电机性能，如图 1 所示。

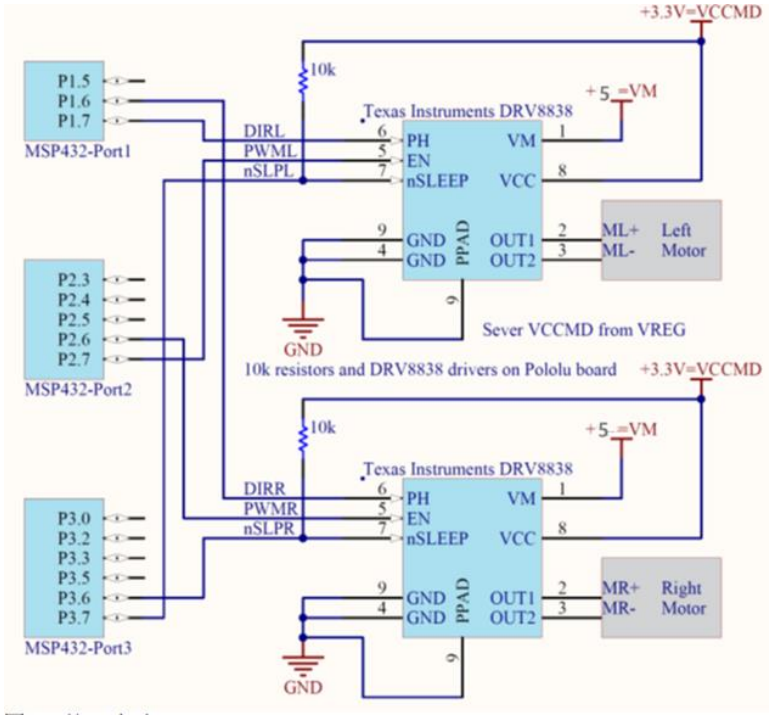

图 1. 接口电路

LaunchPad	MDPDB	DRV8838	描述
P1.6	DIRR	PH	Right Motor Direction
P3.6	nSLPR	nSLEEP	Right Motor Sleep
P2.6	PWMR	EN	Right Motor PWM
P1.7	DIRL	PH	Left Motor Direction
P3.7	nSLPL	nSLEEP	Left Motor Sleep
P2.7	PWML	EN	Left Motor PWM

实验：PWM 控制电机

3.4 实验步骤

3.4.1 底层软件驱动

使用 Timer_A0 创建两个 PWM 输出的功能。这套功能将控制机器人上的两个轮子。当 PWM 激活时，两个电机的 PWM 均为 70Hz（7ms），但具有独立的占空比。驱动程序的原型是：

void Motor_Init(void);
Initializes the 6 lines and Timer A0 and puts driver to sleep
Returns right away

void Motor_Stop(void);
Stops both motors, puts driver to sleep
Returns right away

void Motor_Forward(uint16_t leftDuty, uint16_t rightDuty)
Drives left motor forward at **leftDuty** (0 to 14,998)
Drives right motor forward at **rightDuty** (0 to 14,998)
The motors run until software issues another command
Returns right away

void Motor_Backward(uint16_t leftDuty, uint16_t rightDuty)
Drives left motor backward at **leftDuty** (0 to 14,998)
Drives right motor backward at **rightDuty** (0 to 14,998)
The motors run until software issues another command
Returns right away

void Motor_Left(uint16_t leftDuty, uint16_t rightDuty)
Drives left motor backward at **leftDuty** (0 to 14,998)
Drives right motor forward at **rightDuty** (0 to 14,998)
The motors run until software issues another command
Returns right away

void Motor_Right(uint16_t leftDuty, uint16_t rightDuty)
Drives left motor forward at **leftDuty** (0 to 14,998)
Drives right motor backward at **rightDuty** (0 to 14,998)
The motors run until software issues another command
Returns right away

3.4.2 电机测试

调试电机软件。将机器人放在木块上，使车轮不接触地面，并使用 **Program13_1** 等程序测试低级别电机功能。

```
// 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
  }
}
```

将机器人放在地面并尝试调整 ***Motor_Forward*** 和 ***Motor_Backward*** 调用中的每个 7500 参数，以便机器人沿直线移动。将调用中的 5000 参数调整为 ***Motor_Left*** 和 ***Motor_Right***，并**暂停** 3000 参数，使机器人旋转 90 度。

注： 调整这些参数以运行机器人开环几乎是不可能的。要求您尝试解决一个不可能的问题将激发对输入的需求并创建一个闭环控制器。

3.4.3 周期性中断

编写软件以使用 Timer_A1 创建额外的周期性中断。如果使用 12 MHz SMCLK 并除以 24，则 16 位定时器的时钟频率为 500 kHz。在此时钟速率下，可以创建的最慢中断大约为 130 ms（65535 *2μs）。您可以使用 **Program13_2** 之类的程序来

实验：PWM 控制电机

测试此驱动程序。注意使用位带来移除通常在共享全局上使用读 - 修改 - 写序列发生的临界区。

```
// 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
  }
}
```

使用双跟踪范围观察 P2.0（中断线程）和 P2.1（主线程）。触发中断信号并使用 P2.1 振荡中的间隙来估计服务定时器 A1 中断所需的时间。

在单独测试 PWM 和 Timer A1 之后，将它们组合成一个运行机器人的软件系统，如程序 13.1，但使用周期性中断来检查碰撞开关，在碰撞时停止机器人。

3.5 疑难解答

PWM 或中断是不正确的周期：

- 检查定时器时钟源。
- 确保处理器以 48MHz 运行。

PWM 输出没有发生：

- 运行 **PWMSine** 项目以查看硬件是否正常。
- 使用调试器确保 Timer_A0 寄存器已设置。

中断没有发生：

- 运行 **PeriodicTimerA0Ints** 项目并使用调试器观察 Timer A0 寄存器。运行程序并观察 Timer A1 寄存器。
- 使用调试器观察 NVIC 中的寄存器。
- 通过调用 **EnableInterrupts**（）确保 I 位清零；

3.6 请思考

在本节中，我们列出了完成本实验后要考虑的问题。这些问题旨在测试您对本实验中概念的理解。本模块的目标是让您了解 Timer_A 及其用于 PWM 和周期性中断的用途。

- 软件如何选择 Timer_A 的输入时钟？
- 预分频器为 Timer_A 做了什么？ 为什么预分频器很重要（即更改预分频器时会发生什么？）
- 本实验中生成的 PWM 的精度是多少？
- 如果在 **Program13_2** 主程序中循环执行 **P2->OUT ^= 0x04** 会发生什么情况； 相反呢?
- 您如何使用 Timer A1 每秒执行一次定期任务？
- 什么是函数指针？ 为什么在本实验中使用函数指针？

3.7 其它挑战

在本节中，我们列出了您可以执行的其它活动，以进一步探索此模块的概念。 例如，

- 如果您没有电机板，则必须更改软件的运行方式。幸运的是，可以在 P2.4，P2.5，P2.6 或 P2.7 中的任何一个上创建 PWM 输出。
- 现在可以将实验 5（FSM），实验 4（循线传感器检测黑白线）和本实验结合起来，创建一个循线机器人。

实验：PWM 控制电机

3.8 接下来是哪些模块？

1) 本实验中构思的机器人有两个主要限制。
2) 1）软件消耗所有处理器时间。2）电机的速度取决于许多因素，其中大部分因素无法提前预测。 在余下的实验中，我们将解决这些限制。

模块 4) 组合模块 4，5 和 6 创建一个集成的工程，实现循线迷宫挑战。

模块 8) 使用 ADC 连接距离传感器。 两个距离传感器可用于以固定距离和与墙壁固定的角度驱动机器人。

3.9 您应该已经学会

在本节中，我们将回顾您应该在本单元中学到的重要概念：

- 了解电机的电压，电流和功率。
- 能够使用 PWM 输出来调节电机的功率。
- 了解 H 桥的基本操作和目的。
- 知道如何编写和测试底层软件驱动程序。