

2017

Microchip 第十八届

中国技术精英年会

嵌入式控制工程师的盛会

C18H02 FRM2

像专家一样着手为 PIC16F1xxx编写C程序



目标

完成本课程之后，您将...

- 能够使用**MPLAB® X IDE**创建和编译**PIC16F1xxx**项目
- 使用**MPLAB**代码配置器（**MCC**）和**C**语言创建并运行一个简单的程序
- 在基于**PIC16F1xxx**的应用中使用**GPIO**、定时器、**ADC**、**PWM**和**UART**
- 创建一个基于状态机的应用程序

课程安排

- **PIC16F1xxx架构基础知识**
- **汇编语言和C语言基础知识**
- **实验1：使用MPLAB® X IDE创建一个项目**
- **A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁**
- **B部分：通过按键按下操作使LED开始/停止闪烁**
- **C部分：在应用程序中使用ADC、PWM和UART**
- **总结**

问题？

- 完成了**PIC16F1xxx**项目？
- 从头开始完成了**PIC16F1xxx**项目？
- 用**C**语言完成了**PIC16F1xxx**项目？

挑战

- 阅读**200**页以上的数据手册
- 使用其他人的代码
- 数周的学习和编码
- 最后完成的代码有错误？

解决方案

- 从头开始学习编程
- 使用**MCC**生成经测试/带注释的代码
- 无需阅读数据手册
 - 根据需要阅读数据手册
- 学习创建无错误代码！

课程安排

- **PIC16F1xxx架构基础知识**
- 汇编语言和C语言基础知识
- 实验1：使用MPLAB® X IDE创建一个项目
- A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁
- B部分：通过按键按下操作使LED开始/停止闪烁
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

2017

Microchip 第十八届

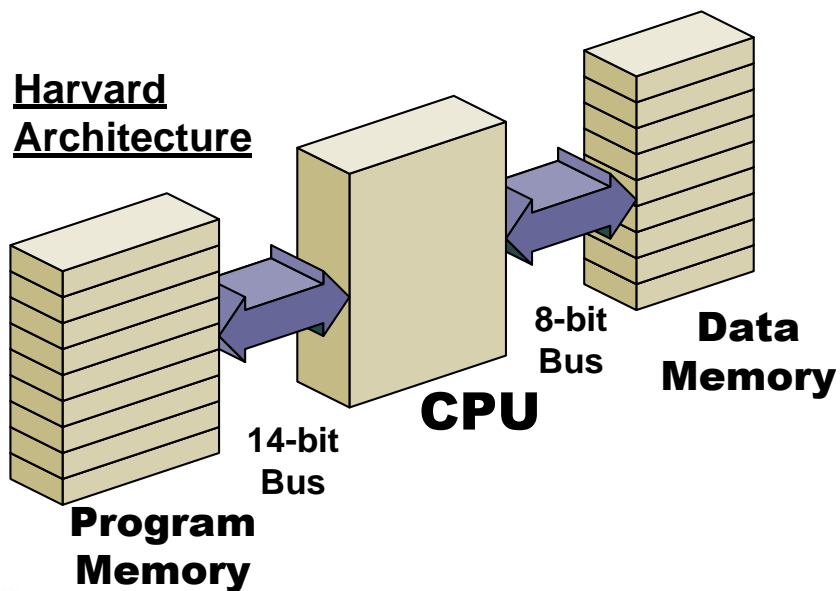
中国技术精英年会

嵌入式控制工程师的盛会

PIC16F1xxx架构基础知识

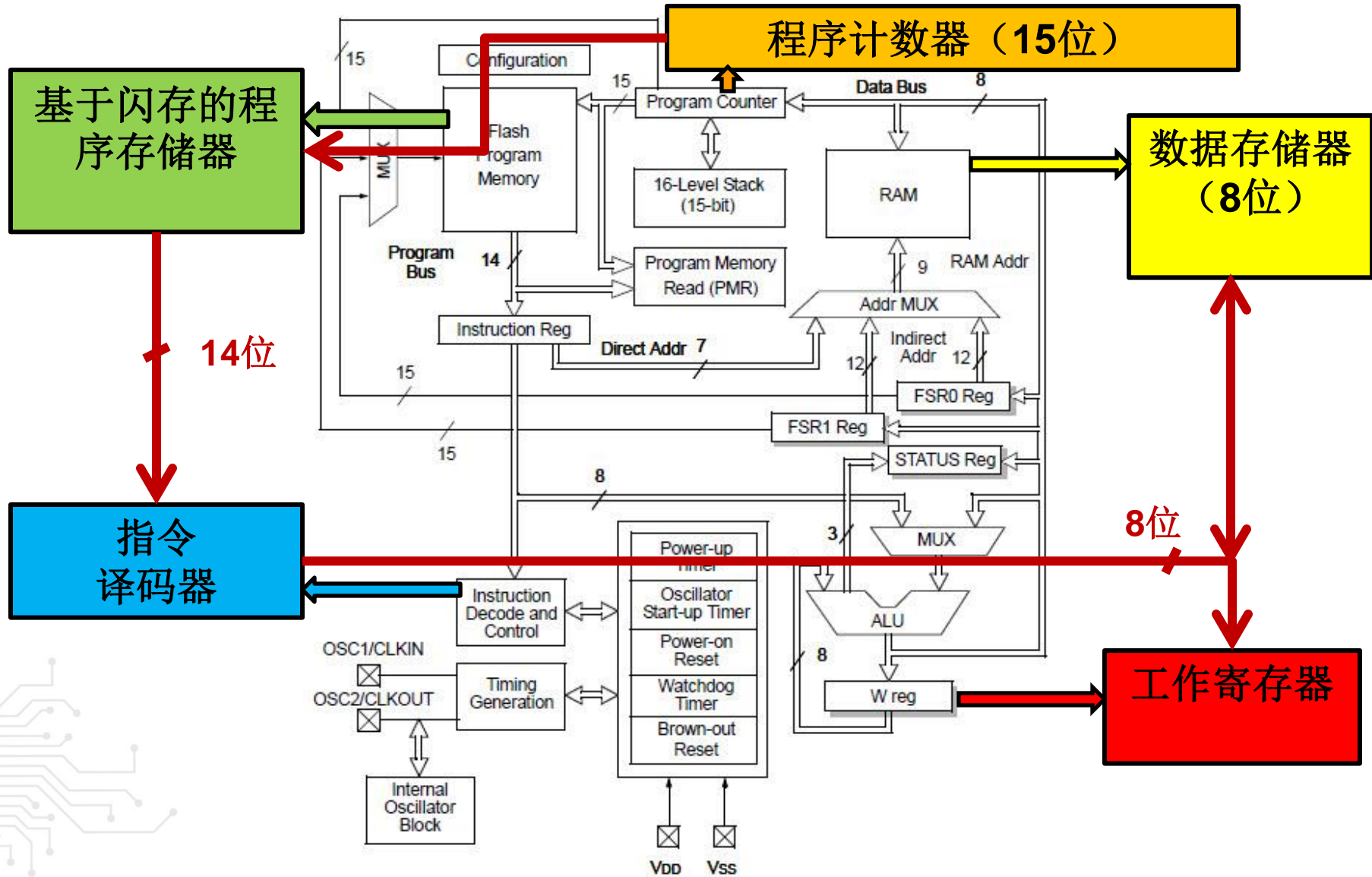


哈佛架构

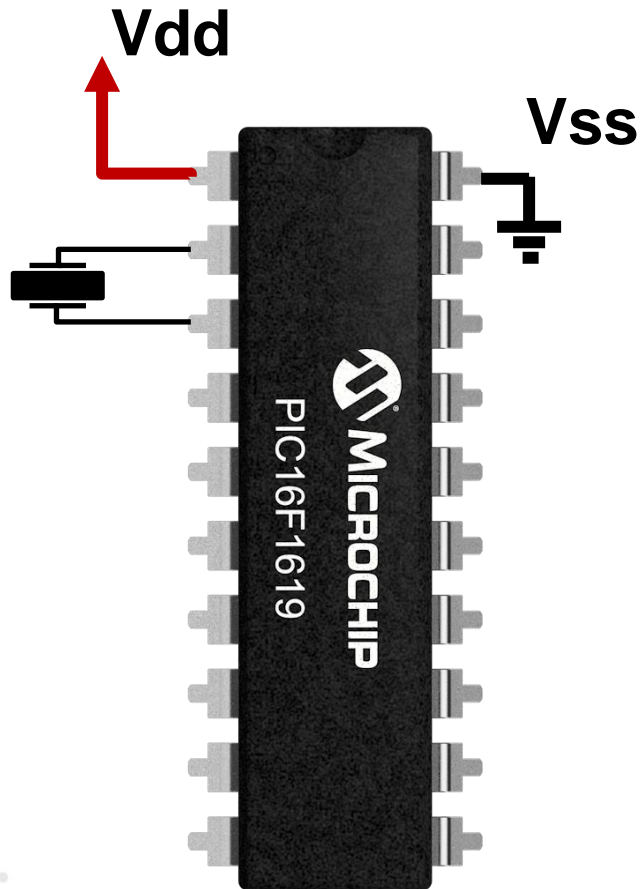


- **哈佛架构：**
 - 程序存储空间和数据存储空间独立
 - 支持不同的总线宽度
 - 提高工作带宽

内核框图

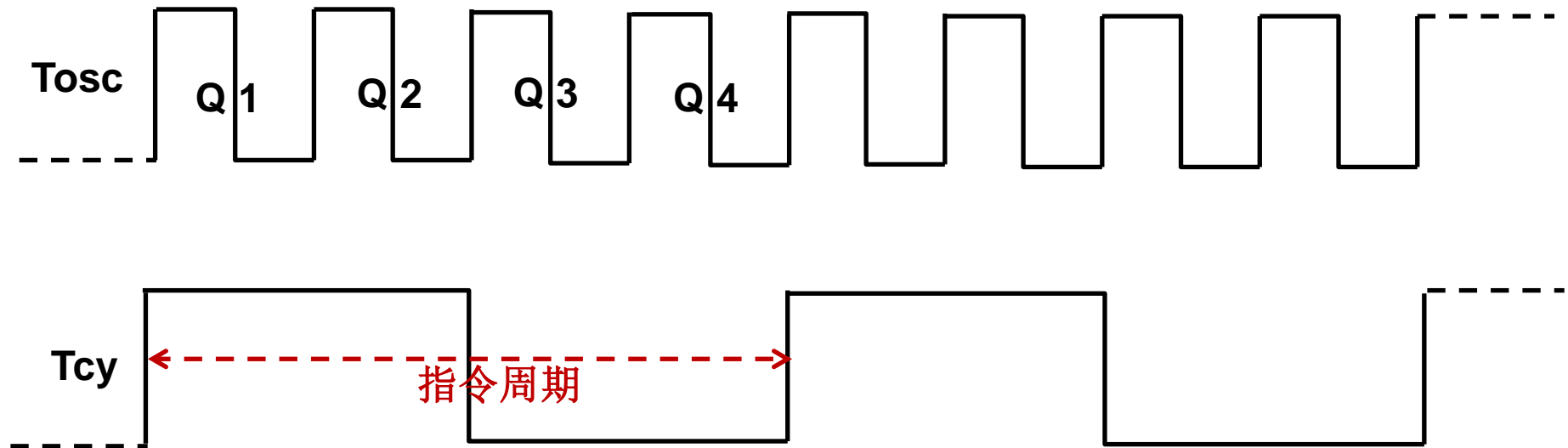


电源和时钟



- 电源——1.8V至5.5V
- 我们使用3.3V
- 时钟——最高32 MHz
- Xtal
- 谐振器
- 内部RC
 - 1%精度
 - 我们使用16 MHz

Tosc和Tcy



- T_{osc} = 振荡器时钟周期
- T_{cy} = 指令周期
- $T_{cy} = 4 \times T_{osc} = 4 \text{ 个 } Q \text{ 周期}$

Tosc和Tcy

- **4 MHz时钟 (Fosc) = 250 ns (Tosc)**
- **在PIC® MCU架构中, 4个时钟周期 = 1个指令周期 (Tcy)**
- **指令周期 (Tcy) = 4x250 ns = 1 μs**
- **大多数指令的执行时间为1 Tcy**
- **修改PC的指令需要2 Tcy**

课程安排

- PIC16F1xxx架构基础知识
- 汇编语言和C语言基础知识
- 实验1：使用MPLAB® X IDE创建一个项目
- A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁
- B部分：通过按键按下操作使LED开始/停止闪烁
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

2017

Microchip 第十八届

中国技术精英年会

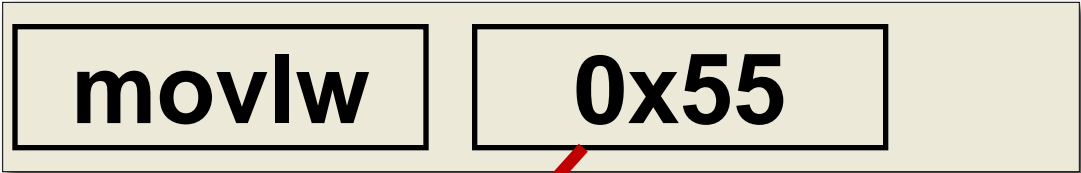
嵌入式控制工程师的盛会

汇编指令和C指令



立即数指令

指令



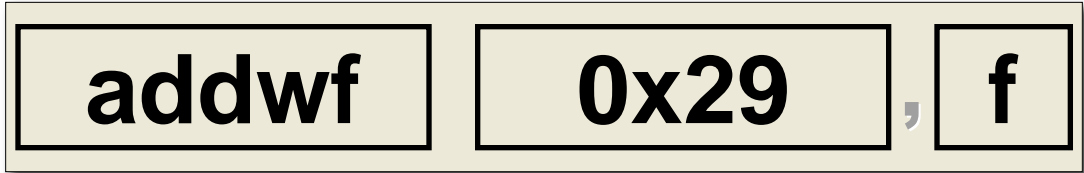
W寄存器



寄存器文件	地址
FF	20h
FF	21h
FF	22h
18	23h
FF	24h
FF	25h
FF	26h
FF	27h
FF	28h
55	29h
FF	2Ah
FF	2Bh

字节指令

指令



W寄存器

0x10

$0x10 + 0x55 = 0x65$

寄存器文件

地址

FF	20h
FF	21h
FF	22h
18	23h
FF	24h
FF	25h
FF	26h
FF	27h
FF	28h
65	29h
FF	2Ah
FF	2Bh

字节指令

指令

decfsz

0x29

f

W寄存器

0x10

→

next: call

getchar

→

decfsz

0x29,f

→

goto

next

→

end: call

print

寄存器文件	地址
FF	20h
FF	21h
FF	22h
18	23h
FF	24h
FF	25h
FF	26h
FF	27h
FF	28h
01	29h
FF	2Ah
FF	2Bh

汇编语言的优缺点

优点	缺点
代码紧凑	非常枯燥
代码执行速度快/效率高	难以调试/结构化
	上市时间长
	不直观；含义模糊

- 替代方法：
 - 用**C**语言编写代码：
 - 更直观
 - 上市时间短
 - 让编译器执行繁琐的工作
 - 我们有**95%**的客户使用**C**语言

C语言

- 与英语类似的语言：
 - `if (a == b) { LED = On; Timer1 = On; }`
`else {LED = Off; Timer2 = Off;}`
 - `While (a == b)`
`Counter++;` // Counter = Counter + 1
 - `for (i = 1; i <= 100; i++)`
`a = a + i;` // do an operation 100 times

C代码和汇编代码

C代码

Var1 = 55; ➡

Var1 = Var2; ➡

汇编代码

movlw 55 ; movwf Var1

movf Var2,w; movwf Var1

C代码和汇编代码

反汇编代码：for循环

```
110 39:                                for (i = 1; i <= 100; i++)
111 01F7 0067        CLRf i
112 01F8 02A7        INCf i, F
113 01F9 0C65        MOVLW 0x65
114 01FA 02A7        INCf i, F
115 01FB 0087        SUBWF i, W
116 01FC 0603        BTFSC STATUS, 0x0
117 01FD 0BF7        GOTO 0x1F7
118 01FE 0BF9        GOTO 0x1F9
119 40:                                a = a + i;
```

- 我们将使用C语言开发代码
- 为什么？


C语言的优点

- 结构化程序
- 逻辑性强/更易于让人理解/读懂
- 更容易调试
- 上市时间短
- 允许C编译器用汇编语言执行繁琐的工作
- **MCHP提供C语言课程！**

知识测验

- **PIC® MCU架构称为哈佛架构。为什么？**
 1. **PIC MCU是在哈佛大学设计的**
 2. **PIC MCU有一条数据/程序总线**
 3. **PIC MCU有独立的数据总线和程序总线**

知识测验

- **PIC® MCU架构称为哈佛架构，为什么？**
 1. PIC MCU是在哈佛大学设计的
 2. PIC MCU有一条数据/程序总线
 3.  **PIC MCU有独立的数据总线和程序总线**

知识测验

- **PIC[®] MCU使用8 MHz晶振运行**
 1. **$T_{cy} = 500 \text{ ns}$**
 2. **$T_{osc} = 125 \text{ ns}$**
 3. **1和2均为真**

知识测验

- **PIC[®] MCU使用8 MHz晶振运行**

1. $T_{cy} = 500 \text{ ns}$

2. $T_{osc} = 125 \text{ ns}$


3.  **1和2均为真**

知识测验

- 汇编语言的特性如下：
 1. 快速且紧凑
 2. 比C语言更容易理解
 3. 易于编写结构化代码

知识测验

- 汇编语言的特性如下：

1.  快速且紧凑
2. 比C语言更容易理解
3. 易于编写结构化代码

知识测验

- 用C语言编写代码：
 1. 编写速度快
 2. 直观
 3. 结构化
 4. 以上全部

知识测验

- 用**C**语言编写代码：

- 1. 编写速度快

- 2. 直观

- 3. 结构化

-  4. 以上全部

课程安排

- PIC16F1xxx架构基础知识
- 汇编语言和C语言基础知识
- **实验1：使用MPLAB® X IDE创建一个项目**
- A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁
- B部分：通过按键按下操作使LED开始/停止闪烁
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

2017

Microchip 第十八届

中国技术精英年会

嵌入式控制工程师的盛会

实验1：使用MPLAB® X IDE创建一个新项目



什么是MPLAB® X IDE?

- 用于开发应用程序的集成开发环境 (IDE)
 - 提供编辑器来编辑代码文件
 - 提供编译功能来编译代码文件
 - 提供模拟器来测试代码
 - 与调试器配合使用来调试/测试代码
 - 编程硬件电路中的PIC®器件
 - Microchip免费提供!

实验1

- 双击图标 ，打开**MPLAB® X IDE**
- 请遵循讲师的指导操作
- 利用实验手册寻求帮助
- 针对**PIC16F1619**创建空白项目

实验1中的操作

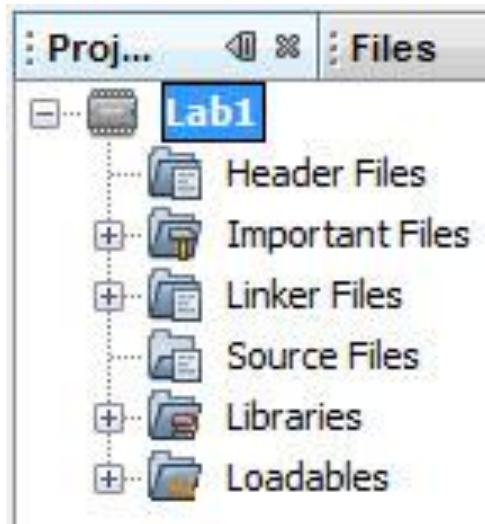
- 在**MPLAB® X IDE**中
 - 选择器件——PIC16F1619
 - 选择实验1的目标位置
 - C:\Masters\20024\Labs\..
 - 选择编译器——XC8
 - 完成并创建空白Lab1.X

实验1：总结

- 在**MPLAB® X IDE**中创建了一个空白项目
- 选择了要使用的**PIC®**器件
- 选择了“调试器/编程器”
- 选择了项目中要使用的编译器

Lab1

- 您将在MPLAB® X IDE中创建一个空白Lab1项目



- 无头文件
- 无源文件

- 使用MCC为Lab1创建代码

课程安排

- PIC16F1xxx架构基础知识
- 汇编语言和C语言基础知识
- 实验1：使用MPLAB® X IDE创建一个项目
- **A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁**
- B部分：通过按键按下操作使LED开始/停止闪烁
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

2017

Microchip 第十八届

中国技术精英年会

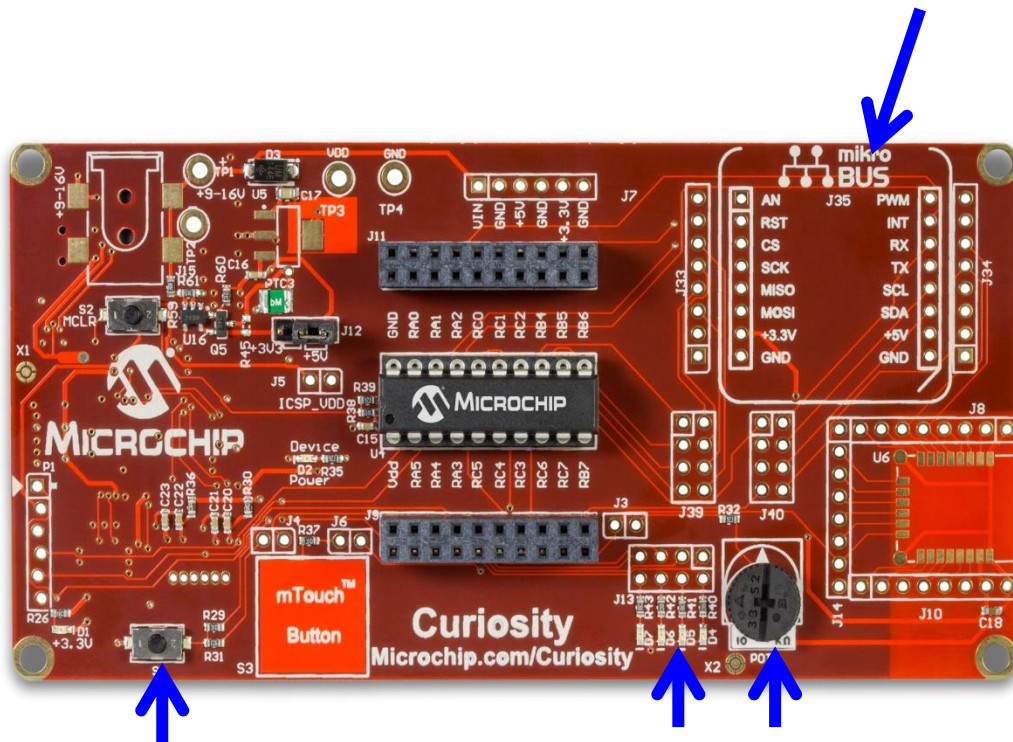
嵌入式控制工程师的盛会

实验1 A部分 使用MCC、TMR1和状态机代码 使LED闪烁



实验中使用的硬件

- **Curiosity开发板:**
 - **PIC16F1619**
 - **4个LED**
 - **开关S1**
 - **电位器**
 - **USB电源**
 - **Mikro总线连接器**
 - **编程器**
 - **调试器**



**Curiosity Development Board
(Part # DM164137)**

PIC16F1619

- **8K字程序存储器和1 KB数据存储器**
- **32 MHz CPU操作**
- **20个引脚，17个GPIO**
- **EUSART、SPI和I2C**
- **12通道10位ADC，8位DAC**
- **2个捕捉/比较/PWM + 2个PWM**
- **4个8位和3个16位定时器**
- **PPS、CWG、CLC、CRC、SMT和ZCD等**

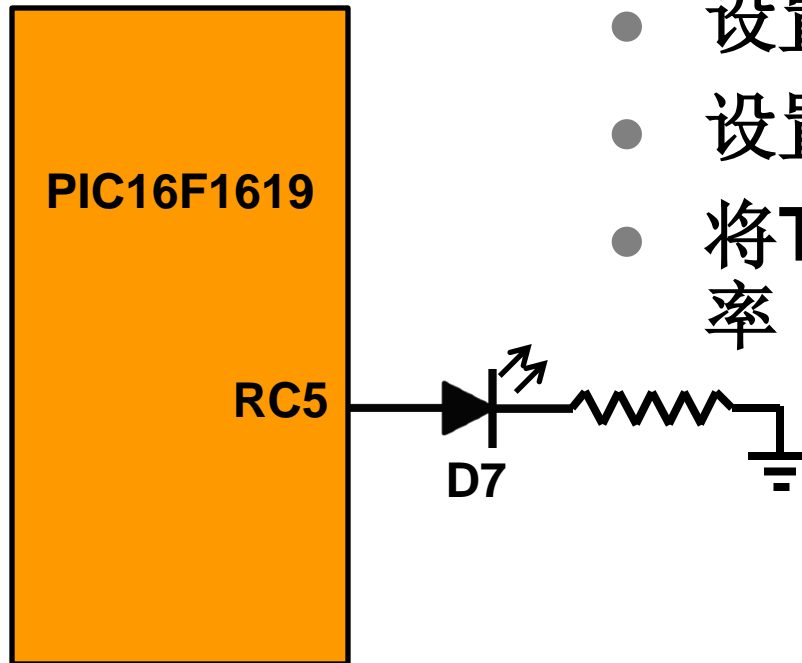
注：附录A中给出了**Microchip**的所有缩略语

什么是MCC?

- 用户友好的**GUI**:
 - 提供按钮和编辑窗口以选择功能
 - 创建代码以初始化外设
 - 提供**C**函数以操作外设
- 经测试的外设功能
- 带注释的**C**代码
- 设置结构化项目
- 为您创建一个**main.c**

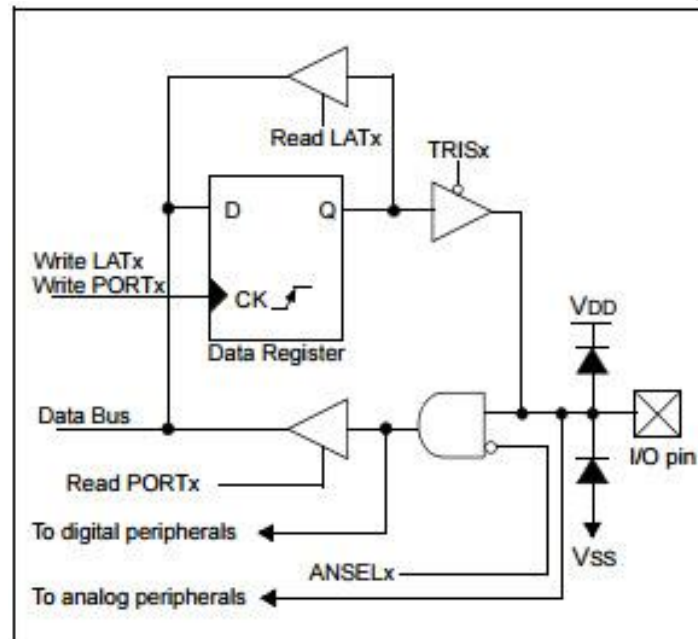
使LED闪烁

- 1个GPIO（RC5）输出
- 设置为高电平点亮
- 设置为低电平熄灭
- 将Timer1设置为1/2秒的速率



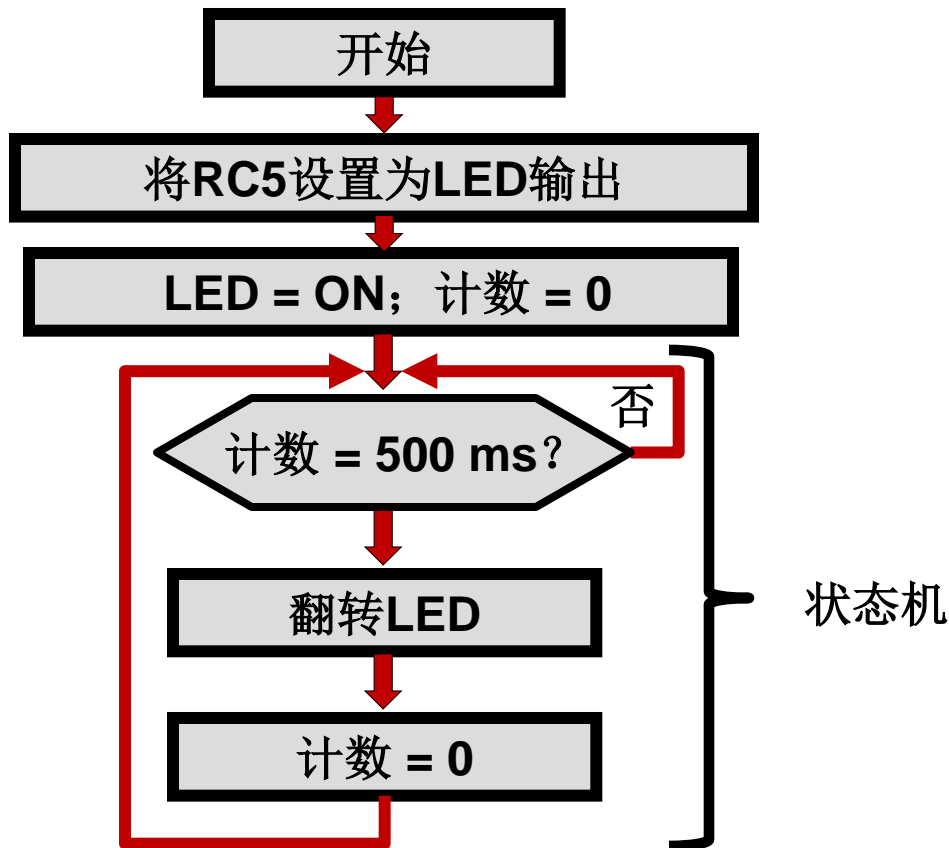
I/O端口

- **50 mA驱动能力**
 - 直接驱动LED
- **方向控制——TRISx**
- **端口和锁存器读写操作：**
 - 写入锁存器或端口
 - 从端口引脚读取
 - 读取锁存器，读取写入锁存器的最后一个值
- **弱上拉电阻和漏极开路能力**
- **ESD保护二极管**
- **启动/复位时默认为输入（高阻抗）**
- **与模拟功能复用的引脚在启动时默认为模拟输入**

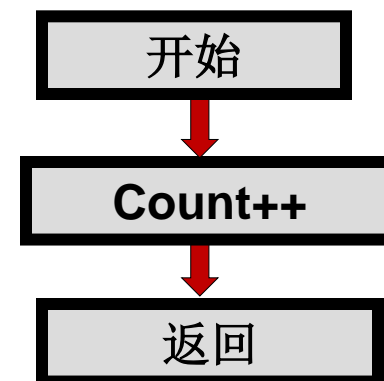


实验1 A部分：程序结构

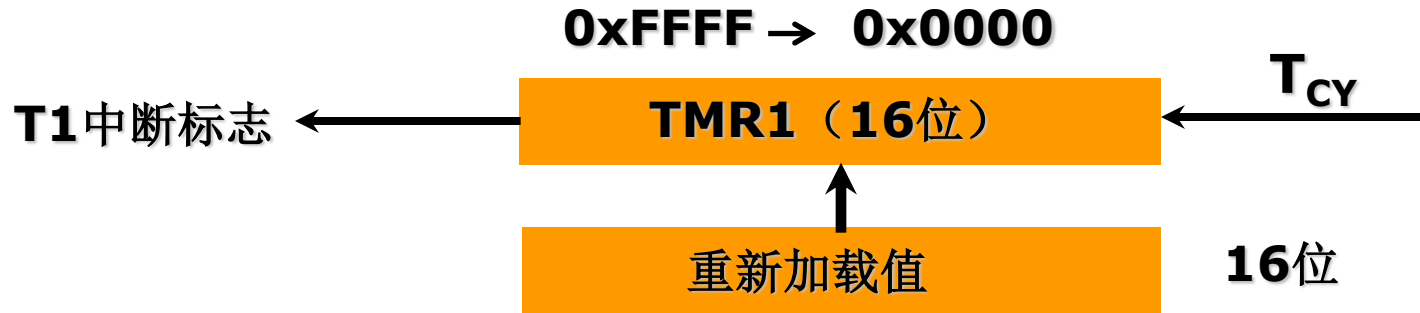
主程序



1 ms中断



定时器1



- TMR1加载初始值
- 内部Tcy使TMR1递增
- 当TMR1从0xFFFF计满返回到0x0000时
 - T1中断标志置1
- TMR1随后重新加载新的16位值
 - 由用户软件完成（MCC为您执行此操作）

TMR1动画

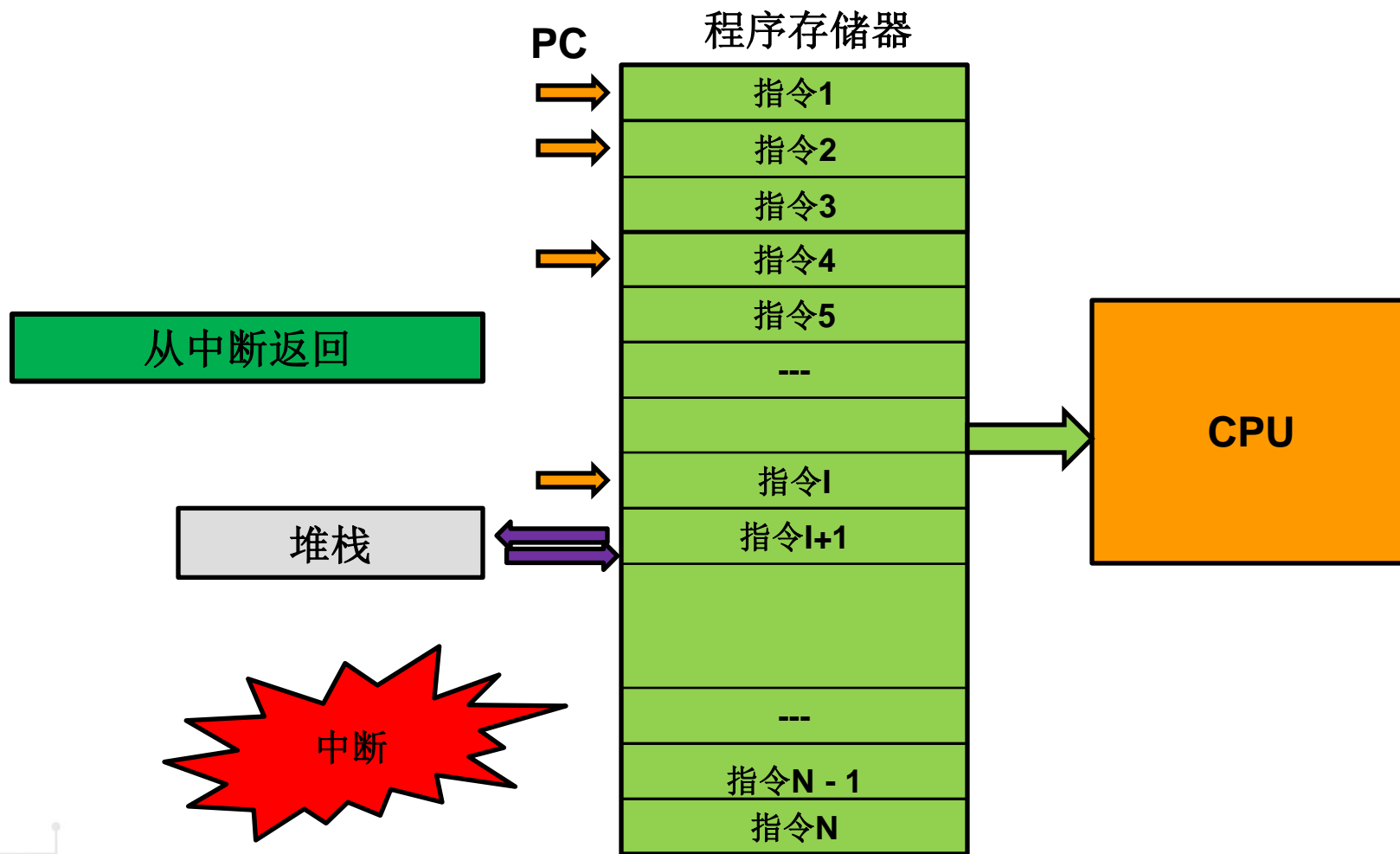


- 在1 ms的中断中，计数递增
- 当计数 == 500时
 - 使LED闪烁，计数 = 0并重复

TMR1的数学运算

- **Tcy = 250 ns** （内部振荡器 = 16 MHz）
- **重载值 = 0xF060** （61536）
- **TMR1 = 0xF060**
- **1 Tcy → TMR1 = 0xF061** （61537）
- **2 Tcy → TMR1 = 0xF062** （61538）
- **3999 Tcy → TMR1 = 0xFFFF** （65535）
- **4000 Tcy → TMR1 = 0x0000** （中断）
- **中断时间 = 250 ns x 4000 = 1 ms**

PIC16F1 中断操作



为什么使用中断？

- 支持立即处理外设
- 基于时间的事件能够立即得到处理
- 中断后正常**CPU**功能继续工作

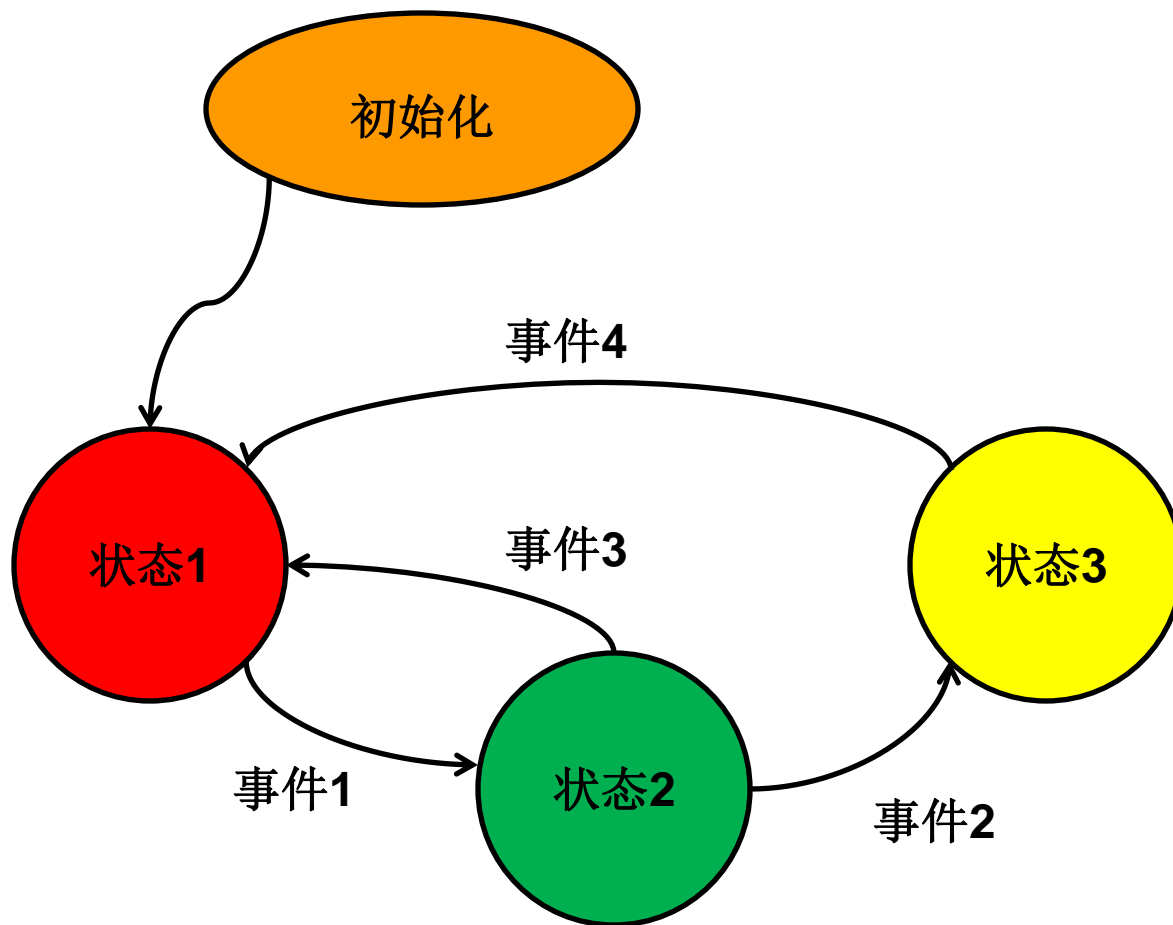
PIC16F1上的中断

- 单个中断向量
 - 位于程序存储器中的存储单元4中
 - 用户设置优先级
- 一个全局中断允许（**GIE**）位
 - 中断发生时， $GIE = 0$
 - 执行RETFIE时， $GIE = 1$
- 一个外设中断允许（**PIE**）位
 - $PIE = 0/1$ ——禁止/允许外设中断
- 每个中断都有自己的允许位和标志位
 - 示例：TMR1IE和TMR1IF
 - 用户必须用软件将中断标志清零
- **MCC负责中断编码**

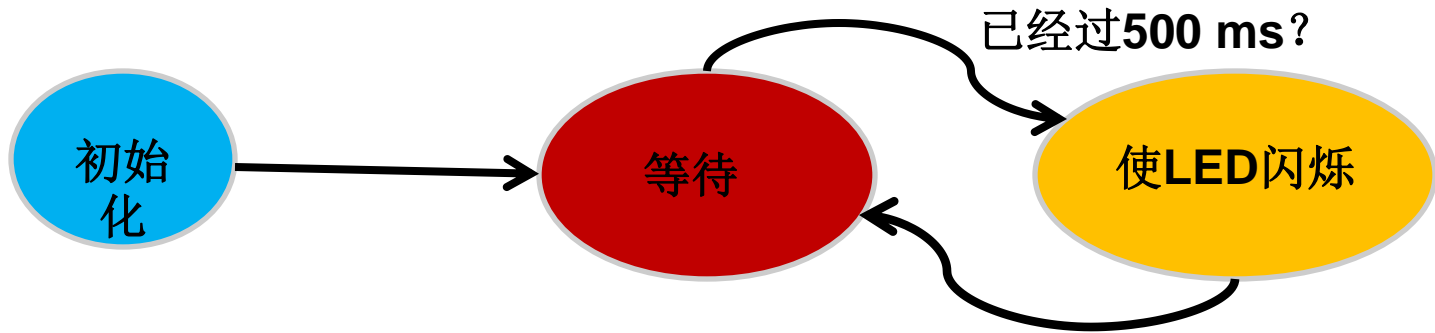
中断的有效利用

- 在外设完成作业/功能时快速指示
- 中断代码尽可能短
- 用常规程序任务函数处理外设
- 使用状态机编码来处理应用程序中的任务

什么是状态机？



LED状态机



- 初始化状态初始化状态参数 → 等待状态
- 等待状态等待500 ms后 → LED闪烁状态
- LED闪烁状态翻转LED → 等待状态

状态机基础知识

- 应用程序有多个状态机：
 - LED状态机
 - 按键状态机
 - UART状态机
- 没有状态机占主导地位
- 所有状态机均在主循环中执行
- 应用程序的所有部分都会执行

状态机代码

典型状态机主代码

```
52 void main(void)
53 {
54     // initialize the device
55     SYSTEM_Initialize();
56     APP_LED_Initialize();    // Initialize LED app
57     APP_KEY_Initialize();    // Initialize KEY app
58     APP_UART_Initialize();   // Initialize UART app
59
60
61     while (1)
62     {
63         APP_LED_Tasks();      // Run LED State machine
64         APP_KEY_Tasks();      // Run KEY state machine
65         APP_UART_Tasks();     // Run UART state machine
66     }
67 }
```

用C语言编写的Switch Case指令

Switch (letter)

```
{  
    case 'a' : printf ("Letter \'a\' found\n"); break;  
    case 'b' : printf ("Letter \'b\' found\n"); break;  
    case 'c' : printf ("Letter \'c\' found\n"); break;  
    default : printf ( " Letter is not a, b or c \n"); break;  
}
```

- **Switch Case**替换了多个“if then else”语句
- 非常适合在状态机代码中使用

LED状态机代码

LED状态机代码

```
146     switch ( appliedData.state )
147     {
148         /* Application's initial state. */
149         case APP_LED_STATE_INIT:
150             appliedData.state = APP_LED_STATE_WAIT;
151             break;
152         /* TODO: implement your application state machine. */
153         case APP_LED_STATE_WAIT:
154             if (appliedData.TimerCount >= 500)
155             {
156                 appliedData.state = APP_LED_STATE_BLINK_LED;
157                 appliedData.TimerCount = 0;
158             }
159             break;
160         case APP_LED_STATE_BLINK_LED:
161             LED_Toggle();
162             appliedData.state = APP_LED_STATE_WAIT;
163             break;
164         /* The default state should never be executed. */
165         default:
166             /* TODO: Handle error in application's state machine. */
167             break;
168     }
```

实验1 A部分

- 请遵循讲师的指导操作
- 使用实验手册寻求帮助
- 在主菜单的**Tools** → **Embedded**（工具 → 已安装工具）下，单击**MPLAB® Code Configurator**（**MPLAB®**代码配置器）
- 我们将使用**MCC**和状态机使**LED**闪烁



实验1 A部分中的操作

- 使用**MCC**:
 - TMR1 = 1 ms中断
 - 将RC5设置为输出D7 ...生成代码
- 编辑**timer1.c**:
 - 在回调函数中: Count++;
 - 注: 剪切并粘贴“**app.h**”文件底部的编辑内容
- 将**app.h**和**app.c**文件包含到项目中
 - 学习LED状态机代码
- 在**main.c**中:
 - 使能GIE和PIE位
 - 添加LED闪烁初始化和任务程序
- 编译代码并运行

实验1 A部分——深入了解

- 打开**app.h**文件
- 向下滚动并查看：
 - APP_LED_DATA结构
- 打开**app.c**文件
- 向下滚动并查看：
 - APP_LED_Initialize()
 - APP_LED_Tasks()

实验1 A部分总结

- 使用**MCC**配置**TMR1 1 ms**中断
- 在回调函数中，**Count++**
- 用于编码的状态机
- 状态机代码使**LED**以**1/2秒**的速率闪烁

课程安排

- PIC16F1xxx架构基础知识
- 汇编语言和C语言基础知识
- 实验1：使用MPLAB® X IDE创建一个项目
- A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁
- **B部分：通过按键按下操作使LED开始/停止闪烁**
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

2017

Microchip 第十八届

中国技术精英年会

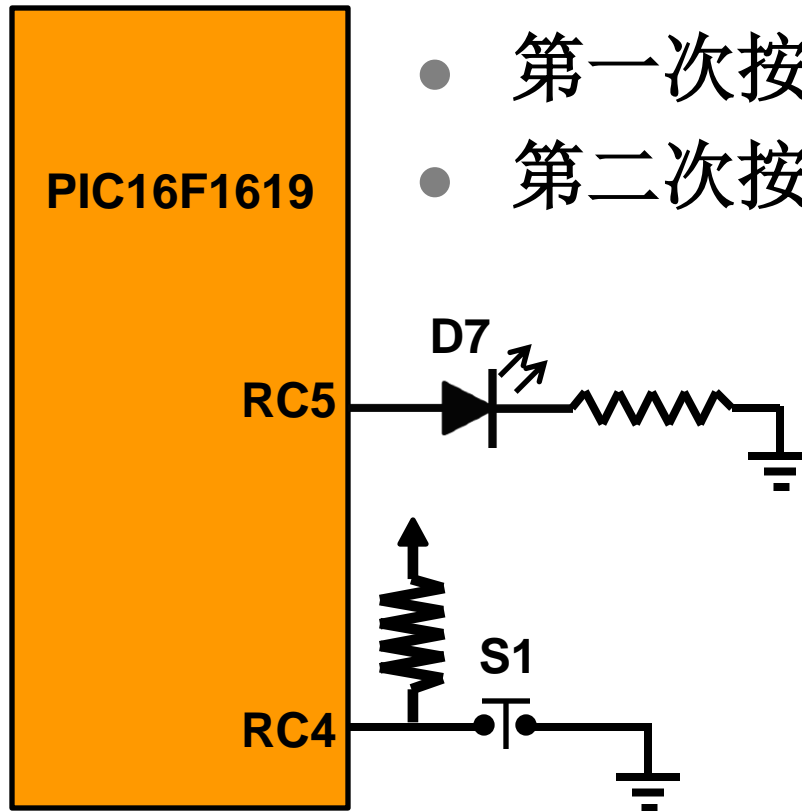
嵌入式控制工程师的盛会

实验1 B部分 通过按键按下操作使LED开始/停止闪烁

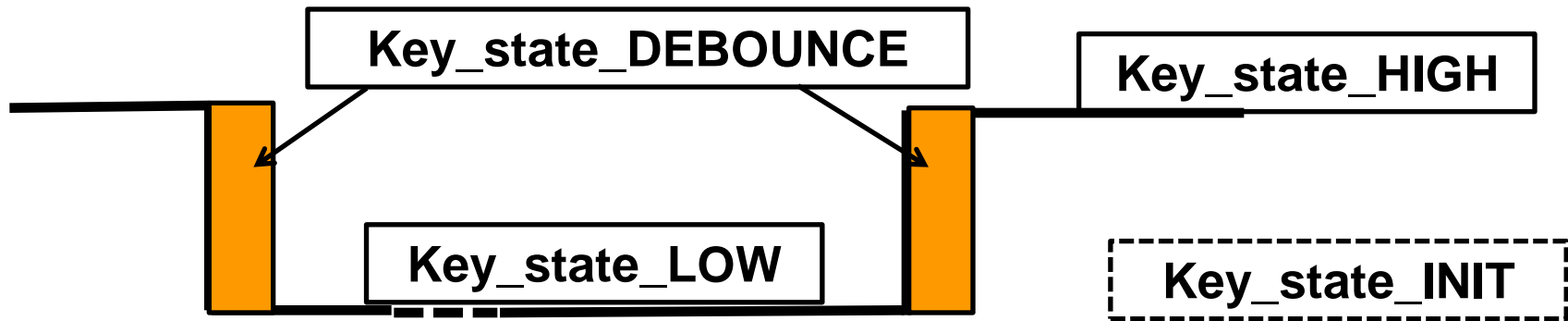


通过按键按下操作使LED开始/停止闪烁

- 1个GPIO（RC4）输入
- 第一次按下按键会使LED停止闪烁
- 第二次按下按键会使LED开始闪烁

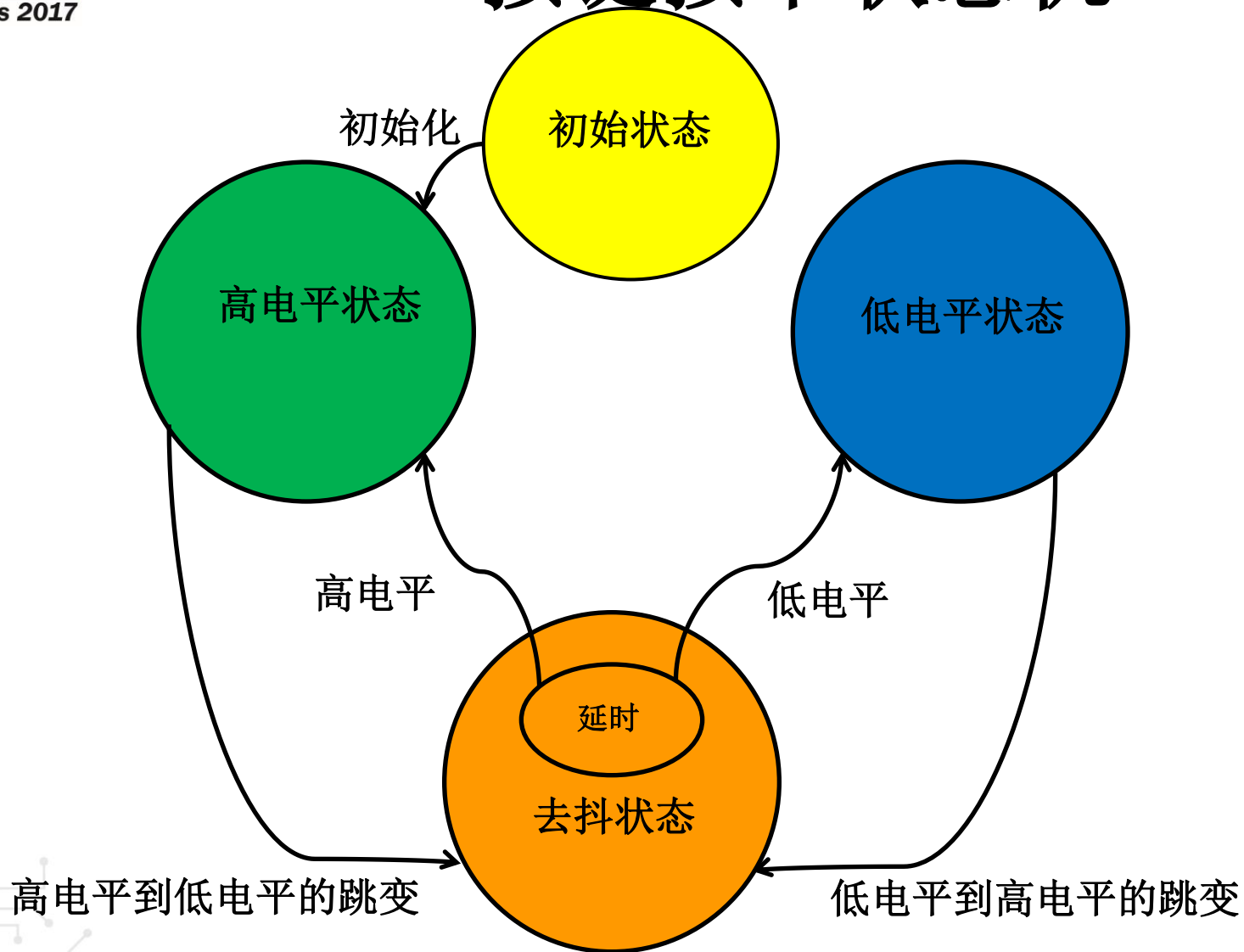


按键按下状态机



- **Key_state_INIT**: 初始化 ⇒ **Key_state_HIGH**
- **Key_state_HIGH**: 正常或初始状态。输入：从高电平切换到低电平 ⇒ **Key_state_DEBOUNCE**
- **Key_state_DEBOUNCE**: 等待去抖时间，之后如果输入为低电平 ⇒ 则为**Key_state_LOW**，否则为**Key_state_HIGH**。
- **Key_state_LOW**: 按下状态。输入：从低电平跳变到高电平 ⇒ **Key_state_DEBOUNCE**

按键按下状态机



实验1 B部分中的操作

- **使用MCC:**
 - 将RC4设置为按键按下输入——S1
 - 生成代码
- **编辑timer1.c:**
 - 在回调函数中，处理去抖时间
- **将app.h和app.c文件包含到项目中**
 - 学习按键按下状态机代码
- **在main.c中:**
 - 初始化并运行新的关键任务
- **编译代码并运行**

实验1 B部分——深入了解

- 打开**app.h**文件
- 向下滚动并查看：
 - APP_KEY_DATA结构
- 打开**app.c**文件
- 向下滚动并查看：
 - APP_LED_Tasks()
 - APP_KEY_Initialize()
 - APP_KEY_Tasks()

实验1 B部分总结

- 使用了**MCC**配置**RC4**输入
- 在回调函数中，处理了去抖计数
- 添加了按键按下状态机
- 通过按键按下状态机代码使**LED**开始/停止闪烁

课程安排

- PIC16F1xxx架构基础知识
- 汇编语言和C语言基础知识
- 实验1：使用MPLAB® X IDE创建一个项目
- A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁
- B部分：通过按键按下操作使LED开始/停止闪烁
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

2017

Microchip 第十八届

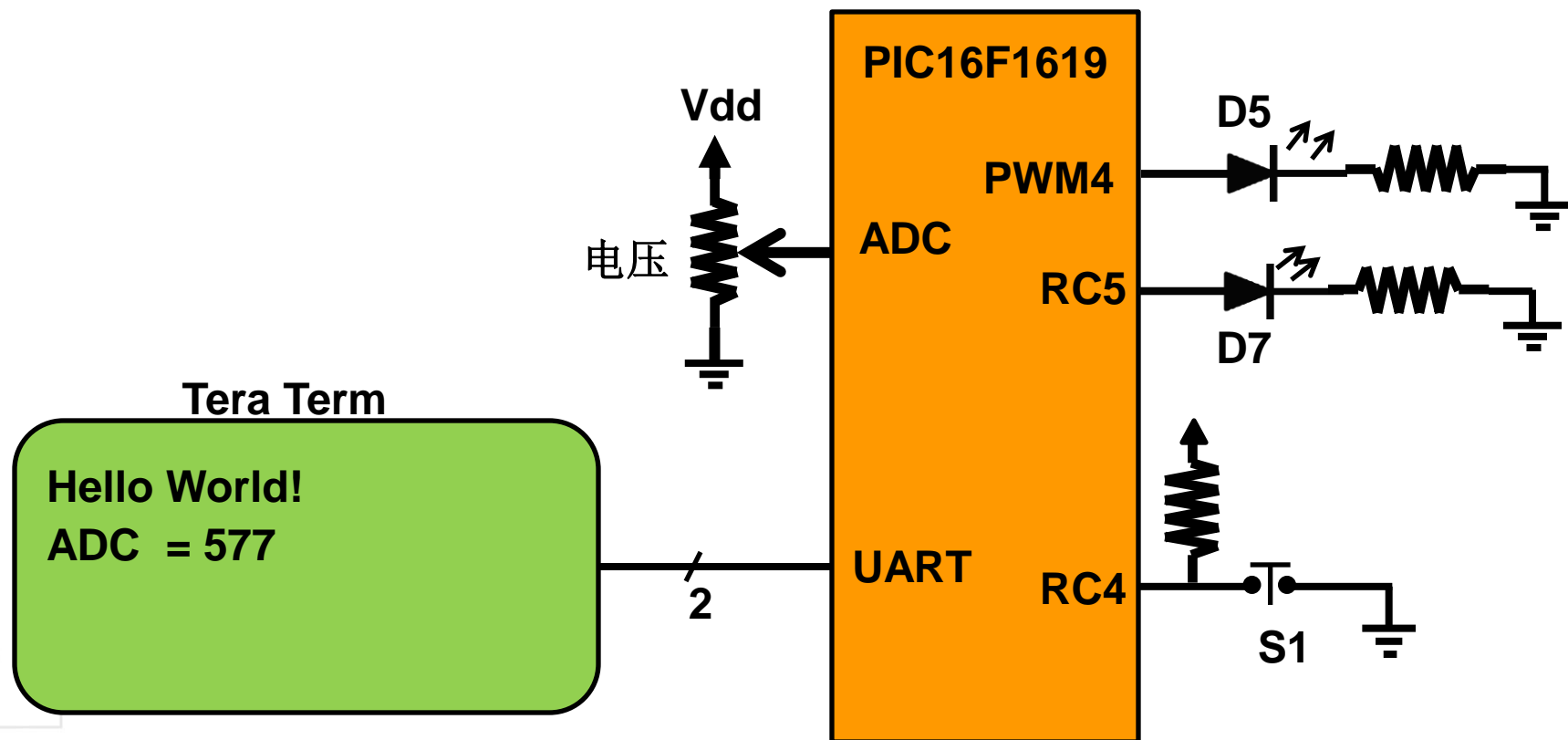
中国技术精英年会

嵌入式控制工程师的盛会

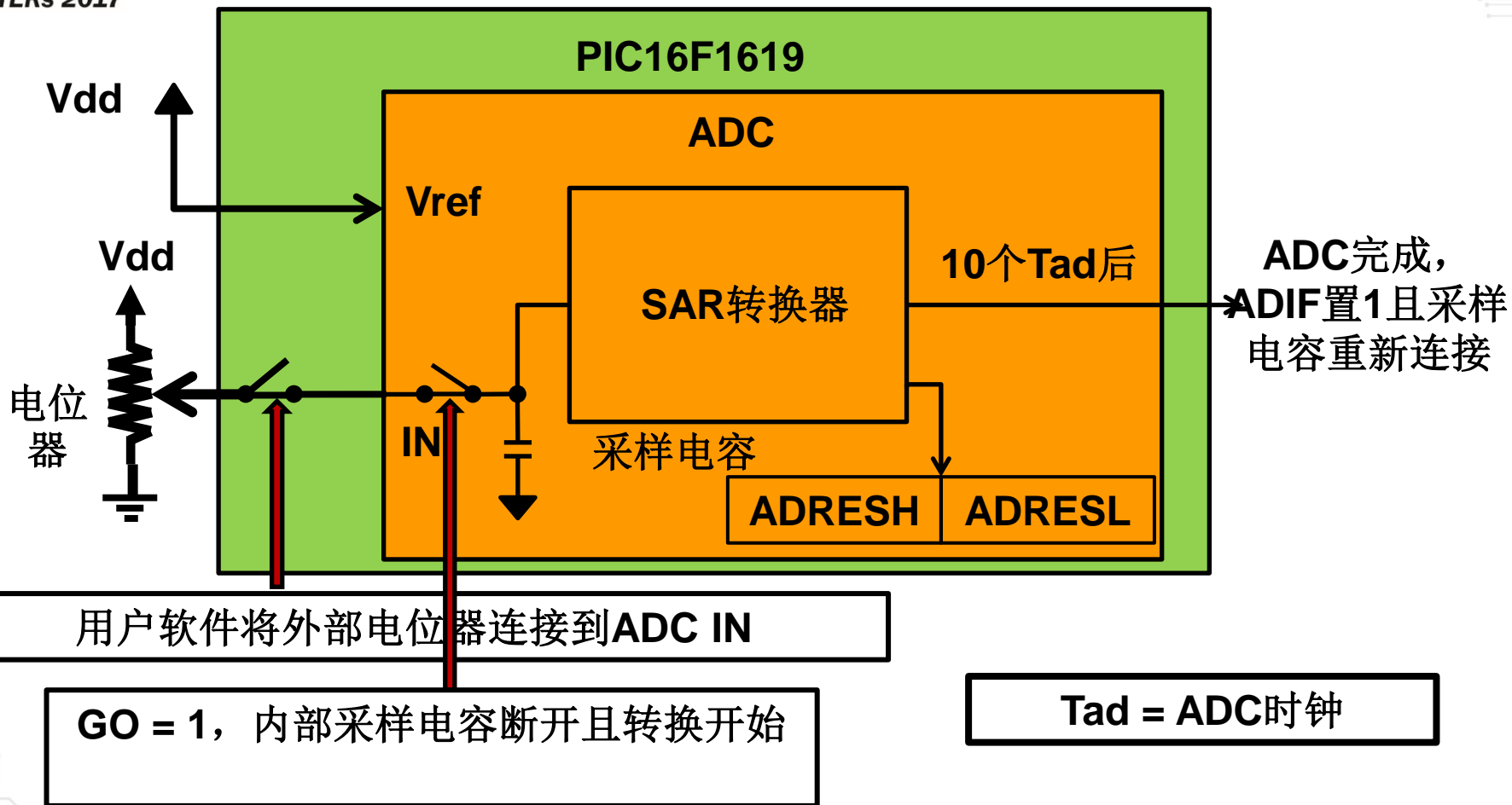
实验1 C部分 将ADC、PWM和UART添加到应 用程序中



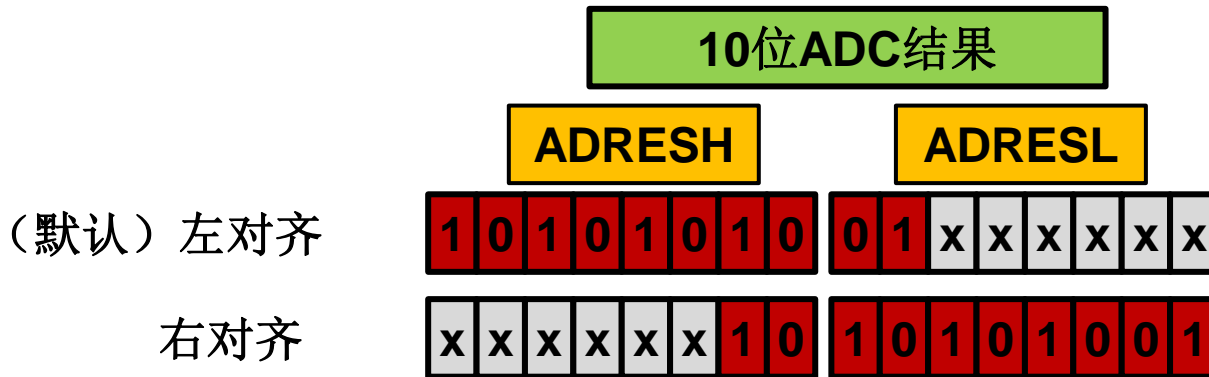
将ADC、PWM和UART添加到项目中



ADC框图



ADC结果的表示



- 在C语言中：ADC值定义为整数
- ADC值的表示需要右对齐

ADC设置

ADCC

Easy Setup Registers Notifications : 3

Hardware Settings

☒ Enable ADC

Operating Basic_mode

▼ ADC

ADC Clock

Clock Source FRC

Clock FOSC/2

1 TAD 1.7 us

Sampling Frequency 51.1509 kHz

Conversion Time = $11.5 * TAD = 19.55 \text{ us}$

Result Alignment right

Positive Reference VDD

Negative Reference VSS

Auto-conversion Trigger disabled

☐ Enable Continuous Operation

☐ Enable Stop on Interrupt

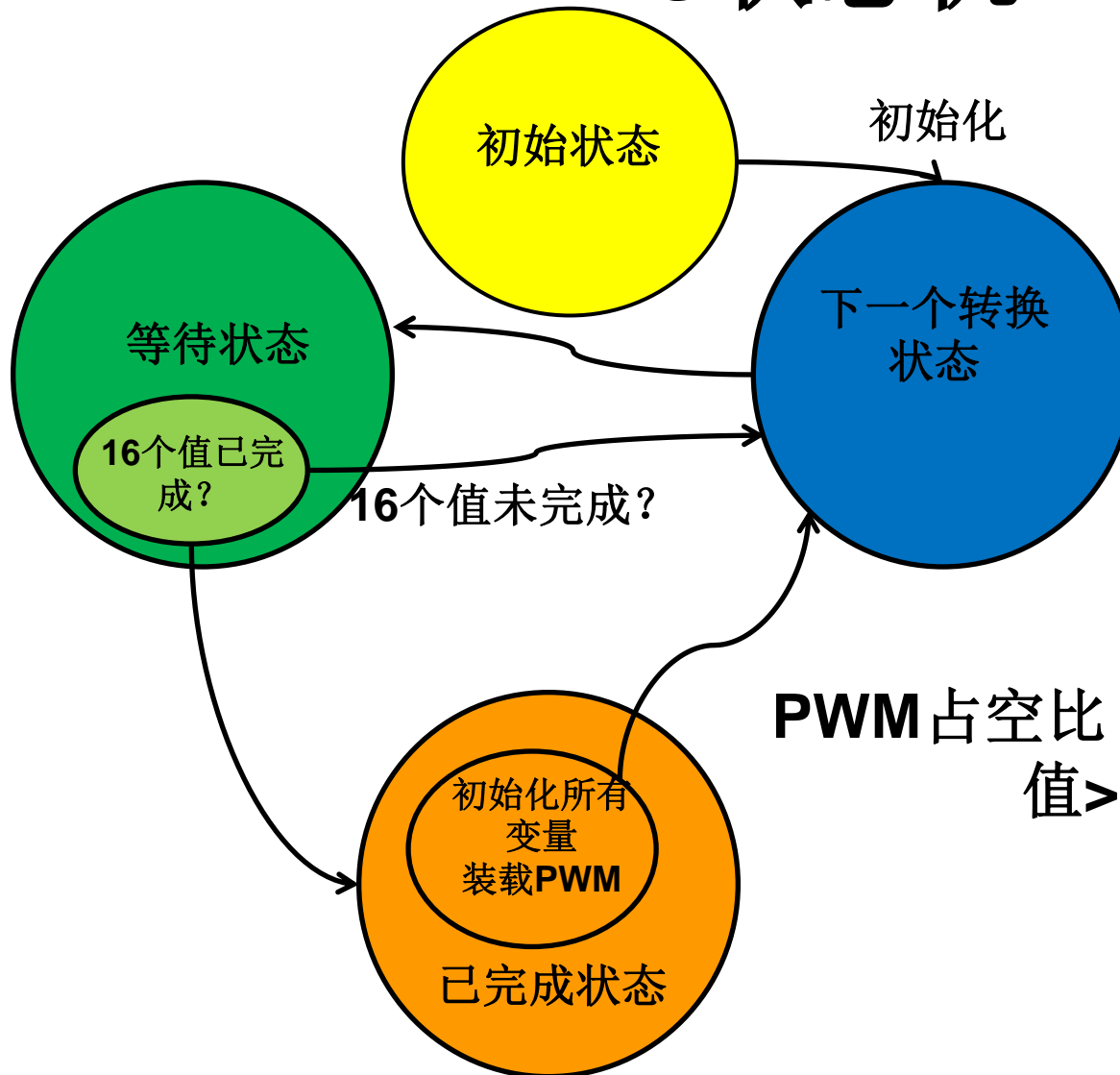
☐ Enable Double Sample

Acquisition Time 0 ≤ 0 ≤ 255

选择FRC

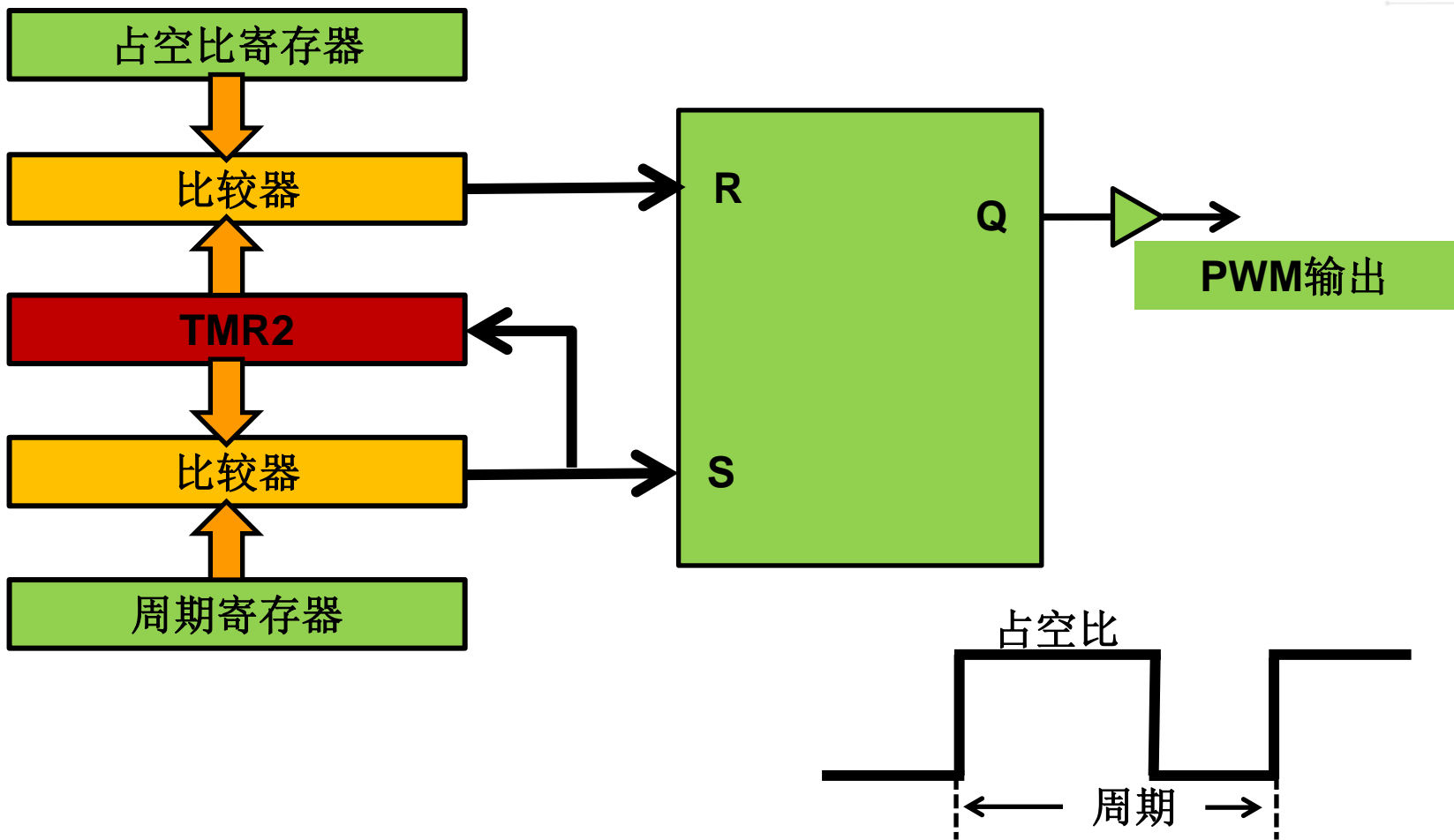
选择右对
齐

ADC状态机



**PWM占空比 = 16个ADC
值>>4**

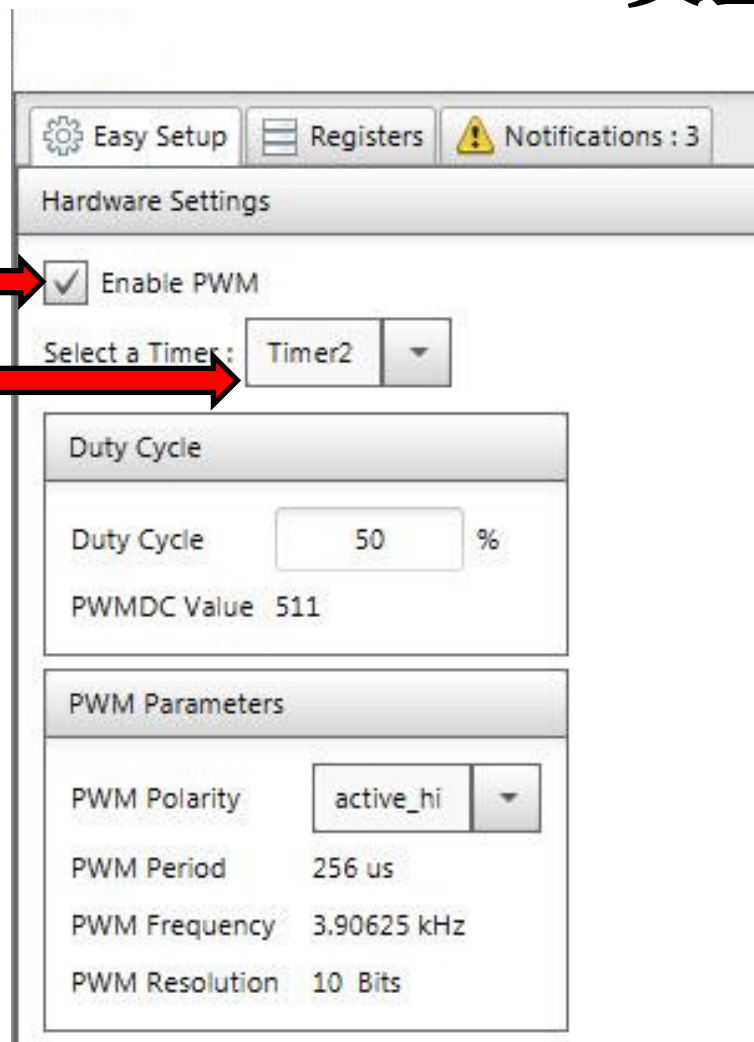
简化PWM



PWM4设置

使能PWM

选择TMR2



Easy Setup Registers Notifications : 3

Hardware Settings

☒ Enable PWM

Select a Timer: Timer2

Duty Cycle

Duty Cycle 50 %

PWMDC Value 511

PWM Parameters

PWM Polarity active_hi




PWM Period 256 us

PWM Frequency 3.90625 kHz

PWM Resolution 10 Bits

TMR2设置

TMR2

 Easy Setup  Registers  Notifications : 4

Hardware Settings

☒ Enable Timer

Timer Clock

Clock Source

FOSC/4

Clock Frequency

32.768 kHz

Postscaler

1:1

Prescaler

1:1

Polarity

Rising Edge

Timer Period

Timer Period

250 ns ≤ 64 us

Actual Period

64 us

(Period calculated via PR Register)

Ext Reset Source

PWM4OUT

Control Mode

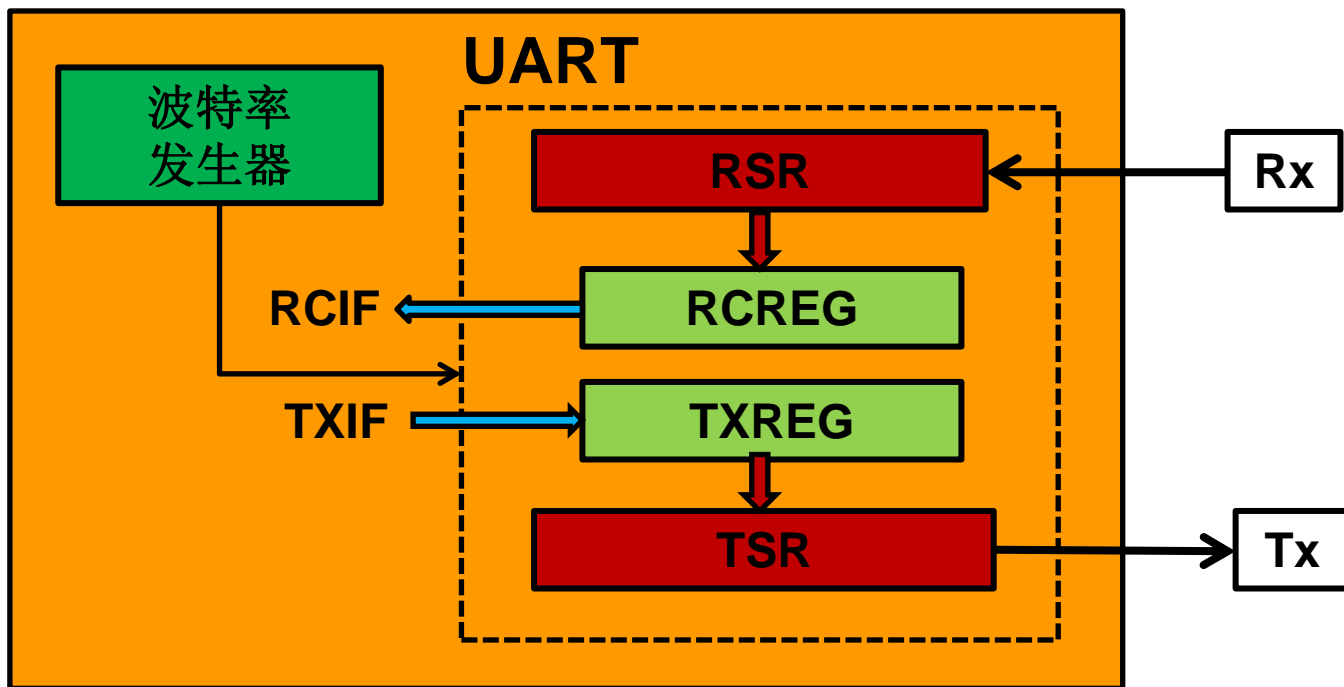
Roll over pulse

Start/Reset Option

Software control

选择PWM4OUT

UART框图



UART设置

EUSART

Easy Setup Registers Notifications : 0

Hardware Settings

Mode asynchronous

☒ Enable EUSART Baud Rate: 9600 Error: 0.160 %

☒ Enable Transmit Transmission Bits: 8-bit

☐ Enable Wake-up Reception Bits: 8-bit

☐ Auto-Baud Detection Clock Polarity: Non-Inverted

☐ Enable Address Detect ☒ Enable Continuous Receive

☐ Enable EUSART Interrupts

Software Settings

☒ Redirect STDIO to USART

Software Transmit Buffer Size 8

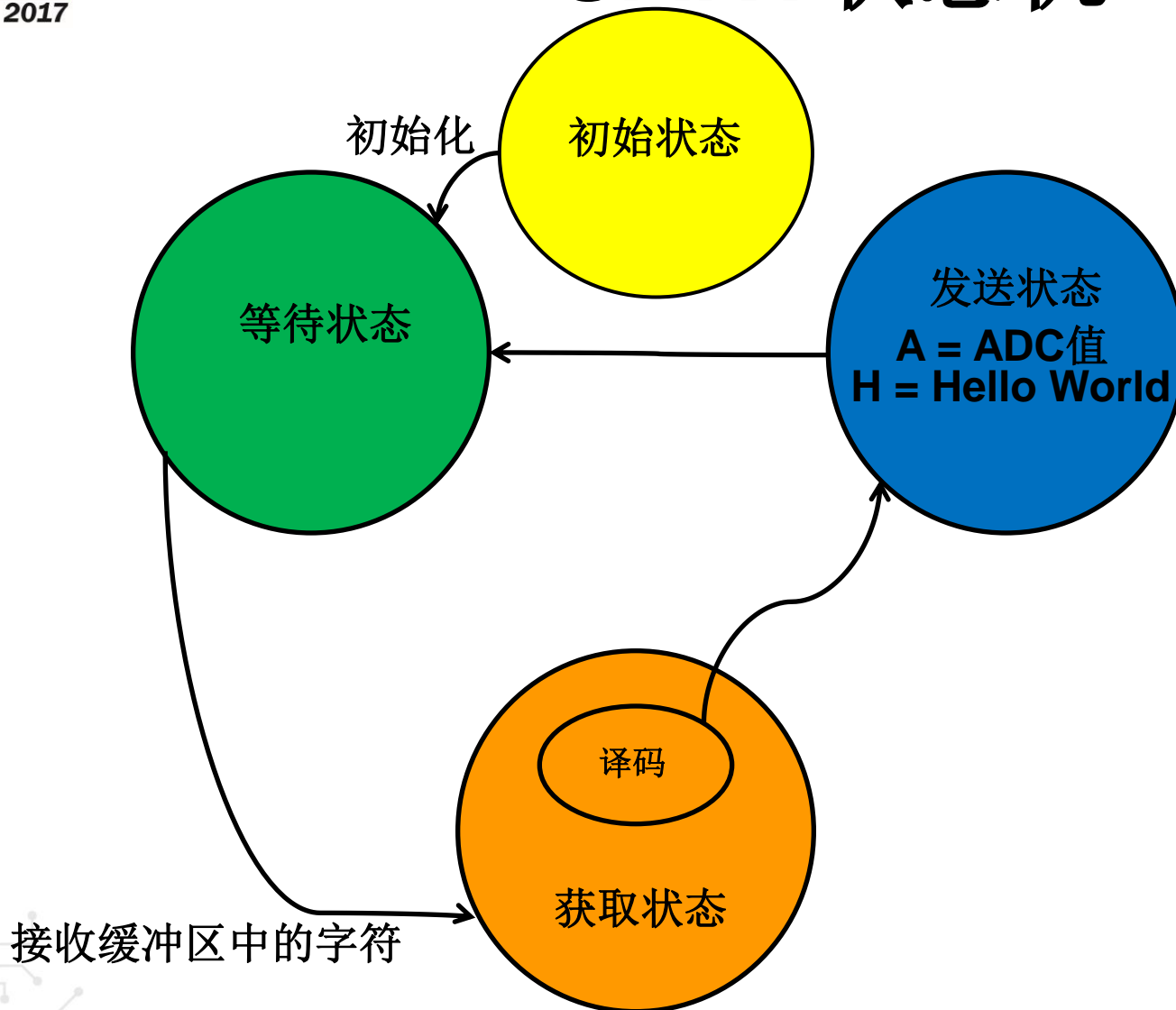
Software Receive Buffer Size 8

使能发送

使能连续接收

将STDIO重定向到
USART

UART状态机



外设引脚选择

- 数字外设连接到不同的I/O引脚
- 示例： **PWM**、**UART**、**SPI**和**I2C**等
- 允许在**PCB**设计中实现灵活性
- 与现有设计兼容

实验1 C部分中的操作

- 使用**MCC**:
 - 使能ADC、PWM、TMR2和UART
 - 设置RC0——电位器输入； RA5——PWM输出； RB7——UART Tx； RB6——UART Rx
 - 生成代码
- 将**app.h**和**app.c**文件包含到项目中
- 在**main.c**中：
 - 初始化并运行新的任务
- 连接**UART**电缆
- 编译代码并运行

实验1 C部分——深入了解

- 打开**app.h**文件
- 向下滚动并查看：
 - APP_XXX_DATA结构
- 打开**app.c**文件
- 向下滚动并查看：
 - APP_XXX_Initialize()
 - APP_XXX_Tasks()

实验1 C部分总结

- 使用了MCC配置ADC、PWM和UART
- 使用了PIC16F1619中的PPS功能将PWM与RA5相连
- 添加了ADC和UART的状态机
- 使用多个外设创建并运行了项目

知识测验！

- **MCC创建：**
 1. 应用程序代码
 2. 初始化外设的代码
 3. 使用外设的代码
 4. 以上全部

知识测验！

- **MCC创建：**

- 1. 应用程序代码

- 2. 初始化外设的代码

- 3. 使用外设的代码

- 4. 以上全部

知识测验

- 状态机代码：
 1. 易于调试
 2. 易于结构化
 3. 易于修改
 4. 以上全部

知识测验

- 状态机代码:

1. 易于调试
2. 易于结构化
3. 易于修改
4. 以上全部



课程安排

- PIC16F1xxx架构基础知识
- 汇编语言和C语言基础知识
- 实验1：使用MPLAB® X IDE创建一个项目
- A部分：使用MCC、TMR1和状态机（SM）代码使LED闪烁
- B部分：通过按键按下操作使LED开始/停止闪烁
- C部分：在应用程序中使用ADC、PWM和UART
- 总结

总结

- 今天介绍了：
 - PIC16架构与基础知识
 - 使用了MPLAB® X IDE和MCC创建并运行一个简单的程序
 - 在我们的代码中使用了状态机
 - 在PIC16F1xxx应用程序中将GPIO、ADC、PWM和UART与状态机代码结合使用

相关课程


- **21013——MPLAB® X IDE入门**
- **21023——C编程语言基础知识**
- **21015——可简化嵌入式开发的MCC**
- **21026——改良的嵌入式C语言**
- **21025——嵌入式固件设计基础知识**
- 此外还提供在线课程
- 请访问[**www.microchip.com**](http://www.microchip.com)

开发人员帮助

- 可随时提供培训
- 作为单一主题或在培训课程内提供
- 多媒体环境——文字、视频、音频和动画
- 适用于开发工具、功能和产品的部分


www.microchip.com/developerhelp

开发人员帮助——主页

**MICROCHIP** Developer Help

Site updated 1 day ago
3100 active pages

[Home](#)
[Training](#)
[Development Tools](#)
[Functions](#)
[Projects](#)
[Products](#)
[Store](#)
[Help](#)


MICROCHIP

What's New?

[Variables to Control Processor Selection](#)
26 May 2017, 10:28

[Variables to Control Special Linking Needs](#)
26 May 2017, 10:27

[Variables to Control Tool Names/Locations](#)
26 May 2017, 10:27

[Variables to Modify Command Lines](#)
26 May 2017, 10:27

[Working Outside of MPLAB® X IDE](#)
26 May 2017, 10:26

[+ More new content...](#)

Tips & Tricks

[SFR Macros](#)
19 May 2017, 12:07

[Concealing Source When Distributing Code](#)
6 Mar 2017, 12:55

[A Bit of Debugging](#)
6 Mar 2017, 12:53

[Performing Rotations in C](#)
5 Dec 2016, 10:25

[Finding the Names of Device Registers](#)
5 Dec 2016, 10:25

[+ More Tips & Tricks...](#)

自学培训课程


MICROCHIP Developer Help

Search This Site


 Search

Site updated 1 day ago


3100 active pages


 Home

 Training

 Self-Paced Training

 [Get Started Here](#)

 Development Tools

 Functions

 Projects

 Products

 Store

 Help



Self-Paced Training

Most of the material you will find in these training modules exists elsewhere on this site in a general reference format. However, the training modules present it in an organized, step-by-step sequence along with commentary from one of our instructors to help you learn the topic from the ground up. In addition, many modules include hands-on exercises to help reinforce the concepts presented.



Click image to enlarge.

2017

Microchip 第十八届

中国技术精英年会

嵌入式控制工程师的盛会

谢谢！



法律声明

软件:

Microchip软件仅允许用于Microchip产品。此外，Microchip软件的使用受软件附带的版权声明、免责声明以及任何授权许可的限制，无论这些内容是在安装各个程序时阐明还是在头文件或文本文件中公告。

尽管有上述限制，但Microchip和第三方提供的软件的某些组件仍可能被“开源”软件许可覆盖，其中包括要求分发者提供软件源代码的许可。在开源软件许可要求的范围内，许可条款将起主导作用。

注意事项和免责声明:

这些材料和随附信息（例如，包括任何软件以及对第三方公司和第三方网站的引用）仅供参考，并且按“现状”提供。Microchip对第三方公司做出的声明或第三方可能提供的材料或信息不承担任何责任。

MICROCHIP不承担任何形式的保证，无论是明示的、暗示的或法定的，包括有关无侵权性、适销性和特定用途的暗示保证。在任何情况下，对于与MICROCHIP或其他第三方提供的材料或随附信息有关的任何直接或间接的、特殊的、惩罚性的、偶然的或间接的损失、损害或任何类型的开销，MICROCHIP概不承担任何责任，即使MICROCHIP已被告知可能发生损害或损害可以预见。请注意，使用此处所述的知识产权时可能需要第三方许可。

商标:

Microchip的名称和徽标组合、Microchip徽标、AnyRate、AVR、AVR徽标、AVR Freaks、BeaconThings、BitCloud、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KEELOQ、KEELOQ徽标、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、MediaLB、megaAVR、MOST、MOST徽标、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32徽标、Prochip Designer、QTouch、RightTouch、SAM-BA、SpyNIC、SST、SST徽标、SuperFlash、tinyAVR、UNI/O及XMEGA均为Microchip Technology Incorporated在美国和其他国家或地区的注册商标。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge和Quiet-Wire均为Microchip Technology Incorporated在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、chipKIT、chipKIT徽标、CodeGuard、CryptoAuthentication、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet徽标、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certified徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PureSilicon、QMatrix、RightTouch徽标、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQL、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA和ZENA均为Microchip Technology Incorporated在美国和其他国家或地区的商标。

SQTP为Microchip Technology Inc.在美国的服务标记。

Silicon Storage Technology为Microchip Technology Inc.在除美国外的国家或地区的注册商标。

GestIC为Microchip Technology Inc.的子公司Microchip Technology Germany II GmbH & Co. KG在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2017, Microchip Technology Incorporated版权所有。

2017

Microchip 第十八届

中国技术精英年会

嵌入式控制工程师的盛会

附录A Microchip缩略词



Microchip 缩略词

- **CIP**——内核独立外设
- **CCP**——捕捉/比较/PWM
- **CWG**——互补波形发生器
- **NCO**——数控振荡器
- **CLC**——可配置逻辑电路
- **ADCC**——具有计算功能的ADC
- **DSM**——数字信号调制器
- **ZCD**——过零检测

Microchip 缩略词

- **CRC**——循环冗余校验
- **PPS**——外设引脚选择
- **WWDT**——窗口看门狗定时器
- **BOR**——欠压复位
- **LPBOR**——低功耗BOR
- **ICSP**——在线串行编程
- **SAR ADC**——逐次逼近ADC
- **TSR/RSR**——发送/接收移位寄存器