AN010 — 用片内定时器实现 16 位 PWM

相关器件

本应用笔记适用于下列器件：

C8051F000、C8051F001、C8051F002、C8051F005、C8051F006、C8051F007、C8051F010、C8051F011、C8051F012、C8051F015、C8051F016、C8051F017、C8051F220、C8051F221、C8051F226、C8051F230、C8051F231 和 C8051F236。

引言

本文说明介绍如何实现一个 16 位的脉冲宽度调制器（PWM）数/模转换器（DAC）。该 PWM 包含两部分：

1. 一个用于产生给定周期和占空度 PWM 的定时器。
2. 一个用于将 PWM 波形转换为模拟电压输出的低通滤波器。

一个 PWM 与一个低通滤波器结合可以用作简单、廉价的数/模转换器（DAC）。其输出可用于驱动电压控制器件，或用于需要模/数转换器（ADC）采样受控参数的反馈控制系统中。PWM 常用于电机控制系统中。

本应用笔记讨论实现 PWM 的软件和硬件。本文提供了用 C8051F226-TB 目标板上的片内定时器和低通滤波器实现 PWM 的例子。该应用示例还将目标板配置为使用片内 ADC 对 PWM 输出进行采样。这一 DAC 实现方案可用于评估 C8051F220/1/6 的 ADC。

关键点

- C8051F2xx 系列 SoC 芯片有三个可用于产生 PWM 的 16 位定时器。本例使用定时器 0 产生 PWM 波形，输出到一个通用端口引脚。
- C8051F2xx 系列 SoC 芯片有一个 8 位 ADC，在本例中用于采样 PWM DAC 的输出。
- C8051F226-TB 目标板提供了一个低通滤波器，该滤波器可很容易地用于 PWM DAC，亦可配置为用片内 ADC 对其采样而不需要另外连线或焊接。本例假定使用目标板。

产生一个 PWM 输入波形

脉冲宽度调制（PWM）是通过改变一个脉冲的宽度或周期波形的占空度来对数据进行编码的一种方法。通过调节波形的占空度，我们能控制低通滤波器的电压输出。我们可以认为这是一个数/模转换器（DAC）。在本例中，我们使用定时器 0 对一个通用端口引脚的高、低电平切换进行定时，以产生所要求的 PWM 波形。

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Email: mcuinfo@silabs.com
Internet: www.silabs.com

新华龙电子有限公司
深圳市福田区华强北路现代之窗大厦 A 座 13F C 室(518013)
电话：0755-83645240 83645242 83645244 83645251
电邮：shenzhen@xhl.com.cn
网址：www.xhl.com.cn

配置定时器 0

为了按用户规定的占空度产生一个 PWM 波形，我们使用定时器 0 的 16 位计数器/定时器方式。为此，我们用下面的代码设置定时器方式寄存器（TMOD）和时钟控制寄存器（CKCON），将定时器 0 配置为使用系统时钟（不分频）：

```
; 设置定时器 0 为 16 位计数器方式
orl         TMOD, #01h
; 设置定时器 0 使用系统时钟/1
orl         CKCON, #08h
```

定时器 0 用于设置一个周期内 PWM 波形为高电平的时间。当定时器溢出时，程序转向中断服务程序（ISR），使某个端口引脚变高或变低，从而产生 PWM 波形。我们通过将 ET0 设置为 1 来允许定时器 0 中断：

```
; 允许定时器 0 中断
setb        ET0
```

另外，还必须置 1 全局中断允许位：

```
; 允许全局中断
setb        EA
```

对定时器 0 进行设置的最后一步是通过置 1 TR0 位来启动定时器：

```
; 启动定时器 0
setb        TR0
```

一个称为 *pulse_width* 的变量定义 PWM 波形的占空度。其值大小决定了在一个周期内波形为高电平的时间，该值被装入到定时器 0。占空度可以以 16 位分辨率设置。但是由于执行定时器 0 中断服务程序需要一定数量的时钟周期（将在后面讨论），可以赋予的最小时钟宽度为 19 个时钟周期。由于同样的原因，中断服务程序将 PWM 波形从高电平变为低电平需要 14 个时钟周期。因此，可以使用的最大值是 65522。变量 *pulse_width* 定义如下：

```
; 定义变量，用户用该变量设置
; 输入到低通滤器的 PWM 波形的占空度
pulse_width         EQU     35000d
```

注意，示例代码中将 *pulse_width* 设置为 35000，该值对应于 53.4%的占空度。占空度按下式计算：

$$占空度\% = \frac{pulsewidth}{65536} \times 100$$

方程 1. 计算占空度

占空度表示波形为高电平的平均时间。这一时间值将在低通滤波器中被转换成电压值。对于一个给定的 *pulse_width* 值，对应的平均输出电压计算如下：

$$Voutput = VDD \times \frac{pulsewidth}{65536}$$

方程 2. 计算平均输出电压

硬件配置

端口引脚 P2.7 用于输出 PWM 波形到 PWM 滤波器。我们通过设置端口 2 配置寄存器（PRT2CF）将 P2.7 设为推挽输出方式。

```
; 设置 P2.7 为推挽方式
orl         PRT2CF, #80h
```

另外，如果使用 Cygnal 的 C8051F226-TB 目标板，必须用跨接片将“PWMIN”跳线短接，以使端口引脚 P2.7 连接到低通滤波器。

等待中断

定时器 0 的 ISR（定时器 0 溢出中断服务程序）用于产生 PWM 波形，这是通过对端口引脚 P2.7 进行电平切换实现的。在完成对各种外设的初始化后，可以用一个简单的原地跳转指令等待中断，这是最常见的作法。但是，当 ISR 用于产生 PWM 波形时，会因中断响应延迟时间的微小变化而出现不希望的微小时序抖动。中断响应延迟时间不同是因为 C8051 在完成当前指令后才能转向中断服务程序。转移到中断服务程序所需的时间取决于中断条件发生时 MCU 处于双周期转移指令的哪一个周期。为避免这种情况，我们可以利用 C8051 MCU 的等待方式。当一个被允许的中断发生时，MCU 会被自动从等待方式“唤醒”。由于 CPU 内核始终处于同一状态，这就消除了中断延迟的不一致性。注意：在等待方式下，所有外设（例如定时器）都照样工作。

将功耗控制寄存器（PCON）中的等待方式选择位置‘1’可使 C8051 进入等待方式。在从中断服务程序返回时，用一条转移语句将程序计数器送回到设置等待方式的那条指令处：

```
; 在等待方式下等待中断
IDLE:
    orl     PCON, #01h
    sjmp    IDLE
```

当从一个 ISR 返回（*reti* 指令）时，MCU 将跳回到 *sjmp* 指令。程序从这条指令又循环回到设置等待方式位的那条指令，然后等待下一个中断条件发生。

用软件（定时器 0 ISR）产生 PWM 波形

PWM 波形是中断服务程序（ISR）通过切换端口引脚电平产生的。该 ISR 是具有两个状态的状态机。在一个状态，输出引脚为高电平（PWM 波形的高电平部分）。在该状态，定时器 0 被装入 *pulse_width* 值，MCU 退出 ISR。在下一个状态，端口引脚变为‘低’电平（通过清零 P2.7 位）。

在低电平状态，定时器 0 被装入*-pulse_width* 值。这就设置了 PWM 波形的低电平时间。在下一次溢出时，测试 P2.7 位，然后将其置 1，进入到下一周期波形的高电平状态。在这种方式下占空度是可以改变的，但 PWM 波形的周期保持不变。

定时器 0 的 ISR 如下：

```
TIMER0_ISR:
; 检查是在波形的低电平还是高电平
jbc     P2.7, LO
setb    P2.7

; 设置 PWM 波形的低电平时间
; 在装入前停止定时器 0
clr     TR0
mov     TH0, #HIGH(-pulse_width)
mov     TL0, #LOW(-pulse_width)

; 重新启动定时器 0
setb    TR0
; 转向 reti 语句
jmp     RETURN

; 设置 PWM 波形的低电平时间
LO:
; 停止定时器 0
clr     TR0
mov     TH0, #HIGH(pulse_width)
mov     TL0, #LOW(pulse_width)
; 重新启动定时器 0
setb    TR0

; 返回到主程序并等待中断
RETURN: reti
```

低通滤波器

按给定占空度生成的 PWM 波形被输入到一个低通滤波器。该滤波器将消除 PWM 波形的大部分高频成分。从时域角度看，RC 电路被充电到一个与 PWM 波形高电平占整个周期百分比（占空度）成正比的电压电平。简言之，低通滤波器将 PWM 波形的高电平时间转换成系统输出端的电压。由于系统输入的是一个数值而输出是一个所要求的电压，因此 PWM 与低通滤波器可以被认为是一种数/模转换器（DAC）。

在示例中，我们使用 C8051F226-TB 目标板上的一阶 RC 滤波器（用跨接片将“PWMIN”跳线短接）。所用的滤波器如图 1 所示。

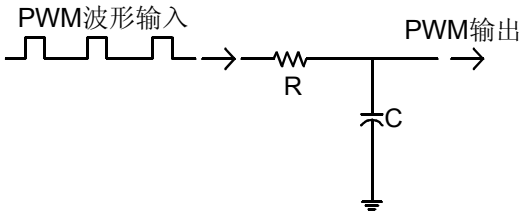

图 1. 低通滤波器

图 1 中的滤波器是简单的一阶滤波器。它的传输函数为：

$$\frac{Vout(s)}{Vin(s)} = \frac{\omega_c}{s+\omega_c}, \left(\omega_c = \frac{1}{RC}\right)$$

方程 3. RC 滤波器传输函数

为了能充分消除波形的高频成分以得到相对平滑的直流电压电平，RC 滤波器必须有相对低的截止频率。但是如果 RC 常数太大，则 RC 电压上升到平滑电平的时间就会过长（即建立时间长）。用计算机模拟或实验测试可以容易地在两者之间进行折衷，选择出合适的电阻/电容值。

该滤波器只有一个极点，因此不能滤出矩形 PWM 波的所有高频成分。电容要经历交替的充电和放电周期，所以输出不可能是一个恒定的直流电压（见图 2）。输出电压中会有一些与滤波器时间常数（τ=RC）相关的“纹波”（图 2 中的 V*ripple*）。在频域，电压纹波可以被认为反映滤波器截止频率（ω=1/R）与 PWM 波形的关系。

在设计低通滤波器时，预测或表征所要求的恒定电压与直流输出电压的偏差可能是很重要的。我们把这一偏差电压称为电压*纹波*（V*ripple*）。为了表征 V*ripple*，我们使用描述 RC 电路中电容两端电压的公式。

图 2 示出输入的 PWM 波形和低通滤波器的输出波形。为了表现 RC 电路中电容的充放电过程，输出波形是经过放大的。对于 50%的占空度（最坏情况的纹波），给定 R、C 及 PWM 波形的周期 T，该滤波器输出的纹波由下式计算：

$$Vripple = VDD\left(1 - \frac{2e^{-\frac{T}{2\tau}}}{1+e^{-\frac{T}{2\tau}}}\right), \tau = RC$$

方程 4. 滤波电路产生的电压纹波

方程 4 是在描述 RC 电路中电容两端电压的公式的基础上，利用方波 PWM 波形（即 50%占空度）的对称性而导出的。注意：最坏情况的纹波由频率（*f* = 1/*T*）和 RC 时间常数（τ）决定。这样说是理由的，因为 RC 组合决定了低通滤波器的截止频率，而 PWM 波形频率已知的，这就可

以表征出有多少方波 PWM 波形的高频成分能够被滤出。

目标板上的 RC 电路使用一个 220kΩ 的电阻和一个 0.47μF 的电容。这些数值的选择是为了在用 8 位 ADC 采样时能得到一个相对平滑的电压电平而又维持一个合理的建立时间。

如果理想输出是一个恒定的直流电压，则输出电压中的纹波可以被认为是误差。为了能在设计滤波器（或求简单 RC 滤波器元件值）时计算这个误差，我们必须知道 PWM 波形的频率和时间常数（τ）。使用目标板上的 RC 取值时，τ=RC=0.1034 秒。如果定时器工作于 16 位方式并且使用 16MHz 的系统时钟速度，则本例中 PWM 周期为：

$$T = \frac{2^{16}}{sysclk} = \frac{65536}{16 \times 10^{6}} \approx 4ms$$

在本例中用方程 4 计算出的预测 V*ripple* 值为 200 mV。

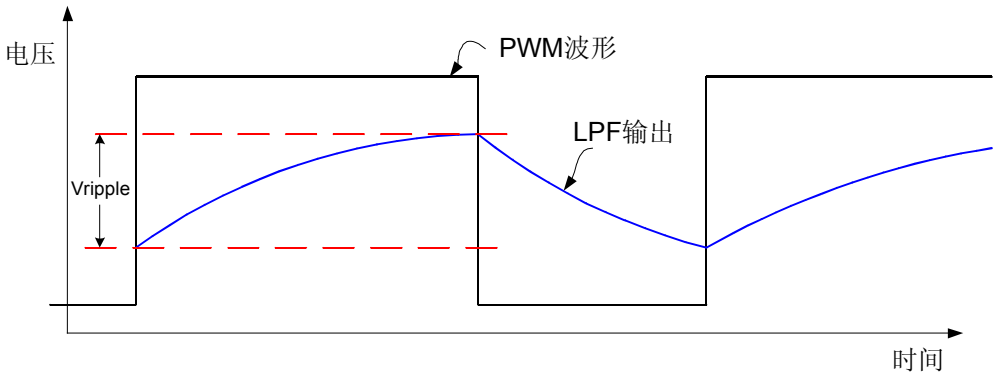

图 2. PWM 波形和滤波器输出

用片内 ADC 采样 PWM 输出

C8051F226-TB 目标板包含一个 C8051F226 SoC，它有一个 8 位模/数转换器（ADC）。在本例中，我们要用该 ADC 对输出电压采样。也可以用电压表对测试点“PWM”上的输出进行测量。为了使用 ADC，我们必须为 ADC 输入配置一个端口并对 ADC 初始化，使其能以我们希望的速率测量 PWM 输出。

配置 ADC

C8051F2xx 系列器件可以用任意一个通用端口引脚作为模拟信号输入。AMX0SL 寄存器用于设置 ADC 的多路开关（AMUX），以选择一个端口引脚作为 ADC 输入。本例中所用的目标板提供的电路能很容易地将 PWM 输出接到端口引脚 P3.0，将 P3.0 设置为 ADC 输入的代码如下：

```
; 允许 AMUX 并将 P3.0 设置为输入
; mov     AMUX0SL, #38h
```

ADC0CF 配置寄存器设置基于系统时钟的 SAR 转换时钟，还设置可编程增益放大器（PGA）的增益。SAR 时钟的最大频率为 2MHz。系统时钟工作于 16MHz，因此将 SAR 转换时钟设置为系统时钟频率的 1/8（即，SAR 转换频率 = sysclk / 8）。我们用如下语句将 PGA 的增益设置为 1：

```
; 设置转换时钟为一个系统时钟并将 PGA 的增益设置为 1
; mov     ADC0CF, #60h
```

ADC0CN 是 ADC 控制器。该寄存器被设置为：在定时器 2 溢出时启动转换，ADC 工作于低功耗跟踪方式（在定时器 2 溢出时启动跟踪）：

```
; SAR 时钟 = SYSCLK / 8
; PGA 增益 = 1
; 定时器 2 溢出
; mov     ADC0CN, #01001100b
```

最后，我们允许 ADC 工作。这一控制位在 ADC0CN 寄存器中，而该寄存器是可以位寻址的，因此我们使用 *setb*：

```
; 允许 ADC
; setb    ADCEN
```

在本例中，我们使用 VDD 电源作为 ADC 的电压基准。这是在 REF0CN 寄存器中设置的：

```
; 设置 ADC 使用 VDD 作为 Vref
; mov     REF0CN, #03h
```

在能利用定时器 2 溢出启动 ADC 转换之前，我们必须设置和启动定时器 2。我们将一个称为 *ADCsampl* 的值装入到定时器 2 对其进行初始化；将同样的值装入定时器 2 的捕获寄存器 RCAP2H:RCAP2L，使其按所希望的采样频率溢出。定时器 2 的自动重装载功能为这一应用提供了方便。采样频率最好与 PWM 波形的频率无关，因为滤波器不是理想的（电容的充电和放电过程导致 V*ripple*），在直流电平中会有一个周期性的偏差电压。采样频率与 PWM 频率不同使我们能通过 ADC 观察这个电压纹波。在本例中我们使用的采样频率为 1.6kHz。

配置定时器 2：

```
; 初始化 T2，使得在 16MHz 系统时钟时 ADC 采样频率为 1.6kHz
mov     TL2, #LOW(ADCsampl)
mov     TH2, #HIGH(ADCsampl)

; 装入对应于 ADC 采样频率的重载值
mov     RCAP2L, #LOW(ADCsampl)
mov     RCAP2H, #HIGH(ADCsampl)
```

```
; 设置定时器 2 使用 sysclk/1
orl         CKCON, #20h

; 启动定时器 2
setb        TR2
```

我们必须允许 ADC 转换结束中断，以便处理 ADC 样本。为了允许 ADC 中断，我们要设置扩展中断允许 2 寄存器（EIE2）：

```
; 允许 ADC 中断
orl         EIE2, #00000010b
```

现在 ADC 被配置为从 P3.0 对输入采样，用定时器 2 设置采样频率。现在所要做的是按下节所述设置端口引脚 P3.0 为模拟输入并将其连到低通滤波器的输出。

配置 ADC 端口

ADC 已经被设置为从 P3.0 输入模拟信号。现在我们必须将该端口设置为模拟用途。

端口引脚在复位后的缺省设置是数字输入方式。我们通过设置端口 3 数字/模拟端口方式寄存器 P3MODE 将端口引脚 P3.0 配置为模拟输入方式：

```
; 配置 P3.0 为模拟输入方式
; orl       P3MODE, #01h
```

注意，我们必须对 PWM 输出和 ADC 输入进行物理连接。可以通过焊线或设计 PCB 来提供这种连接。本例中使用的目标板提供了方便连接的跳线，可以很容易地用跨接片将板上提供的低通滤波器与端口引脚 P3.0 连接在一起。对于本例而言，不需进行焊接或额外的连线。

为了配置外部电路使 PWM 输出信号输入到端口引脚 P3.0（设置为 ADC 输入），用一个短路块放在跳线 J6 上就可以将“PWM”脚连接到“P3.0AIN”。P3.0AIN 与 P3.0 端口引脚已在片内连在一起。

ADC 中断服务程序

在我们的例子中，ADC 中断服务程序只有一个功能：清除 ADC 中断标志，即 ADCINT 位。该标志必须用软件清除，程序如下：

```
ADC_ISR:
clr   ADCINT
reti        ; 中断返回
```

用 ADC_ISR 从 ADC 数据寄存器读取采样数据和进行数据处理是很方便的。在本例中，数据保留在字寄存器（ADC0H）中并可被新样本覆盖。该数据可以用 Cygnal 的集成开发环境（IDE）工具通过查看特殊功能寄存器 ADC0H（该寄存器保持 ADC 转换结果）来观察。

结果阐释

PWM 输出一个与 *pulse_width* 变量（它决定 PWM 波形的占空度）对应的电压电平。如前所述，该输出电压可用第三页中的公式 2 计算。

VDD 是指器件的供电电压。数字 65536 是用 16 位数所能表示的最大值（因为我们的 PWM 定时器是一个 16 位的计数器/定时器）。V*output* 是 PWM 低通滤波器的输出电压。需要注意的是，由于执行定时器 0 中断服务程序需要一定数量的时钟周期，实际上可以使用的最小 *pulse_width* 为 19。因此可以产生的最小 V*output* 为 VDD 的 0.028%。任何小于 19 的 *pulse_width* 值都将产生与 19 相同的结果。类似地，处理 PWM 下降沿的定时器 0 中断服务程序需要 14 个时钟周期。因此，可以使用的 *pulse_width* 的最大值是 65522（65536-14）。对应的输出电压为 VDD 的 99.98%。在 0.028% - 99.98%范围内，除了 16 位定时器分辨率带来的量化误差外，再没有因软件引起的其它限制。例如，如果 VDD=3.0V，则电压分辨率为 46 μV，相应的输出电压值的范围为 0.87V 到 2.9994V。

在我们的例子中，我们用片内 ADC 测量 PWM 的输出。ADC 寄存器（ADC0H）中的结果，将是 0 和 255（8 位 ADC）之间的一个值。本例使用 VDD 作为 ADC 转换的基准电压。ADC 输出值可以表示如下：

$$Vresult = VDD \times \frac{ADC0H}{256}$$

注意，V*result* 不可能与计算得到的 PWM 输出值 V*output* 完全一致。这是因为存在前面讨论过的 V*ripple*（见“低通滤波器”一节）。

软件示例

```
;
; 在目标板 SA_TB4PCB-002 上实现一个 16 位的 PWM，
; 通过采样测试片内的 8 位 A/D 转换器（ADC）。
; 下面的程序配置片内外设并使用目标板上的低通滤波器。
;
; 文件：            PWM.asm
; 器件：            C8051F2xx
; 开发工具          Cygnal IDE, 8051 汇编器 (Metalink)
; 作者：            LS
;-----------------------------------------------------------------
$MOD8F200
;-----------------------------------------------------------------
;
; 复位向量
;
   org      00h
   jmp      MAIN
;
;-------------------------------------------------------------------------------------------------------
;
; ISR 向量
   org      0Bh
   jmp      TIMER0_ISR

   org      7Bh
   jmp      ADC_ISR

;-------------------------------------------------------------------------------------------------------
; 常数
pulse_width     EQU     10000d          ; 定时器 0 的装入值，该值调整 PWM
                                        ; 波形的高电平宽度（占空度），
                                        ; 因而调节低通滤波器输出的直流偏置电平
                                        ; 设置范围为 19-65522d。
                                        ; 32768 = VDD/2

ADCsampl        EQU     55536d          ; 定时器 0 的装入值，用于设置 ADC 采样速率
```

```
; 主程序代码 ----------------------------------------------------------------------------

        org  0B3h

MAIN:
        mov     OSCICN, #07h                ; 配置内部振荡器为 15MHz
        mov     WDTCN, #0DEh                ; 机智看门狗定时器
        mov     WDTCN, #0ADh
        orl     PRT2CF, #80h                ; 配置 P2.7 为推挽方式，
                                            ; 作为低通滤波器的输入
        mov     P3MODE, #0FEh               ; 配置 P3.0  为模拟输入
        orl     CKCON, #28h                 ; 设置定时器 0 和定时器 2 使用系统时钟
        orl     TMOD, #01h                  ; 设置定时器 0 为 16 位计数器方式
        mov     RCAP2H, #HIGH(ADCsampl)     ; 装入设置 ADC 采样速率的自动重装载值
        mov     RCAP2L, #LOW(ADCsampl)      ; 用定时器 2 溢出启动 ADC 转换
        mov     TH2, #HIGH(ADCsampl)        ;初始化 T2，使 ADC 采样频率  =1.6KHz
        mov     TL2, #LOW(ADCsampl)
        orl     AMX0SL, #38h                ; 设置 AMUX，选择 P3.0 为输入/允许 AMUX
        mov     ADC0CF, #60h                ; 设置转换时钟  = SYSCLK/8，PGA 增益=1
        orl     ADC0CN, #00001100b          ; 设置 ADC 在定时器 2 溢出时启动转换
        orl     REF0CN, #03h                ; 选择内部电压基准
        orl     EIE2,#00000010b             ; 允许 ADC 转换结束中断
        setb ET0                            ; 允许定时器 0 中断
        setb EA                             ; 全局中断允许
        setb TR0                            ; 启动定时器 0
        setb TR2                            ; 启动定时器 2
        setb ADCEN                          ; 允许 ADC
IDLE:
        orl     PCON,#01h
        sjmp    IDLE                        ; 中断返回后循环回到置 MCU 为等待方式处

;------定时器 0 ISR----------------------------------------------------
TIMER0_ISR:
        jbc     P2.7,LO                     ; 测试波形处于低/高电平
        setb P2.7                           ; 从低电平变为高电平
        clr     TR0                         ; 在装入期间停止定时器 0
        mov     TH0,#HIGH(-pulse_width)     ; 设置与 DC 偏置电平对应的脉冲宽度
```

```
        mov     TL0,#LOW(-pulse_width)
        setb TR0                                ; 重新启动定时器 0
        jmp     RETURN
LO:     clr     TR0
        mov     TH0,#HIGH(pulse_width)          ; 设置低电平时间
        mov     TL0,#LOW(pulse_width)
        setb    TR0
RETURN:  reti                                   ; 中断返回

;------ADC ISR---------------------------------------------------------------
ADC_ISR:
        clr     ADCINT                          ; ADC 必须用软件清除
        reti                                    ; 中断返回

;-------------------------------------------------------------------------------

;程序结束
END
```