《egPLC5888 用户手册》

Rev3.0

概述

egPLC5888 是一种可编程 PLC/RTU 设备，应用于工业数字化中，用于数据采集和控制驱动，集成信号采样、信号处理、工业控制驱动以及通讯物联于一体。它在低功耗、高效率、抗干扰的环境下运行，能够进行工业过程控制和定制开发设备。egPLC5888 内嵌功能强大的 32 位微处理器，用于应用程序二次开发。通过 RS485 进行通信为网络建设提供了一个灵活、强大的拓扑结构，可用于远程传感和控制。时序数据采集的应用之一为客户提供了有价值的制造过程控制。

egPLC5888 PLC/RTU 支持 27 个物理 IO 端口

-1 个 USB 设备端口，用于 egPLC5888 程序下载和设备的 PC API 访问

-1 个 RS485 主端口，用于连接其他 RS 485 从设备和仪器

-1 个 RS485 Modbus 从端口，用于将远程 485 主端口或 PC API 连接到设备 (可配置为主端口)

-8 个电流 ADC 通道，用于采样 4-20 mA 模拟电流信号（IO 引脚与模拟电压 ADC 通道共享）

-8 个电压 ADC 通道，用于采样 0-5V 模拟电压信号（IO 引脚与模拟电流 ADC 通道共享）

-2 个计数器采样输入通道，用于脉冲计数信号（与数字输入通道 X0/X1 共享的 IO 引脚）

-8 个数字信号输入通道，用于对数字电压信号进行采样

-8 个固态继电器输出通道，用于环境控制驱动。其中 4 个为 4 个 PWM（脉宽调制）通道的公共引脚，可用于高速脉冲输出，Y3/Y2/Y1/Y0。

IO 端口与 CPU 分离，具有光电耦合保护，最大 IO 输入持续电压不低于 24VDC。

egPLC5888 电源采用超宽范围设计，可在 5VDC 至 24VDC 之间正常工作。

图 1 是 egPLC5888 及其 IO 端口的真实照片图像

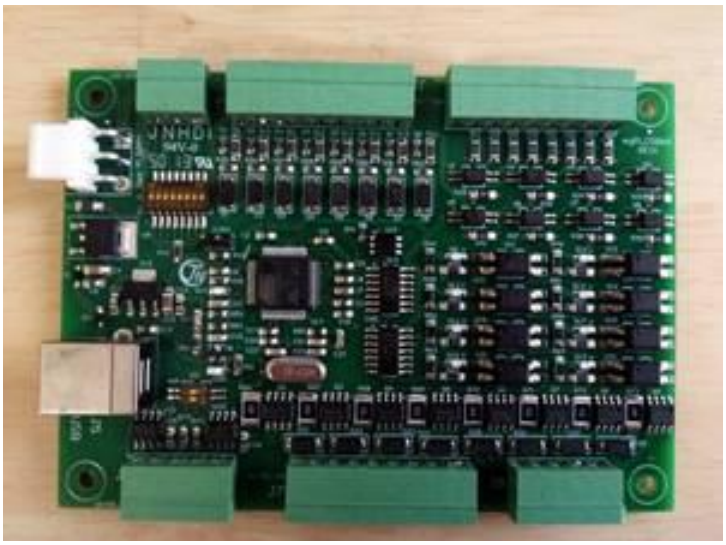

图 1. egPLC5888 的图像

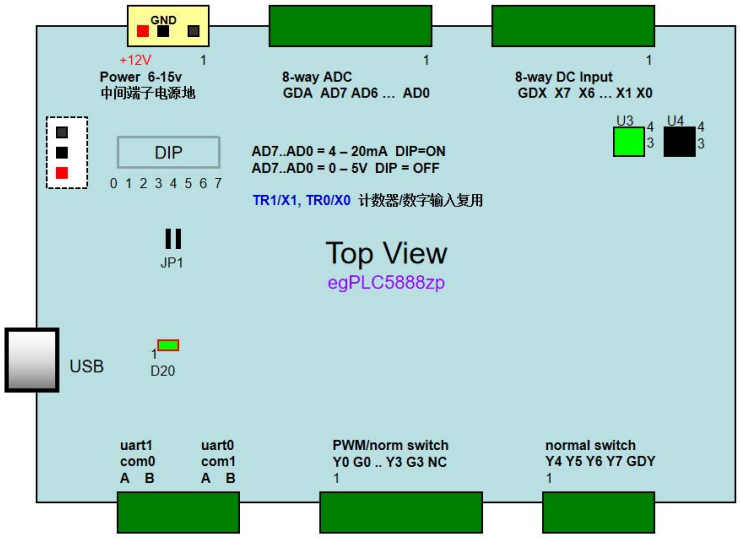

图 2. egPLC5888 及其 IO 端口布局

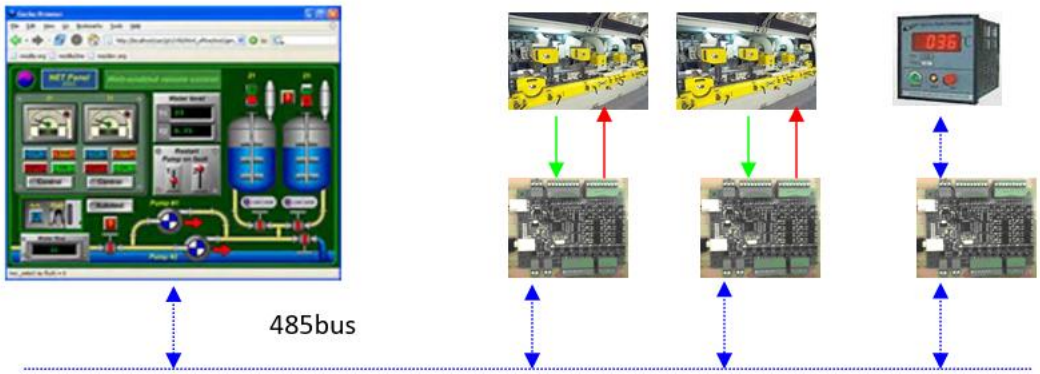

图 3. egPLC5888 网络拓扑的应用示例

egPLC5888 IO 引脚的电气特性

RS485 从端口：差分 IO 信号输入/输出 A 和 B，带 8 位数据帧，2 位停止，无位奇偶校验。通信波特率和其他参数可通过“可视状态机”软件实用程序设置。RS485 从端口嵌入 Modbus 协议，用于远程主机访问任何 egPLC5888 内部寄存器。RS485 端口符合 RS485 Modbus RTU 协议，该协议坚固耐用，可在恶劣环境下操作，用于远程设备通信。

RS485 主端口：差分 IO 信号输入/输出 A 和 B，带 8 位数据帧，2 位停止，无位奇偶校验。通信波特率和其他参数可以通过“可视状态机”软件实用程序和编程方式设置。egPLC5888 可以访问群集在同一 RS485 上的从属 RS485 设备。RS485 主端口在硬件上符合 RS485 总线协议，坚固耐用，可在恶劣环境下进行远程信号通信。

8 个固态继电器输出 Y0..Y7，每个通道维持 60VDC 和 4A 电流脉冲。推荐电源电路建议使用机械中间继电器。典型应用如下（保险丝可选）：

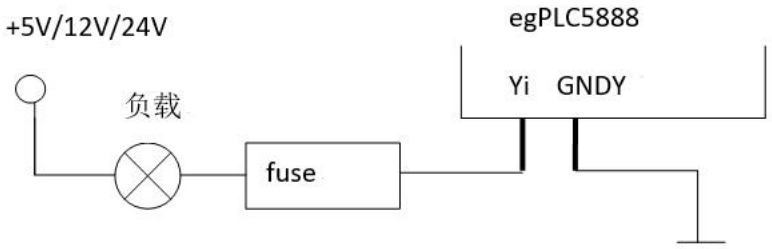

图 4.输出控制的连接图

根据应用要求选择负载。负载的最低驱动电压建议为+5VDC，用户可根据需要提高负载电压和功耗。可选保险丝用于电流过载保护。正确选择此类测量值是为了保护目的。

egPLC5888 有 8 个固态继电器输出，其中 4 个具有高速脉冲 PWM 功能。输出频率和占空比可由 VSM 嵌入式 C 语言控制。要控制 IO 输出，可以通过 C 语言编程下载，也可以通过 PC USB API 和 485 Modbus 远程上位机 API 访问来实现。

egPLC5888 有 8 个数字通道 X0..X7。当外部输入电压大于 3.6V 时，输入数字信号被解释为高，当外部电压信号小于 2V 时，输入数字信号被解释为低。建议的输入信号应大于 5V 以显示高电平，低于 1V 以显示低电平。每个输入可维持高达 36V 的直流电压。

除此功能外，X0 和 X1 还具有脉冲计数能力。该输入脉冲计数功能是一种并行硬件设计，尽管它们在物理上与数字电平输入共用同一引脚。通过嵌入式 C 代码访问连接的内部寄存器，可以读取/设置/重置计数器。此外，用户还可以通过 USB 端口和 RS485 Modbus 从端口以及远程 PC API 访问该硬件功能。

egPLC5888 有 8 个模拟输入通道 AD0..AD7。egPLC5888 具有模拟电流和电压采样功能。可通过 dip 开关配置选择输入类型。出厂时默认配置为电压输入。电压模拟输入范围为 0 到 5VDC，分辨率为 10 位，这意味着测量精度可以达到 5V 的 0.1%。最大耐受输入电压为 20VDC，内部输入电阻为 5K 欧姆。对于通过配置选用模拟电流输入用法，最大耐受输入电压为 10VDC，其内部输入电阻为 250 欧姆。

egPLC5888 IO 空间寄存器地址

egPLC5888 简化了 64K IO 寄存器空间。如果未特别指定，则所有寄存器均为 32 位。所有地址都与 4 字节地址对齐。内存存储器中的所有寄存器都符合 big-endian 格式。

Reg. address (hex)	Reg. Mnemonic	Interpretation	Read/write mode
0000 .. 03FC	IO_MEM	256 个 32 位通用寄存器	读/写
1000	AD0	10 位 ADC 结果寄存器	只读
1004	AD1	10 位 ADC 结果寄存器	只读
1008	AD2	10 位 ADC 结果寄存器	只读
100C	AD3	10 位 ADC 结果寄存器	只读
1010	AD4	10 位 ADC 结果寄存器	只读

1014	AD5	10 位 ADC 结果寄存器	只读
1018	AD6	10 位 ADC 结果寄存器	只读
101C	AD7	10 位 ADC 结果寄存器	只读
2000	TR0	16 位计数器	只读，写入 0 以清除
200C	TR[0].运行	写入 1 开始计数，0 停止	只写
2010	TR1	16 位计数器	只读，写入 0 以清除
201C	TR[1].运行	写入 1 开始计数，0 停止	只写
3000	X0	1 位数字输入寄存器	只读
3004	X1	1 位数字输入寄存器	只读
3008	X2	1 位数字输入寄存器	只读
300C	X3	1 位数字输入寄存器	只读
3010	X4	1 位数字输入寄存器	只读
3014	X5	1 位数字输入寄存器	只读
3018	X6	1 位数字输入寄存器	只读
301C	X7	1 位数字输入寄存器	只读
4000	Y0	1 位数字输出寄存器	读/写
4004	Y1	1 位数字输出寄存器	读/写
4008	Y2	1 位数字输出寄存器	读/写
400C	Y3	1 位数字输出寄存器	读/写
4010	Y4	1 位数字输出寄存器	读/写
4014	Y5	1 位数字输出寄存器	读/写
4018	Y6	1 位数字输出寄存器	读/写
401C	Y7	1 位数字输出寄存器	读/写

5000	PWM[0].频率	16 位 PWM 频率寄存器	读/写
5004	PWM[0].占空比	16 位占空比寄存器 0..100	读/写
5008	PWM[0].POL	波形正/反相输出	读/写
500C	PWM[0].运行	1 位运行/停止寄存器	读/写
5010	PWM[1].频率	16 位 PWM 频率寄存器	读/写
5014	PWM[1].占空比	16 位占空比寄存器 0..100	读/写
5018	PWM[1].POL	波形正/反相输出	读/写
501C	PWM[1].运行	1 位运行/停止寄存器	读/写
5020	PWM[2].频率	16 位 PWM 频率寄存器	读/写
5024	PWM[2].占空比	16 位占空比寄存器。 0..100	读/写
5028	PWM[2].POL	波形正/反相输出	读/写
502C	PWM[2].运行	1 位运行/停止寄存器	读/写
5030	PWM[3].频率	16 位 PWM 频率寄存器	读/写
5034	PWM[3].占空比	16 位占空比寄存器。 0..100	读/写
5038	PWM[3].POL	波形正/反相输出	读/写
503C	PWM[3].运行	1 位运行/停止寄存器	读/写
6000	计时器[0]	32 位时钟，以秒为单位	读/写
6004	计时器[1]	32 位时钟，以秒为单位	读/写
6008	计时器[2]	32 位时钟，以秒为单位	读/写
600C	计时器[3]	32 位时钟，以秒为单位	读/写
6010	计时器[4]	32 位时钟，以秒为单位	读/写
6014	计时器[5]	32 位时钟，以秒为单位	读/写
6018	计时器[6]	32 位时钟，以秒为单位	读/写

601C	计时器[7]	32 位时钟，以秒为单位	读/写
6020..603C	计时器[8..f]	32 位时钟，以秒为单位	读/写
6100	计时器[10]	32 位时钟，单位为毫秒	读/写
6104	计时器[11]	32 位时钟，单位为毫秒	读/写
6108	计时器[12]	32 位时钟，单位为毫秒	读/写
610C	计时器[13]	32 位时钟，单位为毫秒	读/写
6110	计时器[14]	32 位时钟，单位为毫秒	读/写
6114	计时器[15]	32 位时钟，单位为毫秒	读/写
6118	计时器[16]	32 位时钟，单位为毫秒	读/写
611C	计时器[17]	32 位时钟，单位为毫秒	读/写
6120..613C	计时器[18..1f]	32 位时钟，单位为毫秒	读/写
C000/C004	IDAT（com0/com1）	RS485 主读端口	只读
C100..C3FF C400..C7FF	ODAT（com0/com1）	768/1024 字节 RS485 输出发送数据写入端口	读写
C030/C034	CDAT（com0/com1）	要发送的输出字节数	仅发送到触发器
C040/C044	波特率（com0/com1）	接收/发送波特率	仅写，设置波特率
C050/C054	模式（com0/com1）	模式寄存器，参考下表 2	只写
E000	AD0-AD1	RS485 AD 压缩通道（每个 16 位）共 2 个通道	只读
E004	AD2-AD3	RS485 AD 压缩通道（每个 16 位）共 2 个通道	只读
E008	AD4-AD5	RS485 AD 压缩通道（每个 16 位）共 2 个通道	只读
E00C	AD6-AD7	RS485 AD 压缩通道（每个 16 位）共 2 个通道	只读
E010	XY	------X7..X0-----Y7..Y0 32 位，位图为 8 输入 X，	只读

		8 为输出 Y	
F000..F3FC	NVR	256 个 32 位非易失性寄存器	读/写

表 1. egPLC5888 OI 空间定义

egPLC5888 有 2 种变体：egPLC5888xp 有 1 个从端口和 1 个主端口（com1）；egPLC5888zp 有 2 个主端口（com0 和 com1 都是主端口）。

通信方式	定义	描述	示例，数据位：奇偶校验方式：停止位
位<3..0>	传输的数据的停止时间	0：用于 1 位停止	0x8020，8:N:1
		1：用于 1.5 位停止	0x8021，8:N:1.5
		2：用于 2 位停止	0x8022，8:N:2
位<7..4>	字节的奇偶校验模式	0：偶数奇偶校验	0x8002，8:E:2
		1：奇数奇偶校验	0x8012，8:O:2
		2：无奇偶校验	0x8022，8:N:2
位<11..8>	保留		
位<15..12>	每个字节流的位数	8：8 位为基准	0x8022，8:N:2
		7：7 位为基准	0x7022，7:N:2

表 2. RS485 通信模式定义

IO_MEM 寄存器空间：0x0000–0x03FF，共 1K 字节，256 个 32 位寄存器，具有读写功能。这些可通过客户下载的 C 代码或 PC API 访问，用于多端口/多设备访问的数据同步通信。

ADC IO 空间 AD0..AD7:0x1000–0x101F，总共 32 字节，8 个 32 位 ADC 采样寄存器，只读。数据采集过程由硬件自动完成。用户可以随时访问它们，用户读取的是最新的采样结果。

计数器寄存器空间：TR0 和 TR1。它们的 IO 空间地址为 0x1000 和 0x1010。

0x1000 和 0x100C 是第一个计数器寄存器及其启用寄存器；0x1010 和 0x101C 是第二个计数器寄存器及其启用寄存器。启用寄存器可用于启用/清除计数器寄存器。

MOV R7, 1

OUT R7, [0x200C]

IN R6, [0x2000]

X0 信号端口开始接收上升沿计数，寄存器 R6 的内容是启用后捕获的脉冲事件数，即输入信号上升沿脉冲数。

数字信号 IO 空间 X0..X7:0x3000–0x301F，共 32 字节，8 个通道，每个通道 4 字节，只读。

数字信号输出（固态继电器）IO 空间 Y0..Y7:0x4000–0x401F，共 32 字节，8 个通道，每个通道 4 字节，读/写。

PWM 输出（固态继电器）IO 空间：0x5000–0x503F，总共 64 个字节，4 个通道，每个通道 16 个字节，每个通道有 4 个字段寄存器，即频率、占空比、相位极性和启动/停止寄存器。

系统时钟定时器 0..15 是以秒为单位计数的 16 个定时器；TIMER16..TIMER31 是以毫秒为单位计数的 16 个计时器。它们可以在任何时候读/写，并且一直在计数。

RS485_0 IO 空间：0xC000 输入数据读取端口。当输入缓冲区无数据时，读访问返回-1。

0xC100 输出数据写入端口，带 768 字节缓冲区。

0xC030 数据发送端口，仅写，用于指示从输出数据缓冲区发送的字节数。

波特率设置 0xC040 临时存储在寄存器中。

0xC050 模式寄存器、字节的串行数据位、奇偶校验类型和停止位。

RS485_1 IO 空间：0xC004 输入数据读取端口。当输入缓冲区无数据时，读访问返回-1。

0xC400 输出数据写入端口，带 1K 字节缓冲区。

0xC034 数据发送端口，仅写，用于指示从输出数据缓冲区发送的字节数。

0xC044 波特率寄存器，此处的波特率设置临时存储在 RAM 中。

0xC054 模式寄存器、字节的串行数据位、奇偶校验类型和停止位。

IO 空间访问是通过嵌入式微处理器的输入/输出指令，egPLC5888 系统为用户开发提供 C 语言子程序接口。

RS485 Modbus 协议

egPLC5888 RS485 从端口嵌入 Modbus 通信协议，使其他主设备能够使用 485 数据总线与该设备通信。egPLC5888 Modbus 命令简化为两类，即寄存器读取和寄存器写入操作。egPLC5888 RS485 从机接收远程主 PC Modbus 命令，然后对其进行处理并立即返回。实现了读取命令（功能代码 0x03）和写入命令（功能代码 0x10）两种命令格式。

命令和数据帧以 Modbus RTU 格式的字节码形式表示，使用 2 个十六进制数字表示一个字节。在下面的示例中，我们将假设 egPLC5888 485 从机地址为 0xaa。

远程 PC 读取 egPLC5888 寄存器操作={addr，03，sah，sal，regsh，regsl，crcl，crch}

egPLC588 返回={addr，03，nbytes，w1h，w1l，w0h，w0l，…，crcl，crch}，其中连续的 4 个字节四倍构成一个 32 位长字（w1h，w1l，w0h，w0l），在总线传输和寄存器存储状态下均为大端格式。

其中 sah 是目标寄存器地址的高位字节，sal 是目标寄存器地址的低位字节，regsh 是要访问的 16 位寄存器的高位字节数，regsl 是要访问的 16 位寄存器的低位字节数，crcl 是 CRC 的低位字节，crch 是 CRC 的高位字节，w1h 是 4 字节寄存器内容的最高有效字节，w0l 是 4 字节寄存器内容的最低有效字节。

示例 1：主 PC 读取 egPLC5888 Y0 数字输出（固态继电器）状态。

PC 输出数据帧：aa 03 40 00 00 02 c8 10，其中 aa 是 egPLC5888 的 modbus 地址

从 egPLC 数据帧返回：aa 03 04 00 00 00 01 21 39

示例 2：主 PC 读取电流/电压的 egPLC5888 AD0 模拟输入。

PC 输出数据帧：aa 03 10 00 00 02 d9 10，其中 aa 是 egPLC5888 的 modbus 地址

从 egPLC 数据帧返回：aa 03 04 00 00 01 ab a0 d6

远程 PC 写入 egPLC 寄存器操作={addr，10，sah，sal，regsh，regsl，nbytes，datH，datL，…，crcl，crch}

egPLC 返回={addr，10，sah，sal，regsh，regsl，crcl，crch}

其中 sah 是寄存器地址的高位字节，sal 是寄存器地址的低位字节，crcl 是 CRC 的低位字节，crch 是 CRC 的高位字节，regsh，regsl 是访问寄存器的编号（这里每个 32 位寄存器正式表示为 2 个 16 位寄存器），nbytes 是要写入的实际数据字节数。

示例 3：主 PC 将 egPLC5888 Y0 数字输出（固态继电器）状态写入 0（开启）。

PC 输出数据帧：aa 10 40 00 00 02 04 00 00 00 00 e5 4a

从 egPLC 数据帧返回：aa 10 40 00 00 02 4d d3

示例 4：主 PC 将 egPLC5888 Y1 数字输出（固态继电器）状态写入 0（开启）。

PC 输出数据帧：aa 10 40 04 00 02 04 00 00 00 00 e4 b9

从 egPLC 数据帧返回：aa 10 40 04 00 02 0c 12

egPLC5888 RS485 通信波特率可通过“可视状态机”软件实用程序设置。egPLC5888 RS485 主端口和从端口使用相同的默认通信设置。但是，对于 RS485 主端口，可以通过用户嵌入的 C 代码更改通信参数。

egPLC5888 嵌入式 C 语音编程

egPLC5888支持C语言应用程序开发。系统为用户程序堆栈和变量保留32KB的RAM空间，并为二进制代码下载保留128KB的闪存。开发工具链是

C编译器：cc105.exe

汇编语言汇编程序：as105.exe

模拟器：sim105.exe

二进制代码下载程序：fsh4888.exe

用户可以利用这套工具开发egPLC5888嵌入式定制应用程序，实现板级定制。用户还可以使用图形界面IDE实用程序“可视化状态机”来开发定制程序。下面介绍此IDE开发实用程序。

在介绍此IDE开发环境之前，将导出一些egPLC5888 C语言原型（在内置h文件中定义），如下所示：

```
void switch_to(long addr);
```

此函数实现从源状态到目标状态的状态机转换，用户程序从一个状态跳到下一个状态。addr是目标状态的地址，是标签或函数名。

```
void br_delay(int ms);
```

此函数以毫秒为单位实现用户程序控制的延迟。egPLC5888用户延迟分辨率为10毫秒。建议参数“ms”为10的倍数。

```
long br_inport(long addr);
```

此函数用于从IO寄存器空间读取寄存器内容。参数“addr”是IO空间寄存器地址的32位整数。

```
void br_outport(long addr, long dat);
```

此函数写入32位整数以从IO空间注册内容。参数“addr”是目标IO空间寄存器的地址。

```
void qdat_mem_copy(char *des_io, char *src_mem, int cnt) ;
```

此函数用于将数据块从源区域复制到目标区域，地址可以是内存地址和IO缓冲区地址。

开发包提供“syslib.h”和“syslib.asm”，上述 C 函数被视为内部内置 C 函数的一部分，代码体在“syslib.asm”中预编译。以下示例显示了使用该代码控制 Y7 指示灯闪烁。

可视化状态机软件包提供 C 语言编程开发环境。

egPLC 图形用户界面开发工具-可视化状态机

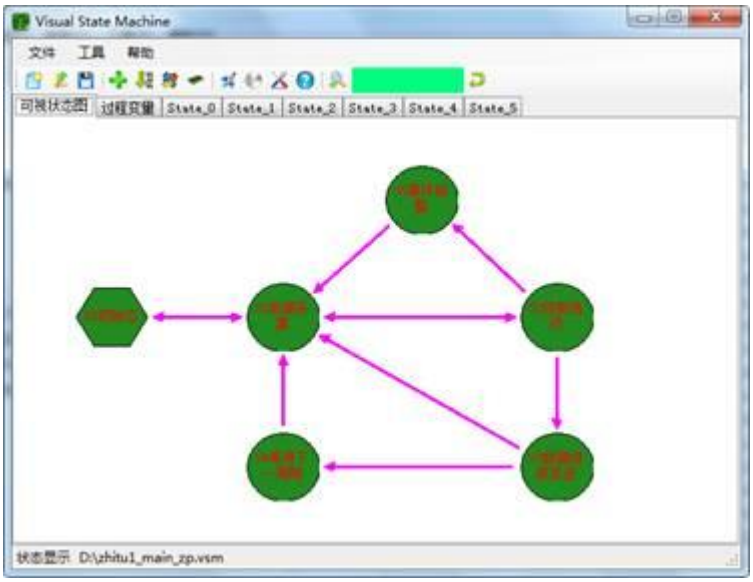

图 5.实际应用程序的状态转换图

可视化状态机（VSM）提供了一个 GUI C 语言开发环境。与其他 IDE C 语言开发环境相比，该软件包为用户提供了更自由、更简单的 C 语言开发环境。用户无需编写完整的 C 程序，只需关注状态转换及其逻辑，有限状态机的完整 C 程序由软件系统自动生成。有限状态机是 egPLC5888 系列产品的软件体系结构，它为编写和定制各种数据采集、控制驱动和通信程序提供了一种清晰、简单的方法。最后，这将为用户提供清晰的逻辑和解决问题的效率改进。

VSM 是专门为 egPLC5888 系列设备准备的，具有内置功能，可在 C 语言中定制板级功能。egPLC5888 非常适用于工业过程控制和远程数据采集，以及所有相同体系结构的独立机电设备。

VSM 将每个状态视为一个子例程。程序从的先前状态转换到下一个状态是通过函数 Switch_to()完成的。前台状态转换图和每个状态的代码体由软件系统在后台用 main() 入口点和<syslib.h>头进行封装和完成。整个程序最终被编译并汇编成二进制机器代码。

VSM 软件功能包括状态图编辑和 C 代码编辑。VSM 还集成了 C 语言开发工具链，包括 C 语法检查、C 程序编译、汇编和二进制代码下载。有关更多详细信息，请参阅 VSM 软件和附录部分，了解更多 egPLC5888 C 语言编程讨论。

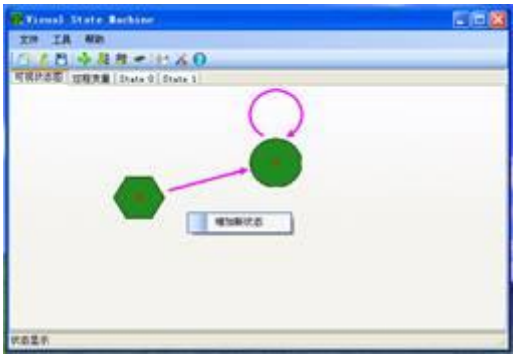

图 6.通过右键单击添加新状态

上面的VSM图显示了新状态被创建并添加到状态机中。“可视状态图”选项卡显示从状态到状态的转换的有限状态机。我们可以编辑每个状态的属性，设置状态名称或切换路径。

图7.更改状态的属性

VSM菜单功能包括来自egPLC5888的清除程序，以及设置RS485参数的波特率和Modbus从机地址。

图8.设置RS485通信参数

创建 VSM 项目文件后，我们可以通过软件提供的菜单功能编辑可视状态图和 C 代码。它们是语法检查、编译、汇编和程序下载。

VSM 编程示例 1 是控制数字输出 Y7，输出“0”（关闭继电器）1 秒，然后输出“1”（打开继电器）0.5 秒。然后重复这个动作。

State	Output reg. addr.	Output data val.	Delay time	Next State	remark
State_0				State_1	initialization
State_1	0x401c	0	1000ms	State_2	relay on
State_2	0x401c	1	500ms	State_1	relay off

每个状态的可视状态图和代码如下所示。

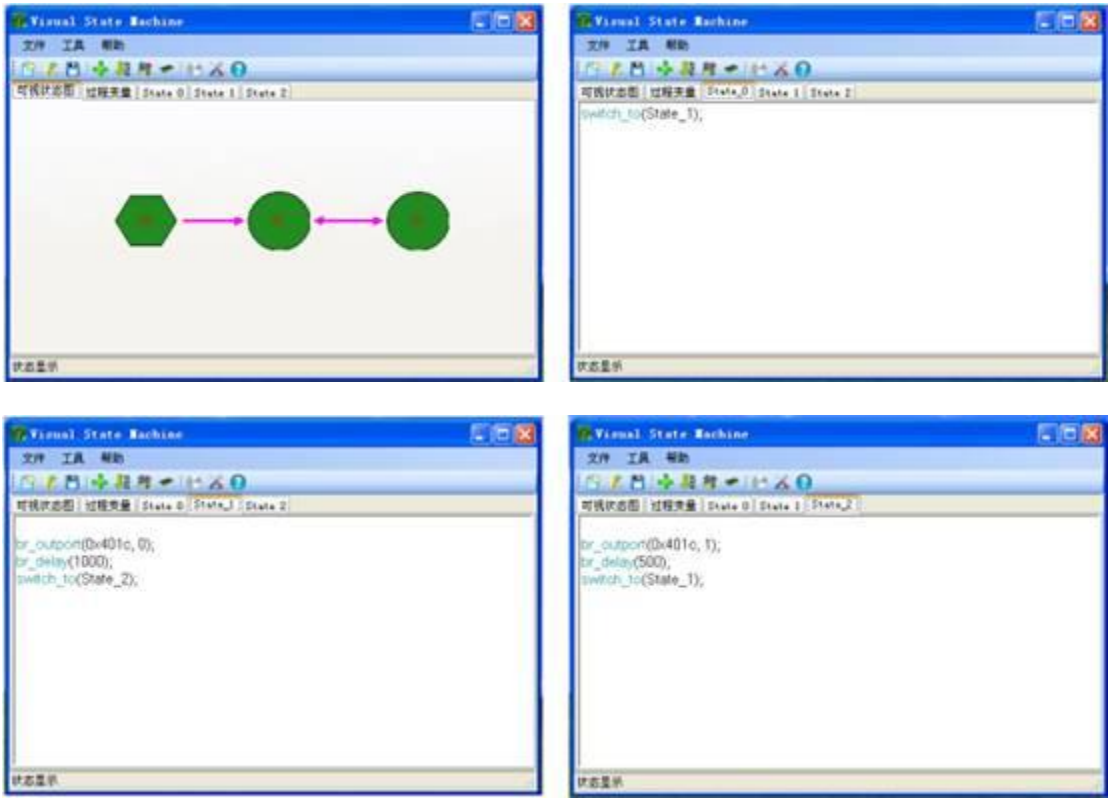

图 9.示例 1，闪烁

请注意数字通道 7（Y7）的 PLC5888 输出指示灯 led

VSM 编程示例 2，观察数字输入通道 X7 的状态和 Y7 的输出。

State	Input reg. addr.	Output reg. addr.	Variable name	remark
State 1	0x301c	0x401c	tmp	

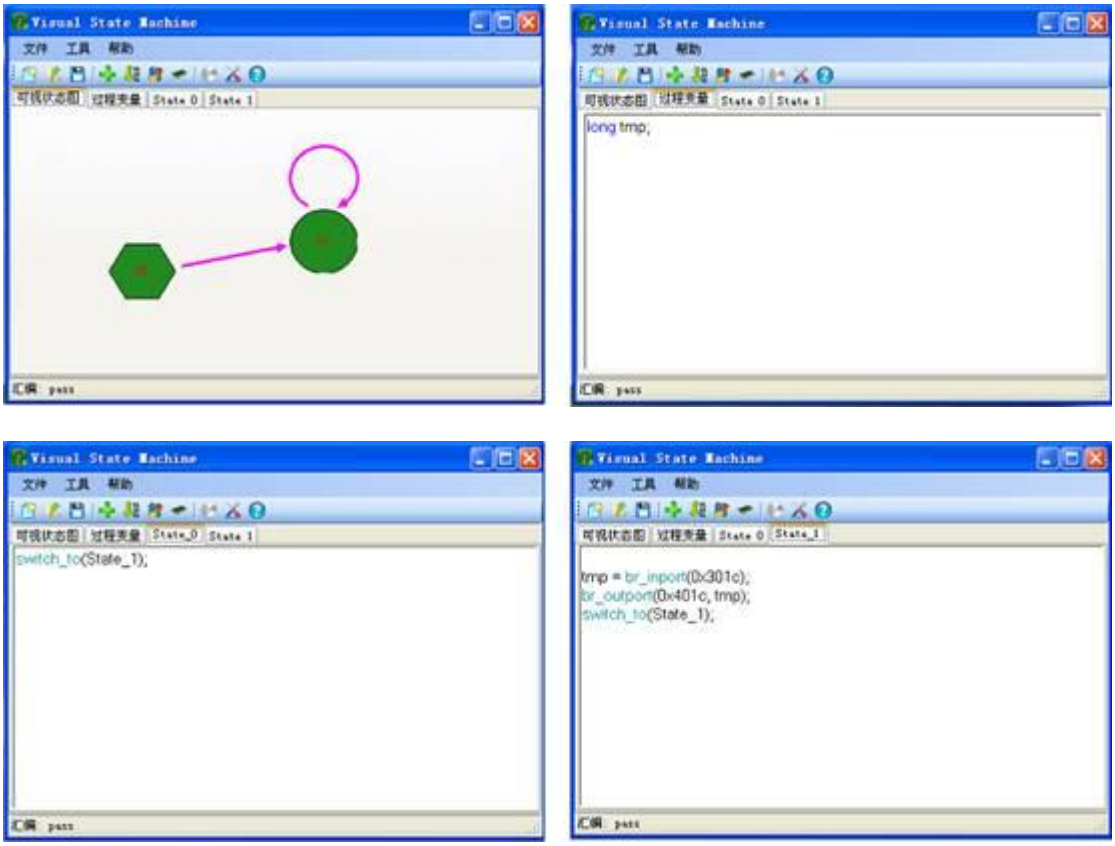

图 10.示例 2，PWM 输出

当输入 X7 连接到 5V 或更高电压时，返回结果为“1”。当 X7 连接到 1V 或更低电压时，返回结果为“0”。输出 Y7 反映输入 X7。

视觉状态机编程示例 3 是通过引脚 Y1 通过 PWM 实现高频信号输出。Y0..Y3 输出引脚可用于正常固态继电器输出，也可用于高速开关频率输出。采用 PWM 接口实现高速开关输出。与电路回路一起工作需要外部电源。

每个 PWM 通道有 4 个寄存器来定义 PWM 字符。这 4 个通道是独立的，同时工作。

输出频率寄存器：地址偏移量为 0x00，用于设置输出波形频率。

占空比寄存器：地址偏移量为 0x04，百分比值范围为 0 到 100

波极性相位寄存器：地址偏移量为 0x08，从低或高开始

运行/停止寄存器：偏移地址为 0x0c，其中 0 表示停止，1 表示开始输出。

这里我们想要的是从输出 Y1 引脚输出频率设置为 2000Hz 的对称方形波形。为了实现这个实验，我们只需将前面示例中的 State_1 代码替换为下面的代码，然后编译、组装和下载程序。

```
br_outport (0x5010,  2000);
br_outport (0x5014,  50);
br_outport (0x5018,  1);
br_outport (0x501c,  1);
while (1) br_delay(1000);
```

请注意 Y1 的 led 指示灯（“0”亮起，“1”熄灭）

如果频率太高，无法观察 led 的变化，则我们将频率调整为低至 1 Hz。

```
br_outport (0x501c,  0);
br_outport (0x5010,  1);
br_outport (0x5014,  50);
br_outport (0x5018,  1);
br_outport (0x501c,  1);
while (1) br_delay(1000);
```

现在请再次观察 Y1 的 led 指示灯，查看波形的输出变化。

通信接口和上位机 PC API

egPLC5888 具有 3 个通信端口、1 个 USB 和 2 个 RS485 端口。主 PC 使用 USB 通信端口通过 API 访问 egPLC5888 IO 空间寄存器。USB 端口还用于下载用户程序和设置 485 通信波特率。RS485 端口包括 1 个主端口和 1 个从端口。从端口嵌入 Modbus 协议，从端口可通过 RS485 Modbus API 访问。无论哪种方式，IO 空间寄存器同时也可以通过 C 编程语言访问。

PC Windows API 包括一组 dll 动态模块和静态模块。它们支持 PC 主机 C/C++，C#，Vb 开发编程接口。API 库支持 USB、RS485（从端口）和 TCP/IP 网络通信。

windows 网络 API 的 3 个类封装在一个库类中，包含 3 个文件：

egCommIO.h、egCommIO.lib、egCommIO.dll

egPLC5888 支持 USB 通信模块和 RS485 通信模块。此产品型号不支持 TCP/IP 端口。

.Net API 封装了 3 个库类，其中包括 egUsbPort、eg485Port 和 egNetPort。egTreeIO.dll 将上述 3 个库类集成到.Net 体系结构下的单个库类中，这为.Net Windows GUI 编程开发提供了一种简单的方法。具体情况如下：

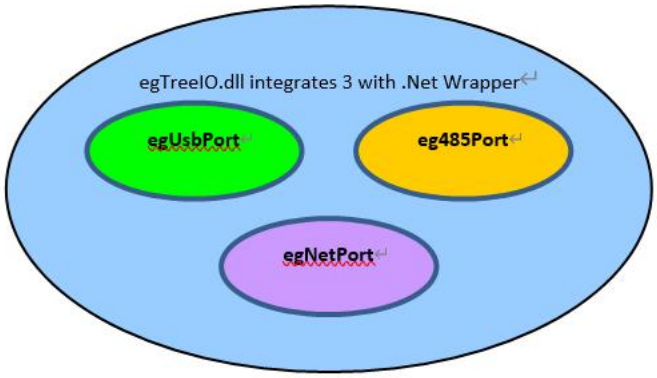

图 11.上位机 PC API 软件组件

有关 egPLC5888 的 Windows PC API 的更多信息，请参阅《egTreeIO API》