

## S7-1200 PLC 的硬件结构

## 4.1 S7-1200 PLC 的硬件结构

西门子可编程控制器家族是一个完整的产品组合，包括 LOGO、S7-200CN、S7-1200、S7-300、S7-400 等。S7-1200 PLC 充分满足了中小型自动化的系统需求，具有集成 PROFINET 接口、强大的集成工艺功能和灵活的可扩展性等特点，为各种工艺任务提供了简单的通信，尤其满足多种应用中完全不同的自动化需求。

所有的 SIMATIC S7-1200 硬件都具有内置安装夹，能够方便地安装在一个标准的 35mm DIN 导轨上。这些内置的安装夹可以咬合到某个伸出位置，以便在需要进行背板悬挂安装时提供安装孔。SIMATIC S7-1200 硬件都可进行垂直安装或水平安装，且配备了可拆卸的端子板，不用重新接线，就能迅速地更换硬件。这些特性为用户安装 PLC 提供了很好的灵活性，同时也使得 SIMATIC S7-1200 成为众多应用场合的理想选择。

## 4.1.1 CPU 模块

## 1. CPU 模块的种类及特征

SIMATIC S7-1200 系统的 CPU 常见的有五种不同型号：CPU 1211C、CPU 1212C、CPU 1214C、CPU 1215C 和 CPU 1217C。各类 CPU 的技术规范如表 4.1 所示。

表 4.1 S7-1200 CPU 技术规范

特性	CPU 1211C	CPU 1212C	CPU 1214C	CPU 1215C	CPU 1217C
本机数字量 I/O 点数	6 入/4 出	8 入/6 出	14 入/10 出	14 入/10 出	14 入/10 出
本机模拟量 I/O 点数	2 入	2 入	2 入	2 入/2 出	2 入/2 出
工作存储器/装载存储器	50kB/1MB	75kB/2MB	100kB/4MB	125kB/4MB	150kB/4MB
信号模块扩展个数	无	2	8	8	8
最大本地数字量 I/O 点数	14	82	284	284	284
最大本地模拟量 I/O 点数	13	19	67	69	69
高速计数器	最多可组态 6 个使用任意内置或信号板输入的高速计数器				
脉冲输出(最多 4 点)	100kHz	100kHz/30 kHz	100kHz/30 kHz	1MHz/100kHz	
上升沿/下降沿中断点数	6/6	8/8	12/12		
脉冲捕获输入点数	6	8	14		

## 2. 扩展 CPU 的能力

S7-1200 系列提供了多种信号模块和信号板用于开关量的输入、输出和模拟量的扩展，还可以安装附加的通信模块以支持其他通信协议，如图 4.1 所示。

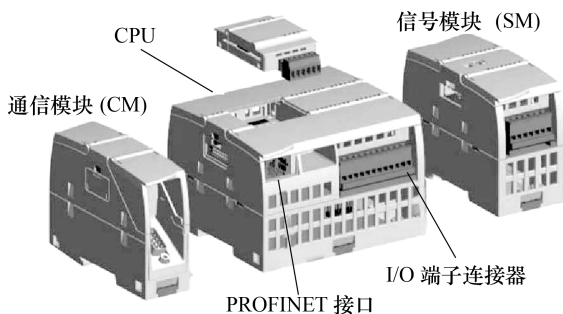


图 4.1 S7-1200 扩展模块及接口

不同型号的 CPU 面板是类似的，如图 4.2 所示为 CPU 1214C 的面板示意图。

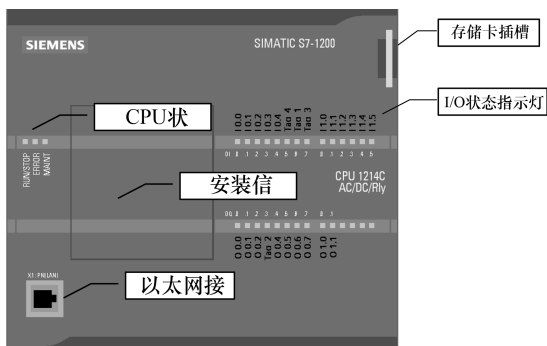


图 4.2 CPU 1214C 面板示意图

CPU 通常有以下三类指示灯，用于提供 CPU 模块的运行状态信息。

### (1) STOP/RUN 指示灯。

该指示灯的颜色为黄色时指示 STOP 模式，绿色时指示 RUN 模式，绿色和黄色交替闪烁指示 CPU 正在起动。

### (2) ERROR 指示灯。

该指示灯为红色闪烁状态时指示有错误，如 CPU 内部错误、存储卡错误或组态错误 (模块不匹配) 等，纯红色时指示硬件出现故障。

### (3) MAINT 指示灯。

该指示灯在每次插入存储卡时闪烁。

CPU 模块上的 I/O 状态指示灯用来指示各种数字量输入或输出的信号状态。

CPU 模块上提供一个以太网通信接口用于实现以太网通信，还提供了两个可以指示以太网状态的指示灯，其中“Link” (绿色) 点亮指示连接成功，“Rx/Tx” (黄色) 点亮指示传输活动。

拆下 CPU 上的挡板可以安装一个信号板(Signal Board)。通过信号板可以在不增加空间的前提下给 CPU 增加 I/O。S7-1200 PLC 任何一种 CPU 都支持扩展最多一个信号板,以扩展数字量或模拟量 I/O,而不必改变控制器的体积。目前信号板有 8 种,包括数字量输入、数字量输出、数字量输入/输出及模拟量输出等类型,S7-1200 PLC 的信号板如表 4.2 所示。

表 4.2 S7-1200 PLC 的信号板

SB 1221 DC 200 kHz	SB 1222 DC 200 kHz	SB 1223 DC/DC 200 kHz	SB 1223 DC/DC
DI 4 × 24V DC	DQ 4 × 24V DC 0.1 A	DI 2 × 24V DC/ DQ 2 × 24V DC 0.1 A	DI 2 × 24V DC/ DQ 2 × 24V DC 0.5 A
DI 4 × 5V DC	DQ 4 × 5V DC 0.1 A	DI 2 × 5V DC/ DQ 2 × 5V DC 0.1 A	AQ 1 × 12 Bit ± 10V DC/0 ~ 20mA

## 4.1.2 信号模块

信号模块包括:数字量输入模块、数字量输出模块、数字量输入/输出模块以及模拟量输入模块、模拟量输出模块、模拟量输入/输出模块等。DI、DQ、AI、AQ 模块统称为信号模块 SM,信号模块应安装在 CPU 模块的右边,最多可以扩展 8 个信号模块。输入模块用来接收和采集输入信号,输出模块用来控制输出设备和执行器。信号模块除了传递信号外,还有电平转换与隔离的作用。S7 - 1200 PLC 提供了各种信号 I/O 模块用于扩展 CPU 的能力,CPU 1211C 不支持信号模块的扩展,CPU 1212 C 支持 2 个,CPU 1214 C 支持最多 8 个,如表 4.3 所示。

表 4.3 S7-1200 PLC 信号模块

信号模块	SM 1221 DC	SM 1221 DC		
数字量输入	DI 8 × 24V DC	DI 16 × 24V DC		
信号模块	SM 1222 DC	SM 1222 DC	SM 1222 RLY	SM 1222 RLY
数字量输出	DO 8 × 24V DC 0.5 A	DO 16 × 24V DC 0.5 A	DO 8 × RLY 30V DC/ 250V AC 2 A	DO 16 × RLY 30V DC/ 250V AC 2 A
信号模块	SM1223 DC/DC	SM1223 DC/DC	SM1223 DC/RLY	SM1223 DC/RLY
数字量 输入/输出	DI 8 × 24V DC/DO 8 × 24V DC 0.5 A	DI 16 × 24V DC/DO 16 × 24V DC 0.5 A	DI 8 × 24V DC/DO 8 × RLY 30V DC/250V AC 2 A	DI 16 × 24V DC/DO 16 × RLY 30V DC/250V AC 2 A
信号模块	SM1231 AI	SM1231 AI		
模拟量输入	AI 4 × 13Bit ± 10V DC/0 ~ 20mA	AI 8 × 13Bit ± 10V DC/0 ~ 20mA		
信号模块	SM1232 AQ	SM1234 AQ		

续 表

模拟量输出	AQ 2 × 14Bit ± 10V DC/0 ~ 20mA	AQ 4 × 14Bit ± 10V DC/ 0 ~ 20mA		
信号模块	SM1234 AI/AQ			
模拟量 输入/输出	AI 4 × 13Bit ± 10V DC/0 ~ 20mA AQ 2 × 14Bit ± 10V DC/0 ~ 20mA			

各数字量信号模块还提供了指示模块状态的诊断指示灯。其中，绿色指示模块处于运行状态，红色指示模块有故障或处于非运行状态。

各模拟量信号模块为各路模拟量输入和输出提供 I/O 状态指示灯。其中，绿色指示通道已组态且处于激活状态，红色指示个别模拟量输入或输出处于错误状态。此外，各模拟量信号模块还提供有指示模块状态的诊断指示灯。其中，绿色指示模块处于运行状态，而红色指示模块有故障或处于非运行状态。

### 4.1.3 通信模块

SIMATIC S7-1200 配备了不同的通信机制：通过通信模块实现点对点连接和集成的 PROFINET 接口。所有的 SIMATIC S7-1200 CPU 都可以配备最多 3 个通信模块(Communication Module, CM)，通信模块连接在 CPU 的左侧(或连接到另一 CM 的左侧)。

#### 1. RS232 和 RS485 通信模块

S7-1200 系列提供了给系统增加附加功能的通信模块 RS232 和 RS485，为点到点的串行通信提供连接，该通信的组态和编程采用扩展指令或库功能、USS 驱动协议、Modbus RTU 主站和从站协议，均包含在 STEP 7 Basic 工程组态系统中。通信模块的特征如表 4.4 所示。

表 4.4 通信模块特征

通讯模块	CM 1241 RS232	CM 1241 RS485
串行通信	1 x 9-pin D-sub 公联接头	1 x 9-pin D-sub 母联接头
供电方式	由 CPU 供电	由 CPU 供电
状态指示	通过 LED 方式动态显示发送和接收	通过 LED 方式动态显示发送和接收

通信模块允许通过点对点连接的通信，任何具备串行接口的设备都能够被连接，如驱动器、打印机、条形码阅读器、调制解调器等。图 4.3 为通过 CM1241 在编程接口模式下的点对点连接。

#### 2. 集成 PROFINET 接口

新型的 SIMATIC S7-1200 配备了集成 PROFINET 接口，提供与下列组件的无缝通信：SIMATIC STEP 7 Basic 工程组态系统(用于编程)、SIMATIC HMI 精简系列面板(用于可视化)、其他控制器(用于 PLC 间的通信)、第三方设备(用于可选的高级集成)。通过开放式

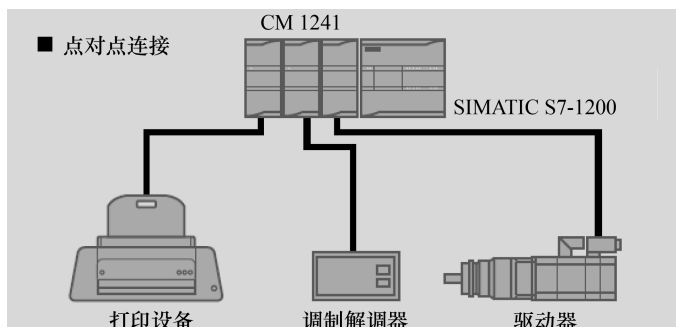


图 4.3 通过 CM1241 在编程接口模式下的点对点连接

工业以太网标准以现有的 TCP/IP 标准，SIMATIC S7-1200 提供的集成 PROFINET 接口可用于编程，实现与 HMI 通信或与其他 PLC 通信，还可通过成熟的 S7 通信协议连接到多个 S7 控制器和 HMI 设备。通过开放式以太网协议 TCP/IP 和 ISO on TCP 可与多个第三方设备进行连接和通信。PROFINET 电缆是带有 RJ45 接口的标准 CAT5 以太网电缆，用于连接 CPU 与计算机或编程设备，通信时将 PROFINET 电缆的一端插入 CPU，将电缆的另一端插入计算机或编程设备的以太网端口。

#### 4.1.4 电源计算

S7-1200 PLC 有一个内部电源，为 CPU、信号模块、信号扩展模块提供电源，并且也可以为用户提供 24V 电源。

CPU 将为信号模块、信号扩展板、通信模块提供 5V 直流电源，不同的 CPU 能够提供的功率是不同的。在硬件选型时，需要计算所有扩展模块的功率总和，检查该数值是否在 CPU 提供功率的范围之内，如果超出范围则必须更换容量更大的 CPU 或减少扩展模块数量。

S7-1200 PLC 也可以为信号模块的 24V 输入点、继电器输出模块或其他设备提供电源（称作传感器电源），如果实际负载超过了此电源的能力，则需要增加一个外部 24V 电源，此电源不可与 CPU 提供的 24V 电源并联。建议将所有 24V 电源的负端连接到一起。

传感器 24V 电源与外部 24V 电源应当供给不同的设备，二组电源的负级应互连。例如，当设计 CPU 为 24V 电源供给、信号模块继电器为 24V 电源供给、非隔离模拟量输入为 24V 电源供给的“非隔离”电路时，所有的非隔离的 M 端子必须连接到同一个外部参考点上。

下面我们通过一个例子说明电源的计算方法。

某工程项目经统计，需要 20 个直流 24V 的数字量输入；10 个数字量输出，其中 8 个为继电器输出，另 2 个必须为直流输出；模拟量方面则需要 1 路输入和 1 路输出。

由于 I/O 点数较多且输出种类必须包含两种不同类型，故选用 CPU 1214C AC/DC/Rly，订货号为 6ES7 214-1BE30-0XB0；选用输入扩展信号模块 SM 1223，订货号为 6ES7 22-1BF300XB0，包含 8 × DC 24V 输入和 8 × DC 24V 输出，一路模拟量输入 CPU 自带，一路模拟量输出可以选用信号板 SB 1232，订货号为 6ES7 2324HA30-0XB0。

电源功率的计算如表 4.5 所示。本例中，CPU 为 SM 提供了足够的 DC 5V 电流，通过传感器电源可以为所有输入和扩展的继电器线圈提供足够的 DC 24V 电流，故不再需要额

外 DC 24V 电源。

表 4.5 电源功率的计算

CPU 功率预算	5V DC	24V DC
CPU 1214C AC/DC/继电器	1600mA	400mA
减		
系统要求	5V DC	24V DC
CPU 1214C, 14 点输入	—	$14 \times 4\text{mA} = 56\text{mA}$
1 个 SM 1223, 5V 电源	145mA	—
1 个 SM 1223, 8 点输入	—	$8 \times 4\text{mA} = 32\text{mA}$
1 个 SM 1223, 8 点继电器输出	—	$8 \times 11\text{mA} = 88\text{mA}$
总要求	145mA	176mA
等于		
电流差额	5V DC	24V DC
总电流差额	1455mA	224mA

## 4.2 S7-1200 PLC 的硬件接线规范

### 1. 安装现场的接线

在安装和移动 S7-1200 模块及其相关设备时，一定要切断所有的电源。S7-1200 设计安装和现场接线的注意事项如下。

- (1) 使用正确的导线，采用  $0.50\text{mm}^2 \sim 1.50\text{mm}^2$  的导线。
- (2) 尽量使用短导线(最长 500 米屏蔽线或 300 米非屏蔽线)，导线要尽量成对使用，用一根中性或公共导线与一根热线或信号线相配对。
- (3) 将交流线和高能量快速开关的直流线与低能量的信号线隔开。
- (4) 针对闪电式浪涌，安装合适的浪涌抑制设备。
- (5) S7-1200 PLC 的供电电源可以是 AC 110V 或 AC 220V，也可以是 DC 24V，但外部电源不要与 DC 输出点并联用作输出负载，这可能导致反向电流冲击输出，除非在安装时使用二极管或其他隔离栅。

### 2. 隔离电路时的接地

使用隔离电路时的接地应遵循以下几点。

- (1) 为每一个安装电路选一个合适的参考点(0V)。
- (2) 隔离元件用于防止安装中的不期望的电流产生。应考虑到哪些地方有隔离元件，些地方没有，同时要考虑相关电源之间的隔离以及其他设备的隔离等。
- (3) 选择一个接地参考点。



(4) 在现场接地时，一定要注意接地的安全性，并且要正确地操作隔离保护设备。

### 3. 电源连接方式

S7-1200 PLC 的供电电源可以是 AC 110V 或 220V 电源，也可以是 DC 24V 电源，接线时要注意如下事项。

#### (1) 交流供电接线。

如图 4.4 所示为交流供电的 PLC 电源接线示意图，其注意事项如下。

- ① 用一个单刀切断开关将电源与 CPU、所有的输入电路和输出(负载)电路隔离开。
- ② 用一台过流保护设备保护 CPU 的电源、输出点以及输入点，也可以为每个输出点加上熔丝进行范围更广的保护。
- ③ 当使用 PLC DC 24V 传感器电源时，可以取消输入点的外部过流保护，因为该传感器电源具有短路保护功能。
- ④ 将 S7-1200 PLC 的所有地线端子同最近接地点相连接，以获得最好的抗干扰能力。建议所有的接地端子都使用 14 AWG 或  $1.5\text{mm}^2$  的电线连接到独立导电点上(也称一点接地)。
- ⑤ 本机单元的直流感应器电源可用来为本机单元的输入。
- ⑥ 扩展 DC 输入以及[g]扩展继电器线圈供电，这一传感器电源具有短路保护功能。
- ⑦ 在大部分的安装中，如果把传感器的供电 M 端子接到地上可以获得最佳的噪声抑制。

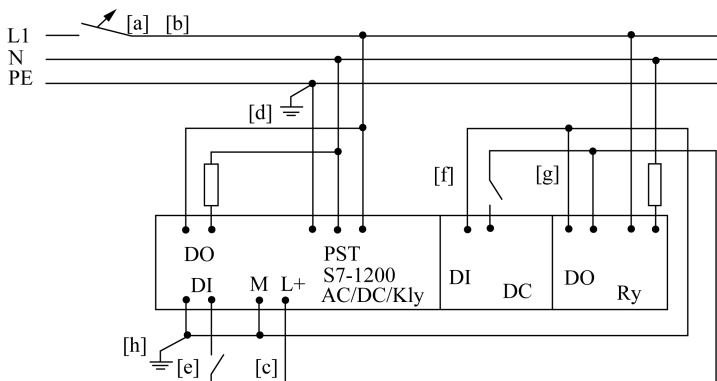


图 4.4 交流供电的 PLC 电源接线示意图

#### (2) 直流供电接线。

如图 4.5 所示为直流供电的 PLC 电源接线示意图，其注意事项如下。

- ① 用一个单刀开关[a]将电源同 CPU、所有的输入电路和输出(负载)电路隔离开。
- ② 用过流保护设备以保护 CPU 电源、[c] 输出点，以及[d]输入点。也可以在每个输出点加上熔丝进行过流保护。当时用 DC 24V 传感器电源时，可以取消输入点的外部过流保护，因为传感器电源内部具有限流功能。
- ③ 确保 DC 电源有足够的抗冲击能力，以保证在负载突变时，可以维持一个稳定的电压，这时需要一个外部电容。
- ④ 在大部分的应用中，把所有的 DC 电源接地可以得到最佳的噪声抑制。在未接地 DC 电源的公共端与保护地之间并联电阻与电容[g]。电阻提供了静电释放通路，电容提供



高频噪声通路，它们的典型值是  $1\text{M}\Omega$  和  $4700\text{pF}$ 。

⑤ 将 S7-200 PLC 所有的接地端子同最近接地点[h]连接，以获得最好的抗干扰能力。建议所有的接地端子都使用 14AWG 或  $1.5\text{mm}^2$  的电线连接到独立导电点上(也称一点接地)。DC 24V 电源回路与设备之间，以及 AC 120/230V 电源与危险环境之间，必须提供安全电气隔离。

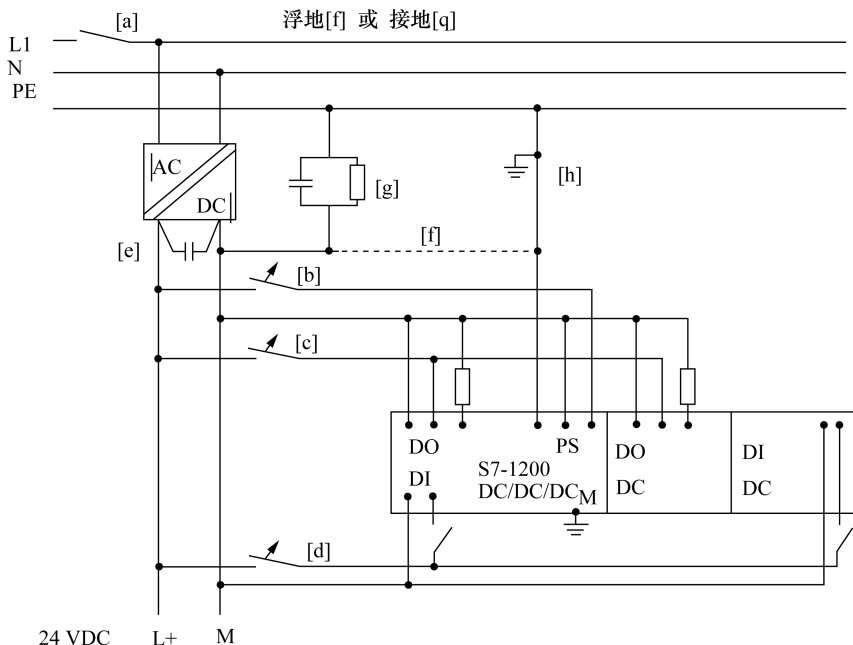


图 4.5 直流供电的 PLC 电源接线示意图

#### 4. 数字量输入接线

数字量输入类型有源型和漏型两种。S7-1200 PLC 集成的输入点和信号模块的所有输入点都既支持漏型输入又支持源型输入，而信号板的输入点只支持源型输入或者漏型输入的一种。

DI 输入为无源触点(行程开关、接点温度计、压力计)时，其接线示意图如图 4.6 所示。

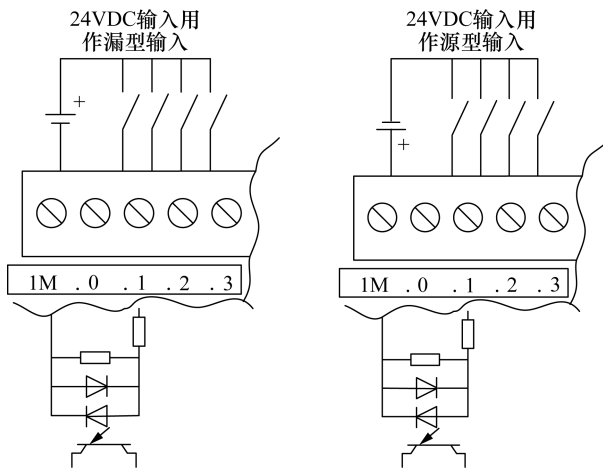


图 4.6 无源触点接线示意图



对于直流有源输入信号，一般都是 5V、12V 或 24V，而 PLC 输入模块输入点的最大电压范围是 30V，当无源开关量信号以及有源直流电压信号混合接入 PLC 输入点时，一定要注意电压的 0V 点要相互连接，如图 4.7 所示。

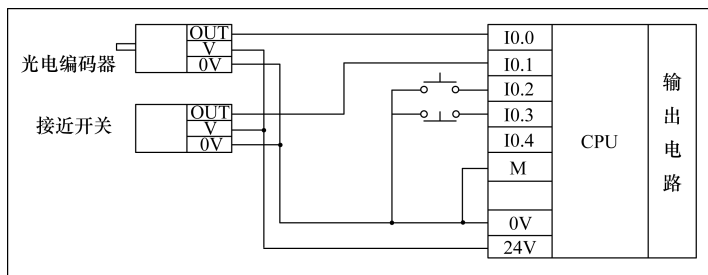


图 4.7 有源直流输入接线示意图

PLC 的直流电源的容量无法支持过多的负载，或者外部检测设备的电源不能使用 24V 电源，而必须使用 5V、12V 时，必须设计外部电源，以为这些设备提供合适的电源，如图 4.8 所示。

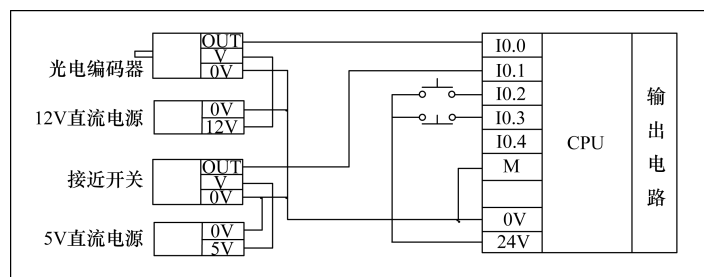


图 4.8 外部不同电源供电示意图

## 5. 数字量输出接线

晶体管输出形式的 DO 负载能力较弱(小型的指示灯、小型的继电器线圈等)，响应相对较快，其接线示意图如 4.9 所示。

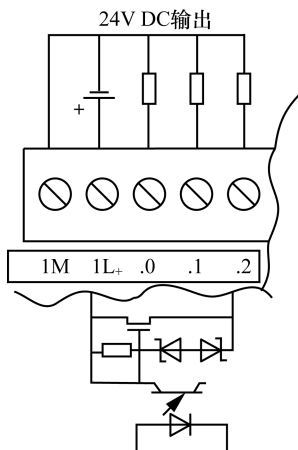


图 4.9 晶体管输出形式的 DO 接线示意图

继电器输出形式的 DO 负载能力较弱(能驱动接触器等), 响应相对较慢, 其接线示意图如图 4.10 所示。

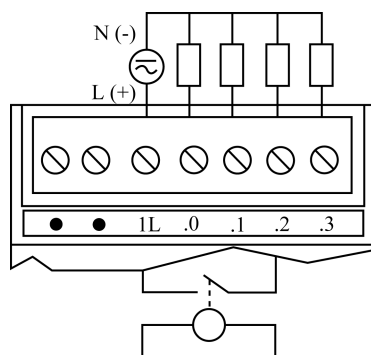


图 4.10 继电器输出形式的 DO 接线示意图

S7-1200 PLC 数字量的输出信号类型, 只有 200kHz 的信号板输出既支持漏型输出又支持源型输出, 其他信号板、信号模块和 CPU 集成的晶体管输出只支持源型输出。

关于 S7-1200 PLC 数字量的输出模块接线的更多类型可参考系统手册。

## 6. 模拟量输入/输出接线

模拟量输入/输出有以下三种接线方式。

(1) **二线制**: 两根线既传输电源又传输信号, 也就是传感器输出的负载和电源是串联在一起的, 电源是从外部引入的, 和负载串联在一起来驱动负载。

(2) **三线制**: 电源正端和信号输出的正端分离, 但它们共用一个 COM 端。

(3) **四线制**: 两根电源线, 两根信号线。电源和信号是分开工作的。

如图 4.11 为各种方式下的传感器接线示意图。

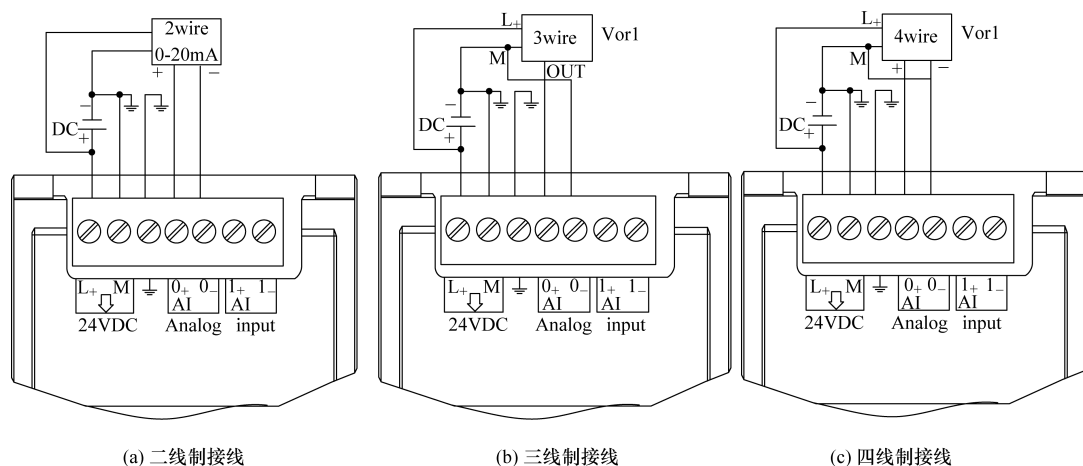


图 4.11 传感器接线示意图



## 习 题

1. S7-1200 的硬件主要由哪几部分组成?
2. 信号模块是哪些模块的总称?
3. 一个控制系统如果需要 14 点数字量输入, 25 点数字量输出, 8 点模拟量输入和 4 点模拟量输出, 那么可以选择 S7-1200 哪种主机型号? 如何选用扩展模块?
4. 数字量输出接线有哪几种方式, 各有什么特点?
5. 模拟量输入/输出接线有几种方式, 各有什么特点?

## 5.1 数制、码制和逻辑运算

### 5.1.1 数制

#### 1. 二进制数

二进制数的 1 位(bit)只能取 0 和 1 这两个不同的值,可以用来表示开关量(或称数字量)的两种不同的状态,如触点的断开和接通、线圈的通电和断电等。如果该位为 1,则表示梯形图中对应的位编程元件(例如位存储器 M 和过程映像输出 Q)的线圈“通电”,常开触点接通,常闭触点断开,以后称该编程元件为 1 状态,或称该编程元件 ON(接通)。如果该位为 0,则对应的编程元件的线圈和触点的状态与上述的相反,称该编程元件为 0 状态,或称该编程元件 OFF(断开)。在编程软件中,位编程元件的 1 状态和 0 状态用 TRUE 和 FALSE 来表示。

计算机和 PLC 内部用多位二进制数来表示数字,二进制数基数为 2,遵循逢二进一的运算规则,从右往左的第  $n$  位(最低位为第 0 位)的权值为  $2^n$ 。

用位置记数法表示为

$$N = \pm (B_{n-1} \times 2^{n-1} + \cdots + B_1 \times 2^1 + B_0 \times 2^0 + B_{-1} \times 2^{-1} + \cdots + B_{-m} \times 2^{-m})$$

$$= \pm \sum_{i=m}^{n-1} (B_i \times 2^i)$$

二进制常数以 2#开始,可用下式计算 2#1100 对应的十进制数,即

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 12$$

#### 2. 十六进制

多位二进制数的书写和阅读并不方便。为了解决这一问题,可以用十六进制数来取代二进制数,二进制整数从最低位(小数从最高位)起,每四位用一位十六进制数表示,十六进制数的 16 个数字是 0~9 和 A~F(对应于十进制数 10~15)。B#16#、W#16#和 DW#16#分别用来表示十六进制字节、字和双字常数,例如 W#16#13AF。在数字后面加“H”也可以表示十六进制数,例如:

$$B3H = 1011\ 0011B$$

1FB.02H = 0001 1111 1011.0000 0010B

F3ADH = 1111 0011 1010 1101B

### 3. 数制转换

#### (1) 十进制→二进制

整数部分：除基取余法——整数连续用基数 2 除，取各次余数，直到商为零。

小数部分：乘基取整法——小数连续乘基数 2，每次取整数部分，直到小数为零；若小数乘 2 无法使尾数为零，则可根据精度要求求出足够位数。

例：分别将 237 和 0.6875 化为二进制数，其过程如下。

2	237	余数	低位		0.6875	整数	高位
2	118	...	1=k <sub>0</sub>	↑	×	2	
2	59	...	0=k <sub>1</sub>			1.3750	...
2	29	...	1=k <sub>2</sub>		×	2	
2	14	...	1=k <sub>3</sub>			0.7500	...
2	7	...	0=k <sub>4</sub>		×	2	
2	3	...	1=k <sub>5</sub>			1.5000	...
2	1	...	1=k <sub>6</sub>		×	2	
	0	...	1=k <sub>7</sub> 高位			1.0000	...
							1=k <sub>4</sub> 低位

其结果是  $(237)_{10} = (1110, 1101)_2$ ， $(0.6875)_{10} = (0.1011)_2$

#### (2) 十进制→十六进制。

整数部分：除基取余法——整数连续用基数 16 除，取各次余数，直到商为零。

小数部分：乘基取整法——小数连续乘基数 16，每次取整数部分，直到小数为零；若小数乘 16 无法使尾数为零，则可根据精度要求求出足够位数。

例：分别将 237 和 0.5429 化为十六进制数。

16	237	余数	低位		0.5429	整数	高位
16	14	...	13(D)=k <sub>0</sub>	↑	×	16	
	0	...	14(E)=k <sub>1</sub> 高位			8.6864	...
					×	16	
						10.9824	...
					×	16	
						15.7184	...
							15(F)=k <sub>3</sub> 低位

其结果是  $(237)_{10} = (ED)_{16}$ ， $(0.5429)_{10} = (0.8AF)_{16}$ ，此时精度到小数点后三位。

#### (3) 十六进制→十进制。

整数部分：按权展开法。

$$\begin{aligned}
 \text{例：} 3D7BH &= 3 \times 16^3 + 13 \times 16^2 + 7 \times 16^1 + 11 \times 16^0 \\
 &= 3 \times 4096 + 13 \times 256 + 7 \times 16 + 11 \times 1 \\
 &= 12288 + 3328 + 112 + 11
 \end{aligned}$$

=15739

小数部分：一般不用，使用时可化成二进制小数再做换算。

二进制→十进制：一般二进制先换成十六进制，再转换成十进制。

十进制→十六进制：一般十进制先转换成二进制，再转换成十六进制。

表 5.1 给出了不同进制数的表示方法，BCD 码将在 5.1.2 节介绍。

表 5.1 给出了不同进制数的表示方法

十进制数	十六进制数	二进制数	BCD 码	十进制数	十六进制数	二进制数	BCD 码
0	0	00000	0000 0000	9	9	01001	0000 1001
1	1	00001	0000 0001	10	A	01010	0001 0000
2	2	00010	0000 0010	11	B	01011	0001 0001
3	3	00011	0000 0011	12	C	01100	0001 0010
4	4	00100	0000 0100	13	D	01101	0001 0011
5	5	00101	0000 0101	14	E	01110	0001 0100
6	6	00110	0000 0110	15	F	01111	0001 0101
7	7	00111	0000 0111	16	10	10000	0001 0110
8	8	01000	0000 1000	17	11	10001	0001 0111

## 5.1.2 码制

带符号数的表示方法：二进制数最高位表示数符，其余位表示数值。

最高位 0：+ 1：-

例：00000100 表示 +4 10000011 表示 -3

### 1. 原码

尾数部分直接表示数值本身绝对值：此称原码表示法。

表达式为

$$[x]_{\text{原}} = \begin{cases} x & (x \geq 0) \\ 2^{n-1} - x & (x \leq 0) \end{cases}$$

例：n=8，x=+4，则 $[x]_{\text{原}}=00000100$

n=8，x=-3，则 $[x]_{\text{原}}=2^7 - (-00000011) = 10000011$

存在两种表示方式： $[x]_{\text{原}}=00000000$  或 x=+0

$[x]_{\text{原}}=10000000$  或 x=-0

### 2. 补码

二数相减，用电路实现减法，结构极为复杂。实际实现时，用加法器就可完成减法。加法器容易制作，故运算更加方便。用 8 位二进制数表示一个带符号数，最高位是符号位，剩下 7 位表示数值。

当 $x \geq 0$ 时， $[x]_{\text{补}}$ 是 x 的本身值，最高位为 0；

当  $x < 0$  时,  $[x]_{\text{补}} = 2^8 + x$ 。

例: ①  $x = +0000101\text{B}$ ,  $[x]_{\text{补}} = 00000101\text{B}$ ;

②  $x = -0000101\text{B}$ ,  $[x]_{\text{补}} = 10000000\text{B} + (-0000101\text{B}) = 11111011\text{B}$ 。

对于负数  $x$ , 求  $[x]_{\text{补}}$  的简单方法: 绝对值变反加 1, 再在最高位添 1。

例: 二进制补码的简明方法。

设  $y = -0000010\text{B}(-2_{+})$   $[y]_{\text{补}} = 11111101\text{B}$ , 再加 1, 得:  $11111110\text{B}$

设  $y = -0000110\text{B}(-6_{+})$   $[y]_{\text{补}} = 11111010\text{B}$

已知  $[X]_{\text{补}}$ , 求  $X$ 。

正数:  $X$  即  $[X]_{\text{补}}$ , 只是最高位 0 变成 +;

负数: 数值变反加 1, 再添负号(-)。

例: 已知  $[X]_{\text{补}} = 11111010\text{B}$  数值变反:  $0000101\text{B}$

加 1:  $0000110\text{B}$

添加符号:  $-0000110\text{B}$

例: 利用补码计算  $14 - 32 = 14 + (-32) = -18$

$x = 14 = +0001110\text{B}$ ,  $[x]_{\text{补}} = 00001110\text{B}$ ;

$y = -32 = -0100000\text{B}$ ,  $[y]_{\text{补}} = 11100000\text{B}$ ;

$[x + y]_{\text{补}} = [x]_{\text{补}} + [y]_{\text{补}} = 00001110\text{B} + 11100000\text{B} = 11101110\text{B}$ 。

$x + y = -0010010\text{B} = -(16 + 2) = -18$ 。

对  $n$  位字长的二进制数, 最高位仍为符号位。-128 是特殊情况: 最高位既是数符位又是数值位。8 位二进制数可表示带符号数数值范围为:

$00000000\text{B} \sim 01111111\text{B}$        $10000000\text{B} \sim 11111111\text{B}$

计算机和 PLC 加减运算均以补码形式出现, 当两个正数相加(或一个正数减去一个负数), 和(或差)不能超过 +127, 否则结果变为负; 当两个负数相加(或一个负数减去一个正数), 和(或差)不能小于 -128, 否则结果变为正, 此时称为溢出。

### 3. BCD 码

实际应用中, 一般计算问题的原始数据大多采用十进制数, 人们为计算机设计了一种用二进制数为它编码, 该编码称 BCD 码(Binary Coded Decimal)。BCD 是二进制编码的十进制数的缩写, BCD 码用四位二进制数表示一位十进制数(如表 5.1 所示), 每一位 BCD 码允许的数值范围为  $2\#0000 \sim 2\#1001$ , 对应于十进制数 0 ~ 9。四位二进制数共有 16 种组合, 有 6 种组合( $2\#1010 \sim 2\#1111$ )没有在 BCD 码中使用。

BCD 码的最高位二进制数用来表示符号, 负数为 1, 正数为 0。BCD 码各位之间的关系是逢十进一。一般令负数和正数的最高 4 位二进制数分别为 1111 和 0000。一个字节的 BCD 码可表示数值范围为 0 ~ 99, 16 位 BCD 码的范围为 -999 ~ +999, 32 位 BCD 码的范围为 -9999999 ~ +9999999。

BCD 码运算时, 每半个字节结果不超过 9 则不用修正, 超过 9 则加 6。



例:      0001 0101    15<sub>+</sub>                      0100 1000    48<sub>+</sub>  
       + ) 0010 0011    23<sub>+</sub>                    + ) 0110 1001    69<sub>+</sub>  
       0011 1000    38<sub>+</sub>                      1011 0001 ←低半字节有进位, 修正  
   ↑高半字节>9, 修正  
       + ) 0110 0110  
       0001 0111    117<sub>+</sub> CY←1  
       注意:BCD 码处理一般均为正数。

### 5.1.3 逻辑运算

在数字量(或称开关量)控制系统中,变量仅有两种相反的工作状态,例如高电平和低电平、继电器线圈的通电和断电,可以分别用逻辑代数中的 1 和 0 来表示这些状态,在波形图中,用高电平表示 1 状态,用低电平表示 0 状态。使用数字电路或 PLC 的梯形图都可以实现数字逻辑运算,图 5.1 的上面是 PLC 的梯形图,下面是对应得数字门电路。

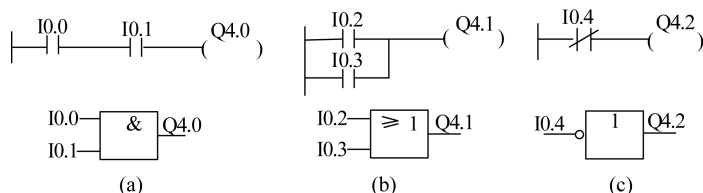


图 5.1 基本逻辑运算

图 5.1 中的 I0.0 ~ I0.4 为数字输入变量, Q4.0 ~ Q4.2 为数字量输出变量,它们之间的“与”“或”“非”逻辑运算关系如表 5.2 所示。表中的 0 和 1 分别表示输入点的常开触点断开和接通,或表示线圈断电和线圈通电。

表 5.2 逻辑运算关系表

与			或			非	
Q4. 0 = I0. 0 · I0. 1			Q4. 1 = I0. 2 + I0. 3			Q4. 2 = $\overline{I0. 4}$	
I0. 0	I0. 1	Q4. 0	I0. 2	I0. 3	Q4. 1	I0. 4	Q4. 2
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

用继电器电路或梯形图可以实现基本逻辑运算,触点的串联可以实现“与”运算,触点的并联可以实现“或”运算,例如图 5.2 中继器控制电路实现的逻辑运算可以用逻辑代数表达式表示为

$$KM = (SB1 + KM) \cdot \overline{SB2} \cdot \overline{FR}$$

式中，加号表示逻辑或；乘号(或星号)表示逻辑与；变量上面的横线表示“非”运算。

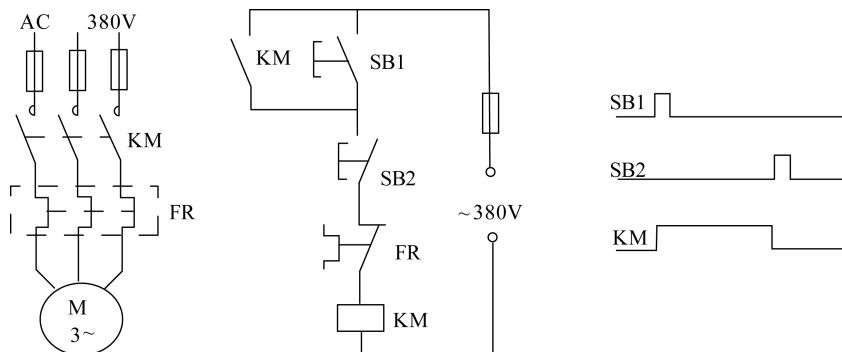


图 5.2 继电器控制电路

## 5.2 S7-1200 PLC 支持的数据类型

数据类型用来描述数据的长度(即二进制的位数)和属性,即用于指定数据元素的大小以及如何解释数据。很多指令和代码块的参数支持多种数据类型,不同的任务使用不同长度的数据对象,例如位指令使用位数据,传送指令使用字节、字和双字。数据类型是唯一声明了允许的数据范围和允许使用的指令,每个指令的参数至少支持一种数据类型,有些参数可以支持多种数据类型。

### 5.2.1 基本数据类型

表 5.3 列出了 S7-1200 PLC 支持的基本数据类型,同时还包括常量输入实例。所有数据类型都可以在 PLC 变量编辑器和块接口编辑器中使用。

表 5.3 S7-1200 PLC 支持的基本数据类型

变量类型	符号	位数	取值范围	常数举例
位	Bool	1	1、0	TRUE、FALSE 或 1、0
字节	Byte	8	16#00 ~ 16#FF	16#12, 16#AB
字	Word	16	16#0000 ~ 16#FFFF	16#ABCD, 16#0001
双字	DWord	32	16#00000000 ~ 16#FFFFFFFF	16#02468ACE
字符	Char	8	16#00 ~ 16#FF	'A', 't', '@'
有符号字节	SInt	8	-128 ~ 127	123, -123
整数	Int	16	-32768 ~ 32768	123, -123
双整数	DInt	32	-2147483648 ~ 2147483647	123, -123

续 表

变量类型	符号	位数	取值范围	常数举例
无符号字节	USInt	8	0 ~ 255	123
无符号整数	UInt	16	0 ~ 65535	123
无符号双整数	UDInt	32	0 ~ 4294967295	123
浮点数(实数)	Real	32	$\pm 1.175495 \times 10^{-38} \sim$ $\pm 3.402823 \times 10^{38}$	12.45, -3.4, -1.2E+12
双精度浮点数	LReal	64	$\pm 2.2250738585072020 \times 10^{-308} \sim$ $\pm 1.7976931348623157 \times 10^{308}$	12345.12345, -1.2E+40
时间	Time	32	T# -24d2h31m23s648ms ~ T# +24d2h31m23s647ms	T#1d_2h_15m_ 30s_45ms

表 5.3 中数据类型的符号有以下特点:

- (1) 字节、字和双字均为十六进制数, 字符又称为 ASCII 码;
- (2) 包含 Int 但无 U 的数据类型为有符号整数, 包含 Int 又有 U 的数据类型为无符号整数;
- (3) 包含 SInt 的数据类型为 8 位整数, 包含 Int 且无 D 和 S 的数据类型为 16 位整数, 包含 Dint 的数据类型为 32 位双整数。

S7-1200 PLC 的 USInt、LReal 具有以下优点:

- (1) 使用短整数数据类型, 可以节约内存资源;
- (2) 无符号数据类型可以扩大正数的数值范围;
- (3) 64 位双精度浮点数可用于高精度的数学函数运算。

## 5.2.2 复杂数据类型

复杂数据类型是由基本数据类型组成的, 不能将任何常量用作复杂数据类型的实参, 也不能将任何绝对地址作为实参传送给复杂数据类型。

### 1. DTL 数据类型

DTL 数据类型是一种 12 个字节的结构, 在预定义的结构中保存日期和时间信息, 包括年、月、日、星期、小时、分、秒和纳秒, 其长度为 12B。可以在全局数据块或块的接口区中定义 DTL 变量, 数据结构 DTL(日期时间)如表 5.4 所示。

表 5.4 数据结构 DTL

数据	字节数	取值范围	数据	字节数	取值范围
年	2	1970 ~ 2554	小时	1	0 ~ 23
月	1	1 ~ 12	分钟	1	0 ~ 59
日	1	1 ~ 31	秒	1	0 ~ 59
星期	1	1 ~ 7(日 ~ 六)	纳秒	4	0 ~ 999 999 999



## 2. String 数据类型

字符串(String)数据类型的变量将多个字符保存在一个字符串中,字符串(ASCII 字符)最多有 254 个字符(Char),最大长度为 256 个字节,其中前两个字节用来存储字符串的长度信息,称为标头。定义字符串的最大长度可以减少它占用的存储空间,例如定义了字符串“Mystring[12]”之后,字符串 Mystring 的最大长度就只有 12 个字符了。如果字符串的数据类型为 String(没有方括号),每个字符串变量将占用 256B。

执行字符串指令之前,首先应定义字符串,但不能在变量中定义字符串,只能在代码块的接口区或全局数据块中定义。将 String 输入和输出数据初始化为存储器中的有效字符串,有效字符串的最大长度必须大于 0 且小于 255。String 只能在块接口编辑器中使用,不能用于 I 或 Q 存储区。

## 3. Array 数据类型

数组(Array)是由相同数据类型的固定个数的多个元素组成。S7-1200 PLC 只能生成一维数组,数组元素的数据类型可以是所有的基本数据类型。在用户程序中,可以创建包含多个基本类型元素的数组。数组可以在组织块(OB)、功能块(FC)、功能块(FB)和数据块(DB)的块接口编辑器中创建,但不能在 PLC 变量编辑器中创建数组。

S7-1200 PLC 支持的数组格式是“ARRAY[lo..hi]”,下标[lo..hi]是在程序中引用的数组元素。lo 是数组的起始(最低)下标,hi 是数组的结束(最高)下标,元素可以是基本数据类型之一,下标可以为负数。例如,[1..10]表示有 10 个元素,第 1 个元素的地址是[1],最后 1 个元素的地址是[10]。除采用[1..10]外,也可以来用[0..9],它只影响元素的访问。在块接口编辑器中创建数组时,选择数据类型“Array [lo..hi]类型”,然后编辑“lo”“hi”和“类型”。可以在块接口编辑器的“名称”(Name)列中为数组命名,如#My\_Bits[3],引用数组“My\_Bits”的第 3 位;Array [1..10],BOOL,数组 Array[1..10] of BOOL 包含 10 个布尔值。

## 4. Struct 数据类型

由固定个数的元素组成的结构,其元素可以具有不同的数据类型。不同的结构元素可以具有不同的数据类型,不能在 Struct 变量中嵌套结构。Struct 变量始终以具有偶地址的一个字节开始,并占用直到下一个字限制的内存,可应用所有数据类型的值范围。

对于一个具体的结构体而言,其元素的数量是固定的,这一点与数组相同,但该结构体中各元素的数据类型可以不同,这是结构体与数组的重要区别。PLC 变量表只能定义基本数据类型的变量,不能定义复杂数据类型的变量。但可以在代码块的接口区或全局数据块中定义复杂数据类型的变量。

### 5.2.3 参数类型

在向 FB 和 FC 的形式参数提供数据时,数据可以是基本数据类型、复杂数据类型、系统数据类型和硬件数据类型。除此之外,还可以使用参数类型。有两个参数类型可供使用,即 Variant 和 Void。Variant 数据类型的参数是指向可变的变量或参数类型的指针,Variant 可以识别结构并指向它们,还可以指向结构变量的单个元件。在存储区中 Variant

参数类型变量不占用任何空间。Variant 参数类型的属性如表 5.5 所示。Void 参数类型不保存任何值，如果某个功能不需要任何返回值，则使用此数据类型。

表 5.5 Variant 参数类型属性

表示法	格式	长度	输入值实例
符号	操作数	0	MyTag
	数据块、操作数名称、元素		MyDB. StructTag. FirstComponent
绝对	操作数		% MW10
	数据块编号、操作数类型长度		P#DB10. DBX10. 0INT12

## 5.2.4 系统数据类型

系统数据类型 (SDT) 由固定个数的元素组成，它们具有不能更改的不同的数据结构。系统数据类型只能用于某些特定的指令，表 5.6 给出了可以使用的系统数据类型。

表 5.6 系统数据类型

系统数据类型	字节数	描述
IEC_Timer	16	用于定时器指令的定时结构
IEC_SCounter	3	用于数据类型为 SInt 的计数器指令的计数器结构
IEC_USounter	3	用于数据类型为 USInt 的计数器指令的计数器结构
IEC_UCounter	6	用于数据类型为 UInt 的计数器指令的计数器结构
IEC_Counter	6	用于数据类型为 Int 的计数器指令的计数器结构
IEC_DCounter	12	用于数据类型为 DInt 的计数器指令的计数器结构
IEC_UDCounter	12	用于数据类型为 UDInt 的计数器指令的计数器结构
ErrorStruct	28	编程 I/O 访问错误的错误信息结构，用于 GET_ERROR
CONDITIONS	52	定义启动和结束数据接收条件，用于 RCV_CFG 指令
TCON_Param	64	用于指定存放 PROFINET 开放通信连接描述的数据块的结构
Void	-	该数据类型没有数值，用于输出不需要返回值的场所，例如可以用于没有错误信息时的 STATUS 输出

## 5.2.5 硬件数据类型

硬件数据类型由 CPU 提供，可用硬件数据类型取决于 CPU 类型，取决于 CPU 类型，根据硬件配置中设置的模块，存储特定硬件数据类型的常量。在用户程序中插入控制或激活模块的指令时，将使用硬件数据类型参数作为指令的参数。

表 5.7 给出了可以使用的硬件数据类型。

表 5.7 硬件数据类型

数据类型	基本数据类型	描述
HW_ANY	Word	用于识别任意的硬件部件，例如模块
HW_IO	HW_ANY	用于识别 I/O 部件
HW_SUBMODULE	HW_IO	用于识别重要的硬件部件
HW_INTERFACE	HW_SUBMODULE	用于识别接口部件
HW_HSC	HW_SUBMODULE	用于识别高速计速器，例如用于 CTRL_HSC 指令
HW_PWM	HW_SUBMODULE	用于识别脉冲宽度调制，例如用于 CTRL_PWM 指令
HW_PTO	HW_SUBMODULE	用于在运动控制中识别脉冲传感器
AOM_IDENT	Dword	用于识别 AS 运动系统中的对象
EVENT_ANY	AOM_IDENT	用于识别任意的事件
EVENT_ATT	EVENT_ANY	用于识别可以动态地指定给一个 OB 的事件，例如用于 ATTACH 和 DETACH 指令
EVENT_HWINT	EVENT_ATT	用于识别硬件中断事件
OB_ANY	Int	用于识别任意的 OB
OB_DELAY	OB_ANY	出现时间延迟中断时，用于识别 OB 调用，例如用于 SRT_DINT 和 CAN_DINT 指令
OB_CYCLIC	OB_ANY	出现循环中断时，用于识别 OB 调用
OB_ATT	OB_ANY	用于识别可以动态地指定给事件的 OB，例如用于 SRT_DINT 和 CAN_DINT 指令
OB_PCYCLE	OB_ANY	用于识别可以指定给循环事件级别的事件 OB
OB_HWINT	OB_ANY	出现硬件中断时，用于识别调用的 OB
OB_DIAG	OB_ANY	出现诊断错误中断时，用于识别调用的 OB
OB_TIMEERROR	OB_ANY	出现时间错误时，用于识别调用的 OB
OB_STARTUP	OB_ANY	出现启动事件时，用于识别调用的 OB
PORT	UInt	用于标识通信接口，用于点对点通信
CONN_ANY	Word	用于识别任意的连接
CONN_OUC	CONN_ANY	用于识别 PROFINET 开放通信的连接

### 5.2.6 数据类型转换

如果在一个指令(分配或块参数)中链接多个操作数，这些操作数的数据类型必须是兼容的。如果操作数不是同一数据类型，则必须执行转换，可选择隐式转换或显式转换方式。

(1)隐式转换。执行指令时自动进行转换，如果操作数的数据类型是兼容的，则自动执行隐式转换。但从 REAL 到 TIME 或从 TIME 到 REAL 的转换例外，它们不能隐式转换。

(2)显式转换。如果操作数不兼容而不能进行隐式转换，则可以使用显式转换指令进行转换。可以在“指令”(Instructions)任务卡的“转换操作”中找到转换指令。显式转换的

优点是可通过输出 ENO 检测到所有超出范围的情况。如图 5.3 所示为一个执行显式数据类型转换的示例。

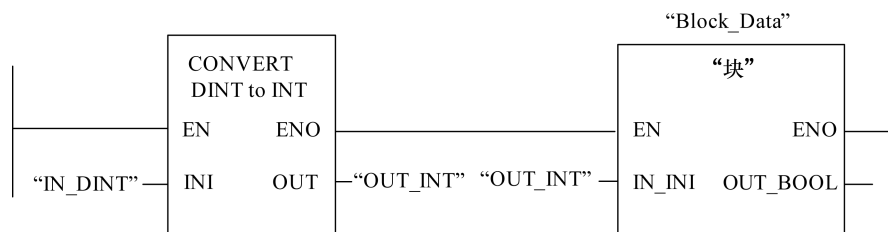


图 5.3 执行显式数据类型转换的示例

## 5.3 S7-1200 PLC 的存储区及寻址

存储器用来存储系统程序、用户程序与数据。数据存储结构是数据结构的实现形式及其在可编程序控制器内的表示。在可编程序控制器中，数据结构用于表达程序设计中的操作对象(数据元素)，以及它们之间的逻辑关系和运算关系等。

### 5.3.1 S7-1200 CPU 的存储器

根据可编程序控制器的工作原理，一般将存储器划分为系统存储器(System Memory)、用户程序存储器和数据存储器三部分。根据不同的用途又分为若干工作区域，如装载存储器(Load Memory)区、工作存储器(Work Memory)区和保持存储器(Non-Volatile memory)区等；根据数据对象的不同，还可分为物理信号输入/输出区、输入/输出映像区、位存储区、定时器区、计数器区、数据区等。S7-1200 CPU 模块中集成了装载存储器(1MB 或 2MB)、工作存储器(25kB 或 50kB)和保持存储器(2kB)。

#### 1. 装载存储器

每个 S7-1200 CPU 都具有内部装载存储器，装载存储器是非易失性的存储器，用于保存用户程序、数据和组态信息。项目被下载到 CPU 后，首先存储在装载存储区中。该存储区位于存储卡(如存在)或 CPU 中，该非易失性存储区中的内容能够在断电后继续保持。存储卡支持的存储空间比 CPU 内置的存储空间更大。

内部装载存储器可以用外部 SIMATIC 存储卡来替代。如果未插入 SIMATIC 存储卡，S7-1200 CPU 将使用内部装载存储器；如果插入了 SIMATIC 存储卡，S7-1200 CPU 将使用该存储卡作为装载存储器。但是，可使用的外部装载存储器的大小不能超过内部装载存储器的大小，即使将 24M 存储卡插于 CPU 的卡槽中，CPU 的装载存储区也达不到 24MB。

SIMATIC 存储卡用作程序卡或传送卡：①无须 STEP 7 Basic 仅使用传送卡就可以将项目复制到多个 CPU 的存储器，复制后必须取出传送卡；②程序卡可以替代 CPU 存储器(所有 CPU 功能由程序卡进行控制)，插入程序卡会擦除 CPU 内部装载存储器的所有内容，包括用户程序和任何强制 I/O。CPU 执行程序卡中的用户程序时，程序卡必须保留在 CPU 中，否则 CPU 切换到 STOP 模式。



## 2. 工作存储器

工作存储器是集成在 CPU 中的高速存取的 RAM，为了提高运行速度，则将用户程序中与程序执行有关的部分，例如组织块、功能块、功能和数据块从装载存储器复制到工作存储器。工作存储器类似于计算机的内存条，断电时工作存储器中的内容将会丢失。

## 3. 保持性存储器

用于在断电时非易失性地存储有限数量的工作存储器的值。保持性存储器发生掉电时，CPU 留出了足够的缓冲时间来保存数量有限的指定单元的值，这些值随后会在上电时自行恢复。

CPU 提供了 2048B 的保持存储器，可以在断电时，将工作存储器的数据（例如数据块或位存储器）的值永久保存在保持存储器中。断电时 CPU 有足够的时间来保存数量有限的、指定的存储单元的值。电源恢复后，系统将保持存储器保存的断电之的工作存储器的数据，恢复到原来的存储单元。需要保存的数据如果超过 2048B，将被拒绝。

## 4. 诊断缓冲区

诊断缓冲区也是系统存储器的一部分，包含由 CPU 或具有诊断功能的模块所检测到的错误。诊断缓冲区可容纳 50 个条目，其中最后（最近的）10 个条目在循环上电后将保留下来。这些条目只能通过将 CPU 恢复到工厂默认设置进行清除，可在“在线和诊断”视图中读取诊断缓冲区的内容。用鼠标右键点击项目树中的某个 PLC，执行出现的快捷菜单中的“Resources”（资源）命令，可以查看当前项目的存储器使用情况。双击项目树中某个 PLC 文件夹内的“Online and Diagnostics”，打开工作区左边窗口的“Diagnostics”（诊断）文件夹，选中“Memory”（存储器），可以查看 PLC 运行时存储器的使用情况。

## 5. 过程映像

用户程序对输入(I)和输出(Q)操作数区域寻址时，不是直接查询或更改数字量信号端口的信号状态，而是访问 CPU 系统存储器中的存储区，该存储区称为过程映像。在一个用户程序循环扫描周期中，CPU 具有一致性的过程信号映像。如果程序执行期间输入数字量信号端的信号状态转换，过程映像中的信号状态保持不变，直到下一个循环扫描周期再次更新过程映像。在用户程序中周期性地循环扫描输入信号的过程，保证了输入信息的一致性，一个用户程序循环扫描周期如图 5.4 所示。

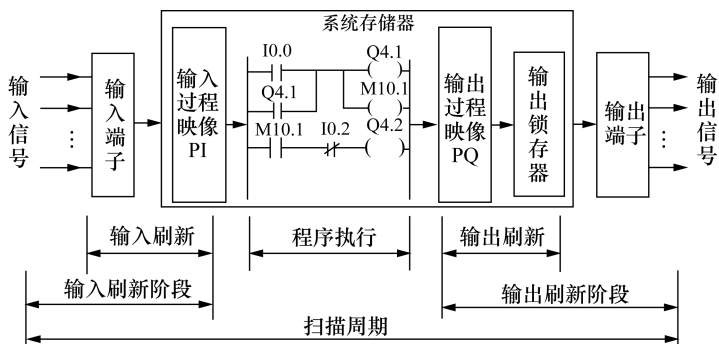


图 5.4 一个用户程序循环扫描周期

PLC 完成一个循环扫描工作的时间称为扫描周期，由上图知，CPU 的扫描周期包括自诊断、信息交换、通信等过程。包括程序扫描时间和 I/O 刷新时间。前者取决于用户程序的长短和采用的指令类型，后者随 CPU 型号不同而不同。

S7-1200 CPU 执行用户程序时，首先读入输入状态，并将它们存入输入过程映像区中；然后开始执行用户程序，执行过程按照自上而下、由左而右的顺序进行，根据第一行梯形图的逻辑运算结果，刷新该逻辑在系统存储区中对应位的状态，以及过程映像区中对应位的状态，但在程序执行阶段，输入过程映像区内的状态是不会发生变化的；程序执行结束后，CPU 就进入输出刷新阶段，按照过程映像区内对应的状态和数据刷新输出映像区，并输出状态转换，这才是 PLC 的真正输出。

S7-1200 CPU 的有些指令可直接读写 I/O。如果指令直接访问输出且输出地址位于过程映像输出中，将更新相关输出的过程映像，如果更新过程映像期间出错(I/O 访问错误)，则 CPU 将以默认的系统响应“STOP”进行响应。

### 5.3.2 用户程序存储区和寻址

#### 1. 用户程序存储区的类型

S7-1200 CPU 通过系统存储器、数据块(DB)和临时存储器处理和存储用户程序和数据。为了在执行用户程序期间存储数据，如表 5.8 所示对 CPU 存储区进行了如下划分。  
①全局存储器：CPU 提供了各种专用存储区，其中包括输入(I)、输出(Q)和位存储器(M)，所有代码块可以无限制地访问该存储器。  
②数据块(DB)：可在用户程序中加入 DB 以存储代码块所需的数据，从相关代码块开始执行一直到结束，存储的数据始终存在。“全局”DB 存储所有代码块均可使用的数据，而背景 DB 存储特定 FB 的数据并且由 FB 的参数进行构造。  
③临时存储器：只要调用代码块，CPU 的操作系统就会分配要在执行块期间使用的临时或本地存储器(L)、代码块执行完成后，CPU 将重新分配本地存储器，以用于执行其他代码块。

表 5.8 CPU 的存储区

存储区	说明	强制	保持性
I 过程映像输入	在扫描周期开始时从物理输入复制	否	否
I <sub>-</sub> : P(物理输入)	立即读取 CPU、SB 和 SM 上的物理输入点	是	否
Q 过程映像输出	在扫描周期结束时复制到物理输出	否	否
Q <sub>-</sub> : P(物理输出)	立即写入 CPU、SB 和 SM 上的物理输出点	是	否
M 位存储器	控制和数据存储器	否	是(可选)
L 临时存储器	存储块的临时数据，这些数据仅在该块的本地范围内有效	否	否
DB 数据块	数据存储器，同时也是 FB 的参数存储器	否	是(可选)



在 STEP 7 软件中采用符号编程,既可在项目变量(数据块)中创建变量,也可在组织块 OB、功能块 FC 或功能块 FB 顶部的接口中创建变量,这些变量包括名称、数据类型、偏移量(Offset)和注释。

### (1) 输入映像存储器(I)。

输入映像存储器(I)是以字节为单位的寄存器,它的每一位对应于一个物理数字量输入接点。CPU 仅在每个扫描周期的循环组织块 OB 执行之前对物理输入点进行采样,并将这些值写入输入映像寄存器。在执行用户程序过程中,不再理会物理输入点的状态,它所处理的数据为输入映像存储器(I)中的值。物理输入点只能以位为单位存取,但可以按位、字节(B)、字(W)或双字(D)访问输入映像寄存器,对过程映像输入点进行读写访问,但过程映像输入点通常为只读。通过在地址后面添加“:P”,就可以立即读取 CPU、SB 或 SM 的数字量和模拟量输入。

使用 I\_:P 访问与使用 I 访问的区别是:前者直接从被访问点而非输入映像存储器获得数据。这种 I\_:P 访问称为“立即读”访问,物理输入点是直接从与其连接的现场设备接收数值,所以不允许对这些点进行写访问。即 I\_:P 访问为只读访问,而 I 访问是可读/写的访问。I\_:P 访问也仅限于单个 CPU、SB 或 SM 所支持的输入大小(向上取整到最接近的字节),如信号板 SB1223 只有 2×DC 24V 输入/2×DC 24V 输出,如果输入被组态为从 I4.0 开始,则可按 I4.0:P 和 I4.1:P 形式,或者按 IB4:P 形式访问输入点。不会拒绝 I4.2:P 到 I4.7:P 的访问形式,但没有任何意义,因为这些输入点未使用。但不允许 IW4:P 和 ID4:P 的访问形式,因为它们超出了与 SB 1223 相关的字节偏移量。使用 I\_:P 访问不会影响存储在输入映像存储器中的相应值。

### (2) 输出映像存储器(Q)。

输出映像存储器(Q)是以字节为单位的寄存器,它的每一位对应于一个物理数字量输出接点。CPU 在执行用户程序过程中,并不把输出信号随时送到物理输出接点,而是送到输出映像寄存器中,只有到了每个扫描周期的末尾,才将输出映像寄存器的输出状态复制到各物理输出接点。物理输出点只能以位为单位存取,但可以按位、字节(B)、字(W)或双字(D)访问输出映像寄存器。输出映像寄存器允许读/写的访问。通过在地址后面添加“:P”,可以立即写入 CPU、SB 或 SM 的物理数字量和模拟量输出。使用 Q\_:P 访问与使用 Q 访问的区别是,前者除了将数据写入输出映像寄存器外,还直接将数据写入被访问点(写入 2 个位置)。这种 Q\_:P 访问称为“立即写”访问,物理输出点是直接控制与其连接的现场设备,所以不允许对这些点进行读访问。即, Q\_:P 访问为只写访问,而 Q 访问是可读/写访问。Q\_:P 访问也仅限于单个 CPU、SB 或 SM 所支持的输出大小(向上取整到最接近的字节),使用 Q\_:P 访问时既影响物理输出,也影响存储在输出过程映像区中的相应值。

### (3) 位存储器(M)。

位存储器(M)又称内部辅助继电器,类似于继电逻辑控制系统的中间继电器,用于实现中间逻辑,存储中间状态或其他控制信息,可以按位、字节(B)、字(W)或双字(D)访问位存储区,允许读/写访问。

#### (4) 临时存储器(L)。

临时存储器(L)用来存储代码块的局部变量、调用功能(FC 或 FB)时要传送的参数、程序中的中间逻辑结果等。CPU 根据需要分配临时存储器,在代码块起动(对于 OB)或被调用(对于 FC 或 FB)时为其分配临时存储器。为代码块分配临时存储器时,可能会重复使用其他 OB、FC 或 FB 先前使用的相同临时存储单元。CPU 在分配临时存储器时不会对其进行初始化,因而临时存储器可能包含任何值。

临时存储器与 M 存储器类似,主要的区别是 M 存储器在“全局”范围内有效,数据可以全局性地用于用户程序中的所有元素,任何 OB、FC 或 FB 都可以访问 M 存储器中的数据。而临时存储器在“局部”范围内有效,只有创建或声明了临时存储单元的 OB、FC 或 FB 才可以访问临时存储器中的数据。只能通过符号寻址的方式访问临时存储器,如当 OB 调用 FC 时,FC 无法访问对其进行调用的 OB 的临时存储器。

#### (5) 数据块(DB)。

数据块(DB)是用户声明的用于存取数据的存储区,可以被打开或关闭。在用户程序中创建数据块(DB)以存储代码块所需的数据,其中包括操作的中间状态或 FB 的其他控制信息参数以及许多指令(如定时器和计数器)所需的数据结构。可以按位、字节(B)、字(W)或双字(D)访问数据块存储器,可以指定数据块为读/写访问还是只读访问。

数据块有全局数据块 DB 和背景数据块 DB 两种类型。用户程序中的任何代码块 OB、FB 或 FC 都可访问全局数据块 DB 中的数据。背景数据块是由编辑器生成,而非用户编辑的。在背景数据块 DB 中仅存储特定功能块(FB)的数据,存放 FB 的部分运行变量,其中数据的结构反映了功能块 FB 的参数(Input、Output 和 InOut)和静态数据。任何代码块都可访问背景数据块 DB 中的数据,FB 的临时存储器不存储在背景数据块 DB 中。每个功能块 FB 都有一个对应的背景数据块,一个 FB 也可以使用不同的背景数据块。用户程序中的相关代码块执行完成后,数据块 DB 中存储的数据不会被删除。

### 2. I/O 寻址

对于 S7-1200 PLC 来说,如果将模块插入设备视图中,其用户数据将位于 CPU 的过程映像中(默认设置),起始地址就是模块的最低字节地址。组态模块时,调整用户程序中所使用的地址与模块起始地址。在模块属性(“I/O 地址”组)中,可更改插入模块后自动分配的起始地址,还可进行设置,以确定地址是否位于过程映像中。采用 STEP 7 软件进行组态和编程时,当添加 CPU 和 I/O 模块时,系统会自动分配默认的 I 地址和 Q 地址,也可以通过在组态画面中选择地址域,并键入新编号更改默认寻址设置。数字量输入和输出按完整的字节方式进行分配,无论模块是否使用所有的字节位,数字量输入和输出可以按字节(B)、字(W)或双字(D)存取。模拟量输入和输出按每组 2 通道(2 个字)的方式进行分配。存储区的每个存储单元都有唯一的地址,用户程序利用这些地址访问存储单元中的信息。

(1) “位”寻址方式。访问一个位也称为“字节. 位”寻址,即位存储单元的地址由字节地址和位地址组成,位寻址是最小存储单元的寻址方式。寻址时,采用存储区关键字 + 字节地址 + 位地址。如地址 I3.4: I 表示过程映像输入存储区,而 3 表示字节 3(第 4 个字节),通过后面的句点(.)与位地址 4(第 5 位)分开。再如 Q4.3: 区域标识符“Q”表示输





出, 4 表示第 5 个字节, 3 表示位地址为 3。字节地址从 0 开始, 最大值由该存储区的大小决定, 位地址的取值范围是 0 ~ 7。如果程序直接访问用户数据(而不是使用过程映像), 则为 I/O 地址附加后缀“: P”。对于物理输入或输出地址, 应在过程映像区地址后面添加“: P”, 如 I0. 3: P、Q1. 7: P 或“Motor\_Start: P”等。

(2) “字节”寻址方式。访问一个 8 位存储区域称为字节寻址。要在存储器中访问数据的字节, 必须以类似于指定位地址的方式指定该地址。该地址包括存储区标识符、数据大小标识以及起始字节地址, 如 MB20、QB5 等。MB20 中, “M”表示位存储区, B 表示字节, 20 表示字节 20, 其中, 最低位的位地址为 M20. 0, 最高位的为 M20. 7。如输入字节 IB3 由 I3. 0 ~ I3. 7 这 8 位组成。

(3) “字”寻址方式。访问一个 16 位存储区域称为字寻址。相邻的两个字节组成一个字, 一个字中两个字节的地址必须连续, 而且低位字节是高 8 位, 高位字节是低 8 位。取值范围为 W#16#0000 ~ W#16#FFFF。寻址时采用存储区关键字 + 字的关键字(W) + 第 1 字节地址结构, 如 IW14 表示由 IB14 和 IB15 组成的 1 个字, IW14 中的 I 为区域标识符, W 表示字(Word), 14 表示从第 14 个字节开始, 包括两个字节的存储空间, 即 IB14 和 IB15。IB14 是高 8 位, IB15 是低 8 位。

(4) “双字”寻址方式。访问一个 32 位存储区域称为双字寻址。相邻的 4 个字节表示一个双字, 4 个字节的地址必须连续。最低位字节在一个双字中是最高 8 位。范围 DW#16#0000\_0000 ~ DW#16#FFFF\_FFFF。寻址时采用存储区关键字 + 字的关键字(D) + 第 1 字节地址结构, 如 ID12 表示由 IB12 ~ IB15 组成的双字, 包括 LB12、LB13、LB14 和 LB15 四个字节; I 为区域标识符, D 表示存取双字(Double Word), 12 为起始字节的地址。ID12 中的 IB12 是最高 8 位, IB15 是最低 8 位。

(5) 符号寻址。符号寻址是利用符号名称和绝对地址访问块。符号寻址是先给需要使用的绝对地址或参数变量声明符号, 然后在程序中使用已声明的符号进行编程寻址。可以声明全局符号和局部符号两种: ① 在 STEP 7 程序编辑器中, 全局符号在“PLC 变量(PLC tags)”编辑器中声明, 适用于所有的程序块, 以双引号表示; ② 局部符号在“块接口(Interface)”中声明, 所定符号只在本程序块中有效, 符号前加#号表示。

(6) 变量的符号声明和寻址。可以符号声明、符号和绝对声明两种不同的方式声明块中的变量。符号声明仅包含一个符号名且块中没有固定寻址, 变量的绝对地址在编译期间动态传递并且不会在块接口中显示。变量存储在可用存储区中, 且只能以符号形式对所声明的变量进行符号寻址, 如可按如下方式访问“Data”数据块中的“Fill Level”变量: “Data”. Fill Level。对变量的纯符号声明的优点是, 数据以最适合所用 CPU 的方式结构化和排列, 这样可提高 CPU 的性能, 可以将特定的各变量定义为有保持性。对于绝对声明, 保持性设置适用于所有块变量。

在块中符号和绝对声明仅包含一个符号名和固定的绝对地址, 如该地址显示在块接口的“偏移量”(Offset)列中。可以符号或绝对方式对这些变量进行寻址, 如可按如下方式访问“Data”数据块中的“Fill Level”变量: “Data”. Fill Level 或 % DB1. DBX0. 0。

(7) 符号编程。符号编程时, 编程期间使用操作数和变量, 如输入、输出和位存储

器。CPU 根据数字绝对地址来识别这些操作数,该绝对地址由操作数 ID 和地址构成(如 Q4.0、I1.1、M2.0)。但如果要使程序布局更加清晰和简化故障排除,在编程时应使用代表操作数和变量的符号名,在 PLC 变量表中为全局变量定义符号名。在块接口中,为代码块或全局数据块中的局部变量指定符号名,如可以将符号名“Motor\_on”分配给输入信号 I1.0。编程时可先使用符号操作数,以后再分配绝对地址,即使分配没有完成,也仍可保存程序。尚未分配绝对地址的操作数会以红色波浪下划线突出显示。要分配绝对地址,选择操作数并单击快捷菜单中的“定义变量”(Define tag)。

在程序编辑器中可以采用符号表示法、绝对表示法及符号和绝对表示法显示操作数。要改变操作数的表示法,在程序编辑器工具栏中单击“启用/禁用绝对操作数”按钮。所有新建的块之后将使用该默认设置来创建,该设置不会影响背景数据块,因为其会接管来自更高级代码块的访问,该设置也不会影响现有的块或移植的块。要为程序中的所有新块设置符号访问,在“选项”菜单中,选择“设置”命令,在区域导航中,选择“PLC 编程”组,然后在“操作数访问方式”中,选中“仅符号访问”复选框。

如果设置为“仅符号访问”,则可以纯符号方式声明块变量,无须指定绝对地址。如果启用了符号访问,则可以设置功能块中各个变量的保持特性。在只能以符号方式访问的块中,可以将各个变量定义为具有保持性。在可通过其他方式访问的块中,不能对块接口中的各个变量进行保持性设置,只能将分配的背景数据块定义为具有保持性,该块包含的所有变量随后会被视为具有保持性。

### 3. 变量声明

变量声明的目的是区分不同的变量类型,通过变量声明定义要在块中使用的变量名称和数据类型。在功能块中,可以为块参数和静态局部数据分配默认值。功能块的变量声明可在背景数据块中保留存储空间。代码块的变量声明可确定程序中功能块的调用接口。功能块的变量声明可确定分配给 FB 的每个背景数据块的数据结构。

在编程系统中,除指令外,变量是最为重要的元素。在用户程序中,对每一个程序组织单元必须声明变量属性,变量属性包括符号名和地址(如 I0.1, MOTOR\_ON)、数据类型(如 BOOL)、保持性和注释等。在程序编辑器中的“PLC 变量(PLC tags)”窗口中声明变量,变量声明窗口分为“变量表”“变量属性”和“变量详细视图”三部分,在“变量属性”中还显示时间,记录变量创建和修改时间等信息,“变量详细视图”相当于一个已声明变量的目录。

全局变量在 PLC 变量表的“PLC 变量”选项卡中声明。在代码块的声明部分声明的变量和参数称为局部变量和参数,局部数据可以是基本数据类型或复合数据类型,也可以是专门用于参数传递的“参数类型”。参数是在调用块和被调用块间传递的数据,是指代码块的变量,参数类型包括定时器、计数器、块的地址等。

形参是指指令上标记该指令要使用的数据位置的标识符(例如 ADD 指令的 IN1 输入)。实参是指包含指令要使用的数据的存储单元或常量(例如% MD400,“Number\_of\_Widgets”)。用户指定的实参的数据类型必须与指令指定的形参所支持的数据类型之一匹配。指定实参时,必须指定变量(符号)或者绝对存储器地址。变量将符号名(变量名)与数据类型、存储区、存储器偏移量和注释关联在一起,并且可以在 PLC 变量编辑器或块(OB、

FC、FB 或 DB)的接口编辑器中进行创建。如果输入一个没有关联变量的绝对地址，使用的地址大小必须与所支持的数据类型相匹配，而默认变量将在输入时创建，还可为许多输入参数输入常数值。

## 5.4 PLC 的程序结构

S7-1200 PLC 程序结构可分为线性化程序结构和结构化程序结构两类，以块形式管理用户程序和数据。在 STEP 7 软件操作的控制下，通过在程序块内部或程序块之间的调用，实现程序运行与控制任务。

### 5.4.1 逻辑块的类型

STEP 7 软件中的逻辑块是包含操作系统功能和部分用户程序的程序块，组织块(OB)、功能(FC)、功能块(FB)和数据块(DB)都属于逻辑块的范畴。表 5.9 列出了各种逻辑块类型。

表 5.9 STEP 7 软件中的逻辑块类型

块类型	简要描述
组织块(OB)	CPU 操作系统与用户程序之间的接口，组织块定义用户程序的结构，基础块为组织块 OB1
功能(FC)	功能是“无存储器”的代码块，功能允许在用户程序中传递参数
功能块(FB)	功能块是一种代码块，它将值永久存储在背景数据块中，即使块执行完后，这些值仍可用
背景数据块	调用背景数据块来存储程序数据时，该背景数据块将分配给功能块
全局数据块	全局数据块是用于存储数据的数据区，任何块都可以使用这些数据

(1) 组织块 OB 定义程序的结构，充当操作系统和用户程序之间的接口。组织块 OB 是由事件驱动的，事件(如诊断中断或时间间隔)会使 CPU 执行组织块 OB。将组织块 OB 插入用户程序中后，需要为其设置参数。有些具有预定义的行为和起动事件，但也可以创建具有自定义起动事件的组织块 OB。

(2) 功能 FC 和功能块 FB 包含与特定任务或参数组合相对应的程序代码，是可从组织块 OB 或其他功能 FC 或功能块 FB 调用的程序代码块，由用户编写，并定义自己的参数。每个功能 FC 或功能块 FB 都提供一组输入和输出参数，用于与调用块共享数据。功能块 FB 还使用相关联的数据块(称为背景数据块)来保存执行期间的值状态，程序中的其他块可以使用这些值状态。由于功能 FC 没有背景数据块的逻辑块，功能 FC 的临时变量保存在本地数据堆栈中，所以不能使用静态变量。

(3) 数据块 DB 分为可由所有代码块访问的全局数据块，以及分配给特定功能块调用的背景数据块。操作系统为参数及静态变量分配的存储空间是背景数据块。功能 FC 不与任何特定数据块 DB 相关联，而功能块 FB 与数据块 DB 直接相关并使用数据块 DB 来传送



参数及存储中间值和结果。如果用户程序的其他元素需要使用 FC 的输出值，那么必须将这些值写入存储器地址或全局 DB 中。

### 5.4.2 组织块(OB)

组织块 OB 包括起动组织块、诊断错误组织块和中断组织块三大类，分别用于起动 CPU，循环或定时执行用户程序、过程出错时调用检错组织块 OB、发生硬件中断时起动中断组织块 OB。通过中断功能组织块 OB，程序段可在特定的时间或间隔内执行或从进程中响应外部信号，循环的用户程序不需要查询中断事件，如果已发生中断，操作系统可将用户程序在中断组织块 OB 中执行，因而，PLC 将按已编写的动作对中断做出反应。

组织块 OB1 通常称为主程序，用于循环执行用户程序的默认组织块，是用户程序的基本结构，也是唯一的用户程序中必须具备的代码块。执行用户程序时，操作系统从起动组织块 OB(可选)开始，然后执行程序循环组织块 OB1。组织块 OB1 也可以与中断事件(可以是标准事件或错误事件)相关联，并在相应的标准或错误事件发生时执行。用户程序、数据及组态的大小受 CPU 中可用装载存储器和工作存储器的限制，在可用工作存储器空间范围内，对所支持的块数量没有限制。

组织块由操作系统调用，控制用户程序执行、中断当前的程序执行、进行错误处理等。中断组织块由事件驱动，由事件进行触发。组织块 OB 包含一些相当于操作系统与程序的起动事件，可根据实际需要调用，但应将一些用于诊断错误的组织块装入 CPU 中。各种组织块由不同的事件起动且具有不同的优先级，每一个组织块 OB 在执行程序的过程中，可以被更高优先级的事件组织块 OB 在指令边界处中断。具有同等优先级的组织块 OB 不能相互中断，而是按照发生的先后顺序执行，组织块 OB 的起动事件如图 5.5 所示。

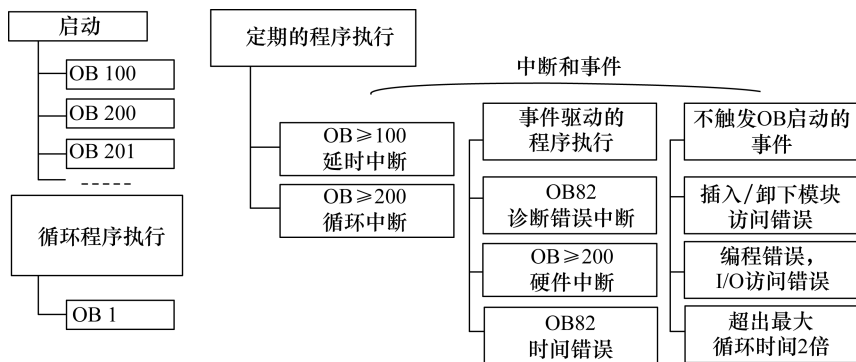


图 5.5 组织块 OB 的起动事件

由图 5.5 可见，当 CPU 上电后或模式选择器由 STOP 切换为 RUN\_P 时，编程设备中的菜单命令发出请求后，在循环程序执行之前执行起动程序组织块 OB100 或 OB200、OB201 等，然后循环程序执行组织块 OB1，根据起动事件、特定的 CPU 及其参数，将调用起动中断和事件组织块 OB。组织块 OB1 中的用户程序从第一条语句开始，执行完毕后，刷新映像区，开始一个新的循环。定期的程序执行组织块 OB 可以根据设定的延时中断、循环中断执行程序。当事件发生后，马上打断循环程序并执行中断程序。延时中断可在—



个过程事件出现后，延时一段时间响应，循环中断可每隔一段预定的时间（如 100ms）执行一次。诊断错误组织块 OB 和时间错误组织块 OB 可以在出现错误时决定系统如何响应。

### 5.4.3 功能 (FC) 和功能块 (FB)

功能 FC 是从另一个代码块 (OB、FB 或 FC) 进行调用时执行的子程序，不具有相关的背景数据块。在用户程序中，调用块将参数传递给功能 FC，其输出值必须写入存储器地址或全局数据块中。可以在程序中的不同位置多次调用同一个功能 FC，FC 适合于对频繁发生的复杂功能进行编程，从而实现程序代码的复用。在功能 FC 执行完后，临时变量中的数据将丢失，对需要永久性存储的数据可使用全局数据块进行存储。调用功能 FC 时需要对所有形参赋值，可将变量或常量指定为实参，但会有一些限制，如：不能给输出和输入/输出参数分配常数值，复杂数据类型的参数不能作为实参分配。可以将调用功能 FC 的形参分配给被调用功能 FC (功能块 FB) 的形参。如果没有给功能 FC 的输出参数赋值，则返回给被调用块的值可以是随机值。对于功能 FC 的输入/输出参数，输出参数不会是随机值，因为如果未对该参数执行写操作，将保留其中的旧输出值或输入值。

功能块 FB 是带有静态数据的代码块，它是从另一个代码块 (OB、FB 或 FC) 进行调用时执行的子程序。在用户程序中，调用块将参数传递到功能块 FB，并标识可存储特定调用数据或该功能块 FB 实例的特定数据块。因为功能块 FB 带有存储器，所以可随时在用户程序中任何地方进行参数访问。功能块 FB 的各个实例部分配有背景数据块，其中包含功能块 FB 使用的数据，可以在程序中的不同位置多次调用同一个功能块 FB。

在功能块 FB 中，访问参数时，使用背景数据块中的实参。如果调用功能块 FB 时，没有传送输入参数或没有写访问输出参数，那么将使用原先保存在背景数据块中存储的值。当数据区 (数据块) 地址或调用块的局部变量用作实参时，实参将临时保存到用于传送参数的调用块的本地数据区中。可以将调用功能块 FB 的形参分配给被调用功能 FC (功能块 FB) 的形参。不能将复杂数据类型的输入或输出参数作为实参分配给被调用功能块 FB 的输入或输出参数。可以给功能块 FB 接口中的形参赋初值，这些值将传送到相关的背景数据块。如果在调用指令中没有将实参分配给形参，将使用当前存储在背景数据块中的值。

### 5.4.4 数据块 (DB)

数据块 DB 用来存储用户程序中逻辑块的变量数据 (如数值)，数据块 DB 定义在 CPU 存储器中，所存储的数据在用户程序中进行处理，用户可在存储器中建立一个或多个数据块 DB，每个数据块 DB 可大可小，但 CPU 对数据块 DB 数量及数据总量有限制。数据块 DB 不包含任何程序段或指令，只用于存储用户数据，仅包含变量声明和用户程序将使用的变量数据。变量声明定义数据块 DB 的结构，数据以用户设定的变量形式存储，是唯一的。如果需要防止在发生电源故障时数据丢失，可以将数据值存储在保持性存储区中。

数据块 DB 有全局数据块和背景数据块两种类型。全局数据块是用户程序的一个可保存的数据区，用于存储全局数据，大小因 CPU 的不同而各异，其数据结构和大小是用户

自己定义的,所有逻辑块都可以访问全局数据块存储的信息。全局数据块仅包含静态变量,不能分配给代码块,可以从任何代码块访问全局数据块的值。可以以自己喜欢的方式定义全局数据块的结构,在声明表中声明全局数据块的变量。如果调用代码块,则代码块可以以数据块的形式打开存储区。当数据块已关闭或相应代码块的执行结束时,数据块中包含的数据不会被删除。即使在退出数据块后,这些数据仍然会保存在其中。可以同时打开一个全局数据块和一个背景数据块。

背景数据块是直接分配给功能块 FB 的块,其结构取决于功能块的接口声明,包含接口中声明的块参数和静态变量。可以在背景数据块中定义实例特定的值,如所声明变量的初始值。背景数据块不是由用户编辑的,而是由编辑器生成的。“背景”即功能块调用,在用户程序中调用一个功能块 FB 几次,就会生成几个该块的背景。可以将背景数据块分配给功能块 FB 调用或功能块的调用层级。每个单独背景都有一个背景数据块,背景数据块可分配给传送参数的每个功能块 FB 调用。创建背景数据块之前,必须存在相应的功能块 FB。创建背景数据块时指定功能块 FB 的编号。

“背景”即功能块调用,一个背景数据块可用于一个功能块 FB 的多个背景(多重背景)。通过调用功能块 FB 的多重背景,一个功能块 FB 可以控制多个设备,即给一个功能块 FB 的多个背景使用一个背景数据块。例如用于电动机类型的功能块 FB,可以在一个背景数据块中同时给多个电动机传送背景数据。为此,必须在另一个功能块 FB 中编程调用电动机控制器,并在调用功能块 FB 的声明部分给单个背景定义数据类型为功能块 FB 声明静态变量,即可以通过给不同电动机使用不同的背景数据集来控制各个电动机。每个电动机的数据(如速度、斜坡、累积操作时间等)可以保存在一个或多个背景数据块中。对于在一个背景数据块中的多重背景,必须在调用功能块 FB 的声明部分给每个单独背景以已调用功能块 FB 的数据类型声明变量。因此,功能块 FB 内的调用不要求背景数据块,只需要变量的符号名。允许在一个背景数据块(多重背景)中嵌套变量和集中背景数据。

#### 5.4.5 块的调用

用户程序的块是用户程序的组成部分,要执行用户程序中的块,必须通过其他块对它们进行调用。只有完成被调用块的执行后,才会继续执行调用块,并继续执行块调用后的操作。块调用的顺序和嵌套称为调用层级(体系)。可嵌套的块数目(嵌套深度)取决于 CPU 型号。如果嵌套太深(嵌套层次过多),则本地数据堆栈可能上溢。调用结构描述了 STEP 7 程序中的块的调用层级。当程序调用另一个块时,执行被调用的块的指令。一旦被调用的块执行结束,程序指针将从被调用的块中返回,并继续执行其程序指令。用户程序中块调用的顺序如图 5.6 所示,在一个执行周期内的块调用顺序和嵌套深度示例如图 5.7 所示。

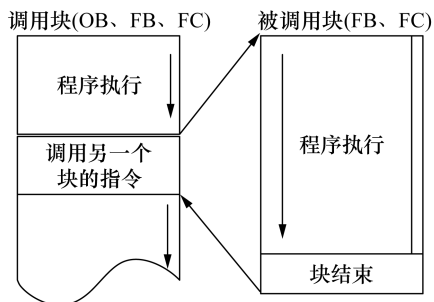


图 5.6 用户程序中块调用的顺序

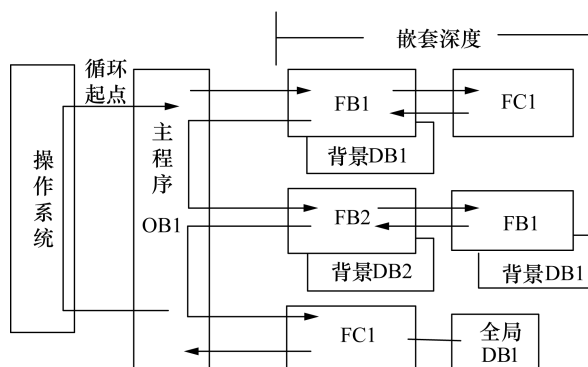


图 5.7 在一个执行周期内的块调用顺序和嵌套深度示例

(1) 插入功能 FC 的调用。使用拖放操作在程序段中插入功能 FC 的调用，将功能从项目树拖到程序段中。这样，功能 FC 及其参数一同被插入，然后可以分配这些参数。

(2) 插入单个背景方式的功能块 FB 调用。以单个背景的方式调用功能块 FB 的情况下，被调用的功能块 (FB) 将其数据存储在本身的功能块 FB 中。使用拖放操作在程序段中插入单个背景方式的功能块 FB 调用，将功能块 FB 从项目树拖到程序段中，将打开“调用选项”(Call Options) 对话框，单击“单个背景”按钮，输入要分配给该功能块的数据块的名称，单击“确定”。这样，功能块 FB 及其参数一同被插入，然后可以分配这些参数。

(3) 插入多重背景方式的功能块 FB 调用。以多重背景的方式调用功能块 FB 的情况下，被调用的功能块 FB 将其数据存储在调用功能块 FB 的背景数据块中。要使用拖放操作在程序段中插入多重背景方式的功能块 FB 调用，将功能块从项目树拖到程序段中，将打开“调用选项”对话框，单击“多重背景”按钮，在“接口中的名称”(Name In the Interface) 文本块中，输入变量的名称，通过该名称将被调用功能块作为调用块接口中的静态变量输入，单击“确定”确认输入。这样，功能块 FB 及其参数一同被插入，然后可以分配这些参数。如果调用功能块时指定了不存在的背景数据块，则将创建该背景数据块。如果以多重背景方式调用了功能块，则会将该功能块作为接口中的静态变量输入。

调用功能或功能块时，不能将复杂数据类型的输入或输出参数作为实参分配给被调用功能块的输入或输出参数。因为功能 FC 没有可以保存块参数值的数据存储器，调用功能时，必须给所有形参分配实参。功能块 FB 的参数值保存在背景数据块中，如果没有给功能块的输入、输出或输入/输出参数赋值，将使用背景数据块中存储的值。可以给功能块接口中的形参赋初值，这些值将传送到相关的背景数据块。如果在调用指令中没有将实参分配给形参，将使用当前存储在背景数据块中的值。可将变量或常量指定为实参，但不能给输出和输入/输出参数分配常数值，因为输出和输入/输出参数的用途是接收变量值(如计算结果)。

## 习 题

## 1. 填空。

(1) 二进制数 2#0100 0001 1000 0101 对应的十六进制数是 16#\_\_\_\_\_，对应的十进制数是\_\_\_\_\_，绝对值与它相同的负数的补码是 2#\_\_\_\_\_。

(2) 二进制补码 2#\_\_\_\_\_对应的十进制数为\_\_\_\_\_。

(3) 每一位 BCD 码用\_\_\_\_\_位二进制数来表示，其取值范围为二进制数 2#\_\_\_\_\_ ~ 2#\_\_\_\_\_。BCD 码 2#0000 0001 1000 0101 对应的十进制数是\_\_\_\_\_。

(4) 数字量输入模块某一外部输入电路接通时，对应的过程映像输入位为\_\_\_\_\_，梯形图中对应的常开触点\_\_\_\_\_，常闭触点\_\_\_\_\_。

(5) 背景数据块中的数据是函数块的\_\_\_\_\_中的参数和数据(不包括临时数据和常数)。

(6) 在梯形图中调用函数和函数块时，方框内是块的\_\_\_\_\_，方框外是对应的\_\_\_\_\_。方框的左边是块的\_\_\_\_\_参数和\_\_\_\_\_参数，右边是块的\_\_\_\_\_参数。

(7) S7-1200 在起动时调用\_\_\_\_\_。

2. S7-1200 的代码块包括哪些块？代码块有什么特点？

3. RAM 与 EEPROM 各有什么特点？

4. 装载存储器和工作存储器各有什么作用？

5. 字符串的第一个字节和第二个字节存放的是什么？

6. 数组元素的下标的下限值和上限值分别为 1 和 10，数组元素的数据类型为 Word，写出数组的数据类型表达式。

7. 在变量表中生成一个名为“双字”的变量，数据类型为 DWord，写出它的第 23 位和第 3 号字节的符号名。

8. I0.3: P 和 I0.3 有什么区别，为什么不能写外设输入点？

9. 函数和函数块有什么区别？

10. 组织块与 FB 和 FC 有什么区别？

11. 怎样实现多重背景？