第 3 章　S7-1200 PLC 编程基础知识

3.1　PLC 的工作原理

3.1.1　过程映像区的概念

当用户程序访问 PLC 的输入（I）信号和输出（Q）信号时，通常不是直接读取输入/输出模块信号的，而是通过位于 PLC 中的一个存储区域对输入/输出模块进行访问的，这个存储区域就是过程映像区。过程映像区分为过程映像输入区和过程映像输出区。

采用过程映像区处理输入/输出信号的好处：在一个 PLC 扫描周期中，过程映像区可以向用户程序提供一个始终一致的过程信号。在一个扫描周期中，如果输入模块的信号状态发生变化，那么过程映像区中的信号状态在当前扫描周期将保持不变，直到下一个 PLC 扫描周期过程映像区才更新，这样就保证了 PLC 在执行用户程序的过程中，过程映像区数据的一致性。

S7-1200 PLC 的数字量模块和模拟量模块的过程映像区的访问方式相同，输入都是以关键字符%I 开头（%表示绝对地址寻址）的，如%I0.5、%IW20；输出都是以关键字符%Q 开头的，如%Q0.5、%QW20。

3.1.2　PLC 的工作模式

PLC 有 3 种工作模式，分别是 STOP 模式、STARTUP 模式和 RUN 模式，CPU 的状态 LED 指示 PLC 的工作状态。S7-1200 CPU 上没有用于更改工作模式的物理开关，需要使用博途软件切换 PLC 的工作模式。

1．STOP 模式

在 STOP 模式下，PLC 将检查所有组态的模块是否可用，如果结果良好，那么 PLC 随后就将输入/输出信号设置为预定义的默认状态。当 PLC 处于 STOP 模式时，PLC 不可以执行用户程序，但可以下载用户程序。

2．STARTUP 模式

STARTUP 模式是 PLC 从 STOP 模式到 RUN 模式的一个过程，在这个过程中，将清除非保持性存储器的内容、过程映像输出，执行一次启动 OB 块，更新过程映像输入等。如果启动满足条件，则 PLC 将进入 RUN 模式。

3．RUN 模式

在 RUN 模式下，PLC 将执行用户程序、更新输入/输出信号、响应中断请求、对故

障信息进行处理等。

3.1.3 程序扫描模式

PLC 在 RUN 模式下，将按照以下机制循环工作。

（1）将输入模块的信号读到过程映像输入区。

（2）执行用户程序，进行逻辑运算，并更新过程映像输出区中的输出值。

（3）将过程映像输出区中的输出值写入输出模块。

上述 3 个步骤是 S7-1200 PLC 的软件处理过程，即程序扫描周期。只要 PLC 处于运行状态，上述步骤就会周而复始地执行。

在程序扫描期间，若有中断请求发生，那么 PLC 将调用中断 OB 块。

3.2 PLC 的存储器

S7-1200 PLC 提供了以下 3 种存储器，用于存储用户程序、数据和组态数据等。

1．装载存储器

装载存储器是一个非易失性存储器，用于存储代码块、数据块、工艺对象和硬件配置等。这些对象被下载到 PLC 中后，首先存储在装载存储器中，然后被复制到工作存储器中运行。

每个 S7-1200 PLC 均有装载存储器，装载存储器的大小取决于使用的 PLC 的型号。

装载存储器可以用外部存储卡来替代，如果未插入存储卡，那么 PLC 将使用内部装载存储器；如果插入了存储卡，那么 PLC 将使用该存储卡作为装载存储器，即使使用大容量的存储卡，也无法扩展装载存储器的容量。

用户程序中的符号名和注释也可以被下载到装载存储器中，方便用户的调试和维护。

2．工作存储器

工作存储器是一个易失性存储器，用于存储与运行相关的用户程序代码和数据，在执行用户程序时，PLC 会将用户程序的一些内容从装载存储器复制到工作存储器中。如果工作存储器断电，那么数据将丢失。

3．保持性存储器

保持性存储器是一个非易失性存储器，当发生电源故障或者断电时，它可以保存有限数量的数据。这些数据必须预先定义为保持功能，如整个 DB 块、DB 块中的部分数据、位存储区、定时器和计数器等。保持性存储器不需要电池供电。

3.3 数据类型

数据类型用于指定数据元素的大小，以及如何解释数据。在定义变量时，需要设置

变量的数据类型，每个指令参数至少支持一种数据类型，有些参数支持多种数据类型。

S7-1200 CPU 分为以下几种数据类型：基本数据类型、复杂数据类型、PLC 数据类型和指针数据类型等。

3.3.1　基本数据类型

基本数据类型如表 3-3-1 所示。

表 3-3-1　基本数据类型

数据类型	长度/位	数值范围	常数示例	地址示例
Bool	1	0 或 1	1	I1.0，Q0.1，M50.7，DB1.DBX2.3，Tag_name
Byte	8	2#0 到 2#1111_1111	2#1000_1001	IB2，MB10，DB1.DBB4，Tag_name
Word	16	2#0 到 2#1111_1111_1111_1111	2#1101_0010_1001_0110	MW10，DB1.DBW2，Tag_name
USInt	8	0 到 255	78，2#01001110	MB0，DB1.DBB4，Tag_name
SInt	8	−128 到 127	+50，16#50	MB0，DB1.DBB4，Tag_name
UInt	16	0 到 65 535	65 295，0	MW2，DB1.DBW2，Tag_name
Int	16	−32 768 到 32 767	−30 000，+30 000	MW2，DB1.DBW2，Tag_name
UDInt	32	0 到 4 294 967 295	4 042 322 160	MD6，DB1.DBD8，Tag_name
DInt	32	−2 147 483 648 到 2 147 483 647	−2 131 754 992	MD6，DB1.DBD8，Tag_name
Real	32	−3.402 823e+38 到 −1.175 495e−38，0，+1.175 495e−38 到 +3.402 823e+38	123.456，−3.4，1.0e−5	MD100，DB1.DBD8，Tag_name
LReal	64	−1.7 976 931 348 623 158e+308 到−2.2 250 738 585 072 014e−308，0，+2.2 250 738 585 072 014e−308 到 +1.7 976 931 348 623 158e+308	12 345.123 456 789e+40，1.2e+40	DB_name.var_name
TIME	32	T#−24d_20h_31m_23s_648ms 到 T#24d_20h_31m_23s_647ms	T#5m_30s T#1d_2h_15m_30s_45ms TIME#10d20h30m20s630ms	—
DATE	16	D#1990-1-1 到 D#2168-12-31	D#2009-12-31 DATE#2009-12-31 2009-12-31	—
Time_of_Day	32	TOD#0:0:0.0 到 TOD#23:59:59.999	TOD#10:20:30.400 TIME_OF_DAY#10:20:30.400	—

续表

数据类型	长度/位	数值范围	常数示例	地址示例
Char	8	16#00～16#FF	'A'，'@'，'ä'，'∑'	MB0，DB1.DBB4，Tag_name
WChar	16	16#0000～16#FFFF	'A'，'@'，'ä'，'∑'，亚洲字符，西里尔字符及其他字符	MW2，DB1.DBW2，Tag_name

1．整数的存储

在计算机系统中，所有数据都是以二进制数的形式存储的，整数一律用补码来表示和存储，并且正整数的补码为原码，负整数的补码为绝对值的反码加 1。USInt、UInt、UDInt 为无符号整型数；SInt、Int、DInt 为有符号整型数，其最高位为符号位，符号位为“0”表示正整数，符号位为“1”表示负整数。

示例：计算短整型数（SInt）78 和−78 对应的二进制值存储值。

（1）正整数的存储。短整型数（SInt）78 将被转换成二进制数 0100 1110 进行存储，该二进制数即正整数 78 的补码（也是原码），其转换方式如图 3-3-1 所示。

*b*7	*b*6	*b*5	*b*4	*b*3	*b*2	*b*1	*b*0
0	1	0	0	1	1	1	0

$78 = 0\times2^7+1\times2^6+0\times2^5+0\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0$

图 3-3-1　短整型数（SInt）78 的转换方式

（2）负整数的存储。短整型数（SInt）−78 将被转换成二进制数 1011 0010 进行存储，其转换过程如图 3-3-2 所示、存储结果如图 3-3-3 所示。

|−78|=78的原码：0100 1110
反码：1011 0001
补码：1011 0010

图 3-3-2　短整型数（SInt）−78 的转换过程

*b*7	*b*6	*b*5	*b*4	*b*3	*b*2	*b*1	*b*0
1	0	1	1	0	0	1	0

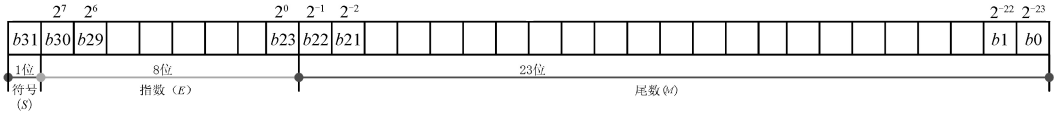
图 3-3-3　短整型数（SInt）−78 的存储结果

2．浮点数的存储

在计算机系统中，浮点数分为 Real（32 位）和 LReal（64 位）两种，不一样的存储长度，其记录的数据值的精度也不一样。浮点数的最高位为符号位，符号位为“0”表示正实数，符号位为“1”表示负实数。

示例：浮点数的存储，计算浮点数（Real）23.5 对应的二进制值存储值。

对于 Real 型浮点数，其数据存储方式和计算公式如图 3-3-4 所示。



$$V = \frac{(-1)^S \cdot (1 + M) \cdot 2^E}{2^{127}}$$

图 3-3-4　Real 型浮点数的储存方式和计算公式

浮点数（Real）23.5 转换成二进制数的计算过程如图 3-3-5 所示。

$$23.5 = \frac{(-1)^S \cdot (1 + M) \cdot 2^E}{2^{127}} \xrightarrow{\text{第一步}} S=0$$

$$\xrightarrow{\text{第二步}} M = \frac{23.5 \cdot 2^{127}}{2^E} - 1$$

$$\xrightarrow{\text{第三步}} M = \frac{23.5 \cdot 2^{127}}{2^E} - 1 \xrightarrow{\because 0 \leqslant M < 1} E = 131 \xrightarrow{\text{除2余1法}} E = 2\#1000\ 0011$$

$$\xrightarrow{\text{代入}E\text{值}} M = 0.46875$$

$$\xrightarrow{\text{第四步}} M \cdot 2^{23} = 393210 = b22 \cdot 2^{22} + b21 \cdot 2^{21} + \cdots + b1 \cdot 2^1 + b0 \cdot 2^0$$

$$\xrightarrow{\text{除2余1法}} M = 2\#011\ 1100\ 0000\ 0000\ 0000\ 0000$$

故 $V = 2\#0100\ 0001\ 1011\ 1100\ 0000\ 0000\ 0000\ 0000$

符号（S）　指数（E）　尾数（M）

图 3-3-5　浮点数（Real）23.5 转换成二进制数的计算过程

3．字符的存储

在计算机系统中，字符的存储采用的是 ASCII 编码方式。ASCII（American Standard Code for Information Interchange，美国信息互换标准代码）是基于拉丁字母的一套计算机编码系统。ASCII 主要用于显示现代英语和其他西欧语言。ASCII 是现今最通用的单字节编码系统，等同于国际标准 ISO/IEC 646，包含所有的大小写字母、数字（0～9）、标点符号等。7 位的 ASCII 表如图 3-3-6 所示。

示例：字符的存储，计算字符“A”对应的二进制值存储值。

通过 7 位的 ASCII 表可知，字符“A”对应的二进制数为 0100 0001。

L \ H	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	)	8	H	X	h	x
1001	HT	EM	(	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

图 3-3-6　7 位的 ASCII 表

3.3.2 复杂数据类型

复杂数据类型主要包括字符串、长日期时间、数组类型、结构类型。

1．字符串

如表 3-3-2 所示，S7-1200 PLC 有两种字符串数据类型：String 数据类型和 WString 数据类型。

表 3-3-2 字符串数据类型

数据类型	长度	范围	常量输入示例
String	（n+2）字节	n=（0～254 字节）	‘ABC’
WString	（n+2）个字	n=（0～65 534 个字）	‘ä123@XYZ.COM’

String 数据类型可存储一串单字节字符。String 数据类型提供了 256 个字节，第一个字节用于存储字符串中最大字符数，第二个字节用于存储当前字符数，接下来的字节最多可存储 254 个字节的字符。String 数据类型中的每个字节都可以是从 16#00 到 16#FF 的任意值。

WString 数据类型可存储单字节/双字节较长的字符串。第一个字节用于存储字符串中最大字符数，第二个字节用于存储当前字符数，接下来的字节最多可存储 65 534 个字节的字符。WString 数据类型中的每个字节都可以是 16#0000 到 16#FFFF 的任意值。

示例 1：String 数据类型和 WString 数据类型在博途软件中的定义方法示例。

字符串可以在 DB 块、OB/FC/FB 块的接口区和 PLC 数据类型中定义，String 数据类型和 WString 数据类型在 DB 块中的定义方法如图 3-3-7 所示。

数据块_1

		名称	数据类型	起始值	保持
1		Static			
2		tag_1	String	'ABC'	
3		tag_2	WString	WSTRING#'Hello'	
4		tag_3	String	''	
5		tag_4	WString	WSTRING#''	

图 3-3-7 String 数据类型和 WString 数据类型在 DB 块中的定义方法

示例 2：字符串的传送方法示例。

用 MOVE 指令和 S_MOVE 指令介绍字符串的传送方法，如图 3-3-8。

（1）MOVE 指令只能完成单字符的传送。

（2）S_MOVE 指令能完成字符串的传送。

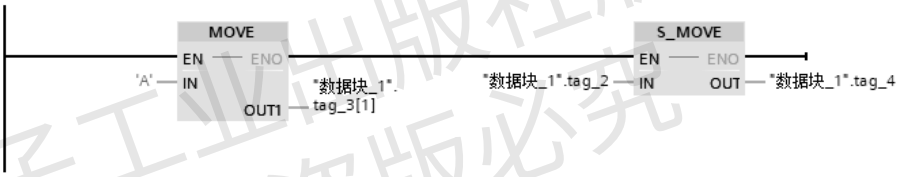


图 3-3-8 字符串的传送方法

2．长日期时间

长日期时间（DTL）数据类型是使用 12 个字节的结构保存日期和时间信息的。可以在 DB 块中定义长日期时间数据类型。长日期时间数据类型及其结构元素分别如表 3-3-3 和表 3-3-4 所示。

表 3-3-3　长日期时间数据类型

数据类型	长度/字节	范围	常量输入示例
DTL	12	最小：DTL#1970-01-01-00:00:00.0 最大：DTL#2554-12-31-23:59:59.999999999	DTL#2008-12-16-20:30:20.250

表 3-3-4　长日期时间数据类型的结构元素

字节	组件	数据类型	值范围
1	年	UInt	1970～2554
2	月	USInt	1～12
3	日	USInt	1～31
4	工作日	USInt	1（星期日）～7（星期六）
5	小时	USInt	0～23
6	分	USInt	0～59
7	秒	USInt	0～59
8	纳秒	UDInt	0～999 999 999

示例：在博途软件中定义长日期时间。

长日期时间可以在 DB 块、OB/FC/FB 块的接口区和 PLC 数据类型中定义，在 DB 块中的定义方法如图 3-3-9 所示。

图 3-3-9　长日期时间在 DB 块中的定义方法

3．数组类型

数组类型是由数目固定且数据类型相同的元素组成的数据结构，数组可以在 DB 块和 OB/FC/FB 块的接口编辑器中定义，但在 PLC 变量编辑器中无法定义数组。

在定义数组时，需要为数组命名并选择数据类型“Array [lo .. hi] of type”，根据如下说明编辑“lo”“hi”“type”。

（1）lo：数组的起始（最低）下标。

（2）hi：数组的结束（最高）下标。

（3）type：数据类型选择，如 Bool、SInt 和 UDInt 等。

示例 1：在博途软件中定义数组变量，如图 3-3-10 所示。

数据块_3

	名称	数据类型	起始值	保持
1	Static			
2	Array_1	Array[0..4] of Byte		
3	Array_1[0]	Byte	16#0	
4	Array_1[1]	Byte	16#0	
5	Array_1[2]	Byte	16#0	
6	Array_1[3]	Byte	16#0	
7	Array_1[4]	Byte	16#0	
8	Array_2	Array[0..4] of Byte		
9	Array_2[0]	Byte	16#0	
10	Array_2[1]	Byte	16#0	
11	Array_2[2]	Byte	16#0	
12	Array_2[3]	Byte	16#0	
13	Array_2[4]	Byte	16#0	

图 3-3-10　定义数组变量

示例 2：数组元素的传送。

在图 3-3-11 中，MOVE 指令将数组“数据块_3”.Array_1[0]的数据移动到数组“数据块_3”. Array_2[0]的地址中。

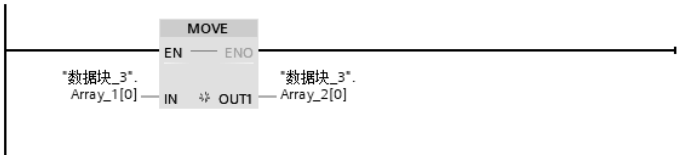


图 3-3-11　数组的寻址方法

4．结构类型

结构（Struct）类型是一种由多个不同数据类型元素组成的数据结构，其元素可以是基本数据类型，也可以是数组等复杂数据类型或者 PLC 数据类型等。结构类型嵌套结构类型的深度限制为 8 级。结构类型的变量在程序中可以作为一个变量整体，也可以作为组成该结构的元素单独使用。结构类型可以在 DB 块、OB/FC/FB 块的接口区、PLC 数据类型中定义。

示例：在 DB 块中定义一个电机变量的结构数据类型，它包含电机启动按钮、电机停止按钮、电机复位按钮、电机急停按钮、电机运行状态、电机故障状态、电机运行电流、电机运行频率和电机设定频率。结构变量定义如图 3-3-12 所示。

数据块_4

	名称	数据类型	起始值	保持
1	Static			
2	Static	Struct		
3	电机启动按钮	Bool	false	
4	电机停止按钮	Bool	false	
5	电机复位按钮	Bool	false	
6	电机急停按钮	Bool	false	
7	电机运行状态	Bool	false	
8	电机故障状态	Bool	false	
9	电机运行电流	Real	0.0	
10	电机运行频率	Real	0.0	
11	电机设定频率	Real	0.0	

图 3-3-12　结构变量定义

3.3.3　PLC 数据类型

PLC 数据类型（User Data Type，UDT）是一种由多个不同数据类型元素组成的数据结构，元素可以是基本数据类型，也可以是结构和数组等复杂数据类型及其他 PLC 数据类型等。PLC 数据类型嵌套 PLC 数据类型的深度限制为 8 级。

PLC 数据类型可以在 DB 块和 OB/FC/FB 块的接口区中定义。

PLC 数据类型可以在程序中被统一更改和重复使用，一旦某 PLC 数据类型被修改，那么在执行程序编译后，将自动更新所有使用该数据类型的变量。

示例：定义一个电机变量的 PLC 数据类型，它包含电机启动按钮、电机停止按钮、电机复位按钮、电机急停按钮、电机运行状态、电机故障状态、电机运行电流、电机运行频率和电机设定频率。

第一步：新建 PLC 数据。

在“项目树”窗格中，选择“PLC 数据类型”选项，双击“添加新数据类型”选项，弹出“用户数据类型_1”编辑框。

第二步：添加变量。

在工作区中，添加变量名和数据类型，如图 3-3-13 所示

第三步：使用 PLC 数据类型。

在 DB 块中使用新添加的 PLC 数据类型，如图 3-3-14 所示。

用户数据类型_1

	名称	数据类型	默认值
1	电机启动按钮	Bool	false
2	电机停止按钮	Bool	false
3	电机复位按钮	Bool	false
4	电机急停按钮	Bool	false
5	电机运行状态	Bool	false
6	电机故障状态	Bool	false
7	电机运行电流	Real	0.0
8	电机运行频率	Real	0.0
9	电机设定频率	Real	0.0

图 3-3-13　添加变量名和数据类型

数据块_5

	名称	数据类型	起始值	保持
1	Static			
2	1#电机控制点表	"用户数据类型_1"		
3	电机启动按钮	Bool	false	
4	电机停止按钮	Bool	false	
5	电机复位按钮	Bool	false	
6	电机急停按钮	Bool	false	
7	电机运行状态	Bool	false	
8	电机故障状态	Bool	false	
9	电机运行电流	Real	0.0	
10	电机运行频率	Real	0.0	
11	电机设定频率	Real	0.0	

图 3-3-14　PLC 数据类型的使用

3.3.4　指针数据类型

VARIANT 类型的参数是一个可以指向不同数据类型变量（而不是实例）的指针。VARIANT 指针可以是基本数据类型（如 Int、Real）的对象，也可以是 String、长日期时

间、结构类型的 Array，或者 PLC 数据类型的 Array。VARIANT 指针可以识别结构，并指向各个结构元素。VARIANT 类型的操作数不占用背景数据块或工作存储器空间，但是占用 CPU 存储空间。

VARIANT 类型的变量不是一个对象，而是对另一个对象的引用。在函数块的块接口中的 VAR_IN、VAR_IN_OUT 和 VAR_TEMP 中，VARIANT 类型的单个元素只能声明为形参。因此，不能在数据块或函数块的块接口静态部分中声明。

表 3-3-5 列出了 VARIANT 指针的属性。

表 3-3-5　VARIANT 指针的属性

长度/字节	表示方式	格式	示例输入
0	符号	操作数	MyTag
		数据块名称.操作数名称.元素	“MyDB”.Struct1.pressure
	绝对	操作数	%MW10
		数据块编号.操作数 类型长度 （仅对可标准访问的块有效）	P#DB10.DBX10.0 INT 12

3.4　地址区及寻址方法

博途 STEP 7 软件支持符号寻址和绝对地址寻址。为了更好地理解 PLC 的存储区结构及其寻址方式，本节对 PLC 变量引用的绝对寻址进行说明。

3.4.1　地址区

S7-1200 CPU 地址区包括过程映像输入（I）区、过程映像输出（Q）区、位存储（M）区和数据块（DB）区等地址区，地址区的说明如表 3-4-1 所示。

表 3-4-1　地址区的说明

地　址　区	可以访问的地址单位	符　　号	说　　明
过程映像输入（I）区	输入位	I	CPU 在循环开始时从输入模块读取输入值并将这些值保存在过程映像输入表中
	输入字节	IB	
	输入字	IW	
	输入双字	ID	
过程映像输出（Q）区	输出位	Q	CPU 在循环开始时将过程映像输出表中的值写入输出模块
	输出字节	QB	
	输出字	QW	
	输出双字	QD	
位存储（M）区	位存储区位	M	此区域用于存储程序中计算出的中间结果
	存储区字节	MB	
	存储区字	MW	
	存储区双字	MD	

续表

地　址　区	可以访问的地址单位	符　　号	说　　明
数据块（DB）区	数据位	DBX	数据块存储程序信息，可以对数据块进行定义以便所有代码块都可以对其进行访问，也可将其分配给特定的 FB 函数块
	数据字节	DBB	
	数据字	DBW	
	数据双字	DBD	
局部数据	局部数据位	L	此区域包含块处理过程中块的临时数据
	局部数据字节	LB	
	局部数据字	LW	
	局部数据双字	LD	
I/O 输入区域	I/O 输入位	<变量>:P	两区域均允许直接访问 I/O 模块
	I/O 输入字节		
	I/O 输入字		
	I/O 输入双字		
I/O 输出区域	I/O 输出位		
	I/O 输出字节		
	I/O 输出字		
	I/O 输出双字		

3.4.2　寻址方法

1．寻址规则

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。

绝对地址由以下元素组成。

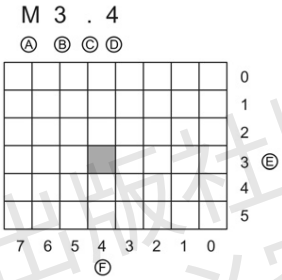
（1）地址区助记符，如 I、Q 或 M。

（2）要访问数据的单位，如 B 表示 Byte，W 表示 Word，D 表示 DWord。

（3）数据地址，如 Byte 3、Word 3。

当访问地址中的位时，不需要输入要访问数据的单位，仅输入数据的地址区助记符、字节位置和位位置（如 I0.0、Q0.1 或 M3.4）即可。

M3.4 寻址方式举例，如图 3-4-1 所示。



Ⓐ—存储器标识符；Ⓑ—字节地址；Ⓒ—分隔符；Ⓓ—位在字节中的位置；Ⓔ—存储区的字节；Ⓕ—字节中的位

图 3-4-1　M3.4 寻址方式举例

2．I 区寻址方法

I 区（过程映像输入区）：CPU 仅在每个扫描周期的循环 OB 块执行之前对外围（物理）输入点进行采样，并将这些值写入 I 区。可以按位、字节、字或双字访问 I 区。I 区通常为只读状态。I 区寻址方法如表 3-4-2 所示。

表 3-4-2　I 区寻址方法

数 据 大 小	表 示 方 法	示　例
位	I[字节地址].[位地址]	I0.1
字节、字或双字	I[大小][起始字节地址]	IB4，IW5 或 ID12

3．Q 区寻址方法

Q 区（过程映像输出区）：CPU 将存储在输出过程映像区中的值复制到物理输出区。可以按位、字节、字或双字访问 Q 区。Q 区允许读访问和写访问。Q 区寻址方法如表 3-4-3 所示。

表 3-4-3　Q 区寻址方法

数 据 大 小	表 示 方 法	示　例
位	Q[字节地址].[位地址]	Q0.1
字节、字或双字	Q[大小][起始字节地址]	QB4，QW5 或 QD12

4．M 区寻址方法

M 区（位存储区）：用于存储操作的中间状态或其他控制信息。可以按位、字节、字或双字访问 M 区。M 区允许读访问和写访问。M 区寻址方法如表 3-4-4 所示。

表 3-4-4　M 区寻址方法

数 据 大 小	表 示 方 法	示　例
位	M[字节地址].[位地址]	M0.1
字节、字或双字	M[大小][起始字节地址]	MB4，MW5 或 MD12

5．DB 区寻址方法

DB 区（数据块区）：DB 区用于存储各种类型的数据，其中包括存储操作的中间状态或 FB 块的背景信息参数等。可以按位、字节、字或双字访问 DB 区。DB 区一般允许读访问和写访问。DB 区寻址方法如表 3-4-5 所示。

表 3-4-5　DB 区寻址方法

数 据 大 小	表 示 方 法	示　例
位	DBX[字节地址].[位地址]	DB1.DBX2.3
字节、字或双字	DB[大小].[起始字节地址]	DB1.DBB4，DB10.DBW2，DB20.DBD8