

第 5 章：应用指令说明

5.1 应用指令的通则

FBs 系列 PLC 的应用指令可分为输入控制、指令号码名称、操作数及功能输出四部分。而各个指令的输入控制、操作数、及功能输出的数目各不相同（请参考各指令说明）。在 FP-07C 程序书写器上除了常用的 T、C、SET、RST 四指令及 SFC 指令有对应的专用按键，可直接按键输入外，其它的应用指令都需以指令号码输入，不能以指令名称输入。如下例：

梯 形 图		FP-07C 简码指令
例 1：单输入指令		
运算控制 —EN— 15 (+1) R 0 —CY— 进位(FO0)		FUN 15 D: R 0
例 2：多输入指令		
计数脉冲 —CK↑— 7.UDCTR CV: R 0 —CUP— 计数到(FO0) 上/下数 —U/D— PV: 10 清除控制 —CLR—		FUN 7 CV: R 0 PV: 10

注：在本手册的简码指令字段中，凡有实线方框框住的字样（如上例 **D:**、**CV:**、**Pr:**等）是 FP-07C 为方便用户输入而自动显示的操作数名称导引字，非用户所键入的。

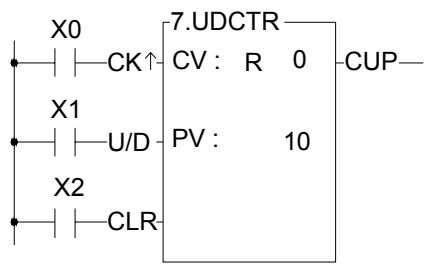
5.1.1 输入控制

FBs-PLC 除 7 个无输入控制的应用指令外，其它应用指令至少有一个输入控制，最多为四个。应用指令是按照输入控制信号的组合来决定该指令是否执行，以及执行何种运算。在 PRO-LADDER 软件包上及梯形图程序印出时，所有的应用指令符号的输入控制及功能输出端子上都有加注英文批注简写，以注明该端子是何种功能控制或输出，以利于记忆和阅读，如上图例 2 第一个输入标注“CK↑”，表示计数脉冲 Clock 由 0→1（升缘）时，该计数器才计数一次，第二个输入标注“U/D”斜线上方 U 表上数 Up，下方 D 表示下数 Down，若此输入为 1 则当计数脉冲 CK↑来时，该计数器值会加 1，反之若为 0 则减 1，第三个输入标示“CLR”，表示清除 Clear，即当此输入为 1 时，该计数器的计数值会被清为 0。其它应用指令的输入控制批注请参考各指令说明。

注：无输入控制指令是指该指令需直接接于母线，不能串接输入控制组件，也无功能输出。该指令本身单独形成一个网络。有 MCE、SKPE、LBL、RTS、RTI、FOR、NEXT 等 7 个无输入控制指令，请参考第 6 及 7 章各该指令的说明。

所有应用指令的各“输入控制”均应有组件连接，否则会出现语法错误。如下图例 3，FUN7 为三输入的应用指令，在 FUN7 指令前面的三个组件（ORG X0, LD X1, LD X2）分别对应到 FUN7 的第一个输入 CK↑，第二个输入 U/D 和第三个输入 CLR。

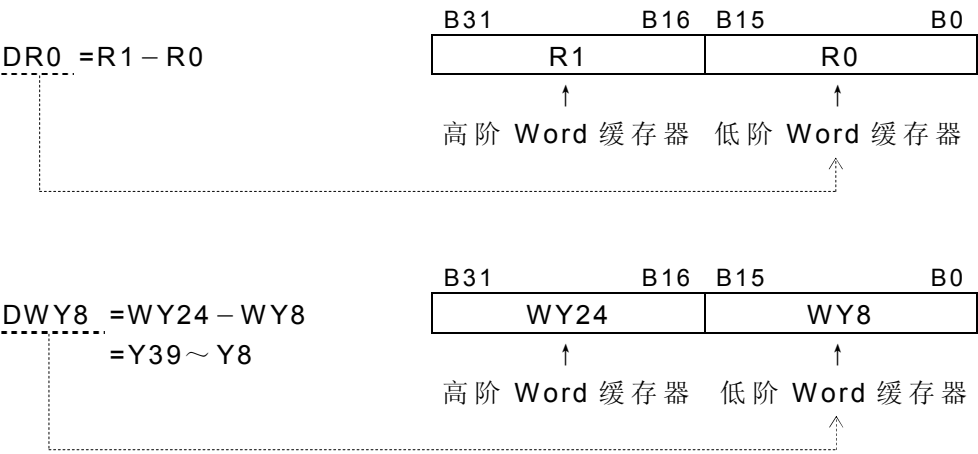
例 3：

梯 形 图	FP-07 简码指令
	ORG X 0 LD X 1 LD X 2 FUN 7 CV: R 0 PV: 10 FUN7 有 3 个输入，故其需要有 3 个组件

5.1.2 指令号码与衍生指令

FP-07C 除前述 9 个指令以专用按键输入外，其它的应用指令都需以“指令号码”来输入，在“指令号码”后，还可加上 D、P 或 DP 等后缀，而衍生出另外三种不同的指令，现叙述如下：

D：表示 Double Word，双字符组（32 位）之意。在 FBs-PLC 中的缓存器都以字符组 WORD（16 位）为基本单位，即所有 R、T、C 缓存器（C200～C255 除外）都为 16 位长度，例如 R0、R1、T0..... 等。如果需 32 位长度的缓存器，则必须由两个连续的 16 位缓存器合并起来而形成如 R1-R0、R3-R2、..... 等，针对这种连续两个 16 位缓存器组成的双字符组缓存器，以该双字符组缓存器的低阶缓存器号码（如 R1-R0 取 R0，R3-R2 取 R2）加上 D 表示（如 DR0 表示 R1-R0，DR2 表示 R3-R2），例如在监视模式(MON)下键入如下的 DR0 或 DWY8，将会显示 32 位（R1-R0，或 WY24-WY8）长度的数值。

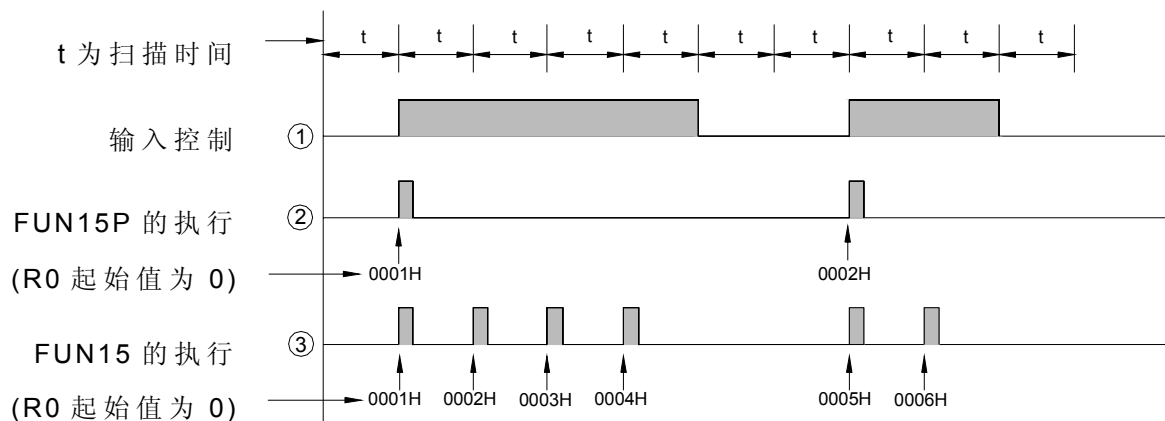


注：在梯形图或简码指令的表示，为方便区别 16 位或 32 位指令，我们在“指令号码”后面加 D 后缀以表示 32 位指令，其来源或目的操作数的长度当然也是 32 位。但在操作数（如 S，D.....）栏上只标示 Double Word 的低阶缓存器号码，如 6-6 页例 4 中“被加数”Sa: R0 因该指令 FUN11D，有 D 后缀，故所有来源或目的操作数都无冠上 D 字，也即 Sa 为 DR0=R1-R0，Sb 为 DR2=R3-R2....但非来源或目的操作数如指标 Pr，数值 N，长度 L.....等则无论 D 或非 D 指令都固定只有 16 位长度，请特别注意。在 16 位指令因其操作数长度只有一个 Word，也正是操作数栏上所标示的缓存器号码，如例 1 中的 D=R0。

P: 表示 Pulse（脉冲）模式运作，也就是每当输入控制由 0→1 瞬间（升缘↑）该指令即执行一次，如上图例 1 若指令号码加上 P 码（即 FUN 15P），则只有在输入控制信号的上升缘（0→1）时 FUN15P 才执行一次。若指令号码后无 P 后缀，则为连续执行模式，即只输入控制为 1，PLC 每次扫描到该指令都会执行一次，一直到输入控制变为 0 为止。（所有脉冲输入端都标有“↑”符号，如 CK↑、EN↑、TG↑.....）。在本手册的应用指令说明中的输入叙述有如下的叙述例：

● 当运算控制“EN”=1 或“EN↑”（**P** 指令）由 0→1 时，.....

前者即表示非 P 指令（连续模式）的执行条件，后者即为 **P** 指令（脉冲模式）的执行条件。下列波形图为上节范例 1（FUN15）工作在 P 模式和非 P 模式下其执行结果(R0) 的比较。



DP: 表示该指令为 32 位指令，且为脉冲模式运作。

注：实际控制应用上大部分的应用指令都可使用 **P** 指令，在程序设计时请尽可能使用 **P** 指令，以节省程序执行时间。

5.1.3 操作数

操作数为指令运算时的参考或写入的对象。可分为只供参考，内容不会因指令运算而改变的来源操作数（Source，简称 S）及用来储存运算结果的目的操作数（Destination 简称 D）两大类。以下就 FBs 系列 PLC 应用指令中，主要的操作数名称及性质作说明，并将可当操作数的接点、线圈或缓存器的类别范围分别叙述如下：

■ 主要操作数名称及性质：

简 写	名 称	说 明
S	来源操作数 (Source)	S 为指令运算中的数据读取、参考的对象，其内容不会因运算而改变，如果不只一个以上，则以脚注区分，如 Sa 、 Sb 。
D	目的操作数 (Destination)	用以存放指令运算结果的区域，其原始数据在运算后会破坏，只有能写入的线圈或缓存器才能当作目的操作数。
L	长 度 (Length)	用来表示一连串资料或列表(Table) 的长度（范围），可以为常数或变量。
N	数 值 (Number)	用来指定次数，个数（如第 N 个位）等的固定数字，若不只有一个，则以脚注区分如 Na 、 Nb 、 Ns 、 Nd 等。
Pr	指 标 (Point)	用来指定一串数据或列表中的某个数据或缓存器，通常 Pr 值为可变，故不能为常数或输入缓存器。
CV	当前值	用在 T 、 C 中，只能为可写入的缓存器。
PV	设定值	用在 T 、 C 中，只供参考比较用。
T	列 表 (Table)	列表是一连续缓存器组合的名称，其运作单位是以字符组或双字符组为单位，如果不只一个，则以脚注区分，如 Ta 、 Tb 、 Ts 、 Td 等。
M	矩 阵 (Matrix)	矩阵也是一连续缓存器组合的名称，但是其运作单位是以位为单位。如果不只一个则以 Ms 、 Md 、 Ma 、 Mb 等表示。

除上述主要操作数外，还有用来指定特定用途的操作数，如 **Fr** 表频率、**ST** 表堆栈、**QU** 表示 **QUEUE** 等，请参阅各指令的说明。

■ 操作数类别与范围：应用指令的操作数类别有 **a.单点(数位)** **b.缓存器** **c.常数** 三种

a.单点（数位）操作数：

在应用指令中，有单点操作数的（即其操作数只影响某一单点的）仅有 **SET**、**RST**、**DIFU**、**DIFD**、**TOGG** 五个指令，而且只能对 **Y△△△**（外部输出）、**M△△△△**（内部及特殊）、**S△△△**（步进）三类型的继电器运作。下表为可当这五个指令的单点操作数的种类及范围，详细解释请参考这五个指令的说明。

范围 操作数	Y	M	SM	S
	Y0 Y255	M0 M1911	M1912 M2001	S0 S999
D	○	○	○*	○

“○”符号表 **D** 可用该类别的线圈当操作数。在 **SM** 字段中“○”上方标有“*”符号，表示在 **SM** 中禁止写入的特殊继电器不得当作 **D** 操作数，请参考 2-3 页“特殊继电器明细”。

b.缓存器操作数：

应用指令中的操作数主要为缓存器操作数。缓存器操作数又分为两类，一为原本就以 **Word** 或 **2 Words** 为单位的缓存器(**R**、**T**、**C**)。另一则为由 **16** 或 **32** 个单点(**X**、**Y**、**M**、**S**)组成 **Word** 或 **2 Words** 的缓存器(**WX**、**WY**、**WM**、**WS**)。下表为在本手册中用来表示各指令的操作数所能允许的缓存器类别及其范围的范例：

操作数 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 WX240	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D4095	16 或 32 位正、 负数	V, Z P0~P9
	S	○	○	○	○	○	○	○	○	○	○*	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○
⋮														

“○”符号表示可以以该类别的缓存器当操作数。在 SR 和 D 交会的字段中，“○”符号上方标有“*”符号，表示 D 操作数若为特殊缓存器 SR 时，应扣除不可以写入的缓存器，请参考 2-7 页“特殊缓存器明细”。

※R5000~R8071 不是规划为只读缓存器时，可当一般缓存器使用（可擦写）

注 1:	凡有 W 开头的缓存器（WX、WY、WM、WS）表示此缓存器是由 16 个单点组成 Word 的缓存器。例如 WX0 表示由 X0（位 0）~X15（位 15）组成的缓存器，WY144 表示由 Y144（位 0）~Y159（位 15）所组成的缓存器。但注意单点的号码必须为 8 的倍数（如 0、8、16、24..... 等）才允许。
注 2:	表中最后一个缓存器（Word），不能当 32 位操作数，因 32 位操作数需有连续两个 Word 的长度才行。
注 3:	TMR（T0~T255）和 CTR（C0~C255）为定时器和计数器专用的缓存器，虽然也可以当一般缓存器使用，但会造成系统复杂，查错困难，因此除 T 或 C 指令外，其它指令应避免写入 TMR 或 CTR。
注 4:	T0~T255 和 C0~C199 都为 16 位长度，而 C200~C255 限定为 32 位长度，故不能当 16 位操作数。
注 5:	缓存器操作数除如上述直接以缓存器号码（地址）来指定外，对于 R0~R8071 范围内的缓存器操作数还可以结合指针缓存器 V、Z 或 P0~P9 来作间接寻址指定。利用指针缓存器（XR）作间接寻址的说明请参考下节（5.2 节）的范例。

c.常数操作数：

在 16 位中的常数范围最大为-32768~32767，32 位的范围为-2147483648~2147483647，而某些指令只能为正常数，因此我们以下列叙述表示 16 或 32 位的常数范围。

常 数 类 别	范 围
16 位正负数	-32768~32767
16 位正数	0~32767
32 位正负数	-2147483648~2147483647
32 位正数	0~2147483647
16 / 32 位正负数	-32768~32767 或 -2147483648~2147483647
16 / 32 位正数	0~32767 或 0~2147483647

此外有某些特定的操作数长度大小不一（如长度 L、位数..... N 等），将在各该操作数的字段上直接标示范围，请参考个别的指令说明。

5.1.4 功能输出(FO)

简称 FO(Function Output)为应用指令运算结果或状态的输出，如同“输入控制”一样。在 WinProladder 及程序印出的梯形图应用指令上，其功能输出上也有英文批注说明该 FO 为何种功能，如上图例 1 的 CY，例 2、例 3 的 CUP 及下图例 4 的 D=0、CY、BR 都是。功能输出 FO 最多只有 4 个（即 FO0~FO3），其编号顺序是由上而下，第一个 FO 为 FO0，第二个为 FO1，最后一个为 FO3。FO 状态的取出必须用 FO 指令（在 FP-07C 程序书写器上有 FO 专用按键），不使用的功能输出可空着不接任何组件，如下图例 4 的 FO1(CY)即是。

例 4：

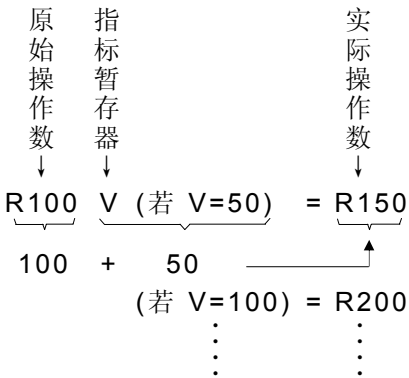
梯 形 图	简码指令
	<pre>ORG X 0 FUN 11D Sa: R 0 Sb: R 2 D: R 4 FO 0 OUT Y 0 FO 2 OUT Y 1</pre>

当 M1919=0 时，FO 状态只有在该指令被执行时才会更新，然后就一直保持至该指令下一次被执行时（记忆保持），始由新产出的 FO 状态所更新。
当 M1919=1 时，应用指令不执行时，FO 状态清除为 0（无记忆保持）。

5.2 利用指针缓存器（XR）作间接寻址

在 FBs-PLC 应用指令中，有些操作数可以结合指针缓存器（V 或 Z）而作间接寻址的指定（在每个指令说明栏的操作数叙述中会注明）但能够结合指针缓存器作间接地址指定的操作数只限定为 R0~R8071 范围内的缓存器（其它操作数如单点、常数、D0~D3071 等都不能作间接地址指定）。

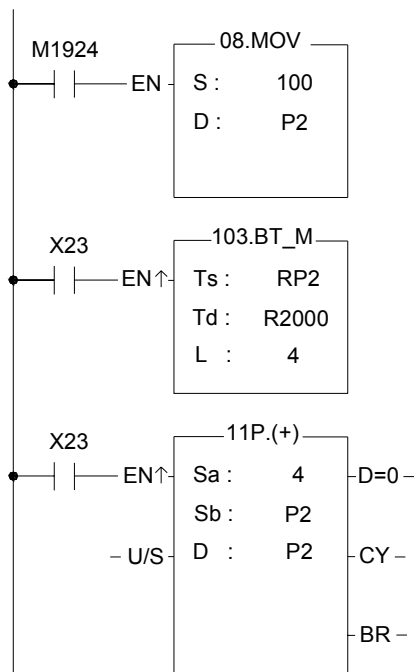
指针缓存器 XR 共有 12 个（V、Z、P0~P9），实际上在 FBs-PLC 的 V 缓存器就是特殊缓存器（R3840~R4167）中的 R4164，而 Z 缓存器则为 R4165。操作数结合指针缓存器作间接寻址的表示方式是原操作数后紧接 V 或 Z 来表示：



如上图示，只要变更 V 的值即可变更操作数的地址，利用此功能结合 FBS-PLC 的应用指令，可以用极简易的指令，实现功能强大、极具效率的控制应用，如下图程序例，只须以一个区块搬移指令（BT_M）即达成诸如停车管理系统的动态区块数据显示。

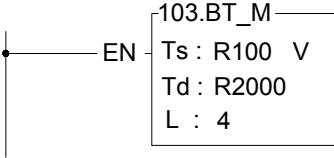
指针缓存器 P0~P9 应用说明：

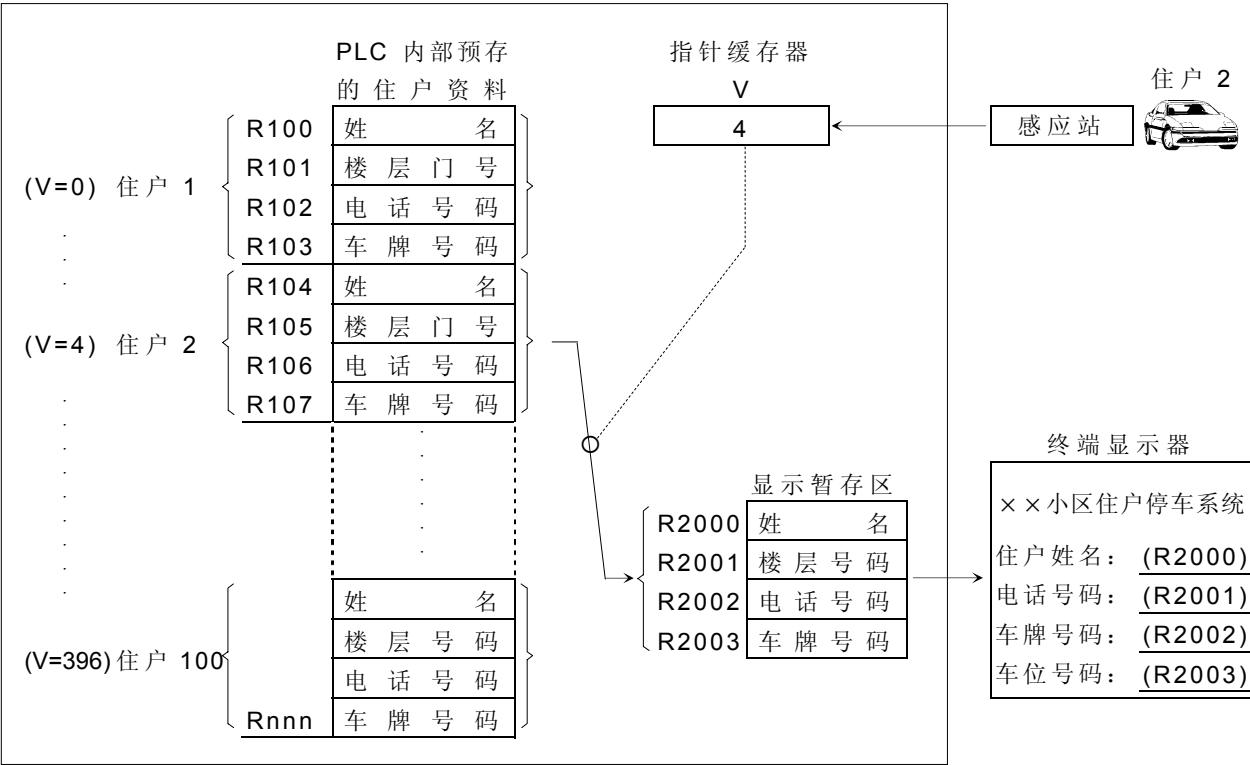
- 在间接寻址应用中，RXXXX 缓存器可以结合指针缓存器 V、Z 和 P0~P9 作间接寻址应用；DXXXX 缓存器不可以结合指针缓存器 V、Z 作间接寻址应用，但可以结合 P0~P9 作间接寻址应用。
- 当 RXXXX 缓存器要结合 V、Z 作间接寻址应用时，例如 R0 要结合 V、Z 做间接寻址应用，则所输入的格式为 R0V(当 V=100 时，则指向 R100)或 R0Z(当 Z=500 时，则指向 R500)；而要结合 P0~P9 作间接寻址应用时，则所输入的格式为 RPn (n 为 0~9)或为 RPmPn (m,n 为 0~9)，例如 RP5 (若 P5 内容为 100，则指向 R100) 或 RP0P1(若 P0 内容为 100，P1 内容为 50，则指向 R150)。
- 当 DXXXX 缓存器要结合 P0~P9 作间接寻址应用时，则所输入的格式为 DPn (n 为 0~9)或为 DPmPn (m,n 为 0~9)，例如 DP3 (若 P3 内容为 10，则指向 D10) 或 DP4P5(若 P4 内容为 100，P5 内容为 1，则指向 D101)。
- P0~P9 指针缓存器可同时结合运用，例如 P2=20、P3=30，当 RXXXX 或 DXXXX 缓存器一次结合两个指针缓存器时，RP2P3 就会指向 R50，DP2P3 就会指向 D50；也就是说两个指标值之间的关系是相加的。



1. 开机时 M1924 开始脉冲将 100 搬入指标暂存器 P2 中。
2. 当 X23 由 0→1 时，Fun103 将由 R100(因为 P2=100)开始，一次 4 个暂存器的长度，按照顺序搬到 R2000。也就是说第一次将 R100~R103 搬到 R2000~R2003，第二次将 R104~R107 搬到 R2000~R2003，第三次将 R108~R111 搬到 R2000~R2003 依次类推。
3. Fun11 用来将指标每次增加 4 个 word 用，也就是 X23 每"ON"一次，P2 的指标值就加上 4。

间接定位程序范例:

梯 形 图	简码指令
	ORG SHORT FUN 103 Ts: R100V Td: R2000 L: 4



程序说明: 上例假设某小区住户的自动化停车场管理系统, 共有 100 个住户停车位, 每个住户都有 1 组基本资料, 分别为住户姓名, 电话号码, 车牌号码, 停车位号码等 4 个, 每组数据如上图标占用连续 4 个 PLC 内部缓存器, 共占用 R100~R499 等 400 个缓存器, 每个住户都有一个不同卡号的卡片, 以供进出口门禁及停车场进出的感应通行使用, 其卡号为 0, 4, ……, 396 等 100 种, 在 PLC 感应到卡号后, 将它存入指针缓存器“V”, 而在管理员处的终端机 (LCD 或 CRT) 显示数据则固定由 PLC 内部的 R2001~R2003 来抓取并显示, 如本例感应到住户 2 的卡片, 其值=4, 因此 V 缓存器=4, PLC 立即将 R104~R107 的数据搬移到显示暂存区 (R2000~R2003), 因此管理员处的终端机可在感应到住户 2 的卡片的同时, 立即将其数据显示在终端机上。



警告

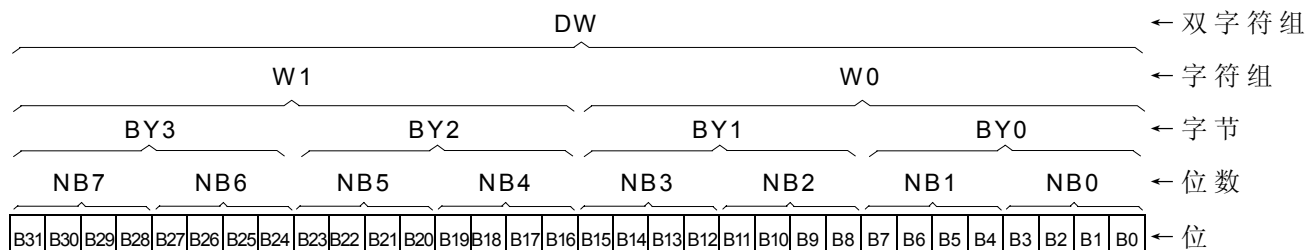
1. 运用指针缓存器作间接寻址的应用虽然功能强大、弹性方便，但相对地 V、Z 内容值的任意变化可能对正常数据区的错误写入有着极大的杀伤力，因此用户在使用时应特别小心。
2. 在间接寻址所能寻址的数据缓存器(R0~R8071)范围中，缓存器 R3840~R4167 等的 328 个缓存器（即 IR，OR，SR）为系统或 I/O 用的重要缓存器，任意对此等缓存器的写入，将可能使系统或 I/O 错误，造成重大的灾害。鉴于 V、Z 值对缓存器地址变化的灵活变化，用户可能不易察觉或掌握，因此 FBS-PLC 对间接寻址的写入动作会自动检查写入目的（Destination）是否在上述的 R3840~R4067 范围内，若是则不执行该写入动作，并将“间接寻址不合法写入”旗标 M1969 设为 1。若应用上确实需要对 R3840~R4067 的缓存器作写入，请使用非间接寻址的指令来执行。

5.3 数目系统

5.3.1 二进制数值及其术语

二进制(Binary)为数字计算器的基本数目系统，PLC 是由数字计算器所构成，自然也采用二进制，为便于表示及掌握二进数值，首先需了解如下的术语：

- 位：(Bit 简写 B，如 B0，B1.....) 位为二进制数值的最基本单位，其状态非 1 即 0。
- 位数：(Nibble 简写 NB，如 NB0，NB1.....) 由连续的 4 个位所组成（如 B3~B0）可用来表示一个位数的 10 进制数字 0~9 或 16 进制的 0~F。
- 字节：(Byte 简写 BY，如 BY0，BY1，.....) 是由连续的两个位数所组成（也就是 8 个位，例如 B7~B0）。可表示 16 进制的两个位数值 00~FF。
- 字符组：(Word 简写 W，如 W0，W1，.....) 是由连续的两个位所组成（也就是 16 个位例如 B15~B0）可表示 16 进制的 4 个位数值 0000~FFFF。
- 双字符组：(Double word 简写 DW，如 DW0，DW1.....) 是由连续的两个字字节所组成（即 32 个位，例如 B31~B0）可表示 16 进制的 8 个位数值 00000000~FFFFFFFF。



- 浮点数：(Floating Point Number) 也是由连续的两个字字节所组成。浮点数所能表示的最大范围为 $\pm(1.8 \times 10^{-38} \sim 3.4 \times 10^{38})$ ，有关详细的格式说明请参考 5.3.6 节。

5.3.2 FBS-PLC 的数码

FBS-PLC 内部的数值运算或储存全部采用二进制 (Binary)，因此外界输入 PLC 内部的数值必须转换成二进制码 PLC 才能处理，同样地从 PLC 内部取出的数值结果也都为二进制，因此无论 FP-07C 或 WinProladder 其所有数目最终都须化成二进制才能输入 PLC。但因二进制极难输入和阅读，因此 FP-07C 或 WinProladder 在人机界面 (数值输入或显示) 部分都提供用户以人们熟悉的 10 进制 (Decimal) 或 16 进制 (Hexadecimal) 来输入或显示，但实际上的数值处理全部都以二进制码来进行。

注：若您的数值输入或显示不是通过 FP-07C 或 WinProladder (例如以指拨开关或 7 段显示器通过 I/O 点而输入 PLC 或从 PLC 取出)，那么就须自己借着梯形图程序指令来处理二进制和 10 进制之间的转换，这样虽不通过 FP-07C 或 WinProladder 也能以 10 进制来输入及得到 10 进制的输出显示，请参考 FUN20(BIN→BCD)和 FUN21(BCD→BIN)的说明。

5.3.3 数值的范围

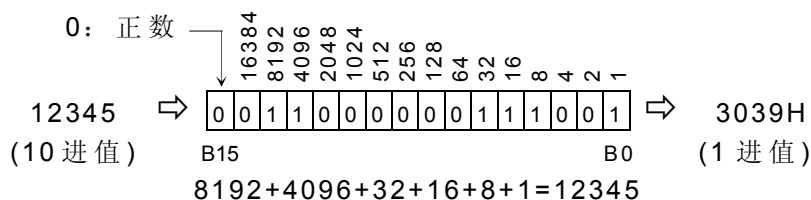
如前面叙述 FBS-PLC 内部全部采用二进制 (BCD 值只是为了适合人们习惯，而由二进制转成适合人们阅读的显示用数码而已)。在 PLC 的数值有 16 位、32 位以及浮点数三种数值，分别能表示如下的范围。

16 位	-32768~32767
32 位	-2147483648~2147483647
浮点数	$\pm(1.8 \times 10^{-38} \sim 3.4 \times 10^{38})$

5.3.4 数值的表示 (初学者请跳过本节)

以下各节将叙述 16 位及 32 位数值的表示方式与格式。以供用户能深入了解数值的运算过程及结果而能应付各种复杂的应用需求。

无论是 16 位或 32 位的数值，都以其最高位 MSB (16 位的 B15, 32 位的 B31) 表示该数值的正负 (0: 正数, 1: 负数)，剩下的位 (B14~B0 或 B30~B0) 才真正用以表示数值大小，现以 16 位为例说明如下：(32 位的作法也一样，只是长度倍增而已)。



如上例，无论 16 位或 32 位，其二进制的位由最低位 LSB (B0) 开始，B0 代表 1，B1 代表 2，B2 代表 4，B3 代表 8，..... 依此类推，而其数值则为所有为 1 的位所代表数值的总和。

5.3.5 负数的表示及取得 (初学者请跳过本节)

如前述当 MSB 为 1, 则此数为负数, FBs-PLC 的负数是以“2 的补码”(2'S Complement)来表示。所谓 2 的补码, 是将等值正数的所有位 (B15~B0 或 B31~B0) 倒相 (为 1 的位变 0, 为 0 的位变 1, 也就是所谓 1 的补码), 然后再加上 1 即变成 2 的补码, 现以上例正数 12345, 取其 2 的补码 (即 -12345) 为例说明如下:

$$\begin{array}{lcl}
 12345 & \Rightarrow & \boxed{0011000000111001} \Rightarrow 3039H \\
 12345 \text{ 的 } 1 & \Rightarrow & \boxed{1100111111000110} \Rightarrow CFC6H \\
 \text{的补码} & & \\
 + & & 1 \\
 \hline
 12345 \text{ 的 } 2 & \Rightarrow & \boxed{1100111111000111} \Rightarrow CFC7H \\
 \text{的补码} & & \\
 (-12345) & &
 \end{array}$$

5.3.6 浮点数的表示 (初学者请跳过本节)

永宏 FBs 浮点数格式同 IEEE-754 所制定的标准。储存的格式共占用 32 个位(双字符组)其说明如下所示:

$$\text{浮点数} = (\text{正/负})\text{数} + \text{指数} + \text{小数}$$

正/负数	指数	小数
b ₃₁	b ₃₀ ~ b ₂₃	b ₂₂ ~ b ₀
1 bit	8 bits	23 bits

32 bits

- ▲ 若(正/负)位的值为 0, 表示此浮点数为正值, 反之如果(正/负)位的值为 1, 代表该浮点数为负值。
- ▲ 指数表示法为超 127 法, 举例来说, 如果指数的值为 128 就代表 1 次方, 指数的值为 129 就代表 2 次方 依此类推。如果想表示指数为负的值, 则 126 就是 -1 次方, 125 为 -2 次方 依此类推。
- ▲ 小数字数有 23bits, 用来存放小数点以下的数字, 正规化格式要求小数点之前一个位数, 必须为 1, 且不必储存(称为隐藏位)。

- 整数转换浮点数的规则为如下:

$$N = (-1)^S * 2^{(E-127)} * (1.M) \quad 0 < E < 255$$

转换范例 1:

$$1 = (-1)^0 * 2^{(01111111)} * (1.000..... 0)$$

此范例中(正/负)数字元为 0, 指数部分为 127(超 127 法)=01111111, 隐藏位为 1, 而小数字数则全部为 0, 因此经过转换后的浮点数表示法如下所示:

$$\text{Code}(1) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \hline s & e & e & e & e & e & e & e & e & m & m & m & m & m & m & \dots & m & m & m \\ \hline \end{array}$$

$$= 3F800000H$$

转换范例 2:

$$0.5 = (-1)^0 * 2^{(01111110)} * (1.000\dots 0)$$

此范例中(正/负)数字元为 0, 指数部分以超 127 法表示为 126=01111110(因指数部分为 -1, 126-127=-1, 故指数部份在超 127 法中要转换成 126), 隐藏位为 1, 而小数字数则全部为 0, 因此经过转换后的浮点数表示法如下所示:

$$\text{Code}(0.5) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \hline s & e & e & e & e & e & e & e & e & m & m & m & m & m & m & m & \dots & m & m & m \\ \hline \end{array}$$

$$= 3F000000H$$

转换范例 3:

$$-500.125 = (-1)^1 * 2^{(10000111)} * (1.111101000010000000000000)$$

此范例中(正/负)数字元为 1, 指数部分以超 127 法表示为 135=10000111 (因指数部分为 8, 135-127=8, 故指数部份在超 127 法中要转换成 135), 隐藏位部分为 1, 而小数字数则为 111101000010000000000000, 因此经过转换后的浮点数表示法如下所示:

$$\text{Code}(-500.125) = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hline s & e & e & e & e & e & e & e & e & m & m & m & m & m & m & m & m & m & m & m & m & m & m & m & m & m & m & m \\ \hline \end{array}$$

$$= C3FA1000H$$

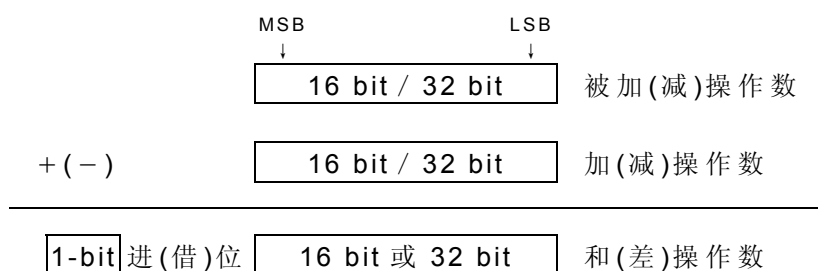
5.4 操作数递增(减)的溢位与欠位 (初学者请跳过本节)

16 位及 32 位操作数所能表示的数字极限正数最上限为 32767 和 2147483647, 而负数的最下限为 -32768 和 -2147483648。当对一个操作数作递增或递减动作 (例如计数器上/下数或使缓存器值 +1 或 -1 时, 若结果使其数值超过该操作数正数的最上限, 则称为溢位 (Overflow: OVF), 溢位的结果会使数值循环至负数的最下限 (例如 16 位上限 32767 再加 1, 变成 -32768)。若递增(减)结果使其数值小于负数最下限, 则称为欠位 (Underflow: UDF), 而欠位的结果会使数值循环至正数的最上限 (例如 -32768 若再减 1 变成 32767), 如下表所示。在 FBs-PLC 的递增指令的功能输出 (FO) 都有溢位或欠位的旗号输出, 可作串联 (Cascade) 应用而获得超过 16 或 32 位的运算结果。

递增(减) 溢/欠位	16 bit 操作数	32 bit 操作数
递增	↑ ↶ -32767 ↶ -32768 OVF=1 ↶ 32767 ↶ 32766 ↶ 32765 ↑	↑ ↶ -2147483646 ↶ -2147483647 OVF=1 ↶ -2147483648 ↶ 2147483647 ↶ 2147483646 ↑
递减	↓ ↷ -32767 ↷ -32768 UDF=1 ↷ 32767 ↷ 32766 ↷ 32765 ↓	↓ ↷ -2147483647 ↷ -2147483648 UDF=1 ↷ 2147483647 ↷ 2147483646 ↷ 2147483645 ↓

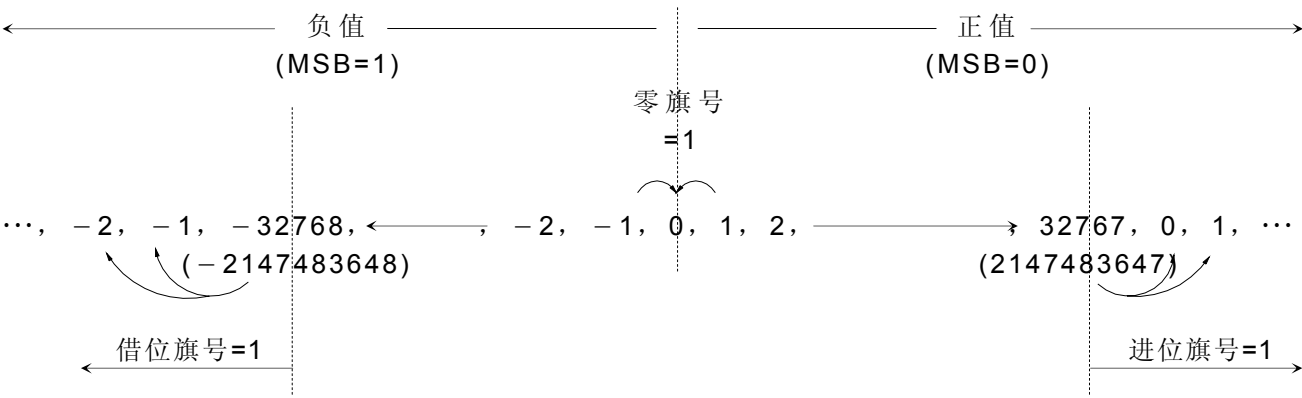
5.5 加/减运算的进位与借位

溢/欠位的发生是针对单一个操作数的递增/减致使该操作数的值超出其所能表示的正/负值极限时，产生溢/欠位旗号。而进/借位则不同于溢/欠位，首先其必有两个操作数作加(减)运算，而得到和(差)结果与进/借位旗号。因被加(减)、加(减)及和(差)的位数(bit数)都一样(16 bit 或 32bit)，因此相加(减)的结果将可能造成和(差)的值超出16或32 bit，因此需以进(借)位旗号配合和(差)操作数来表示真正的数值。而进位旗号发生在加(减)结果超出该和(差)操作数所能表示的正值最大极限(32767 或 2147483647)时，而借位则发生在加(减)结果超出该和(差)操作数所能表示的负值最大极限(-32768 或 -2147483648)时，因此加(减)运算后的真正结果为进(借)位再加上和(差)操作数的值，FBs-PLC 的加减指令的功能输出(FO)都有进位与借位旗号输出，可供您获得真正的结果。



因 FBs-PLC 的数值运算都采用 2 的补码，因此加(减)运算所得的和(差)值的负值

的表示将不同于我们一般习惯的负值表示方式。运算结果为负值时，其和（差）操作数将永远不可能出现 0 值。进位旗号代表正值 32768(2147483648)，借位旗号则代表负值 -32768(-2147483648)。



			MSB		LSB		
C=1	B=0	Z=0	0	0	0	0	32769
C=1	B=0	Z=0	0	0	0	0	32768
C=0	B=0	Z=0	0	1	1	1	32767
C=0	B=0	Z=0	0	1	1	1	32766
C=0	B=0	Z=0	0	1	1	1	32765
C=0	B=0	Z=0	0	0	0	0	2
C=0	B=0	Z=0	0	0	0	0	1
C=0	B=0	Z=1	0	0	0	0	0
C=0	B=0	Z=0	1	1	1	1	-1
C=0	B=0	Z=0	1	1	1	1	-2
C=0	B=0	Z=0	1	0	0	0	-32766
C=0	B=0	Z=0	1	0	0	0	-32767
C=0	B=0	Z=0	1	0	0	0	-32768
C=0	B=1	Z=0	1	1	1	1	-32769
C=0	B=1	Z=0	1	1	1	1	-32770