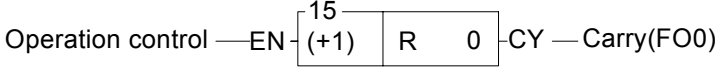
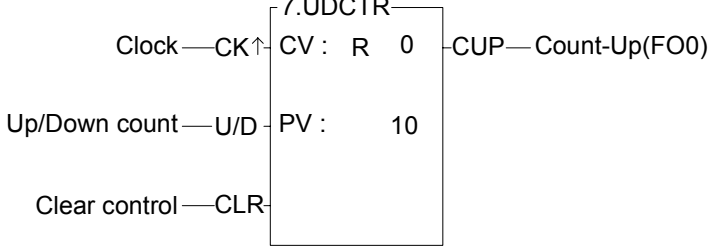


## Chapter 5 Descriptions of Function Instructions

### 5.1 The Format of Function Instructions

In this chapter we will introduce the function instructions of FBs-PLC in details. All the explanations for each function will be divided into four parts including input control, instruction number/name, operand and function output. If use the FP-07 to input the mnemonic instruction, except for the T, C, SET, RST and SFC instructions that can be entered directly by pressing a single key stroke on FP-07, other function instructions must be entered by key in the instruction number rather than the instruction name. An example is shown in below.

Ladder Diagram	FP-07 Mnemonic code
<p>Example 1: Single input instruction</p>  <p>Operation control — EN — <span style="border: 1px solid black; padding: 2px;">15 (+1) R 0</span> — CY — Carry(FO0)</p>	<p>FUN 15  <span style="border: 1px solid black; padding: 2px;">D</span>: R 0</p>
<p>Example 2: Multiple input instruction</p>  <p>Clock — CK↑ — <span style="border: 1px solid black; padding: 2px;">7.UDCTR CV: R 0 PV: 10</span> — CUP — Count-Up(FO0)</p> <p>Up/Down count — U/D —</p> <p>Clear control — CLR —</p>	<p>FUN 7  <span style="border: 1px solid black; padding: 2px;">CV</span>: R 0  <span style="border: 1px solid black; padding: 2px;">PV</span>: 10</p>

Remark : The words inside the hollow box in mnemonic code field are the prompting message from FP-07 such as D, CV, and Pr and are not entered by the user.

#### 5.1.1 Input Control

Except for the seven function instructions that do not have input control, the number of the input control of other FBs-PLC function instructions can be ranged from one to four. Execution of the instructions and operations is dependent on the input control signal or the combinations of the several input control signals. The ladder programming software for FACON PLC - Winproladder can help user to complete the complex design and document works. In the ladder program window we can see all the function instructions were displayed by blocks surrounded with abbreviated words for ease of comprehension, include inputs, outs, function name, and parameter names. As shown in example 2 above, the first input mark "CK ↑" indicates when the "CK ↑" input changes from 0 to 1 (rising edge) the counter will be increased or decreased by 1 (depending on the "U/D" status). The second input mark "U/D" with a status of 1 represents the word above slash ("U") and the status 0 represents the word under slash ("D"), that is second input "U/D" states =1, the counter will be increased by 1 when "CK ↑" input from 0 to 1, and when "U/D"=0, the counter will be decreased by 1. The third input mark "CLR" indicates when this input is 1, the counter will be cleared to 0. Chapter 8~9 give the descriptions of input control of each function instruction.

Remark: There are total of seven instructions whose input control should be directly connected to the origin-line those are MCE, SKPE, LBL, RTS, RTI, FOR, and NEXT. Please refer to chapter 6 and 7 for more detailed explanations.

All input controls of the function instructions should be connected by the corresponding elements, otherwise a syntax error will occur. As shown in example 3 below, the function instruction FUN7 has three inputs and three elements before FUN7. ORG X0, LD X1 and LD X2 corresponds to the first input CK ↑, second input U/D and third input CLR.

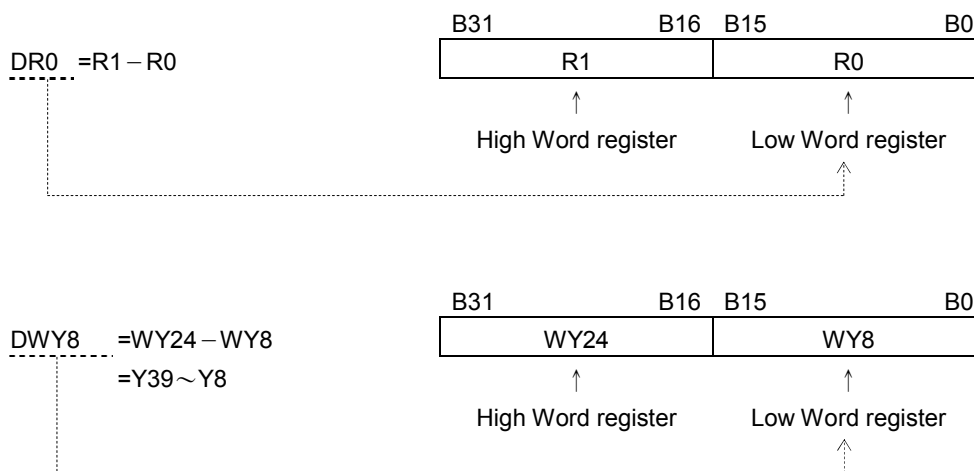
Example 3:

Ladder Diagram	FP-07 Mnemonic code
	<pre> ORG   X 0 LD    X 1 LD    X 2 FUN   7       CV: R 0       PV: 10 </pre> <p>FUN7 need three elements because it has three inputs</p>

### 5.1.2 Instruction Number and Derivative Instructions

As mentioned before, except for the nine instructions that can be entered using the dedicated keys on the keyboard, other function instructions must be entered using the "instruction number". Follow the instruction number there are postfixes D, P, DP can be added which can derive three additional function instructions.

D: Indicates a Double Word (32-bit). The 16-bit word is the basic unit of the registers in FBs-PLC. The data length of R, T and C (except C200~C255) registers are 16-bit. If a register with 32-bit data length is required, then it is necessary to combine two consecutive 16-bit registers together such as R1-R0, R3-R2 etc. and those registers are represented by prefix a D letter before register name such as DR0 represents R1-R0 and DR2 represents R3-R2. If you enter DR0 or DWY8 in the monitor mode of FP-07, then a 32-bit long value (R1-R0 or WY24-WY8) will be displayed.

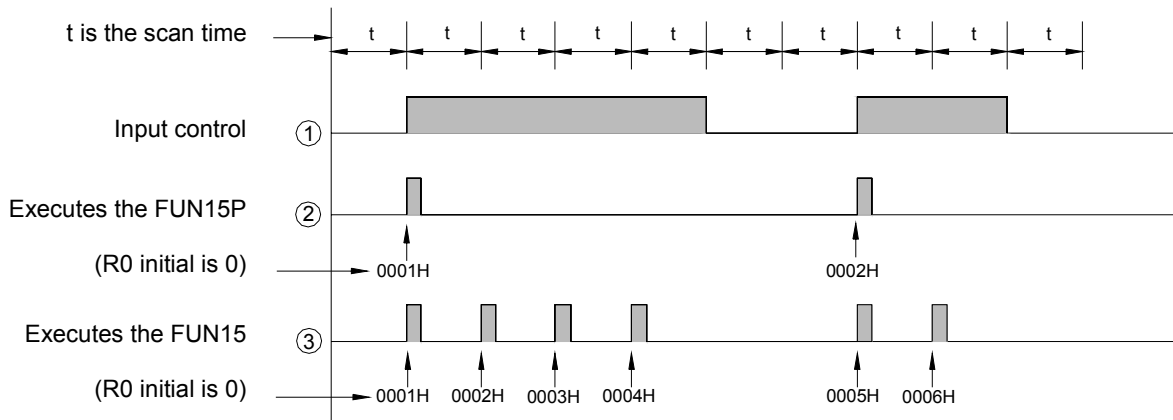


Remark: In order to differentiate between 16-bit and 32-bit instructions while using the ladder diagram and mnemonic code, we add the postfix letter D after the "Instruction number" to represent 32-bit instructions and the size of their operand are 32-bit as shown in example 4 on P.6-6. The instruction FUN 11D has a postfix letter D, therefore the source and destination operands need to prefix a letter D as well, such as the augend Sa : R0 is actually Sa=DR0=R1-R0 and Sb=DR2=R3-R2. Please also pay special attention to the length of the other operands except source and destination are only one word whether 16-bit or 32-bit instructions are used.

P: indicates the pulse mode instruction. The instruction will be executed when the status of input control changes from 0 to 1 (rising edge). As shown in example 1, if a postfix letter P is added to the instruction (FUN 15P), the instruction FUN 15P will only be executed when the status of input control signal changes from 0 to 1. The execution of the instruction is in level mode if it does not have a P postfix, this means the instruction will be executed for every scan until the status of input control changes from 1 to 0. The pulse input is indicated by a symbol "↑", such as CK ↑, EN ↑, TG ↑ etc.. In this operation manual, an example of the operation statement of a function instruction is shown below.

● When the operation control "EN" =1 or "EN ↑" (P instruction) from 0→1, .....

The first one indicates the execution requirement for non-P instruction (level mode) and the second one indicates the execution requirement for P instruction (pulse mode). The following waveform shows the result (R0) of FUN15 and FUN15P under the same input condition.



DP: Indicates the instruction is a 32-bit instruction operating with pulse mode.

Remark: P instruction is much more time saving than level instruction in program scanning, So user should use P instruction as much as possible.

### 5.1.3 Operand

The operand is used for data reference and storage. The data of source (S) operand are only for reference and will not be changed with the execution of the instruction. The destination (D) operand is used to store the result of operation and its data may be changed after the execution of the instruction. The following table illustrates the names and functions of FACON PLC function instruction's operands and types of contacts, coils, or registers that can be used as an operand.

■ The names and functions of the major operands:

Abbreviation	Name	Descriptions
S	Source	The data of source (S) operand are only for reading and reference and will not be changed with the execution of the instruction. If there are more than one source operands, each operand will be identified by the footnote such as Sa and Sb.
D	Destination	The destination (D) operand is used to store the result of operation. The original data will be changed after operation. Only the coils and registers which are not write prohibited can be the destination operand.
L	Length	Indicates the data size or the length of the table, usually are constants.
N	Number	A constant most often used as numbers and times. If there are more than one constant, each constant will be identified by the footnotes such as Na, Nb, Ns etc..
Pr	Pointer	Used to point to a specific a block of data or a specific data or register in a table. Generally the Pr value can be varied, therefore cannot be constant or input register. (R3840~R3847)
CV	Current value	Used in T and C instruction to store the current value of T or C
PV	Set value	Used in T and C instructions for reference and comparison
T	Table	A combination of a set of consecutive registers forms a table. The basic operation units are word and double word. If there is more than one table, each table will be identified by footnotes such as Ta, Tb, Ts and Td etc..
M	Matrix	A combination of a set of consecutive registers forms a matrix. The basic operation unit is bit. If there is more than one matrix, each matrix will be identified by footnotes such as Ma, Mb, Ms and Md etc..

Besides the major operands mentioned above, there are other operands which are used for certain special purposes such as the operand Fr for frequency, ST for stack, QU for Queue etc.. Please refer to the instruction descriptions for more details.

■ The types of the operand and their range: The types of operand for the function instructions are discrete, register and constant.

a) Discrete operand :

There are total five function instructions that reference the discrete operand, namely SET, RST, DIFU, DIFD and TOGG. Those five instructions can only be used for operations of Y (external output), M (internal and special) and S (step) relays. The table shown below indicates the operands and ranges of the five function instructions.

Range	Y	M	SM	S
Oper- rand	Y0   Y255	M0   M1911	M1912   M2001	S0   S999
D	○	○	○*	○

Symbol "O" indicates the D (Destination operand) can use this type of coils as operands. The "\*" sign above the "O" shown in SM column indicates that should exclude the write prohibited relays as operands. Please refer to page I2-8 for introduction of the special relays.

b) Register operand :

The major operand for function instructions is register operand. There are two types of register operands: the native registers which already is of Words or Double Words data such as R, D, T, C. The other is derivative registers (WX, WY, WM, WS) which are formed by discrete bits. The types of registers that can be used as instruction operands and their ranges are all listed in the following table:

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Oper- and	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	16/32-bit +/- number	V、Z P0~P9
	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D4095		
S	○	○	○	○	○	○	○	○	○	○	○*	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○
⋮														

The "○" symbol in the table indicates can apply this kind of data as operand. The "○\*" symbol indicates can apply this kind of data except the write prohibited registers as operand. To learn more about write prohibited registers please refer to page 3-6 for introduction of the special register.

When R5000~R8071 are not set to be read only registers, can used as normal registers (read, and write)

Remark 1: The registers with a prefix W, such as WX, WY, WM and WS are formed by 16 bits. For example, WX0 means the register is formed by X0(bit 0)~X15(bit 15). WY144 means the register is formed by Y144(bit 0)~Y159(bit 15). Please note that the discrete number must be the multiple of 8 such as 0, 8, 16, 24....

Remark 2: The last register (Word) in a table can not be represented as a 32-bit operand in the function because 2 Words are required for a 32-bit operand.

Remark 3: TMR ( T0~T255 ) and CTR ( C0~C255 ) are the registers of timers and counters respectively. Although they can be used as general registers, they also complicate the systems and make debugging more difficult. Therefore you should avoid writing anything into the TMR or CTR registers.

Remark 4: T0~T255 and C0~C199 are 16-bit register. But C200~C255 are 32-bit register, therefore can't be used as 16-bit operands.

Remark 5: Apart from being directly appointed by register's number (address) as the foregoing discussions, the register's operand in the range of R0~R8071 can be combined with pointer register V or Z to make indirect addressing. Please refer to the example in the next section (Section 5.2) for the description of using pointer register (XR) to make indirect addressing.

c) Constant operands :

The range of 16-bit constant is between -32768~32767. The range of 32-bit constant is between -2147483648~2147483647. The constant for several function instructions can only be a positive constant. The range of 16-bit and 32-bit constants are listed in the table shown below.

Classification	Range
16-bit signed number	-32768~32767
16-bit un-signed number	0~32767
32-bit signed number	-2147483648~2147483647
32-bit un-signed number	0~2147483647
16/32-bit signed number	-32768~32767 or -2147483648~2147483647
16/32-bit un-signed number	0~32767 or 0~2147483647

It is possible that the length and size of a specific operand, such as L, bit size, N etc., are different, and the differences are all directly marked at the operand column. Please refer to the explanations of function instructions.

### 5.1.4 Functions Output (FO)

The "Function Output" (FO) is used to indicate the operation result of the function instruction. Like control input, each function outputs shown in the screen of programming software are all attached with a word which comes from the abbreviation of the output functionality. Such as CY derived from CarrY. The maximum number of function outputs is 4 and those are denoted as FO0, FO1, FO2, FO3 respectively. The FO status must be taken out by FO instruction (there is a FO special key on FP-07 program writing device). The unused FO may be left without connecting to any elements, such as FO1 (CY) shown in Example 4 below.

Example 4 :

Ladder Diagram	Mnemonic Codes
	<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>

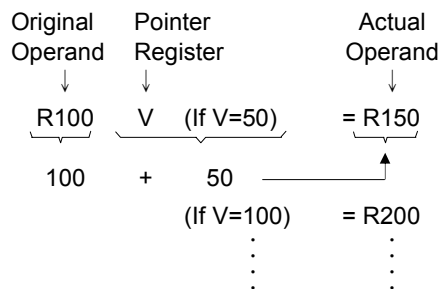
When M1919=0, the FO status will only be updated if the instruction is executed. It will keep the same status until a new FO status is generated after the instruction is executed again (memory keeping).

When M1919=1, the FO status will be reset to 0 (no memory keeping) if the instruction is not executed.

### 5.2 Use Index Register(XR) for Indirect Addressing

In the FBs-PLC function instructions, there are some operands that can be combined with pointer register (V · Z · P0~P9) to make indirect addressing (will be shown in the operand table if it applicable). However, only the registers in the range R0~R8071 can be combined with an pointer register to perform indirect addressing (other operands such as discrete, constant and D0~D3071 cannot be used for indirect addressing).

There are twelve pointer registers XR (V · Z · P0~P9). The V register in fact is the R4164 of special registers (R3840~R4167) , the Z register is the R4165 and the P0~P9 register is the (D4080~D4089). The actual addressed register by index addressing is just offset the original operand with the content of the index register.



As shown in the above diagram, you only need to change the V value to change the operand address. After combining the index addressing with the FBs-PLC function instructions, a powerful and highly efficient control application can be achieved by using very simple instructions. Using the program shown in the diagram below as an example, you only need to use a block move instruction (BT\_M) to achieve a dynamic block data display, such as a parking management system.

### Index Register(P0~P9) Introduction

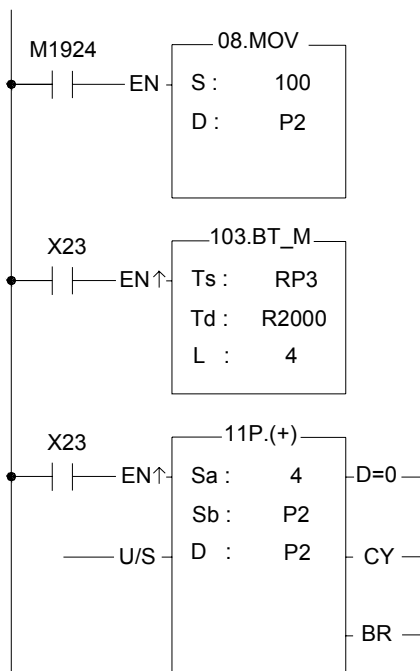
In indirect addressing application, Rxxxx register can combine V · Z & P0~P9 for index addressing; Dxxxx register can't combine V · Z for index addressing, but P0~P9 are allowed.

When V · Z index register being combined with the Rxxxx register,

for example, R0 with V · Z, the instruction format is R0V(where V=100, it means R100) or R0Z(where Z=500, it means R500); when P0~P9 index register being combined with the Rxxxx register, the instruction format is RPn (n=0~9) or RPmPn (m,n=0~9), for example RP5 (where P5=100, it means R100) or RP0P1(where P0= 100, P1=50, it means150).

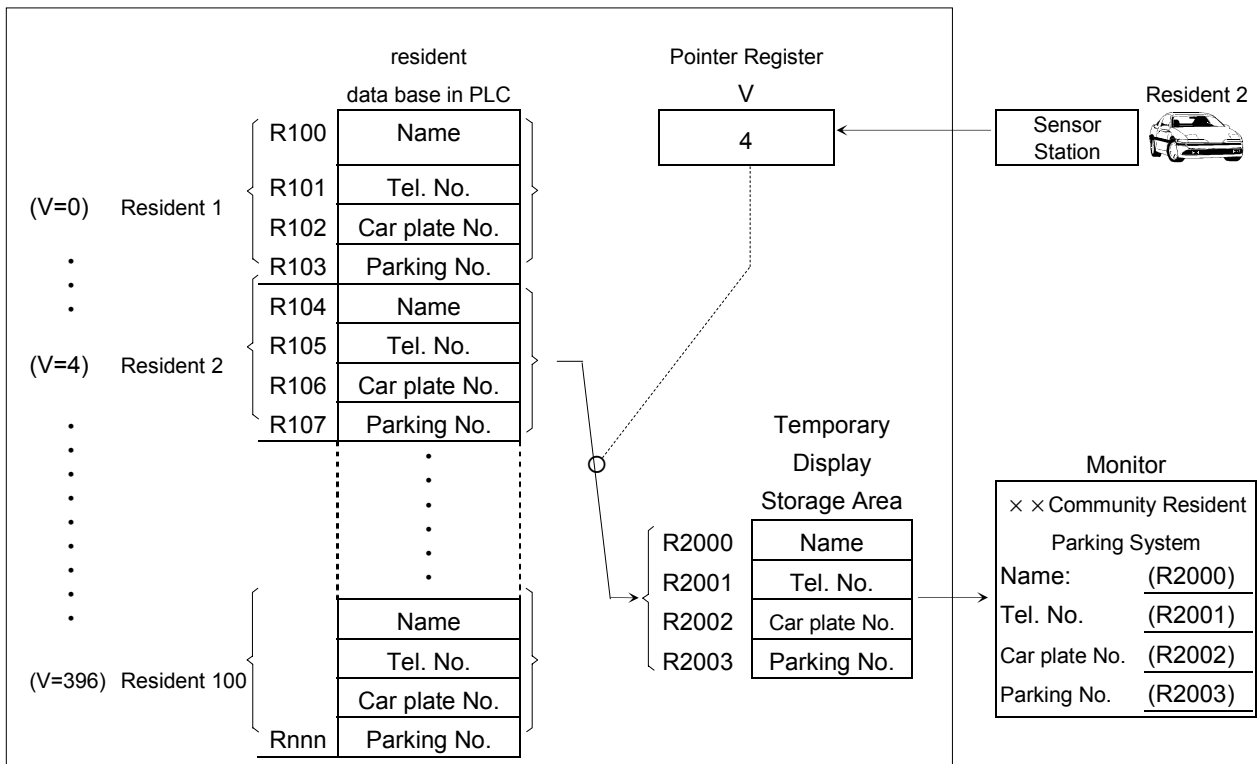
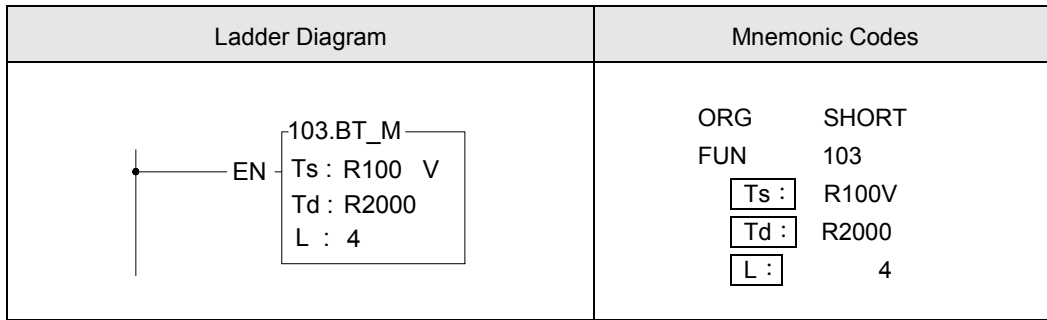
When P0~P9 index register being combined with the Dxxxx register, the instruction format is DPn (n=0~9) or DPmPn (m,n=0~9), for example DP3 (where P3=10, it means D10) or DP4P5 (where P4=100, P5=1, it means D101).

It can combine both P0~P9 index register, for example P2=20, P3=30, when Rxxxx or Dxxxx register combines both index register, RP2P3 will point to R50, DP2P3 will point to D50, it means the summation of both index register for indirect addressing.



1. Index register P2=100 while power up or first run.
2. When X23 changes from 0→1, FUN103 will perform the table movement, the source starts from R100 (P2=100), the destination starts from R2000, the amount is 4. Copying the content of R100~R103 for R2000~R2003 at first execution, copying the content of R104~R107 for R2000~R2003 at second execution...
3. Increasing the P2 index register by 4 to point to next 4

**Indirect addressing program example**



**Description**

Suppose that there are 100 resident parking spaces available in a parking management system for community residents. Each resident has a set of basic information including name, telephone number, number plate and parking number, that occupy four consecutive PLC registers as shown in the above diagram. A total of 400 registers (R100~R499) are occupied. Each resident is given a card with a unique card number (the number is 0 for resident 1, 4 for resident 2 etc.. ) for the sensing pass of the main entrance and parking lot. The card number will be sensed by the PLC and stored into the pointer register "V". The attendant's monitor (LCD or CRT) will only display the data grasped by R2001~R2003 in the PLC. For example, the card of residence 2 with the card number 4 is sensed, then the register V=4 and the PLC will immediately move the data in R104~R107 to the temporary display storage area (R2000~R2003). Hence, the attendant's monitor can display the data of residence 2 as soon as its card is sensed.

**Warning**

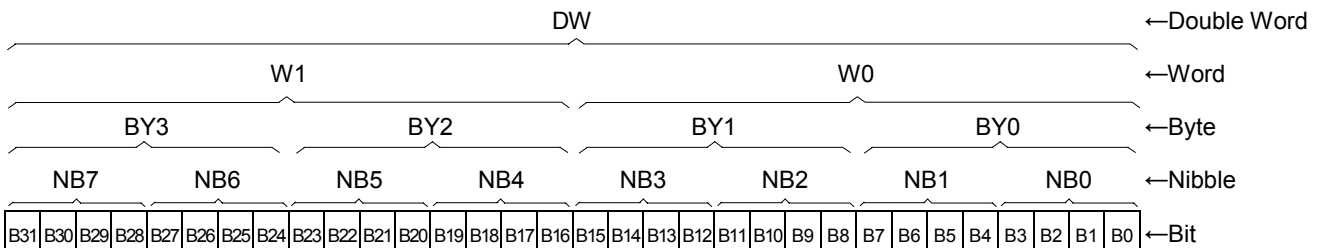
1. Although using pointer register for indirect addressing application is powerful and flexible, but changing the V and Z values freely and carelessly may cause great damages with erroneous writing to the normal data areas. The user should take special caution during operation.
2. In the data register range that can be used for indirect addressing application (R0~R8071), the 328 registers R3840~R4167 (i.e. IR, OR and SR) are important registers reserved for system or I/O usage. Writing at-will to these registers may cause system or I/O errors and may result in a major disaster. Due to the fact that users may not easily detect or control the flexible register address changes made by the V and Z values, FBs-PLC will automatically check if the destination address is in the R3840~R4067 range. If it is, the write operation will not be executed and the M1969 flag "Illegal write of Indirect addressing" will be set as 1. In case it is necessary to write to the registers R3840~R4067, please use the direct addressing.

### 5.3 Numbering System

#### 5.3.1 Binary Code and Related Terminologies

Binary is the basic numbering system of digital computer. Since the PLC operates with discrete ON/OFF values, it is natural to use binary codes. The following terminologies should be fully understood before go to further topic of numbering system.

- Bit: (Abbreviated as B, such as B0, B1, and so on) It is the most basic unit of binary value. The status of bit is either "1" or "0".
- Nibble: (Abbreviated as NB, such as NB0, NB1, and so on) It is formed by four consecutive bits (e.g. B3~B0) and can be used to represent a decimal number 0~9 or a hexadecimal number 0~F.
- Byte: (Abbreviated as BY, such as BY0, BY1, and so on) It is formed by two consecutive nibbles (or 8 bits, such as B7~B0) and can be used to represent a 2-digit hexadecimal number 00~FF.
- Word: (Abbreviated as W, such as W0, W1, and so on) It is formed by two consecutive bytes (or 16 bits, such as B15~B0) and can be used to represent a 4-digit hexadecimal number 0000~FFFF.
- Double Word: (Abbreviated as DW, such as DW0, DW1, and so on) It is formed by two consecutive words (or 32 bits, such as B31~B0) and can be used to represent an 8-digit hexadecimal number 00000000~FFFFFFFF.





### 5.3.2 The Coding of Numeric Numbers for FBs-PLC

FBs-PLC use the binary numbering system for its internal operations that is the data of external BCD inputs must be converted to binary number before the PLC can process. As we know the binary code is very difficult to read and input to the PLC for human, therefore FP-07 and WinProladder use the decimal unit or hexadecimal unit to input or to display the data. But in reality, all the operations taking place in the PLC are performed with binary code.

Remark: If you input or display the data without going through the FP-07 or WinProladder (For instance, input data into or take out data from PLC through the I/O terminals using thumb wheel switch or seven segment display), then you have to use the Ladder program to perform the Decimal to Binary conversion. This enables you to input and display data without using the FP-07 and WinProladder. Please refer to FUN20(BIN→BCD) and FUN21(BCD→BIN).

### 5.3.3 Range of Numeric Value

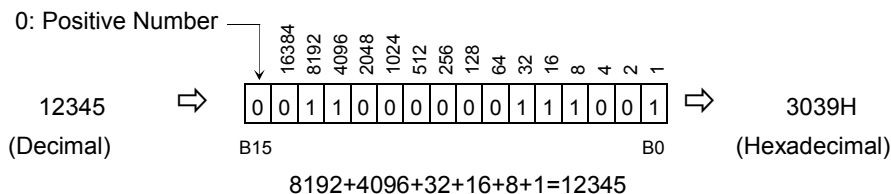
As we have mentioned before that FBs-PLC uses binary numbers for its internal operations. 16-bit and 32-bit are three different numeric data of FBs-PLC. The ranges of the three numeric values are shown below.

16-bit	- 32768 ~ 32767
32-bit	- 2147483648 ~ 2147483647
Floating point number	$\pm(1.8 \times 10^{-38} \sim 3.4 \times 10^{38})$

### 5.3.4 Representation of Numeric Value (Beginners can skip this section)

The representation and specification of 16-bit and 32-bit numeric values are provided below to enable the user to further understand the numeric value operation for more complicated applications.

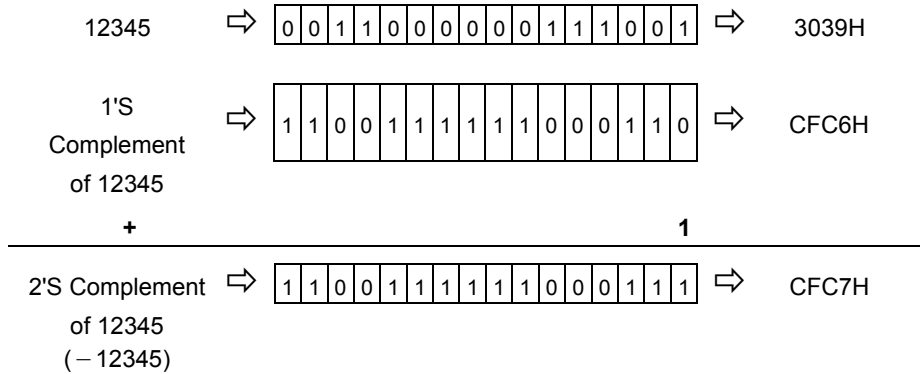
The most significant bits MSB of 16-bits and 32-bits (B15 for 16-bit and B31 for 32-bit) are used to identify positive and negative numbers (0: positive and 1: negative). The remaining bits (B14~B0 or B30~B0) represent the magnitude of the number. The following example uses 16-bit for further explanations. Please note that everything also applies to 32-bit numbers and the only difference is the length.



In the above example, regardless of its size (16-bit or 32-bit), and starting with the least significant bit LSB (B0). B0 is 1, B1 is 2, B2 is 4, B3 is 8, and so on. The number represented by the neighboring left bit will double its value (1, 2, 4, 8, 16, and so on) and the value is the sum of the numbers represented by the bits that are equal to 1.

### 5.3.5 Representation of Negative Number (Beginners should skip this section)

As prior discussion, when the MSB is 1, the number will be a negative number. The FBs-PLC negative numbers are represented by 2'S Complement, i.e. to invert all the bits (B15~B0 or B31~B0) of its equivalent positive number (The so-called 1'S Complement is to change the bits equal 1 to 0 and the bits equal 0 to 1) then add 1. In the above example, the positive number is 12345. The calculation of its 2'S Complement (i.e. -12345) is described below:



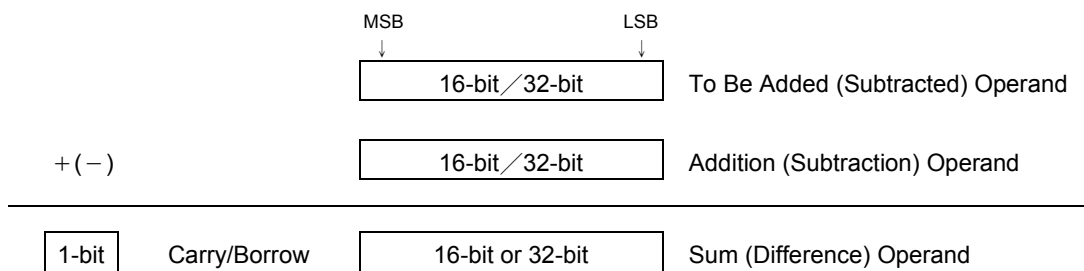
### 5.4 Overflow and Underflow of Increment (+1) or Decrement (-1) (Beginners should skip this section)

The maximum positive value that can be represented by 16-bit and 32-bit operands are 32767 and 2147483647, respectively. While the minimum negative values that can be represented by 16-bit and 32-bit operands are -32768 and -2147483648, respectively. When increase or decrease an operand (e.g. when Up/Down Count of a counter or the register value is +1 or -1), and the result exceeds the value of the positive limit of the operand, then "Overflow" (OVF) occurs. This will cause the value to cycle to its negative limit (e.g. add 1 to the 16-bit positive limit 32767 will change it to -32768). If the result is smaller than the negative limit of the operand, then "Underflow" (UDF) occurs. This will cause the value to cycle to its positive limit (e.g. deducting 1 from the negative limit -32768 will change it to 32767) as shown in the table below. The flag output of overflow or underflow exists in the FO of FBs-PLC and can be used in cascaded instructions to obtain over 16-bit or 32-bit operation results.

Increase (Decrease) Result Overflow/ Underflow	16-bit Operand	32-bit Operand
Increase	OVF=1 <div style="display: flex; justify-content: center; align-items: center;"> <div style="margin-right: 10px;">{</div> <div style="text-align: left;">           - 32767            - 32768            32767            32766            32765         </div> </div>	OVF=1 <div style="display: flex; justify-content: center; align-items: center;"> <div style="margin-right: 10px;">{</div> <div style="text-align: left;">           - 2147483646            - 2147483647            - 2147483648            2147483647            2147483646         </div> </div>
Decrease	UDF=1 <div style="display: flex; justify-content: center; align-items: center;"> <div style="margin-right: 10px;">{</div> <div style="text-align: left;">           - 32767            - 32768            32767            32766            32765         </div> </div>	UDF=1 <div style="display: flex; justify-content: center; align-items: center;"> <div style="margin-right: 10px;">{</div> <div style="text-align: left;">           - 2147483647            - 2147483648            2147483647            2147483646            2147483645         </div> </div>

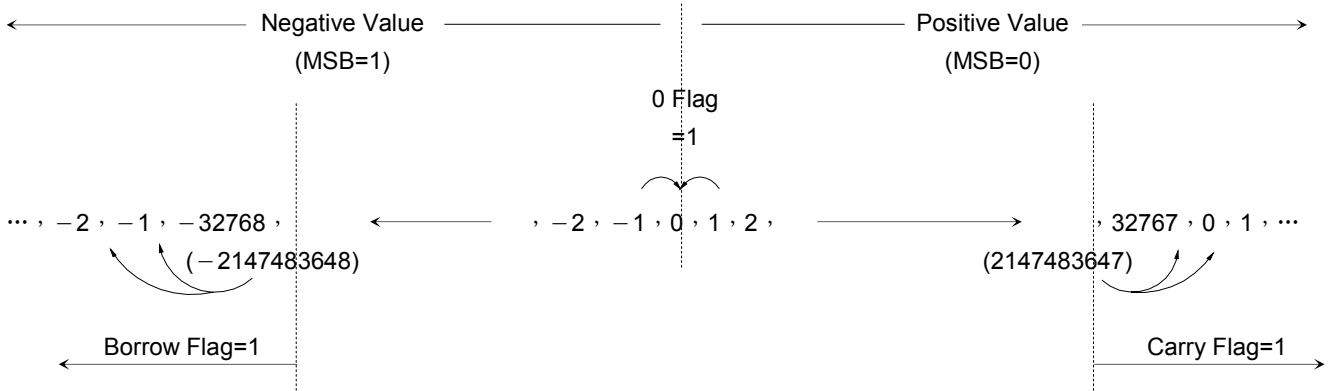
### 5.5 Carry and Borrow in Addition/Subtraction

Overflow/Underflow takes place when the operation of increment/decrement causes the value of the operand to exceed the positive/negative limit that can be represented in the PLC, consequently a flag of overflow/underflow is introduced. Carry/Borrow flag is different from overflow/underflow. At first, there must be two operands making addition (subtraction) where a sum (difference) and a flag of carry/borrow will be obtained. Since the number of bits of the numbers to be added (subtracted), to add (subtract) and of sum (difference) are the same (either 16-bit or 32-bit), the result of addition (subtraction) may cause the value of sum (difference) to exceed 16-bit or 32-bit. Therefore, it is necessary to use carry/borrow flag to be in coordination with the sum (difference) operand to represent the actual value. The carry flag is set when the addition (subtraction) result exceeds the positive limit (32767 or 2147483647) of the sum (difference) operand. The borrow flag is set when addition (subtraction) result exceeds the negative limit (-32768 or -2147483648) of the sum (difference) operand. Hence, the actual result after addition (subtraction) is equal to the carry/borrow plus the value of the sum (difference) operand. The FO of FBs-PLC addition/subtraction instruction has both carry and borrow flag outputs for obtaining the actual result.



While all FBs-PLC numerical operations use 2'S Complement, the representation of the negative value of the sum

(difference) obtained from addition (subtraction) is different from the usual negative number representation. When the operation result is a negative value, 0 can never appear in the MSB of the sum (difference) operand. The carry flag represents the positive value 32768 (2147483648) and the borrow flag represents the negative value -32768 (-2147483648).



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

C = Carry      B = Borrow      Z = Zero