分類/Classification	☐ tDS	☐ tGW	☐ PETL/tET/tPET	☐ DS/PDS/PPDS	☐ tM-752N
	☑ I/O Card		☐ VXC Card	☐ VxComm	☐ Other
作者/Author	Albert	日期/Date	2015-04-02	編號/No.	FAQ-022

Q: How to process data transmission from multiple DI/DO channels?

A: General Digital I/O cards have 8 channels per port, and each uses hexadecimal format, regardless of whether they are input or output. The following will discuss how the channel number corresponds to a hexadecimal value.

When an 8-Bit DO port is configured, the data format can be expressed using the following equation:

D/O byte = 0000 0000 (bit) = 0x00 (hex) (see Appendix 1).

It should be noted that the data information is read from left to right, respectively, i.e., Bit 7 to Bit 0, where Bit 7 ~ Bit 4 is combined as a group, and Bit 3 ~ Bit 0 is combined as another group, as shown in the table below. Refer to the settings for the channels on each card for the channel definition of each port number.

	Bit 7 ~ Bit 4	Bit 3 ~ Bit 0
D/O Byte	0 0 0 0	0 0 0 0

- ## Operation Examples

➢ Digital Output

Step 1: Output "Bit 2": D/O = 0000 0100 (bit) = 0x04 (hex).
Step 2: Output "Bit 5 and Bit 6": D/O = 0110 0100 (bit) = 0x64 (hex).
Step 3: Output "Bit 7" and "Bit 2" are not output: D/O = 1110 0000 (bit) = 0xE0 (hex).

➢ Digital Input

1. Assuming the D/I port receives the data set 0x3D.
   => D/I = 0x3D (hex) = 0011 1101 (bit).
   => Indicates that Bits 0, 2, 3, 4, and 5 have a value, which is means that Input channels 0,2,3,4, and 5 have data.

2. Assuming D/I port receives the data set 0x80.
   => D/I = 0x80 (hex) = 1000 0000 (bit).
   => Indicates that only Bit 7 has a value, which means that only Input channel 7 has data.

Pros: When reading or writing data, multi-channel data transmission allows at least 8 channels of data to be processed simultaneously, which can effectively improve processing efficiency.

Cons: If only a value from a single channel is to be processed, value must be converted. Refer to the following examples for more details.

- **Binary Operation (for C/C++)**

Mask Computing:

Mask = 0000 0001 left-shift 3 bits => 0000 1000

Mask = Invert Mask => 1111 0111

Mask Off for Bit N:

Mask = ~ (1 << N)

Result = Data & Mask

Mask Off for Bit 3

Data	1010	1010
AND	1111	0111
Result	1010	0010

Mask Computing:

Mask = 0000 0001 left-shift 3 bits => 0000 1000

Set On for Bit N :

Mask = 1 << N

Result = Data | Mask

Set On for Bit 3

Data	1010	0010
OR	0000	1000
Result	1010	1010

Get Status of Bit 3 (Data = 1010 1010)

Data right-shift 3 bits = 0001 0101

Data Mask Off with 0000 0001

0001 0101 & 0000 0001 = 0000 0001

Get Status of Bit N:

Result = (Data >> N) & 1

Get Status of Bit 3

Data	1010	0010
Shift	0001	0101
AND	0000	0001
Result	0000	0001

- **Binary Operation (for VB)**

Mask Computing:

Mask = 0000 0001 left-shift 3 bits => 0000 1000

Mask = Invert Mask => 1111 0111

Mask Off for Bit N :

Mask = Not (2 ^ N)

Result = Data and Mask

Mask Off for Bit 3

Data	1010	1010
AND	1111	0111
Result	1010	0010

Mask Computing:

Mask = 0000 0001 left-shift 3 bits => 0000 1000

Set On for Bit N:

Mask = 2 ^ N

Result = Data or Mask

Set On for Bit 3

Data	1010	0010
OR	0000	1000
Result	1010	1010

Get Status of Bit 3 (Data = 1010 1010)

Data right-shift 3 bits = 0001 0101

Data Mask Off with 0000 0001

0001 0101 & 0000 0001 = 0000 0001

Get Status of Bit N:

Result = (Data \ (2 ^ N)) and 1

Get Status of Bit 3

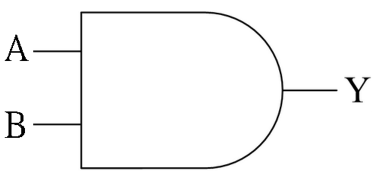
Data	1010	0010
Shift	0001	0101
AND	0000	0001
Result	0000	0001

Appendix 1: Binary Transfer Hexadecimal

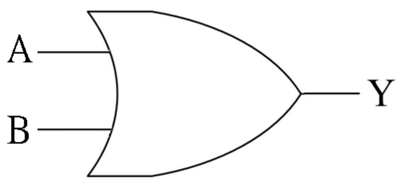
Binary	Hexadecimal	Binary	Hexadecimal
0 0 0 0	0x0	1 0 0 0	0x8
0 0 0 1	0x1	1 0 0 1	0x9
0 0 1 0	0x2	1 0 1 0	0xA (10)
0 0 1 1	0x3	1 0 1 1	0xB (11)
0 1 0 0	0x4	1 1 0 0	0xC (12)
0 1 0 1	0x5	1 1 0 1	0xD (13)
0 1 1 0	0x6	1 1 1 0	0xE (14)
0 1 1 1	0x7	1 1 1 1	0xF (15)

Appendix 2: Binary Operation (Bitwise)

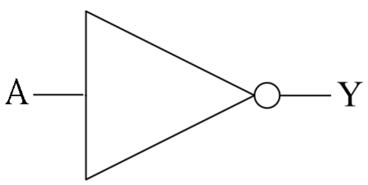
AND	0	1
0	0	0
1	0	1



OR	0	1
0	0	1
1	1	1



NOT	
0	1
1	0



XOR	0	1
0	0	1
1	1	0