

## Data Array Formats

When defining the data arrays in the FieldServer it is necessary to declare the data format of each of the data arrays to facilitate correct storage of the relevant data. The user should declare the data type of the data array in accordance to the data type of the data points being cached in the array. To facilitate the choice of data type, each of the data types available are described below.

- 1. Float. -** Data Array Format used to store Floating Point Analog values. (e.g. temperature, volts ). Each point in the array represents one 32 bit Floating Point value.
- 2. Bit.-** Format for storing Binary Data. Each point in the array represents one bit.
- 3. Packed\_Bit-** The Packed Bit format is used for extracting Binary values from 16 Bit registers, or inversely for creating 16 bit registers from Binary values. Using this data format, the server side of the FieldServer is able to provide 16 bit registers to the server side driver from bit data points on the client side, or vice versa. Each side of the FieldServer (client/server) merely views the data array as having a different format, and hence a different length. The length specified in the data array definition must therefore be the number of bits in the array.
- 4. Byte -** Format for storing Bytes of data. Each point in the Array represents one Byte.
- 5. Packed\_Byte** – The Packed Byte format allows you to extract Bytes out of Registers, or inversely for creating 16 bit registers from Bytes. Using this data format, the server side of the FieldServer is able to provide 16 bit registers to the server side driver from byte data points on the client side, or vice versa. Each side of the FieldServer (client/server) merely views the data array as having a different format, and hence a different length. The length specified in the data array definition must therefore be the number of bytes in the array. In the diagram, the 2 bytes are moved into a 16 bit Unsigned Integer using Packed\_byte.
- 6. Swapped\_Byte** – The Swapped\_Byte format is similar to the Packed Byte Format. The Difference is the Byte order is reversed.
- 7. SInt16** – Signed 16 bit Integer. Range: -32768 to 32767, discrete. Each point in the array represents one integer.
- 8. UInt16** – Unsigned 16 bit Integer. Range: 0 to 65 535, discrete. Each point in the array represents one integer.
- 9. SInt32** – Signed 32 bit Integer. Range: -2147483648 to 2147483647, discrete. Each point in the array represents one integer.
- 10. UInt32** – Unsigned 32 bit Integer. Range: 0 to 4294967295, discrete. Each point in the array represents one integer.

### **Transparency of data formats through the FieldServer.**

In transferring data points from one protocol to another via the data arrays in the FieldServer, the intention would be to maintain the integrity of the data format unless specific requirements dictate otherwise (e.g. packed bit applications). Therefore in most cases the FieldServer will maintain the data type of the point being transferred through the FieldServer, regardless of what is defined for the data array. For example, if a point representing a bit data type is transferred into a data array of type Float, the value in the data array will be a 32 bit floating point value that will only take on the values of zero and one. This could be transferred from the data array to an integer point in another protocol, and the value will still only ever take on the values of zero and one despite the type conversions. This explains the necessity for packed\_bit and packed\_byte, which do not maintain the integrity of the data type as described above.