When the s1[p]= syntax is used, the right-hand side of the string assignment *is inserted before* the byte identified by p after p is clipped to 0 to n.

The subrange assignment just described for string variables is not supported when a text wave is the destination. To assign a value to a range of a text wave element, you will need to create a temporary string variable. For example:

```
Make/O/T tw = {"Red", "Green", "Blue"}
String stmp= tw[1]
stmp[1,2]="XX"
tw[1]= stmp;
```

```
Print tw[0],tw[1],tw[2]                  prints        Red GXXen Blue
```

The indices in these examples are byte positions, not character positions. See **Characters Versus Bytes** on page III-483 for a discussion of this distinction.

String Substitution Using $

Wherever Igor expects the literal *name* of an operand, such as the name of a wave, you can instead provide a string expression preceded by the $ character. The $ operator evaluates the string expression and returns the *value* as a *name*.

For example, the Make operation expects the name of the wave to be created. Assume we want to create a wave named wave0:

```
Make wave0              // OK: wave0 is a literal name.

Make $"wave0"           // OK: $"wave0" evaluates to wave0.

String str = "wave0"
Make str                // WRONG: This makes a wave named str.
Make $str               // OK: $str evaluates to wave0.
```

$ is often used when you write a function which receives the name of a wave to be created as a parameter. Here is a trivial example:

```
Function MakeWave(wName)
   String wName      // name of the wave

   Make $wName
End
```

We would invoke this function as follows:

```
MakeWave("wave0")
```

We use $ because we need a wave name but we have a string containing a wave name. If we omitted the $ and wrote:

```
Make wName
```

Igor would make a wave whose name is wName, not on a wave whose name is wave0.

String substitution is capable of converting a string expression to a *single* name. It can not handle multiple names. For example, the following will *not* work:

```
String list = “wave0;wave1;wave2”
Display $list
```

See **Processing Lists of Waves** on page IV-198 for ways to accomplish this.

See **Converting a String into a Reference Using $** on page IV-62 for details on using $ in a user-defined function.