# Chapter 3—strxxx Class Reference

The strxxx class gives your external components convenient ways to manipulate strings. Once your string is encapsulated inside the string class, it can be passed back and forth to OMNIS or have various string operations performed on it.

The string class is split into three real classes, each derived from a base class strxxx. You should not need to access the strxxx base class directly. Three classes are derived from strxxx: str15, str80, and str255. Each can hold the maximum number of characters as specified by the class name.

Characters in the string class are indexed using a range 1 to n. Index 0 is used to store the real length of the string.

## Member Functions strxxx Class

### strxxx::strxxx()

The strxxx class has various constructors called from the three derived classes.

### strxxx::assign()

|  |
| --- |
| void strxxx::assign( const strxxx& pAssignFrom ) |

Assigns one strxxx class to another.

1. **pAssignFrom** - The string to be copied into this object.

### strxxx::compare()

|  |
| --- |
| void strxxx::compare( const strxxx& pCompare ) |

Compares two strings, this string and the string passed.

1. **pCompare** - This string to compare against.

**return** - This function returns:   
 0 if the strings match.   
 1 if this string is greater than pCompare.   
 -1 if this string is less than pCompare.

### strxxx::concat()

|  |
| --- |
| void strxxx::concat( const strxxx& pNewString ) |

Concatenates two strings together.

1. **pNewString** - String to be concatenated on to this string.

### strxxx::concat()

|  |
| --- |
| void strxxx::concat( qchar pChar ) |

Concatenates a single character on to this string.

1. **pChar** - The character to be concatenated on to this string.

### strxxx::concat()

|  |
| --- |
| void strxxx::concat(const strxxx& pString1, const strxxx& pString2 )  void strxxx::concat(const strxxx& pString1, const strxxx& pString2, const strxxx& pString3 )  void strxxx::concat(const strxxx& pString1, const strxxx& pString2, const strxxx& pString3, const strxxx& pString4 ) |

Concatenates a group of strings together on to this string.

1. **pString1**- String 1 to be concatenated.
2. **pString2**- String 2 to be concatenated.
3. **pString3**- String 3 to be concatenated.
4. **pString4**- String 4 to be concatenated.

### strxxx::copy()

|  |
| --- |
| void strxxx::copy( const strxxx& pExtractFrom, qshort pStart, qshort pLen ) |

Copies a ranges of characters from the passed string, and uses them to set the contents of this.

1. **pExtractFrom** - The string to extract characters from.
2. **pStart** - The starting index in **pExtractFrom.**
3. **pExtractFrom** - The number of characters to copy from **pExtractFrom.**

### strxxx::cString()

|  |
| --- |
| qchar\* strxxx::cString() |

Returns the address of a c-style string. This function converts this string into a c-style string first. A c-style string uses a null terminator, character 0x0 to represent the end of the strings data.

1. **return** - The address to a c-style string.

### strxxx::deleet()

|  |
| --- |
| void strxxx::deleet( qshort pPos, qshort pLen ) |

Deletes a range of characters from a starting point in the string. (Note it is named deleet so it does not interfere with the default C++ delete.)

1. **pPos** - The starting index to delete from.
2. **pLen** - The number of characters to be deleted.

### strxxx::insert()

|  |
| --- |
| void strxxx::insert(const strxxx& pInsertString ,qshort pPos ) |

Inserts a string at an index position.

1. **pInsertString** - The string to be inserted.
2. **pPos** - The index at which to insert the string.

### strxxx::insert()

|  |
| --- |
| void strxxx::insert( qchar pInsertChar ,qshort pPos ) |

Inserts a single character at an index position.

1. **pInsertChar** - The character to be inserted.
2. **pPos** - The index at which to insert the character.

### strxxx::insertStr()

|  |
| --- |
| void strxxx::insertStr(const strxxx& pInsertString ) |

Searches the string for a ‘$’ and inserts a sub-string **pInsertString** replacing the ‘$’.

1. **pInsertString** - The string to be inserted.

### strxxx::insertStr0()

|  |
| --- |
| void strxxx::insertStr0( const strxxx& pInsertString ) |

Similar to strxxx::insertStr(), except that it searches for the character 0x0 instead of ‘$’.

**pInsertString** - The string to be inserted.

### strxxx::length()

|  |
| --- |
| qshort strxxx::length() |

Returns the length of the string stored in the object.

1. **returns** - The length of the string.

### strxxx::maxLength()

|  |
| --- |
| qshort strxxx::maxLength() |

Returns the maximum length that can be stored in the string.

1. **returns** - The maximum length of the string.

### strxxx::operator !()

|  |
| --- |
| qbool strxxx::operator ! () |

Test is this string is not empty.

1. **return** - Returns qtrue if the string contains some data.

### strxxx::operator != ()

|  |
| --- |
| qbool strxxx::operator !=(const strxxx& pCompare ) |

Compares two strings.

1. **return** - qtrue if the strings do not match and qtrue if the strings are the same.

### strxxx::operator [ ]()

|  |
| --- |
| qchar& strxxx::operator [ ] ( qshort pIndex ) |

Returns the character from the string at the passed index.

1. **pIndex** - The index to return a character from.
2. **return** - The character from index **[pIndex].**

### strxxx::operator <()

|  |
| --- |
| qbool strxxx::operator < ( const strxxx& pCompare ) |

Compares two strings.

1. **return** - qtrue if this string is less than **pCompare**.

### strxxx::operator <=()

|  |
| --- |
| qbool strxxx::operator <=( const strxxx& pCompare ) |

Compares two strings.

1. **return** - qtrue if this string is less than or equal to **pCompare**.

### strxxx::operator =()

|  |
| --- |
| void strxxx::operator = ( const strxxx& pNewString ) |

Assigns a string.

1. **pNewString** - Assigned **pNewString** to this string.

### strxxx::operator =(qniltype qnil)

|  |
| --- |
| void strxxx::operator =(qniltype qnil1) |

Sets the length of the string to 0.

### strxxx::operator ==()

|  |
| --- |
| qbool strxxx::operator ==( const strxxx& pCompare ) |

Compares two strings.

1. **return** - qtrue if the strings match and qfalse if the strings are different.

### strxxx::operator >()

|  |
| --- |
| qbool strxxx::operator >( const strxxx& pCompare ) |

Compares two strings.

1. **return** - qtrue if this string is greater than **pCompare**.

### strxxx::operator >=()

|  |
| --- |
| qbool strxxx::operator >=( const strxxx& pCompare ) |

Compares two strings.

1. **return** - qtrue if this string is greater than or equal to **pCompare.**

### strxxx::pos()

|  |
| --- |
| qshort strxxx::pos(const strxxx& pFind ) |

Looks for the string pFind inside this.

1. **pFind** - The string to search for.
2. **returns** - The index if the string is found. 0 is returned if the string is not found.

### strxxx::pos()

|  |
| --- |
| qshort strxxx::pos(qchar pFindChar ) |

Looks for the first occurrence of pFindChar inside this.

1. **pFindChar** - The character to search for.
2. **returns** - The index if the string is found. 0 is returned if the string is not found.

### strxxx::pString()

|  |
| --- |
| qchar\* strxxx::pString() |

Returns the address of a Pascal-style string. This function converts this string into a Pascal string first. A Pascal-style string uses the first byte of the string, index 0 as a length byte. The following characters, index 1 to n, are string data.

1. **return** - The address to a Pascal string.

### strxxx::repWith0()

|  |
| --- |
| void strxxx::repWith0() |

Replaces all ‘$’ characters with a 0x0 character.

### strxxx::upps()

|  |
| --- |
| void strxxx::upps() |

Converts this to uppercase.

### strxxx::uprCmp()

|  |
| --- |
| void strxxx::uprCmp( const strxxx& pCompare ) |

Performs a case-insensitive comparison.

1. **pCompare** - This string to compare against.
2. **return** - This function returns:   
    0 if the strings match.   
    1 if this string is greater than pCompare.   
    -1 if this string is less than pCompare.

## Member Functions str15 Class

### str15::str15()

|  |
| --- |
| str15::str15() |

Constructor for an empty str15 string class.

### str15::str15()

|  |
| --- |
| str15::str15( const str15& pCopyFrom ) |

Constructor for a new str15 object duplicating the contents of another str15 object.

**pCopyFrom** - The string to copy the initial value from.

### str15::str15()

|  |
| --- |
| str15::str15( const strxxx& pCopyFrom ) |

Constructor for a new str15 object duplicating the contents of another strxxx object.

1. **pCopyFrom** - The string to copy the initial value from to a maximum of 15 characters.

### str15::str15()

|  |
| --- |
| str15::str15(const void\* pData ) |

Constructor for a new str15 object setting an initial value.

1. **pData** - This must be a null-terminated, c-style string. The new string has stores a maximum of 15 characters.

### str15::str15()

|  |
| --- |
| str15::str15(qshort pLen, const void\* pData ) |

Constructor for a new str15 object setting an initial value.

1. **pLen**  - The number of characters to copy from pData.
2. **pData** - The source of the initial data for the new string.

### str15::str15()

|  |
| --- |
| str15::str15(qchar pChar ) |

Constructor for a new str15 object setting an initial value.

1. **pChar** - The initial value for the new string.

## Member Functions str80 Class

### str80::str80()

|  |
| --- |
| str80::str80() |

Constructor for an empty str80 string class.

### str80::str80()

|  |
| --- |
| str80::str80( const str80& pCopyFrom ) |

Constructor for a new str80 object duplicating the contents of another str80 object.

**pCopyFrom** - The string to copy the initial value from.

### str80::str80()

|  |
| --- |
| str80::str80( const strxxx& pCopyFrom ) |

Constructor for a new str80 object duplicating the contents of another strxxx object.

1. **pCopyFrom** - The string to copy the initial value from to a maximum of 80 characters.

### str80::str80()

|  |
| --- |
| str80::str80(const void\* pData ) |

Constructor for a new str80 object setting an initial value.

1. **pData** - This must be a null-terminated, c-style string. The new string has a maximum of 80 characters.

### str80::str80()

|  |
| --- |
| str80::str80(qshort pLen, const void\* pData ) |

Constructor for a new str80 object setting an initial value.

1. **pLen**  - The number of character to copy from pData.
2. **pData** - The source of the initial data for the new string.

### str80::str80()

|  |
| --- |
| str80::str80(qchar pChar ) |

Constructor for a new str80 object setting an initial value.

**pChar** - The initial value for the new string.

## Member Functions str255 Class

### str255::str255()

|  |
| --- |
| str255::str255() |

Constructor for an empty str255 string class.

### str255::str255()

|  |
| --- |
| str255::str255( const str255& pCopyFrom ) |

Constructor for a new str255 object duplicating the contents of another str255 object.

**pCopyFrom** - The string to copy the initial value from.

### str255::str255()

|  |
| --- |
| str255::str255( const strxxx& pCopyFrom ) |

Constructor for a new str255 object duplicating the contents of another strxxx object.

1. **pCopyFrom** - The string to copy the initial value from to a maximum of 255 characters.

### str255::str255()

|  |
| --- |
| str255::str255(const void\* pData ) |

Constructor for a new str255 object setting an initial value.

1. **pData** - This must be a null-terminated, c-style string. The new string has a maximum of 255 characters.

### str255::str255()

|  |
| --- |
| str255::str255(qshort pLen, const void\* pData ) |

Constructor for a new str255 object setting an initial value.

1. **pLen**  - The number of character to copy from pData.
2. **pData** - The source of the initial data for the new string.

### str255::str255()

|  |
| --- |
| str255::str255(qchar pChar ) |

Constructor for a new str255 object setting an initial value.

1. **pChar** - The initial value for the new string.

## Other Functions

### qlongToString()

|  |
| --- |
| void qlongToString(qlong pVal, strxxx& pString ) |

Converts a numeric value into a string value.

1. **pVal** - The number to convert.
2. **pString** - The string to receive the converted result.

### qrealToString()

|  |
| --- |
| void qrealToString(qreal pVal, qshort pDecimalPlace, strxxx& pString, qshort pSigDecimalPlace ) |

Converts a numeric value into a string value.

1. **pVal** - The number to convert.
2. **pDecimalPlace** - The number of decimal places to convert to.
3. **pString** - The string to contain the converted result.
4. **pSigDecimalPlace** - This is the number of significant digits the string is converted to if the decimal places passed is larger than or equal to 24.

### stringToQlong()

|  |
| --- |
| qbool stringToQlong(const strxxx& pString, qlong& pVal ) |

Converts a string into a numeric value.

1. **pString** - The string to convert.
2. **pVal** - The numeric result.
3. **returns** - qtrue if the string could be converted, and qfalse if the string could not be converted.

### stringToQreal()

|  |
| --- |
| qbool stringToQreal(const strxxx& pString, qreal& pVal, qshort& pDecimalPlace ) |

Converts a string into a numeric value.

1. **pString** - The string to convert.
2. **pVal** - The numeric result.
3. **pDecimalPlace**  - Returns the number of decimal the converted value has.
4. **returns** - qtrue if the string could be converted, and qfalse if the string could not be converted.

### lowC()

|  |
| --- |
| qchar lowC( qchar pChar ) |

Converts a single character to lowercase.

1. **pChar** - The character to be converted.
2. **returns** - The new lowercase character.

### uppC()

|  |
| --- |
| qchar uppC( qchar pChar ) |

Converts a single character to uppercase.

1. **pString** - The character to be converted.
2. **returns** - The new uppercase character.

### uppC()

|  |
| --- |
| void uppC( qchar\* pAddress, qlong pLen ) |

Converts a range of characters to uppercase.

1. **pAddress** - The address of a buffer of characters to be uppercased.
2. **pLen** - The number of characters to uppercase.

### uprCmp()

|  |
| --- |
| qshort uprCmp( qchar\* pAddress, qchar\* pAddress2, qlong pLen ) |

Performs a case insensitive comparison on two buffers for a specified length.

1. **pAddress1** - The address to a buffer of characters.
2. **pAddress2** - The address to a buffer of characters.
3. **pLen** - The number of characters to uppercase in both strings.
4. **return** - This function returns:   
    0 if the strings match.   
    1 if this string is greater than pCompare.   
    -1 if this string is less than pCompare.