

BSTR Wide String Creation

Macros

```
#define BSTR_CONTAINER(varname_, bufcount_)
    Create a BSTR container.

#define INITIALIZED_BSTR_CONTAINER(varname_, bufcount_, ...)
    Create an initialized BSTR container.

#define MAKE_BSTR(varname_, bufcount_)
    Declare a BSTR variable.

#define MAKE_INITIALIZED_BSTR(varname_, bufcount_, ...)
    Declare and initialize a BSTR variable.
```

Detailed Description

Create a BSTR of wide characters with automatic or static storage duration.

Macro Definition Documentation

◆ BSTR_CONTAINER

```
#define BSTR_CONTAINER( varname_,
                      bufcount_ )
```

Value:

```
INTERNAL_BSTR_CONTAINER__(varname_, (bufcount_) * sizeof(WCHAR))
```

Create a BSTR container.

The BSTR_CONTAINER macro creates a BSTR container on the stack frame (where it is uninitialized) or in static storage (where it is zero-initialized by default).

Parameters

varname_ Name of the container to be instantiated.

bufcount_ Size of the buffer, in wide characters, that must be large enough for the string to represent, including the null-terminating character.

◆ INITIALIZED_BSTR_CONTAINER

```
#define INITIALIZED_BSTR_CONTAINER( varname_,
                                    bufcount_,
                                    ... )
```

Value:

```
BSTR_CONTAINER(varname_, bufcount_) = { .prefix = { .length = ((bufcount_) - 1) * sizeof(WCHAR) }, .bstr = __VA_ARGS__ }
```

Create an initialized BSTR container.

Aim of the INITIALIZED_BSTR_CONTAINER macro is both the creation and the initialization of a BSTR container on the stack frame or in static storage.

Parameters

varname_ Name of the container to be instantiated.

bufcount_ Size of the represented string, in wide characters, including the null-terminating character. This might not be the length of the initializer, but must meet the total length of the string before used.

... Variadic expression to initialize the string buffer.

This can be a string literal or brace-enclosed list of characters that fill the whole buffer.

This can be L"" or { 0 } to zero-initialize the buffer in order to copy characters into it later.

This can be a substring like L"ab" or { L'a', L'b' } to which remaining characters are appended later.

◆ MAKE_BSTR

```
#define MAKE_BSTR ( varname_,  
                  bufcount_ )
```

Value:

```
BSTR varname_;                                \  
do {                                         \  
    static BSTR_CONTAINER(bstr_container_##varname_, bufcount_); \  
    varname_ = bstr_container_##varname_.bstr;           \  
} while (0)
```

Declare a BSTR variable.

The MAKE_BSTR macro declares a BSTR variable in the current scope but restricts the visibility of the container implementation to the body block of a wrapping while-loop. The container object has static storage duration and is therefore zero-initialized.

For the description of the parameters, see [BSTR_CONTAINER\(\)](#).

◆ MAKE_INITIALIZED_BSTR

```
#define MAKE_INITIALIZED_BSTR ( varname_,  
                             bufcount_,  
                             ... )
```

Value:

```
BSTR varname_;                                \  
do {                                         \  
    static INITIALIZED_BSTR_CONTAINER(bstr_container_##varname_, bufcount_, __VA_ARGS__); \  
    varname_ = bstr_container_##varname_.bstr;           \  
} while (0)
```

Declare and initialize a BSTR variable.

The MAKE_INITIALIZED_BSTR macro declares a BSTR variable in the current scope but restricts the visibility of the container implementation to the body block of a wrapping while-loop. The container object has static storage duration and the variadic arguments are used to initialize it.

For the description of the parameters, see [INITIALIZED_BSTR_CONTAINER\(\)](#).