BSTR Byte String Creation

Macros

`#define BSTR_BYTE_CONTAINER(varname_, bufsize_)`	
	Create a BSTR container for binary data.
`#define INITIALIZED_BSTR_BYTE_CONTAINER(varname_, bufsize_, ...)`	
	Create an initialized BSTR container for binary data.
`#define MAKE_BSTR_BYTE(varname_, bufsize_)`	
	Declare a BSTR variable containing binary data.
`#define MAKE_INITIALIZED_BSTR_BYTE(varname_, bufsize_, ...)`	
	Declare and initialize a BSTR variable containing binary data.

Detailed Description

Create a BSTR for binary data with automatic or static storage duration.

Macro Definition Documentation

◆ BSTR_BYTE_CONTAINER

```
#define BSTR_BYTE_CONTAINER ( varname_,
                              bufsize_ )
```

Value:

```
INTERNAL_BSTR_CONTAINER__(varname_, bufsize_)
```

Create a BSTR container for binary data.

The BSTR_BYTE_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.

bufsize_ Size of the buffer, in bytes, that must be large enough for the data to represent, including the null-terminating character.

◆ INITIALIZED_BSTR_BYTE_CONTAINER

```
#define INITIALIZED_BSTR_BYTE_CONTAINER ( varname_,
                                          bufsize_,
                                          ... )
```

Value:

```
BSTR_BYTE_CONTAINER(varname_, bufsize_) = { .prefix = { .length = (bufsize_) - 1 }, .bytestr = __VA_ARGS__ }
```

Create an initialized BSTR container for binary data.

Aim of the INITIALIZED_BSTR_BYTE_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.

bufsize_ Size of the represented data, in bytes, including the null-terminating character. This might not be the length of the initializer, but must meet the total length of the data 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 "" or { 0 } to zero-initialize the buffer in order to copy characters into it later.
This can be a substring like "ab" or { 'a', 'b' } to which remaining bytes are appended later.

◆ MAKE_BSTR_BYTE

```
#define MAKE_BSTR_BYTE ( varname_,
                         bufsize_ )
```

Value:

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

Declare a BSTR variable containing binary data.

The MAKE_BSTR_BYTE 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_BYTE_CONTAINER()**.

◆ MAKE_INITIALIZED_BSTR_BYTE

```
#define MAKE_INITIALIZED_BSTR_BYTE ( varname_,
                                     bufsize_,
                                     ... )
```

Value:

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

Declare and initialize a BSTR variable containing binary data.

The MAKE_INITIALIZED_BSTR_BYTE 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_BYTE_CONTAINER()**.