

symbol	equ	expression
symbol	=	expression
symbol	textequ	expression

The purpose of the "=" directive is to define symbols that have an integer (or single character) quantity associated with them. This directive does not allow real string or text operands. This is the primary directive you should use to create numeric symbolic constants in your programs

The **EQU** directive provides almost a superset of the capabilities of the "=" and TEXTEQU directives. It allows operands that are numeric text or string literal constants.

The **TEXTEQU** directive defines a text substitution symbol. The expression in the operand field must be a text constant delimited with the "<" and ">" symbols. Whenever MASM encounters the symbol within a statement it substitutes the text in the operand field for the symbol. Programmers typically use this equate to save typing or to make some code more readable:

```
COUNT = 600
```

```
array DWORD COUNT DUP(0)
```

```
COUNT = 5
```

```
mov al,COUNT
```

```
COUNT = 10
```

```
mov al,COUNT
```

```
COUNT = 100
```

```
mov al,COUNT
```

Re-define COUNT allowed

; AL = 5

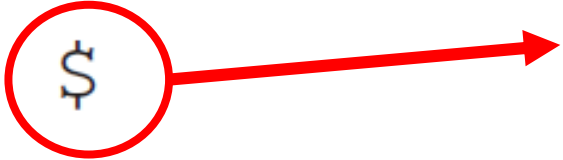
; AL = 10

; AL = 100

```
list BYTE 10,20,30,40
```

```
ListSize = ($ - list)
```

```
selfPtr DWORD $
```



The \$ operator (current location counter) returns the offset associated with the current program statement.

```
list BYTE 10,20,30,40
```

```
var2 BYTE 20 DUP(?)
```

```
ListSize = ($ - list)
```

```
myString BYTE "This is a long string, containing"
```

```
          BYTE "any number of characters"
```

```
myString_len = ($ - myString)
```

EQU vs TEXTEQU

- EQU is more general in that it allows numeric constants as well as text constants. EQU also explicitly states that a text value can be changed after declaration.
- TEXTEQU, on the other hand, only deals with text literals.
 - double quoted text,
 - literals preceded by % , and
 - the values of macros.

```
name EQU expression
name EQU symbol
name EQU <text>
```

Unlike the = directive, a symbol defined with EQU cannot be redefined in the same source code file.

```
matrix1 EQU 10 * 10
matrix2 EQU <10 * 10>
.data
M1 WORD matrix1
M2 WORD matrix2
```

The assembler produces different data definitions for **M1** and **M2**. The integer expression in **matrix1** is evaluated and assigned to **M1**. On the other hand, the text in **matrix2** is copied directly into the data definition for **M2**:

```
M1 WORD 100
M2 WORD 10 * 10
```

```
rowSize = 5  
count    TEXTEQU    %(rowSize * 2)  
move     TEXTEQU    <mov>  
setupAL  TEXTEQU    <move al,count>
```

Therefore, the statement

```
setupAL
```

would be assembled as

```
mov al,10
```

A symbol defined by TEXTEQU can be redefined at any time.