

## Preliminary Specification for Literals in Objective Modula-2

Document version: 1.01 -- Date: 2005-08-13 -- Author: BK

### 1) 7-bit ASCII Character Code Literals

Range checking for ASCII character code literals depends on whether or not code points in the range of ASCII 128 to ASCII 255 are allowed. The compiler directive `M2_STRING_COMPAT` will control the allowed range. By default `M2_STRING_COMPAT` is off and only 7-bit code points are accepted without warning, code points in the range of ASCII 128 to ASCII 255 will be accepted but generate a compiler warning and code points above ASCII 255 will generate a compiler error. If `M2_STRING_COMPAT` is turned on, 8-bit code points are accepted without warning. If `M2_STRING_COMPAT` is turned on, its state will not persist beyond the scope of the current compilation unit.

```
ascii-char = octal-digit { octal-digit } "C" .
```

Example: `137C` (\* underscore \*)

The use of ASCII code literals for code points above the 7-bit ASCII range (128-255) is discouraged, because it cannot be guaranteed what character will actually be produced without specifying an encoding scheme. Unichar literals should be used instead.

The resulting character will be of type `CHAR`.

### 2) Unicode Literals

```
unicode-char = digit { hex-digit } "U" .
```

Example: `20ACU` (\* Euro currency symbol \*)

The resulting character will be of a new built-in base type `UNICHAR`.

### 3) Quoted Character Literals

The compile time evaluation of character literals will depend on whether or not backslash escaping is turned on or off. The compiler directive `M2_STRING_COMPAT` will control the state of backslash escaping. By default `M2_STRING_COMPAT` is off and backslash escaping is enabled. If `M2_STRING_COMPAT` is turned on, backslash escaping is disabled. If `M2_STRING_COMPAT` is turned on, its state will not persist beyond the scope of the current compilation unit.

#### a) when backslash escaping is enabled (default)

```
character = single-quote non-single-quote-char single-quote .
```

```
non-single-quote-char = printable-single-quoted-char |  
                      backslash-escaped-char .
```

```
printable-single-quoted-char =  
    any printable character other than single-quote and backslash
```

```
backslash-escaped-char = backslash backslash-escape-code .
```

```
backslash-escape-code = octal-char-code | hex-char-code |
```

```

        uchar-code | control-char-code .

octal-char-code = octal-digit [ octal-digit [ octal-digit ] ] .

hex-char-code = "x" hex-digit [hex-digit] .

uchar-code = "u" hex-digit [hex-digit [hex-digit [hex-digit]]] .

control-char-code = "a" | "b" | "f" | "n" | "r" | "t" | "v" |
                    backslash | double-quote | single-quote .

single-quote = "'" .

backslash = "\" .

```

Examples: `'\t'` (\* horizontal tab \*)    `'\\'` (\* backslash \*)

The resulting character will be either of type **CHAR** or **UNICHAR**, depending on context:

If the literal is an escaped uchar-code (`'\unnnn'`) the resulting character will be of type **UNICHAR**. In all other cases the resulting character will be of type **CHAR** if the left side of the assignment is of type **CHAR**, and of type **UNICHAR** if the left side of the assignment is of type **UNICHAR**, otherwise the assignment will be illegal, unless of course **CAST** or **VAL** is used.

#### b) when backslash escaping is disabled

```

single-quoted-string = single-quote { non-single-quote-char }
                        single-quote .

non-single-quote-char =
    any printable character other than single-quote

single-quote = "'" .

```

Example: `'this is a string'`

## 4) String Literals

The compile time evaluation of character literals will depend on whether or not backslash escaping is turned on or off. The compiler directive `M2_STRING_COMPAT` will control the state of backslash escaping. By default `M2_STRING_COMPAT` is off and backslash escaping is enabled. If `M2_STRING_COMPAT` is turned on, backslash escaping is disabled. If `M2_STRING_COMPAT` is turned on, its state will not persist beyond the scope of the current compilation unit.

#### a) when backslash escaping is enabled (default)

```

double-quoted-string = double-quote { non-double-quote-char }
                        double-quote .

non-double-quote-char = printable-double-quoted-char |
                        backslash-escaped-char .

printable-double-quoted-char =
    any printable character other than double-quote and backslash

backslash-escaped-char = backslash backslash-escape-code .

backslash-escape-code = octal-char-code | hex-char-code |

```

```

        unichar-code | control-char-code .

octal-char-code = octal-digit [ octal-digit [ octal-digit ] ] .

hex-char-code = "x" hex-digit [hex-digit] .

unichar-code = "u" hex-digit [hex-digit [hex-digit [hex-digit]]] .

control-char-code = "a" | "b" | "f" | "n" | "r" | "t" | "v" |
                    backslash | double-quote | single-quote .

double-quote = '"' .

backslash = "\" .

```

Example: "this is a string with a linefeed\n"

### b) when backslash escaping is disabled

```

double-quoted-string = double-quote { non-double-quote-char }
                      double-quote .

non-double-quote-char =
    any printable character other than double-quote

double-quote = '"' .

```

Examples: "the backslash '\' is not treated as an escape symbol"

The resulting string will be assignment compatible with **ARRAY OF CHAR** and **ARRAY OF UNICHAR**.

## 5) Octal Number Literals

```

octal-number-literal = octal-digit { octal-digit } "B" .

octal-digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" .

```

Example: 136247B (\* octal number 136247<sub>8</sub> \*)

## 6) Decimal Number Literals

```

dec-number-literal = integer-number | real-number .

integer-number = digit { digit } .

real-number = digit { digit } "." digit { digit }
             [ "E" ["+" | "-"] digit { digit } ] .

digit = octal-digit | "8" | "9" .

```

Examples: 159800 (\* integer \*)    1.395E12 (\* real \*)

## 7) Hexadecimal Number Literals

```

hex-number-literal = digit { hex-digit } "H" .

```

hex-digit = digit | "A" | "B" | "C" | "D" | "E" | "F" .

Example: 1F0D7B0H (\* hexadecimal number 1F0D7B0<sub>16</sub> \*)

## 8) Binary Coded Decimal Number Literals

bcd-number-literal = digit { [ group-separator ] digit }  
[ "." digit { [ group-separator ] digit } ] "D".

group-separator = "`" .

Examples: 1`458`208`576D (\* integer BCD \*)    0.049`657`21D (\* real BCD \*)

END OF DOCUMENT