Visual SlickEdit – Modern Editor Theme for OS/2

Version

0.6

Table of Contents

Editor Status on the OS/2 platform..1
Goal of the Modern Editor Theme for OS/2...........................1
User Customizable Controls..3
Implementation Review..5
    Colour Scheme Configuration File..5
    Colour Coding Scheme Configuration File......................................5
Visual SlickEdit Modern Theme for OS/2 “Look'n'Feel”.........5
    Sample main() Declaration...6
    Sample #INCLUDE and #DEFINE Declaration...............................6
Colour Scheme Options...7
Coloured Words Configuration...8
    Keywords...8
    CS keywords...9
    PP keywords...9
    Symobl 1...9
    Symbol 2...9
    Symbol 3...9
    Symobl 4...9
Colour Settings..9
#define Pragma Usage Discussion...9
OS/2 Development Toolkit Integration..................................10
    Sample KEYWORD / SYMBOL Entry..11
    SOM vs Non-SOM Definitions...11
IBM C/C++ Compiler Toolset Integration.............................12
    Sample KEYWORD / SYMBOL Entry..12
GCC Integration...12
Installation...13
    File Locations..13
    SOM vs non-SOM...13

Future Enhancements..13

Visual SlickEdit Programmer's Editor

Editor Status on the OS/2 platform

Visual SlickEdit is a premiere Programmer's Editor. The product has a long history, it was available on the OS/2 platform between 1993 – 1998. In February of 1999 MicroEdge Inc. released the final patch for the product on the OS/2 platform, bringing it to version 4.0b. No further upgrades and/or patches have been released since.

In the meantime, the product itself has continued to evolve and is currently available across multiple platforms. The most recent product release level is SlickEdit 2020. Trial versions can be downloaded from here.

Goal of the Modern Editor Theme for OS/2

Utilized in many source code editors, or general applications for that matter, a concept of a User Presentation Space Theme is present in Visual SlickEdit as well. The 'Color Scheme' functionality in SlickEdit allows you, the end user, to adjust various colour mappings. Additional parts of the editor's configuration allow you to set fonts, certain key mappings, and so forth. However, due to SlickEdit's age, these remain individual settings which are rather unwieldly to deal with. If the default settings are not acceptable, the changes require detailed manual configuration steps for each programming language one intends to utilize.

Therefore, the concept of a Theme is of particular interest when it comes to an implementation within a programmer's editor. After all, such an editor will most likely be utilized to support source code writing in multiple languages. Further on, the various programming languages utilize their own (and specific) syntax, which at times may make it difficult to interpret the logic behind the source code when one simply looks at a stream of flat text. Ultimately, Theme implementation may provide not just a visually more appealing interface, but it can in fact make the programmer more productive by simplifying and speeding up the manipulation of the source code.

In the case of the latest publicly available release of Visual SlickEdit for OS/2, there are several built-in themes present in the editor itself. These, while being incredibly configurable due to SlickEdit's flexiblity, are primarily limitted to font and colour substitutions.

Further on, Visual SlickEdit's ability to support (syntax recognition, etc.) the OS/2 APIs is limitted to a pre-defined subset of such APIs. This once again is entirely driven by the product's age and the published API information at the time the last version of SlickEdit was available for the OS/2 platform. What this means is that as the platform continued to change and new libraries, and therefore APIs and programming elements, became available on OS/2, they are not readily recognized within SlickEdit's environment.

Therefore, the goal of 'Visual SlickEdit - Modern Editor Theme for OS/2' is to close some of these existing gaps by introducing:

1) modern syntax highlighting for the C/C++ language

2) recognition of the wider OS/2 Development Toolkit APIs and elements

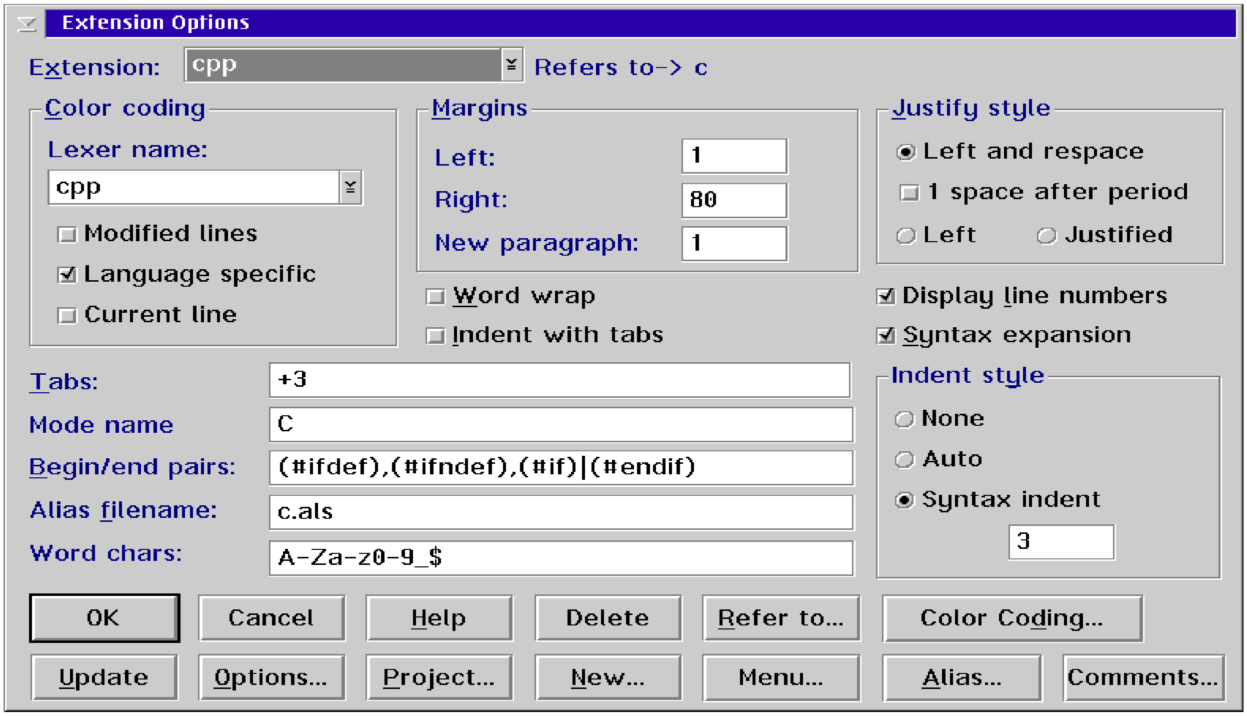
3) packaging of the individual SlickEdit's configuration options into a pre-made set of application configuration files

Visual SlickEdit's Configuration Options

User Customizable Controls

This section provides a high-level summary of the SlickEdit's built-in configuration options. This is done primarily so that as specific changes are highlighted later on, the reader has a good understanding of what changes are being made. No specific application navigation paths are provided given the assumption that you are familiar enough with the editor to otherwise implement the required configuration manually.

A great starting point is the 'Extension Options' configuration screen in SlickEdit, see Fig 2.1 below:



Extension Options

Extension: cpp Refers to-> c

Color coding
Lexer name: cpp
☐ Modified lines
☑ Language specific
☐ Current line

Margins
Left: 1
Right: 80
New paragraph: 1

Justify style
● Left and respace
☐ 1 space after period
○ Left ○ Justified

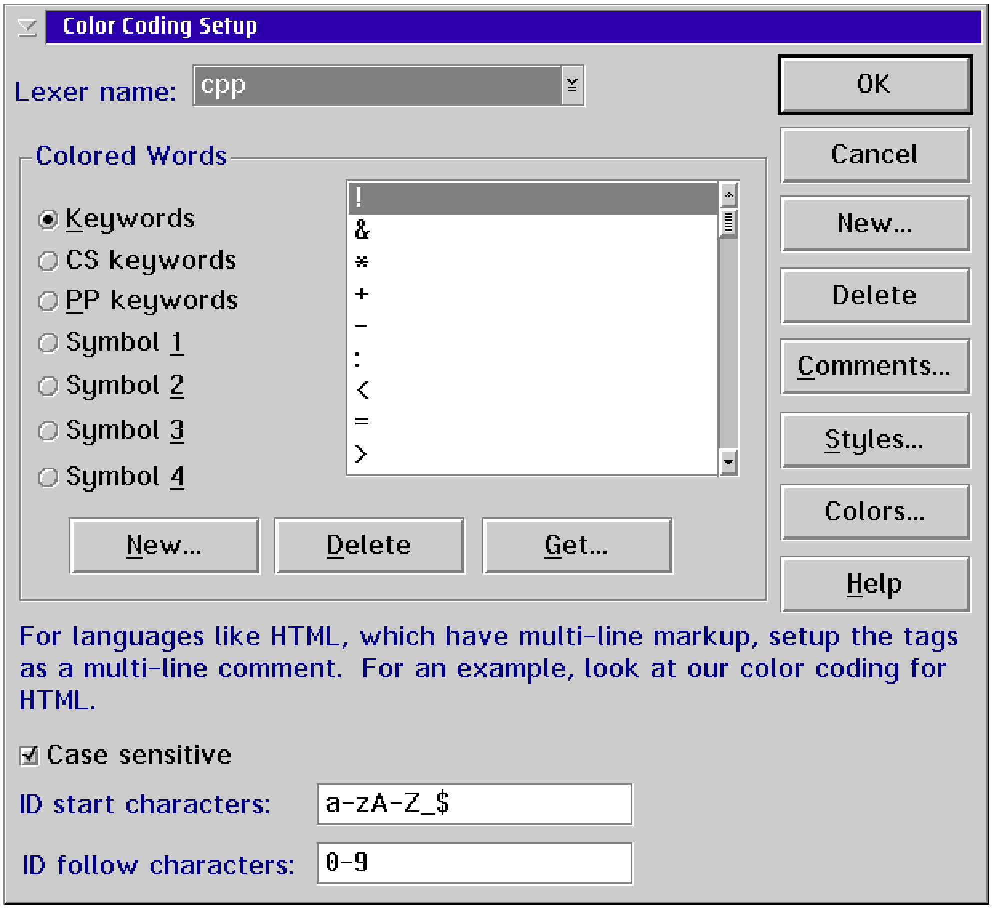
☐ Word wrap
☐ Indent with tabs
☑ Display line numbers
☑ Syntax expansion

Tabs: +3
Mode name: C
Begin/end pairs: (#ifdef),(#ifndef),(#if)|(#endif)
Alias filename: c.als
Word chars: A-Za-z0-9_$

Indent style
○ None
○ Auto
● Syntax indent
3

FIG 2.1 – Extension Options screen

All of the options listed above are of course configurable, however for our purposes we will be looking at implementing the changes applicable to the editor functionality controlled through 'Color Coding...' button, see Fig 2.2 below:



Lexer name: cpp

● Keywords

For languages like HTML, which have multi-line markup, setup the tags as a multi-line comment. For an example, look at our color coding for HTML.

☑ Case sensitive

ID start characters: a-zA-Z_$

ID follow characters: 0-9

FIG 2.2 – Color Coding Setup screen

Specifically since we are only adjusting the C/C++ configuration one should look up the appropriate Lexer name, most likely both C and C++ are mapped to the same Lexer name, that being 'cpp', as shown in the above Figure.

Here we have the option to change the 'Colored Words' definitions, as well as the specific 'Colors' that are to be used. Once implemented, the 'Colored Words' definitions will map specific language keywords, structures and environment variables to a particular 'Colors...' definitions. This is the power of SlickEdit's user extendible configuration which enables us to play a bit of a catch-up game on the OS/2 platform.

The below section will review the changes implemented by the 'Visual SlickEdit - Modern Editor Theme for OS/2', how each of these editor elements is adjusted, and present a visual comparison of a sample source code (DISKIO utility) showcasing the source code rendering using both the default editor settings and those provided by the 'Visual SlickEdit - Modern Editor Theme for OS/2'.

Visual SlickEdit Modern Theme Changes

Implementation Review

The changes packaged into the 'Visual SlickEdit - Modern Editor Theme for OS/2' utilize the above mentioned editor customization controls. However, as opposed to spelling out the details that one must implement manually, the configuration is captured using a set of standard Visual SlickEdit configuration files. These files are the USER customization files, which are normally created by the editor whenever you manually change any of the default configurations. Therefore, it is important that you first check whether your installation already has these present. If they are, you may want to merge your existing changes along with the Modern Theme changes. If you do not, the Modern Theme changes will overwrite your previous customizations. Either way, backup would be prudent.

Colour Scheme Configuration File

The *VSSCHEME.INI* file is the default SlickEdit 'Color Scheme' file. The user defined changes are stored in the *USCHEME.INI* file.

Colour Coding Scheme Configuration File

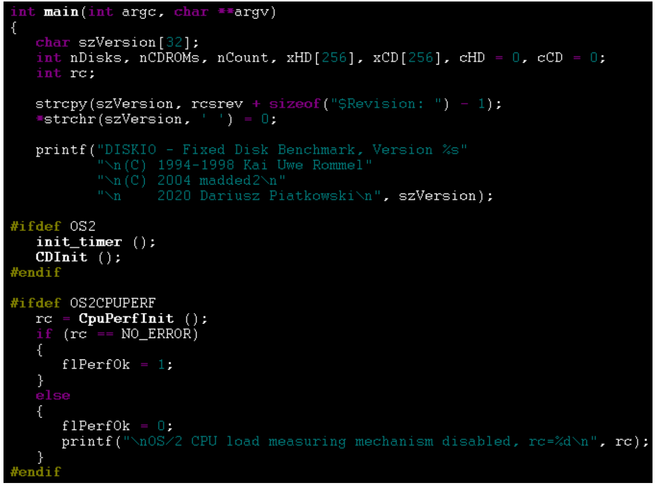
The *VSLICK.VLX* file is the default SlickEdit 'Color Coding Scheme' file. The user defined changes are stored in the *USER.VLX* file.

Visual SlickEdit Modern Theme for OS/2 “Look'n'Feel”

Following are a few examples of what a C/C++ source code rendering with the default SlickEdit configuration, as well as the modified 'Visual SlickEdit - Modern Editor Theme for OS/2' configuration, look like.

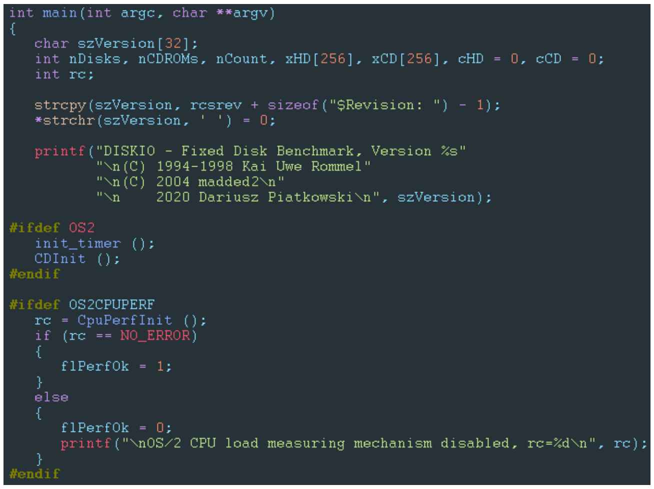
MODERN PROGRAMMER'S EDITOR THEME

Sample main() Declaration



```
int main(int argc, char **argv)
{
   char szVersion[32];
   int nDisks, nCDROMs, nCount, xHD[256], xCD[256], cHD = 0, cCD = 0;
   int rc;

   strcpy(szVersion, rcsrev + sizeof("$Revision: ") - 1);
   *strchr(szVersion, ' ') = 0;

   printf("DISKIO - Fixed Disk Benchmark, Version %s"
          "\n(C) 1994-1998 Kai Uwe Rommel"
          "\n(C) 2004 madded2\n"
          "\n    2020 Dariusz Piatkowski\n", szVersion);

#ifdef OS2
   init_timer ();
   CDInit ();
#endif

#ifdef OS2CPUPERF
   rc = CpuPerfInit ();
   if (rc == NO_ERROR)
   {
      flPerfOk = 1;
   }
   else
   {
      flPerfOk = 0;
      printf("\nOS/2 CPU load measuring mechanism disabled, rc=%d\n", rc);
   }
#endif
```

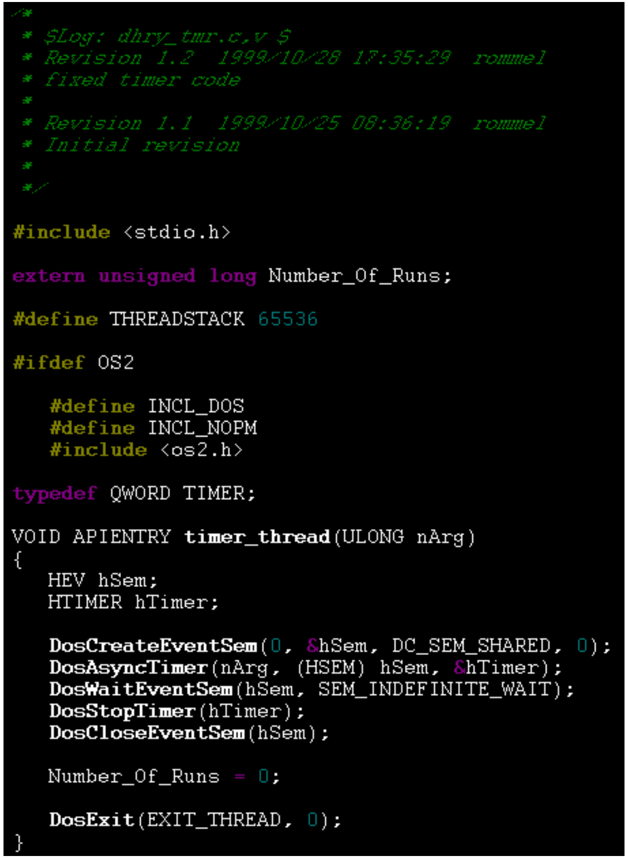
FIG 3.1 – DEFAULT View



```
int main(int argc, char **argv)
{
   char szVersion[32];
   int nDisks, nCDROMs, nCount, xHD[256], xCD[256], cHD = 0, cCD = 0;
   int rc;

   strcpy(szVersion, rcsrev + sizeof("$Revision: ") - 1);
   *strchr(szVersion, ' ') = 0;

   printf("DISKIO - Fixed Disk Benchmark, Version %s"
          "\n(C) 1994-1998 Kai Uwe Rommel"
          "\n(C) 2004 madded2\n"
          "\n    2020 Dariusz Piatkowski\n", szVersion);

#ifdef OS2
   init_timer ();
   CDInit ();
#endif

#ifdef OS2CPUPERF
   rc = CpuPerfInit ();
   if (rc == NO_ERROR)
   {
      flPerfOk = 1;
   }
   else
   {
      flPerfOk = 0;
      printf("\nOS/2 CPU load measuring mechanism disabled, rc=%d\n", rc);
   }
#endif
```

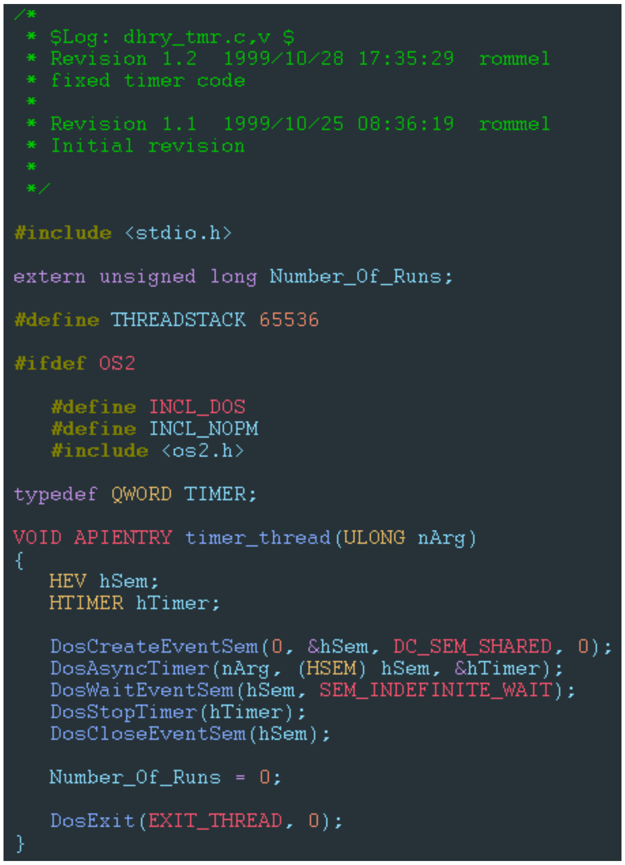
FIG 3.2 – Modern Theme View

Sample #INCLUDE and #DEFINE Declaration



```
/*
 * $Log: dhry_tmr.c,v $
 * Revision 1.2  1999/10/28 17:35:29  rommel
 * fixed timer code
 *
 * Revision 1.1  1999/10/25 08:36:19  rommel
 * Initial revision
 *
 */

#include <stdio.h>

extern unsigned long Number_Of_Runs;

#define THREADSTACK 65536

#ifdef OS2

   #define INCL_DOS
   #define INCL_NOPM
   #include <os2.h>

typedef QWORD TIMER;

VOID APIENTRY timer_thread(ULONG nArg)
{
   HEV hSem;
   HTIMER hTimer;

   DosCreateEventSem(0, &hSem, DC_SEM_SHARED, 0);
   DosAsyncTimer(nArg, (HSEM) hSem, &hTimer);
   DosWaitEventSem(hSem, SEM_INDEFINITE_WAIT);
   DosStopTimer(hTimer);
   DosCloseEventSem(hSem);

   Number_Of_Runs = 0;

   DosExit(EXIT_THREAD, 0);
}
```

FIG 3.3 – DEFAULT View



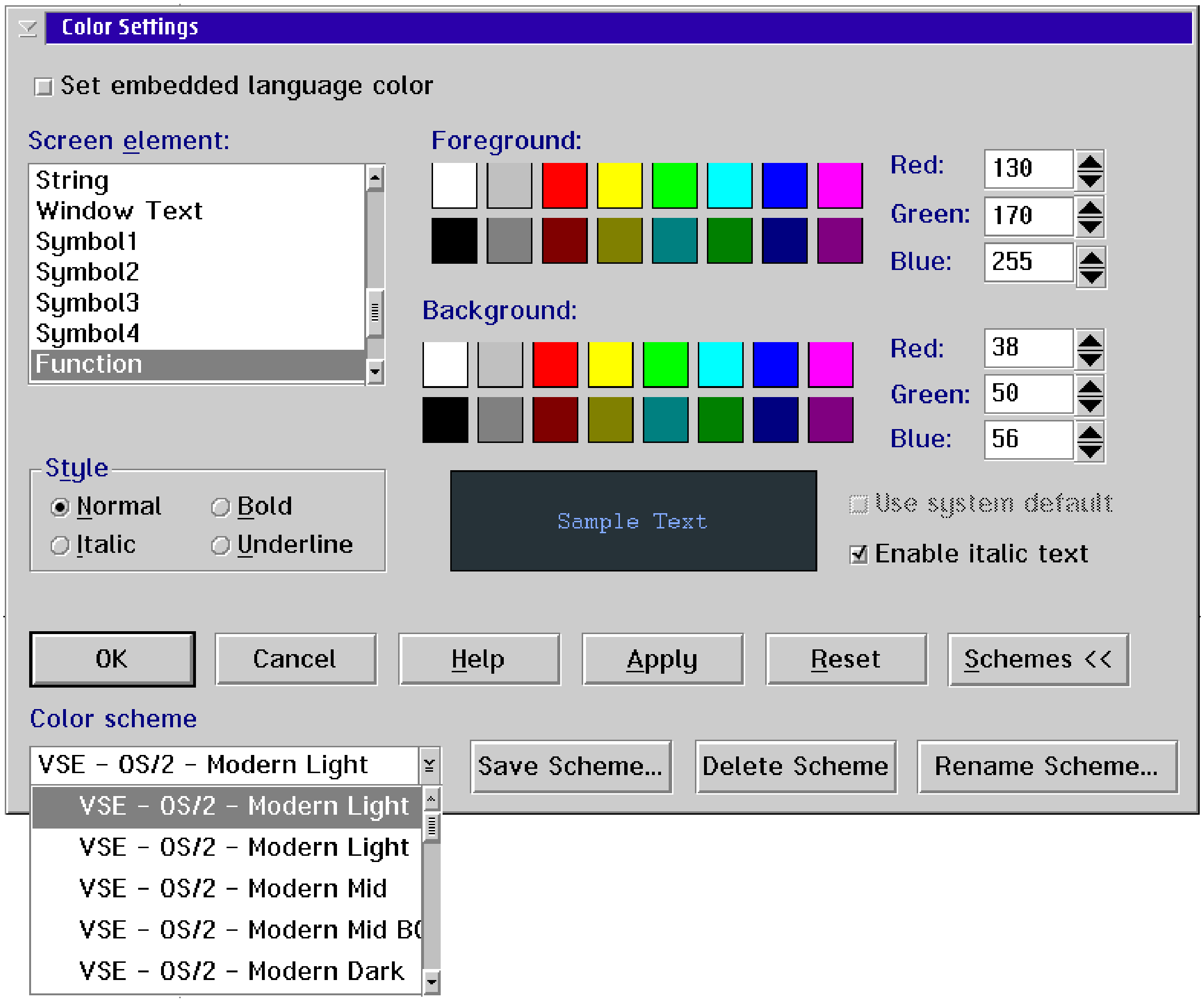
```
/*
 * $Log: dhry_tmr.c,v $
 * Revision 1.2  1999/10/28 17:35:29  rommel
 * fixed timer code
 *
 * Revision 1.1  1999/10/25 08:36:19  rommel
 * Initial revision
 *
 */

#include <stdio.h>

extern unsigned long Number_Of_Runs;

#define THREADSTACK 65536

#ifdef OS2

   #define INCL_DOS
   #define INCL_NOPM
   #include <os2.h>

typedef QWORD TIMER;

VOID APIENTRY timer_thread(ULONG nArg)
{
   HEV hSem;
   HTIMER hTimer;

   DosCreateEventSem(0, &hSem, DC_SEM_SHARED, 0);
   DosAsyncTimer(nArg, (HSEM) hSem, &hTimer);
   DosWaitEventSem(hSem, SEM_INDEFINITE_WAIT);
   DosStopTimer(hTimer);
   DosCloseEventSem(hSem);

   Number_Of_Runs = 0;

   DosExit(EXIT_THREAD, 0);
}
```

FIG 3.4 – Modern Theme View

Colour Scheme Options

The 'Visual SlickEdit – Modern Editor Theme for OS/2' includes several NEW colour schemes. These have been either custom developed, or were based on various on-line definitions (in such a case, a URL reference is provided in the pertinent section below).

There are a total of 15 new colour schemes, as shown below in Fig 3.5:



Color Settings

☐ Set embedded language color

Screen element:
String
Window Text
Symbol1
Symbol2
Symbol3
Symbol4
Function

Foreground:
Red: 130
Green: 170
Blue: 255

Background:
Red: 38
Green: 50
Blue: 56

Style
● Normal
○ Bold
○ Italic
○ Underline

☐ Use system default
☑ Enable italic text

Color scheme: VSE - OS/2 - Modern Light
VSE - OS/2 - Modern Light
VSE - OS/2 - Modern Light
VSE - OS/2 - Modern Mid
VSE - OS/2 - Modern Mid B(
VSE - OS/2 - Modern Dark

FIG 3.5 – Custom Colour Schemes

The naming convention follows a standard where the word “BOLD” appended to the end of the scheme name signifies all known language FUNCTION keywords being rendered in **BOLD** font style.

The following are Colour Schemes included in the 'Visual SlickEdit – Modern Editor Theme for OS/2' package:

MODERN PROGRAMMER'S EDITOR THEME

1) VSE – OS/2 – Modern Light
2) VSE – OS/2 – Modern Light BOLD
3) VSE – OS/2 – Modern Mid
4) VSE – OS/2 – Modern Mid BOLD
5) VSE – OS/2 – Modern Dark
6) VSE – OS/2 – Modern Dark BOLD
7) VSE – OS/2 – Dark
8) VSE – OS/2 – White (based on forum discussion)
9) VSE – OS/2 – White BOLD (based on forum discussion)
10) VSE – OS/2 – Borland Classic (based on forum discussion)
11) VSE – OS/2 – Borland Classic BOLD (based on forum discussion)
12) VSE – OS/2 – VIM (based on forum discussion)
13) VSE – OS/2 – NetBeans (based on forum discussion)
14) VSE – OS/2 – WekeRoad Ink BOLD (based on forum discussion)
15) VSE – OS/2 – Default

Note

The only differences in the 'VSE – OS/2 – Modern Light/Mid/Dark...' colour schemes is the different brightness of the background. Dark is nearly black, Mid is a bit brighter, Light is nearly greeniesh grey.

Coloured Words Configuration

There are several choices listed on this screen, see Fig 3.6 below. We will review each one and explain how these are used.

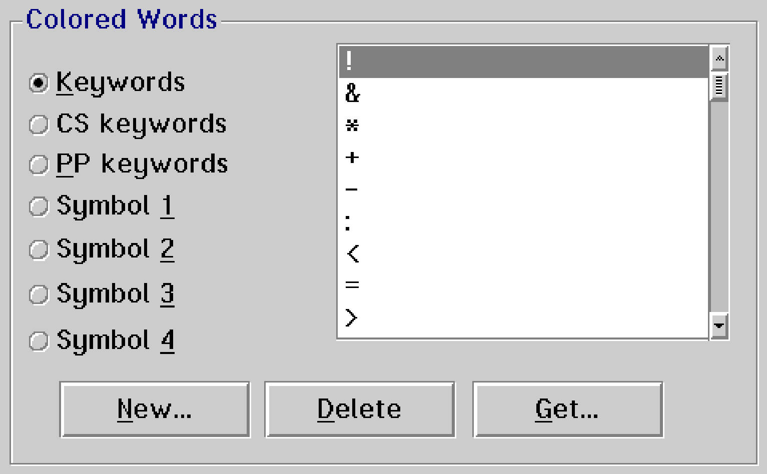


FIG 3.6 – Colored Words screen

Keywords

These are language specific keywords which are going to use a particular colour.

CS keywords

These are language specific keywords that are Case Sensitive, they always remain case sensitive and will use a particular colour.

PP keywords

These are language specific keywords that are Pre-Processor constructs, and will use a particular colour.

Symobl 1

These are language specific keywords that are mapped to the OS/2 Development Toolkit Macro definitions, and will use a particular colour.

Symbol 2

These are language specific keywords that are intended to display using an easily identifiable colour.

Symbol 3

These are language specific keywords that are mapped to the OS/2 Development Toolkit **DEFINE** definitions, and will use a particular colour.

Symobl 4

These are language specific keywords that are mapped to the OS/2 Development Toolkit Macro **TYPEDEF** or **STRUCT** definitions, and will use a particular colour.

Colour Settings

The colour settings implemented by the 'Visual SlickEdit - Modern Editor Theme for OS/2' are stored in a SlickEdit 'Color scheme' named “VSE – OS/2 – Modern Dark”.

Beyond the changes introduced by the 'Visual SlickEdit - Modern Editor Theme for OS/2', this scheme can be further modified as required.

#define Pragma Usage Discussion

The ***#define*** pragma can be used in a multitude of ways to simplify code development. While there are no hard-set rules to how one should do so (compare this to the concept of well recognized coding styles), there are a couple of points worth noting as these specifically pertain to how 'Visual SlickEdit - Modern Editor Theme for OS/2' classifies the ***#define*** statement.

Specifically, the following three distinct definitions are considered:

1) STATIC – this would be a regular #define which simply declares a static value, such as:

   #define G_IMPORTANT_VALUE 999

   These would be classified as SYMBOL3, which is a STATIC value.

2) MACRO – this would be a #define which actually declares a macro, such as:

#define assert(expr) ((expr)? (void)0: _assert(#expr, __FILE__, __LINE__))

These would be classified as SYMBOL1, which is a MACRO.

3) PREDEFINED – this would be a #define which is specifically established by the Toolset, in which case there exist two such unique sets of definitions that are recognized by the IBM C/C++ Compiler Toolset:

1. IBM C and C++ Compilers Predefined Macros

__32BIT__ __64BIT__ _AIX _AIX32 _AIX41 _AIX43 __ANSI__ _ARCH_X86
_ARCH_486 _ARCH_PENTIUM2 _CHAR_SIGNED _CHAR_UNSIGNED
__COMPAT__ __DBCS__ __DDNAMES__ __TOS_AIX__ __TOS_OS2__
__TOS_WIN__ _WIN32 __WINDOWS__ __xlC__ __DEBUG_ALLOC__
__DIGRAPHS__ __DLL__ _DLL __EXTENDED__ _FP_INLINE_ __FUNCTION__
__HHW_INTEL__ __HHW_RS6000__ __HOS_AIX__ __HOS_OS2__ __HOS_WIN__
__IBMC__ __IBMCPP__ __IMPORTLIB__ _LONG_LONG __LONGDOUBLE80
__LONGDOUBLE128 _M_I386 _M_IX86 __MATH__ __MULTI__ _MT
__NO_DEFAULT_LIBS__ __OS2__ _POWER __SAA__ __SAAL2__ __SPC__
_STDCALL_SUPPORTED __STR__ __TEMPINC__ __THW_INTEL__
__THW_RS6000__ __TILED__

2. ISO/ANSI Standard Predefined Macro Names

__cplusplus __FILE__ __DATE__ __LINE__ __STDC__ __TIME__ __TIMESTAMP__

3. Both of these would be classified as SYMBOL1, which is a MACRO.

OS/2 Development Toolkit Integration

The updated colour scheme is not the only enhancement. The 'Visual SlickEdit - Modern Editor Theme for OS/2' also brings close integration with the officially available OS/2 Development Toolkit. Specifically what this means is that the various macros, TYPEDEFs, DEFINEs and STRUCTs that are created within the Toolkit header files are accounted for, and allow Visual SlickEdit to readily recognize such elements.

In combination with the colour scheme settings, this allows the programmer to easily pin-point code elements that are specific to the source being worked on, and which do NOT belong to the Toolkit itself. Such an approach enables quicker differentiation of project source code elements from the existing Toolkit elements, hopefully simplifying the task of both reading the existing code, as well as developing new code.

Note

What the above paragraph means is that all of the source files contained in the OS/2 Development Toolkit have been scanned and the applicable element definitions captured in the USER.VLX configuration file. The placement of each element definition is further detailed in the 'Colored Words Configuratoin' section of this document.

Sample KEYWORD / SYMBOL Entry

Each KEYWORD and/or SYMBOL in the control file is identified as belonging to this enhancement. Further on, specific comments are provided to highlight the intended use of such an entry.

Here is an example of the [cpp] language "SYMBOL1" section:

; OS2_DevToolkit Specific keywords - START

symbol1= ERRORIDERROR ERRORIDSEV FIELDOFFSET

; OS2_DevToolkit Specific keywords - END

You will notice that ALL new *keywords* and *symbol* entries which are part of this enhancement are wrapped by the "**OS2_DevToolkit...START**" and "**OS2_DevToolkit...END** " identifiers.

SOM vs Non-SOM Definitions

The OS/2 Development Toolkit header files contain numerous references to the SOM compiler generated definitions. This is a fairly sizeable list of elements and if one does not wish to get visibility to such programming constructs, these can be readily excluded from the *USER.VLX* file.

The control file SOM entries are appropriately marked up as follows:

; WPS - SOM Compiler Headers START here!

symbol1=....

; WPS - SOM Compiler Headers END here!

These can be safely removed, thus saving a good amount of Visual SlickEdit's runtime memory requirements since the non-SOM version of the configuration file is about 65% smaller in size. The 'Visual SlickEdit – Modern Editor Theme for OS/2' package already provides such a pre-configured solution. Please see the details for which *USER.VLX* configuration file to pick in the below 'Installation' section.

IBM C/C++ Compiler Toolset Integration

Similarly to the above mentioned integration of the OS/2 Development Toolkit, as of v0.6 of this enhancement the support for the IBM C/C++ Compiler Toolset is also provided.

In the case of the IBM C/C++ Toolset it must be noted that the IBM product comes with it's own set of base C/C++ header files, as well as the extensive header files which define the numerous IOC library classes. Both sets of header files have been scanned and added to the previously captured definitions, which were based on the OS/2 Development Toolkit header files.

This means that the 'Visual SlickEdit – Modern Editor Theme for OS/2' enhancement is a cumulative result of the Toolkit and Toolset specific implementations. Therefore, in order to differentiate the building blocks contributed by each specific toolset, the sections which are derived from each toolset are properly marked with a START and END comments. These comments are found in the USER.VLX configuration file, where the IBM C/C++ Compiler specific markers are shown below.

Sample KEYWORD / SYMBOL Entry

Each KEYWORD and/or SYMBOL in the control file is identified as belonging to this enhancement. Further on, specific comments are provided to highlight the intended use of such an entry.

Here is an example of the [cpp] language "SYMBOL1" section:

```
; IBMCPP_DevToolkit Specific keywords - START
symbol1 = assert _fsin _fcos _fptan _fpatan _facos _fasin
; IBMCPP_DevToolkit Specific keywords - END
```

You will notice that ALL new *keywords* and *symbol* entries which are part of this enhancement are wrapped by the "**IBMCPP_DevToolkit...START**" and "**IBMCPP_DevToolkit...END** " identifiers.

GCC Integration

As of v0.6 the standard GCC Development Toolkit Integration is not provided. What this means is that none of the header files that are usually present in a typical GCC installation, such as those that can be found in the *\usr\include** path, have been scanned and are recognized by the 'Visual SlickEdit – Modern Editor Theme for OS/2' package.

> **Note**
>
> The above does not mean that those Toolkit code elements won't be recognized by Visual SlickEdit, in fact many will. However, it does mean that if you are developing pure GCC programs, those unknown header file contructs will not have automatic formatting applied

to them. Therefore, from that perspective the 'Visual SlickEdit – Modern Editor Theme for OS/2' package does not provide a substantial amount of functionality.

Installation

The installation process is manual. This is largely because of the pre-requisite review of your existing Visual SlickEdit customizations, and the need to check whether the configuraiton files provided as part of the Modern Theme already exist in your installation.

File Locations

The files already mentioned in this document (see 'Implementation Review' section above) simply need to be copied from the ZIP file to the location where your Visual SlickEdit is installed. The SlickEdit program should of course be closed first, and must not executing during the deployment of the configuration files.

SOM vs non-SOM

If you do not wish to have the visibility to the SOM compiler header entries, you can pick the non-SOM version of the *USER.VLX* file.

There are two versions of this file included in the 'Visual SlickEdit – Modern Editor Theme for OS/2' package:

1) *USER.VLX* – this is the full listing, it includes the SOM compiler header file constructs

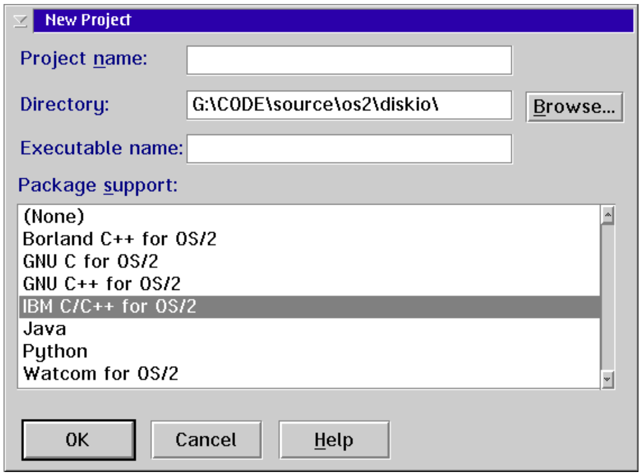
2) *USER.VLX.non-SOM* – this is the sub-set listing, it does NOT include the SOM compiler header file constructs

Simply pick the option you wish to utilize and deploy the matching file to your Visual SlickEdit installation path. If you chose the non-SOM version, the *USER.VLX.non-SOM* has to be renamed to *USER.VLX* .

Future Enhancements

Visual SlickEdit supports multiple programming languages. As such, given the changing OS/2 platform configuration, the existing editor functionality can be further enhanced to support either new programming languages, or different programming environments. In a sense, Visual SlickEdit can execute as a nearly full IDE (Integrated Development Environment).

Within SlickEdit this extensible configuration relies on the concept of 'Project', see Fig 3.7 below:



New Project

Project name:

Directory: G:\CODE\source\os2\diskio\

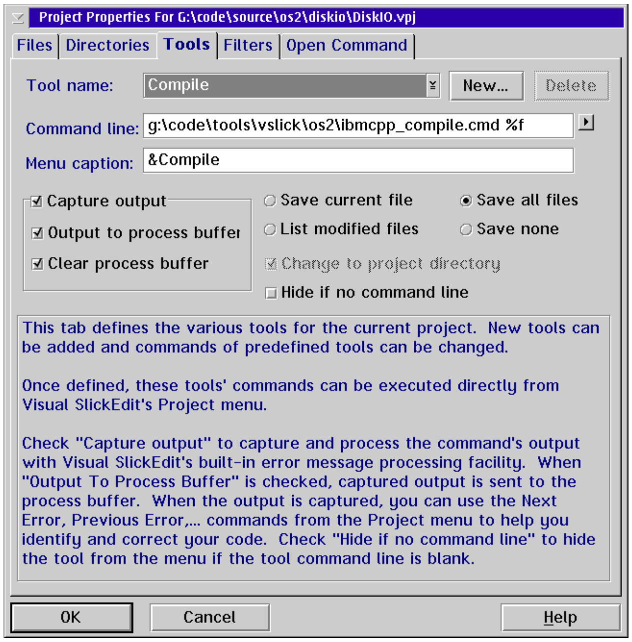
Executable name:

Package support:

- (None)
- Borland C++ for OS/2
- GNU C for OS/2
- GNU C++ for OS/2
- IBM C/C++ for OS/2
- Java
- Python
- Watcom for OS/2

FIG 3.7 – Visual SlickEdit Projects

Each project action, such as 'Compile', 'Build', or 'Build All' are explicitely defined in the 'Project Properties', see Fig 3.8 below:



Project Properties For G:\code\source\os2\diskio\DiskIO.vpj

Tool name: Compile

Command line: g:\code\tools\vslick\os2\ibmcpp_compile.cmd %f

Menu caption: &Compile

☑ Capture output
☑ Output to process buffer
☑ Clear process buffer

○ Save current file
● Save all files
○ List modified files
○ Save none
☑ Change to project directory
☐ Hide if no command line

This tab defines the various tools for the current project. New tools can be added and commands of predefined tools can be changed.

Once defined, these tools' commands can be executed directly from Visual SlickEdit's Project menu.

Check "Capture output" to capture and process the command's output with Visual SlickEdit's built-in error message processing facility. When "Output To Process Buffer" is checked, captured output is sent to the process buffer. When the output is captured, you can use the Next Error, Previous Error,... commands from the Project menu to help you identify and correct your code. Check "Hide if no command line" to hide the tool from the menu if the tool command line is blank.

FIG 3.8 – Project Properties

The next version of the 'Visual SlickEdit – Modern Editor Theme for OS/2' enhancement will contain the pre-defined Project configurations capable of supporting multiple OS/2 compilers. A working sample of matching CMD files will be provided as well. These will enable the execution of the specific Project functions, as detailed in the 'Project Properties' definition.

Document History

Revision	**Date**	**Author**	**Description**
0.1	September 21, 2021	Dariusz Piatkowski	Initial version of the document.
0.2	September 24, 2021	Dariusz Piatkowski	Corrected the project branding to consistently reference 'Visual SlickEdit – Modern Editor Theme for OS/2'.
0.3	September 27, 2021	Dariusz Piatkowski	- Added a separate section to detail the custom 'Color Scheme' choices - Added the 'Future Enhancements' section, which will deal with Project Support for multiple OS/2 compilers
0.4	September 28, 2021	Dariusz Piatkowski	Added some details on how the actual KEYWORDs are described in the USER.VLX file and how the OS/2 SOM compiler definitions are tracked within that file.
0.5	November 4, 2021	Dariusz Piatkowski	Imported the IOC – IBM Open Class Libraries.
0.6	November 6, 2021	Dariusz Piatkowski	Clarified the definitions of '#define', including examples of when a '#define' is considered to be a MACRO as opposed to being a static #define.