THE NEW FREEBASIC 8-BIT

PALETTE MACHINE

Version 1.1

For FreeBASIC v0.12 (beta version)/later
on MS-Windows 9x/ME/2000/NT/XP

Adigun Azikiwe Polack

PRESENTED TO YOU BY ADIGUN AZIKIWE POLACK.

---

*Feel absolutely free to use this library for your projects in FreeBASIC. If you do use ANY of these routines within this original library, then please give me credit for them, otherwise DO NOT EVEN USE THEM AT ALL, *period*. Thank you so rather much for your honest patronage and cooperation!! ;) !*

TABLE OF CONTENTS

4 - INTRODUCTION
6 - HOW TO INSTALL
8 - BLACK PALETTE ROUTINES
9 - SOLID COLOR PALETTE ROUTINES
11 - PALETTE FADING/ROTATION ROUTINES
  11 - *Supporting the 256-color* GFXlib 2 *default palette*
  49 - *Supporting custom 768-byte palettes*
  90 - *Supporting PixelPlus 256 (or PP256) palettes*
131 - PALETTE CROSSFADING ROUTINES
  131 - *Between a 256-color* GFXlib 2 *default palette and a 768-byte palette*
  139 - *Between TWO different 768-byte palettes*
  145 - *Between TWO different PixelPlus 256 (or PP256) palettes*
  151 - *Between a 256-color* GFXlib 2 *default palette and a PP256 palette*
  159 - *Between a 768-byte palette and a PP256 palette*
167 - COLOR ADJUST ROUTINE
168 - APPENDIX A: 768-BYTE PALETTES
170 - APPENDIX B: CONSTANTS
171 - APPENDIX C: PROGRAM EXAMPLE INDEX
172 - CHANGELOG
173 - FINAL WORDS

—— INTRODUCTION

Or, how this first-ever original palette library for FreeBASIC sprang into birth and being!!

HELLO, AND MY MOST BLESSED GREETINGS TO YOU ALL!!! **:D** *I so happily welcome you on purpose to the "***The New FreeBASIC 8-Bit Palette Machine***", the very newest and potentially the hottest incarnation of my "*8-Bit Palette Machine*" ever since its original QuickBASIC 4.5/7.1 debut for Future.Library that was first presented for the "QuickBASIC Caliber Programming Compo – Summer & Autumn 2003" from during the Summer of 2003!!*

What you are about to experience for yourself is the FIRST-EVER 8-Bit (or 256-color) Palette Library for FreeBASIC ever in existence, spanning to an awesomely jaw-dropping total of **129 PALETTE ROUTINES** *— including plenty of routines supporting 768-byte palettes and even PixelPlus 256 (or PP256) .PAL-based palette files as well — and *that* is in this very first version of this* alone*!! I'm a tell you right now, you QuickBASIC/QBasic fans just are gonna love and even enjoy the rather heck out of this one, believe me!!!* **^_-=b**

First of all, ***BIGGEST*** *Special Thanks go to Almighty Jehovah God for first inspiring me to do this one just to be a rather special blessing to the whole entire QB45/QB71/FB community and to YOU, the FreeBASIC (FB) programmer, too!!!* **d=^-^=b !!**

Here now is the real story of how this very first palette lib for FB got started.

*I was looking to implement some *real* good custom palette fading/manipulation routines for the wildly* ***AWESOME*** *QB-like graphics library by Angelo Mottola entitled* GFXlib 2 *for FreeBASIC. I tested several conversions of QB code sources based on fading in/out palettes and stuff, but they ended up failing so miserably due to either improper color fades and/or messed-up colors upon conversion to FB. Then, lo and* *behold, somewhere on the internet, I discovered the QB source code by P. Bindels about his palette fading routines (BIG thanks go to him for his awesome coding ideas to me here!!), and I was _so_ inspired then with the idea to just go right ahead and code in my custom palette routines here for* GFXlib 2 *for FreeBASIC!! So why not, hmmmm?* ***;*)***

*Also, BIG thanks go to Sterling Christensen for his awesome coding ideas on how to *actually* code a greyscale palette in FB as part of my original routines here, too!!*

*Last (but **CERTAINLY** not even least), thanks also to Richard Eric M. Lope (aka Relsoft) and to Chris Chadwick for the PP256 palette loading code presented in this original FreeBASIC lib, and also to Steve Nunnally of Acid Works Software for the 768-byte palette loading code that I have successfully implemented in this very lib, too!!!*

With all of that said......................................

Do enjoy this wonderful and original palette library for FreeBASIC!!! ^_-=b !!

WISHING YOU SUCH EXCELLENT SUCCESS AND MASTERPIECES,

Adigun Azikiwe Polack

Adigun Azikiwe Polack
Official Creator of "The New FreeBASIC 8-Bit Palette Machine"
March 9, 2005

HOW TO INSTALL

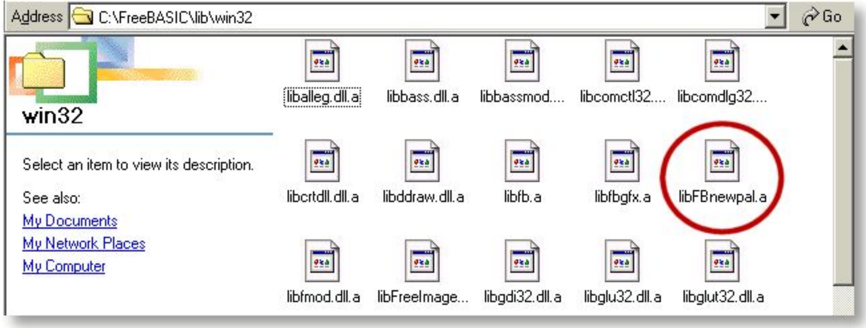
Or, here's some simple instructions on how to get this biggest 8-bit palette block party on!!

Here is how to set up *"The New FreeBASIC 8-Bit Palette Machine v1.1"* on FreeBASIC:

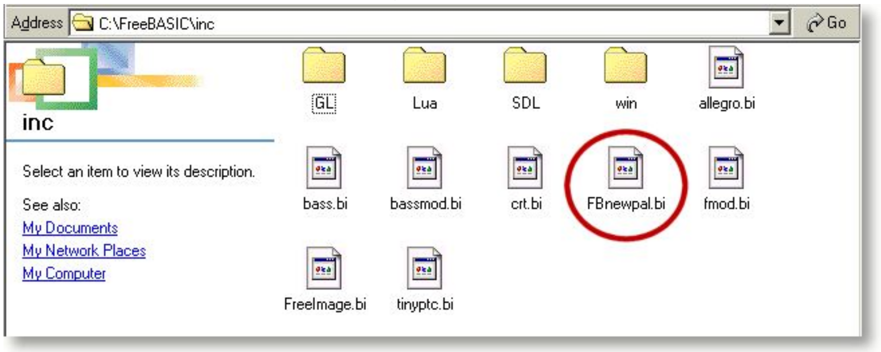
1. Extract all of your files from "*FBPMv1_1.zip*" into the main directory where FreeBASIC is located (ie *C:\FreeBASIC\*).

2. In the main directory where FreeBASIC is located, please run the batch file called "FBPM_createlib.bat". It will then create its own library file "*libFBnewpal.a*" for you to work for your current version of FreeBASIC (recommended FB version is ***0.12b or newer***!).

3. Now, please place the file "libFBnewpal.a" in the "lib\win32" directory *within* the main directory where FreeBASIC is located (ie *C:\FreeBASIC\lib\win32*).



4. Next, place the file "FBnewpal.bi" in the "inc" directory *within* the main directory where FreeBASIC is located (ie *C:\FreeBASIC\inc*).



5. To include this lib in your FB programs, just add this metacommand in at the beginning of your code:

```
'$include: "Fbnewpal.bi"
```

Now you are ready to rock your 256-color world in FreeBASIC!!! ;*) !

Wait a Sec!! Just one important note for you before getting started:

Recommendedly, this lib is best to be used in conjunction with Angelo Mottola's *GFXlib 2* for FB **ONLY**! It will not work using the Allegro game library, unfortunately. However, especially (and even mainly) using *GFXlib 2*, it **WILL** work on *any* 256-color (or 8-bit) graphics mode of your choice, fullscreen or windowed!!! d=^_^=b !

And now, onward we go *straight* into the main meat here: the 8-bit palette routines for this original FB library! HERE WE GO!!!

BLACK PALETTE ROUTINES

Or, routines to just throw the palette into total darkness!

ZeroPal

SUB DESCRIPTION:

```
Sub ZeroPal ()
```

NOTES ON THIS COMMAND:

When this command is called in your FreeBASIC programs, you can actually switch the entire palette to black INSTANTANEOUSLY!

ZeroPalRange

SUB DESCRIPTION:

```
Sub ZeroPalRange (StartColor, EndColor)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

NOTES ON THIS COMMAND:

When this command is called in your FreeBASIC programs, you can actually switch the color-order range within your screen to black INSTANTANEOUSLY! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one.*

SolidColorPal

SUB DESCRIPTION:

```
    Sub SolidColorPal (R, G, B)
```

R = the **red** shade level (0 to 63).

G = the **green** shade level (0 to 63).

B = the **blue** shade level (0 to 63).

NOTES ON THIS COMMAND:

When this command is called in your FreeBASIC programs, you can actually switch the entire palette to the color of your choice easily and INSTANTANEOUSLY!!

SolidColorPalRange

SUB DESCRIPTION:

```
    Sub SolidColorPalRange (StartColor, EndColor, R, G, B)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R` = the **red** shade level (0 to 63).

`G` = the **green** shade level (0 to 63).

`B` = the **blue** shade level (0 to 63).

NOTES ON THIS COMMAND:

When this command is called in your FreeBASIC programs, you can actually switch the color-order range within your screen to the color of your choice easily and INSTANTANEOUSLY! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one.*

PALETTE FADING/ROTATION ROUTINES

Or, how to teach your old 256-color palette the newest mind-blowing tricks and then some!!!

NOTE: THESE NEXT ROUTINES SUPPORT THE DEFAULT 256-COLOR GFXLIB 2 PALETTE ONLY!!

When these following commands are applied in your FB programs, the colors can **automatically** *change to the shades of the default 256-color GFXlib 2 palette, so please be *very* careful if you are using any custom 256-color palette(s) of your own choice! ;*) !*

FadeIn.DefaultPal

SUB DESCRIPTION:

```
    Sub FadeIn.DefaultPal (millisec)
```

`millisec` = the amount of milliseconds determining the speed of fading in from black to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade from black all the way to the default 256-color GFXlib 2 palette, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

FadeOut.DefaultPal

SUB DESCRIPTION:

```
    Sub FadeOut.DefaultPal (millisec)
```

`millisec` = the amount of milliseconds determining the speed of fading from the default *GFXlib 2* palette all the way to black.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade from the default 256-color GFXlib 2 palette all the way out to pitch blackness, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

FadeInX.DefaultPal

SUB DESCRIPTION:

```
    Sub FadeInX.DefaultPal (R.from, G.from, B.from, millisec)
```

`R.from` = the **red** shade level (0 to 63) to fade from.

`G.from` = the **green** shade level (0 to 63) to fade from.

`B.from` = the **blue** shade level (0 to 63) to fade from.

`millisec` = the amount of milliseconds determining the speed of fading in from the selected color to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

On this command, you can actually determine just how fast or slow you want the entire screen to fade from the color of your choice all the way to the default 256-color GFXlib 2 palette, all in ONE single pass! Higher milliseconds = slower fades; while lower milliseconds = faster fades.

FadeOutX.DefaultPal

SUB DESCRIPTION:

```
Sub FadeOutX.DefaultPal (R.to, G.to, B.to, millisec)
```

`R.to` = the **red** shade level (0 to 63) to fade to.

`G.to` = the **green** shade level (0 to 63) to fade to.

`B.to` = the **blue** shade level (0 to 63) to fade to.

`millisec` = the amount of milliseconds determining the speed of fading out the default *GFXlib 2* palette to the selected color.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade the default 256-color GFXlib 2 *palette right into the color of your choice, all in ONE single pass! Higher milliseconds = slower fades; while lower milliseconds = faster fades.*

FadeInRange.DefaultPal

SUB DESCRIPTION:

```
   Sub FadeInRange.DefaultPal (StartColor, EndColor, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading in from black to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from black all the way to the default 256-color GFXlib 2 *palette, all in ONE single pass! Useful for fading certain parts of the screen in, too!! Remember, for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeOutRange.DefaultPal

Sub Description:

```
Sub FadeOutRange.DefaultPal (StartColor, EndColor, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the default *GFXlib 2* palette all the way to black.

Notes on this Command:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from the default 256-color GFXlib 2 *palette all the way out to pitch blackness, all in ONE single pass! Useful for fading out certain parts of the screen, too!! Remember, for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeInRangeX.DefaultPal

SUB DESCRIPTION:

```
    Sub FadeInRangeX.DefaultPal (StartColor, EndColor, R.from, G.from, B.from,
     millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to fade from.

`G.from` = the **green** shade level (0 to 63) to fade from.

`B.from` = the **blue** shade level (0 to 63) to fade from.

`millisec` = the amount of milliseconds determining the speed of fading in from the selected color to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from your chosen color all the way to the default 256-color GFXlib 2 palette, all in ONE single pass! Useful for fading certain parts of the screen in, and for doing some real cool palette-lighting effects, too!! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeOutRangeX.DefaultPal

SUB DESCRIPTION:

```
Sub FadeOutRangeX.DefaultPal (StartColor, EndColor, R.to, G.to, B.to,
 millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.to` = the **red** shade level (0 to 63) to fade to.

`G.to` = the **green** shade level (0 to 63) to fade to.

`B.to` = the **blue** shade level (0 to 63) to fade to.

`millisec` = the amount of milliseconds determining the speed of fading out the default *GFXlib 2* palette to the selected color.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade the default 256-color GFXlib 2 palette right into the color of your choice, all in ONE single pass! Useful for fading certain parts of the screen in, and for doing some real cool palette-lighting effects, too!! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

Please turn to the very next page for an FB program example of this using the commands FadeInRangeX.DefaultPal and FadeOutRangeX.DefaultPal!!

PROGRAM EXAMPLE # 1:

(This example uses FadeInRangeX.DefaultPal *and* FadeOutRangeX.DefaultPal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from the default GFXlib 2 palette.
'--------------------------------------------------------------------------
For DrawPalette = 0 to 255
  Line (DrawPalette, 8)-(DrawPalette, 200), DrawPalette
Next


'Now, let's work some color fades, shall we?  ^_^ !
'--------------------------------------------------

Color 15, 1
Locate 1, 1: ? "Let's work some palette magic!!!"


'--- We fade color entries 20-100 of the default GFXlib 2 palette right to
'    yellow in a single pass, measuring 30 millisecs. per fade-step, and
'    then we fade it back again using that same color and speed!

FadeOutRangeX.DefaultPal 20, 100, 63, 63, 0, 30
FadeInRangeX.DefaultPal 20, 100, 63, 63, 0, 30


'--- Let's try it the same way with purple, except this time, we use color
'    entries 110-255 and with a faster fade speed of 15 millisecs. per
'    fade-step!

FadeOutRangeX.DefaultPal 110, 255, 63, 0, 63, 15
FadeInRangeX.DefaultPal 110, 255, 63, 0, 63, 15


'--- How 'bout some chillin' this time with a dash of blue using color
'    entries 0-158 and with a *much* slower fade speed of 70 milliseconds
'    per fade-step, hmmm?  ;)

FadeOutRangeX.DefaultPal 0, 158, 0, 0, 63, 70
FadeInRangeX.DefaultPal 0, 158, 0, 0, 63, 70


'--- Or get some red in and out with color entries 129-203 using a MUCH
'    faster fade speed of only 4 milliseconds/fade-step!

FadeOutRangeX.DefaultPal 129, 203, 63, 0, 0, 4
FadeInRangeX.DefaultPal 129, 203, 63, 0, 0, 4


'--- Finally, let's wrap this up now by fading the ENTIRE default GFXlib 2
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.DefaultPal 63
```

FadeCtrl.DefaultPal

Sub Description:

```
    Sub FadeCtrl.DefaultPal (FadeIn.Grade)
```

`FadeIn.Grade` = the custom fade-in level between a pitch-black palette and the default *GFXlib 2* palette.
(63 = fully faded in to palette; 0 = fully faded out to black)

Notes on this Command:

With this command, you have FULL and free control of all 64 of the fade levels between a black screen and the default 256-color GFXlib 2 palette! Perfect for doing fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;) !*

FadeCtrlX.DefaultPal

SUB DESCRIPTION:

```
   Sub FadeCtrlX.DefaultPal (R.from, G.from, B.from, FadeIn.Grade)
```

`R.from` = the **red** shade level (0 to 63) to use as your “fade-out”-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your “fade-out”-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your “fade-out”-based color shade.

`FadeIn.Grade` = the custom fade-in level between an entire palette of a resulting color here and the default *GFXlib 2* palette. (63 = fully faded in to palette; 0 = fully faded out to that chosen color)

NOTES ON THIS COMMAND:

With this command, you have FULL and free control of all 64 of the fade levels between a whole palette of the color of your choice and the default 256-color GFXlib 2 *palette! Perfect for doing such “color-to-default-palette”-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;*) !*

FadeCtrlRangeX.DefaultPal

SUB DESCRIPTION:

```
Sub FadeCtrlRangeX.DefaultPal (StartColor, EndColor, R.from, G.from, B.from,
 FadeIn.Grade)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to use as your “fade-out”-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your “fade-out”-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your “fade-out”-based color shade.

`FadeIn.Grade` = the custom fade-in level between a palette of a resulting color here and the default *GFXlib 2* palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to that chosen color)

NOTES ON THIS COMMAND:

*With this command, you again have FULL and free control of all 64 of the fade levels between the color of your choice and the default 256-color GFXlib 2 palette, except this time, it is for **any part of the palette**!!! Especially an awesome thing for doing such “color-to-default-palette”-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;*) ! Keep in mind though that the “EndColor” number **must** be higher than the “StartColor” one in order for this command to work the best!*

PalRotate.DefaultPal

SUB DESCRIPTION:

```
   Sub PalRotate.DefaultPal (StartColor, EndColor, Rotate.Level)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the entire default *GFXlib 2* palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

NOTES ON THIS COMMAND:

In this command, you can truly custom-rotate the many colors of the default GFXlib 2 *palette forwards or backwards, whether you do part of the palette, or even all of it! It is up to you!! :D Remember now that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalRotateFC.DefaultPal

SUB DESCRIPTION:

```
    Sub PalRotateFC.DefaultPal (StartColor, EndColor, Rotate.Level,
     FadeIn.Grade)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`FadeIn.Grade` = the custom fade-in level between a pitch-black palette and the default *GFXlib 2* palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to black)

NOTES ON THIS COMMAND:

In this command, not only can you truly custom-rotate the many colors of the default GFXlib 2 *palette forwards or backwards — whether you do part of the palette or even all of it — but also, you have FULL and free control of all 64 of the fade levels as well within your color range, too!! An awesomely great recommendation for working on your games and graphics demos, I must say!!!* ^-^=b ! *Remember, in order for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one!*

PalRotateFCX.DefaultPal

Sub Description:

```
   Sub PalRotateFCX.DefaultPal (StartColor, EndColor, R.from, G.from, B.from,
    Rotate.Level, FadeIn.Grade)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to use as your "fade-out"-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your "fade-out"-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your "fade-out"-based color shade.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`FadeIn.Grade` = the custom fade-in level between a palette of a resulting color here and the default *GFXlib 2* palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to that selected color)

Notes on this Command:

*Very much the same as PalRotateFC.DefaultPal, except that you are now allowed *as well* custom fades from any single color you wish to that entire palette, back again, and somewhere in-between, too!!! Now, be sure to try that in your games and graphics demos, as it will do you real good here!!* ^-^ ! *Remember, in order for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one!*

Please turn to the very next page for an FB program example of this using the commands PalRotate.DefaultPal, PalRotateFC.DefaultPal, and PalRotateFCX.DefaultPal!!

PROGRAM EXAMPLE #2:

(This example uses PalRotate.DefaultPal, PalRotateFC.DefaultPal, *and* PalRotateFCX.DefaultPal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries for the default GFXlib 2 palette.
'------------------------------------------------------------------------
For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, we will rotate the palettes around and around!!  ;)
'--------------------------------------------------------

Color 15, 4
Locate 1, 1: ? "Let's spin that GFXlib 2 palette wheel!!"

Color 15, 0
Locate 2, 1: ? "- Now using PalRotate.DefaultPal... -"


'--- Using the color entries 20-235 of the default GFXlib 2 palette, we
'    rotate the colors of the entire palette forward for a short moment,
'    and then backwards for that very same moment.

Rota = 0
DO
  PalRotate.DefaultPal 20, 235, Rota
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotate.DefaultPal 20, 235, Rota
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "- Now using PalRotateFC.DefaultPal... -"


'--- Next, we use the same color entries of the default GFXlib 2 palette
'    to once again rotate the colors of the entire palette forward for a
'    short moment, and then backwards for that same moment.  BUT, we do
'    it now with fade-ins/fade-outs as you will see right here!  ;*)
```

Continues on next page.......

“Program Example #2” continued from last page........

```
Rota = 0
DO
  PalRotateFC.DefaultPal 20, 235, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFC.DefaultPal 20, 235, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "- Now using PalRotateFCX.DefaultPal... -"


'--- And now, we do the same thing as the second one here, only this time,
'    with color-based fade-ins/fade-outs that will interest you real
'    good now, so get psyched for this one!!  :D

Rota = 0
DO
  PalRotateFCX.DefaultPal 20, 235, 54, 10, 32, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFCX.DefaultPal 20, 235, 10, 59, 27, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Rota = 0
DO
  PalRotateFCX.DefaultPal 20, 235, 5, 6, 63, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFCX.DefaultPal 20, 235, 63, 63, 63, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "This now concludes the test.  Thank you!"


'--- We now end this test indeed, as always, by fading out the palette
'    to nothing and exiting this program.  ^_^=b !

FadeOut.DefaultPal 63

'--- Good night!  z_z
```

PalNeg.DefaultPal

SUB DESCRIPTION:

```
Sub PalNeg.DefaultPal (Switch)
```

`Switch` = the operation of whether or not to switch the screen to a negative (or inverted-colors) version of the default *GFXlib 2* palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts the palette back to the normal *GFXlib 2* default one automatically.

NOTES ON THIS COMMAND:

With this command, you can actually switch the screen to a "negative"-based effect of the default GFXlib 2 *palette* *INSTANTANEOUSLY!*

PalNegRange.DefaultPal

SUB DESCRIPTION:

```
Sub PalNegRange.DefaultPal (StartColor, EndColor, Switch)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Switch` = the operation of whether or not to switch the selected order of colors to a negative (or inverted-colors) version of original colors of the default *GFXlib 2* palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the order indeed becomes switched like that, otherwise this command reverts the specified color-order range back to the original colors of the normal *GFXlib 2* default palette automatically.

NOTES ON THIS COMMAND:

*Same workings as PalNeg.DefaultPal, except that you can do *any* part of the palette you wish at anytime! :D Keep in mind now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalNegRotate.DefaultPal

SUB DESCRIPTION:

```
   Sub PalNegRotate.DefaultPal (StartColor, EndColor, Rotate.Level)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole inverted version of the default *GFXlib 2* palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

NOTES ON THIS COMMAND:

In this command here, you can truly transform the many colors of the default GFXlib 2 *palette into an inverted-colors version of it, and at the same time custom-rotate it forwards or backwards, whether you do part of the palette, or even all of it!! Awesome!!! :*) Remember now that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalFadeToNega.DefaultPal

SUB DESCRIPTION:

```
    Sub PalFadeToNega.DefaultPal (millisec)
```

`millisec` = the amount of milliseconds determining the speed of fading from the default *GFXlib 2* palette all the way to a negative (or inverted-colors) version of it.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from the whole default 256-color GFXlib 2 *palette all the way to an inverted version of it, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeFromNega.DefaultPal

SUB DESCRIPTION:

```
    Sub PalFadeFromNega.DefaultPal (millisec)
```

`millisec` = the amount of milliseconds determining the speed of fading from a negative (or inverted-colors) version of the entire default *GFXlib 2* palette all the way back to a normal version of it.

NOTES ON THIS COMMAND:

Does the exact opposite of PalFadeToNega.DefaultPal, *in that it lets you fade from an inverted version of the entire* GFXlib 2 *palette right back into a normal version of it once more. Again here, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

PalFadeRangeToNega.DefaultPal

SUB DESCRIPTION:

```
   Sub PalFadeRangeToNega.DefaultPal (StartColor, EndColor, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the selected order of colors from the default *GFXlib 2* palette all the way to a negative (or inverted-colors) version of the colors from that very same palette.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want the screen to fade from *any* part of the default 256-color GFXlib 2 palette all the way to an inverted version of that same palette, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades. Also, for this command to work the best, the "EndColor" number* ***must*** *be higher than the "StartColor" one.*

PalFadeRangeFromNega.DefaultPal

SUB DESCRIPTION:

```
Sub PalFadeRangeFromNega.DefaultPal (StartColor, EndColor, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the selected order of colors from the negative (or inverted-colors) version of the default *GFXlib 2* palette all the way back to the normal colors from that very same palette.

NOTES ON THIS COMMAND:

Does the exact opposite of PalFadeRangeToNega.DefaultPal*, in that it lets you fade from *any* part of the inverted version of the default 256-color* GFXlib 2 *palette all the way back to the normal version of that same palette! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades. Also, for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one.*

Please turn to the very next page for an FB program example of this using the commands PalNeg.DefaultPal, PalNegRange.DefaultPal, PalNegRotate.DefaultPal, PalFadeToNega.DefaultPal, PalFadeFromNega.DefaultPal, PalFadeRangeToNega.DefaultPal, *and* PalFadeRangeFromNega.DefaultPal*!!*

PROGRAM EXAMPLE #3:

(This example uses PalNeg.DefaultPal, PalNegRange.DefaultPal, PalNegRotate.DefaultPal, PalFadeToNega.DefaultPal, PalFadeFromNega.DefaultPal, PalFadeRangeToNega.DefaultPal, *and* PalFadeRangeFromNega.DefaultPal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries for the default GFXlib 2 palette.
'------------------------------------------------------------------------
For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, let's test out some "negative color"-based palette routines!!
'------------------------------------------------------------------


Line (0, 0)-(319, 15), 3, BF
Color 15, 3
Locate 1, 1: ? "We inverse the GFXlib 2 default palette"
Locate 2, 1: ? "and back again in *many* exciting ways!!"
sleep 4500


'--- First off, we switch the full entire GFXlib 2 default palette to a
'    negative of it *instantly*.  Then, let's wait just five (5) seconds
'    before shutting that palette right back to its normal state again and
'    waiting two (2) more seconds.

PalNeg.DefaultPal Yes
sleep 5000
PalNeg.DefaultPal No
sleep 2000


'--- Let's do the exact same thing here using ONLY color entries 16-173.

PalNegRange.DefaultPal 16, 173, Yes
sleep 5000
PalNegRange.DefaultPal 16, 173, No
sleep 2000


'--- Now, we actually fade the default GFXlib 2 palette into a full
'    negative of it in a single pass, measuring just 50 milliseconds per
'    fade-step, and then we fade it right back to normal again using that
'    same exact speed!  ;*)

PalFadeToNega.DefaultPal 50
PalFadeFromNega.DefaultPal 50
```

Continues on next page.......

"Program Example #3" continued from last page........

```
'--- Let's do the exact same thing here once again using ONLY color
'    entries 64-214, measuring only 22 milliseconds per fade-step!

PalFadeRangeToNega.DefaultPal 64, 214, 22
sleep 5000
PalFadeRangeFromNega.DefaultPal 64, 214, 22
sleep 2000


Line (0, 0)-(319, 15), 3, BF
Color 15, 3
Locate 1, 1: ? "Now, let's rotate around and around the"
Locate 2, 1: ? "inversed-color version of this palette!!"


'--- Using the color entries 20-235 now, we rotate the inversed colors of
'    the entire GFXlib 2 default palette forward for a moment, and then
'    backwards for that very same moment.  Amazing, huh?  ^_^

Rota = 0
DO
  PalNegRotate.DefaultPal 20, 235, Rota
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalNegRotate.DefaultPal 20, 235, Rota
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Thank you so much for viewing this fine"
Locate 2, 1: ? "little demonstration.  See you again!!!"


'--- Finally, let's wrap this up now by fading the ENTIRE default GFXlib 2
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.DefaultPal 63


'--- Thank 'ya, thank you very much!!  ^_-=b !
```

PalNegaFadeCtrl.DefaultPal

SUB DESCRIPTION:

```
    Sub PalNegaFadeCtrl.DefaultPal (FadeToNega.Grade)
```

`FadeToNega.Grade` = the custom fade-in level between the default *GFXlib 2* palette and a negative (or inverted-colors) version of that very same palette.
(63 = fully faded in to the negative of *GFXlib 2* palette; 0 = fully faded out to normal *GFXlib 2* palette)

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between the default 256-color GFXlib 2 *palette and an inverted version of it! Perfect for doing "negative color"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is* ***still*** *going, be it a game or graphics demo or whatever it is!! ;*) !*

PalNegaRangeFadeCtrl.DefaultPal

SUB DESCRIPTION:

```
Sub PalNegaRangeFadeCtrl.DefaultPal (StartColor, EndColor, FadeToNega.Grade)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToNega.Grade` = the custom fade-in level between the selected order of colors from default *GFXlib 2* palette and a negative (or inverted-colors) version of those colors from that very same palette.
(63 = fully faded in to the negative of *GFXlib 2* palette; 0 = fully faded out to normal *GFXlib 2* palette)

NOTES ON THIS COMMAND:

*Same drill as the command PalNegaFadeCtrl.DefaultPal, only it lets you do ANY part of the palette right as you please, too, whenever you like!! ^-^ ! A *MUST* for doing "negative color"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! Do not forget, the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best now.*

PalGreyScl.DefaultPal

SUB DESCRIPTION:

```
   Sub PalGreyScl.DefaultPal (Switch, NegaPalSwitch)
```

`Switch` = the operation of whether or not to switch the screen to a greyscale version of the default *GFXlib 2* palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command simply reverses the greyscale effect of the *GFXlib 2* default one automatically.

`NegaPalSwitch` = the operation of whether or not to switch the screen *also* to either a greyscale negative version of the default *GFXlib 2* palette or just a regular color negative of that very same palette, depending on the "*Switch*" setting that you have just specified using this command. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts the *GFXlib 2* default palette away from a negative one automatically.

NOTES ON THIS COMMAND:

With this command, you can actually switch the screen to a "black-and-white"-based effect of the default GFXlib 2 *palette INSTANTANEOUSLY! At the same time, you can even make a negative of the very same palette *in addition* to that, as well!!* ^-^ !!

PalGreySclRange.DefaultPal

SUB DESCRIPTION:

```
Sub PalGreySclRange.DefaultPal (StartColor, EndColor, Switch, NegaPalSwitch)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Switch` = the operation of whether or not to switch the selected order of colors to a greyscale version of the colors from the default *GFXlib 2* palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command simply reverses the greyscale effect of those selected colors from the *GFXlib 2* default palette automatically.

`NegaPalSwitch` = the operation of whether or not to switch the selected order of colors *also* to either a greyscale negative version of the default *GFXlib 2* palette or just a regular color negative of those original colors from that very same palette, depending on the "Switch" setting that you have just specified using this command. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that same order of colors of the *GFXlib 2* default palette away from a negative one automatically.

NOTES ON THIS COMMAND:

*Same deal as PalGreyScl.DefaultPal, except that you can use *any* part of the palette you wish at anytime! :D Keep in mind now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalGreySclRotate.DefaultPal

Sub Description:

```
    Sub PalGreySclRotate.DefaultPal (StartColor, EndColor, Rotate.Level,
     NegaPalSwitch)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole greyscale version of the default *GFXlib 2* palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`NegaPalSwitch` = the operation of whether or not to switch that same order of colors *also* to a negative (or inverted-colors) of the greyscale of the original colors from the default *GFXlib 2* palette that is being rotated around. Pass <u>**1**</u> or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that same selected order of colors of the *GFXlib 2* default palette <u>away</u> from a negative one automatically, leaving those selected colors just regular greyscale-based ones of that palette indeed.

Notes on this Command:

*In this command here, you can <u>truly</u> transform the many colors of the default GFXlib 2 palette into a greyscale version of it, and at the same time custom-rotate it forwards or backwards, whether you do part of the palette, or even all of it!! In addition, you can even make a negative of a greyscale of the GFXlib 2 default palette as well **<u>while rotating it</u>**!!! Rather white-hot stuff, wouldn't you say? :*D !! Remember now that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalFadeToGreyScl.DefaultPal

SUB DESCRIPTION:

```
   Sub PalFadeToGreyScl.DefaultPal (millisec, NegaPalSwitch)
```

`millisec` = the amount of milliseconds determining the speed of fading from the default *GFXlib 2* palette all the way to a greyscale version of it.

`NegaPalSwitch` = the operation of whether or not to fade the screen to *rather* a negative (or inverted-colors) of the greyscale version of the default *GFXlib 2* palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen will become like that, otherwise this command just lets you fade the *GFXlib 2* default palette into a normal greyscale version of it automatically.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from the whole default 256-color GFXlib 2 *palette all the way to a greyscale version of it, all in ONE single pass! Alternatively, on that same pass, you can even fade to a negative of a greyscale of that very palette, too!!* ^_^=b ! *Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeFromGreyScl.DefaultPal

SUB DESCRIPTION:

```
Sub PalFadeFromGreyScl.DefaultPal (millisec, NegaPalSwitch)
```

`millisec` = the amount of milliseconds determining the speed of fading from the greyscale version of the default *GFXlib 2* palette all the way to a normal version of it.

`NegaPalSwitch` = the operation of whether or not to fade the screen from *rather* a negative (or inverted-colors) of the greyscale version of the default *GFXlib 2* palette straight to the original "normal colors"-based version of that palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen will become like that, otherwise this command just lets you fade the normal greyscale version of the *GFXlib 2* default palette back to its original colors automatically.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from the greyscale version of the entire GFXlib 2 *default palette all the way back to its original color-based state, all in ONE single pass! Alternatively, on that exact same pass, you can even fade from a greyscale negative of that very palette over to its normal (not negative) and original colors, too!!* ^_^=b ! *Remember, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

PalFadeRangeToGreyScl.DefaultPal

SUB DESCRIPTION:

```
    Sub PalFadeRangeToGreyScl.DefaultPal (StartColor, EndColor, millisec,
     NegaPalSwitch)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the default *GFXlib 2* palette all the way to a greyscale version of it.

`NegaPalSwitch` = the operation of whether or not to fade the selected order of colors to *rather* a negative (or inverted-colors) of the greyscale version of the default *GFXlib 2* palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and that color order indeed will become like that, otherwise this command just lets you fade those same selected colors from the original *GFXlib 2* default palette into a normal greyscale version of it automatically.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want the screen to fade from*any* part of the default 256-color* GFXlib 2 *palette all the way to a greyscale version of it, all in ONE single pass! Alternatively, on that same pass, you can even fade any part of that very same palette to a greyscale negative of it, too!!* ^_^=b ! *Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeRangeFromGreyScl.DefaultPal

SUB DESCRIPTION:

```
Sub PalFadeRangeFromGreyScl.DefaultPal (StartColor, EndColor, millisec,
 NegaPalSwitch)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the greyscale version of the default *GFXlib 2* palette all the way to a normal version of it.

`NegaPalSwitch` = the operation of whether or not to fade the selected order of colors from *rather* a negative (or inverted-colors) of the greyscale version of the default *GFXlib 2* palette straight to the original "normal colors"-based version of that same palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the color order here will become like that, otherwise this command just lets you fade those exact same selected colors from the normal greyscale version of the *GFXlib 2* default palette back to its original colors automatically.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want to fade from *any* part of the greyscale version of the GFXlib 2 default palette all the way back to its original color-based state, all in ONE single pass! Alternatively, on* ***any*** *portion of that same palette on a single pass, you can even fade from a greyscale negative of it over to its normal (not negative) and original colors, too!!* ^_^=b ! *And do not forget here, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

Please turn to the very next page for an FB program example of this using the commands PalGreyScl.DefaultPal, PalGreySclRange.DefaultPal, PalGreySclRotate.DefaultPal, PalFadeToGreyScl.DefaultPal, PalFadeFromGreyScl.DefaultPal, PalFadeRangeToGreyScl.DefaultPal, and PalFadeRangeFromGreyScl.DefaultPal!!

PROGRAM EXAMPLE #4:

(This example uses PalGreyScl.DefaultPal, PalGreySclRange.DefaultPal, PalGreySclRotate.DefaultPal, PalFadeToGreyScl.DefaultPal, PalFadeFromGreyScl.DefaultPal, PalFadeRangeToGreyScl.DefaultPal, *and* PalFadeRangeFromGreyScl.DefaultPal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries for the default GFXlib 2 palette.
'------------------------------------------------------------------------
For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, do some exciting greyscale palette routines here!!  ;*)
'------------------------------------------------------------

Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Let's make the GFXlib 2 default palette"
Locate 2, 1: ? "greyscale and back again and more!!!"
Sleep 4500


'--- First off, we switch the full entire GFXlib 2 default palette to a
'    greyscale of it *instantly*.  Then between brief moments, we inverse
'    it into a greyscale-based negative, then a color negative, and then
'    back to a normal version of it again.

PalGreyScl.DefaultPal Yes, No
Sleep 3000
PalGreyScl.DefaultPal Yes, Yes
Sleep 3000
PalGreyScl.DefaultPal No, Yes
Sleep 3000
PalGreyScl.DefaultPal No, No
Sleep 3000



'--- Let's do the exact same thing here using ONLY color entries 16-200.

PalGreySclRange.DefaultPal 16, 200, Yes, No
Sleep 3000
PalGreySclRange.DefaultPal 16, 200, Yes, Yes
Sleep 3000
PalGreySclRange.DefaultPal 16, 200, No, Yes
Sleep 3000
PalGreySclRange.DefaultPal 16, 200, No, No
Sleep 3000
```

Continues on next page.......

“Program Example #4” continued from last page........

```
'--- Now, we actually fade the default GFXlib 2 palette into a full
'    greyscale of it in a single pass, measuring just 45 milliseconds per
'    fade-step, wait two seconds, and then we fade it right back to normal
'    again using that same exact speed!  ;*)

PalFadeToGreyScl.DefaultPal 45, No
sleep 2000
PalFadeFromGreyScl.DefaultPal 45, No


'--- Let's do it again, but this time, we fade to an entire *negative
'    greyscale* of the GFXlib 2 default palette and then back to normal
'    again in just the next two passes, measuring at the same exact speed
'    per fade-step.

PalFadeToGreyScl.DefaultPal 45, Yes
sleep 2000
PalFadeFromGreyScl.DefaultPal 45, Yes


'--- We repeat the "greyscale/negative greyscale"-based fades once more
'    here, only this time, we use color entries 1-166 of the default
'    GFXlib 2 palette now, measuring only 25 milliseconds per fade-step.

PalFadeRangeToGreyScl.DefaultPal 1, 166, 25, No
sleep 2000
PalFadeRangeFromGreyScl.DefaultPal 1, 166, 25, No

PalFadeRangeToGreyScl.DefaultPal 1, 166, 25, Yes
sleep 2000
PalFadeRangeFromGreyScl.DefaultPal 1, 166, 25, Yes



Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Now rotating around and around the"
Locate 2, 1: ? "greyscale version of this palette!!"


'--- Using the color entries 20-235 now, we rotate the greyscaled colors of
'    the entire GFXlib 2 default palette forward for a moment, and then
'    backwards for that very same moment.  Amazing, huh?  ^_^

Rota = 0
DO
  PalGreySclRotate.DefaultPal 20, 235, Rota, No
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalGreySclRotate.DefaultPal 20, 235, Rota, No
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Rota = 0
Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Let's also do the same for the negative"
Locate 2, 1: ? "greyscale version of this palette!!!"
```

Continues on next page.......

“Program Example #4” *still* continued from last page........

```
'--- Same thing again, but this time rotating around and around the
'    negative greyscale of the entire GFXlib 2 default palette!!

DO
  PalGreySclRotate.DefaultPal 20, 235, Rota, Yes
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalGreySclRotate.DefaultPal 20, 235, Rota, Yes
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Catch you later now, and thank you!"
Locate 2, 1: ? "Hope you have enjoyed the show!!!"


'--- Finally, let's close this now by fading the ENTIRE default GFXlib 2
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.DefaultPal 63


'--- See you again!!  ^-^ !
```

PalGreyFadeCtrl.DefaultPal

SUB DESCRIPTION:

```
Sub PalGreyFadeCtrl.DefaultPal (FadeToGrey.Grade, NegaPalSwitch)
```

`FadeToGrey.Grade` = the custom fade-in level between the default *GFXlib 2* palette and a greyscale version of that very same palette.
(63 = fully faded in to the greyscale of *GFXlib 2* palette; 0 = fully faded out to normal *GFXlib 2* palette)

`NegaPalSwitch` = the operation of whether or not to set the custom fade-in level to *rather* fade between the default *GFXlib 2* palette and a negative greyscale version of it. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the entire palette will become like that, otherwise this command just sets the custom fade-in level to fade between the default *GFXlib 2* palette and a normal greyscale version of it.

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between the default 256-color GFXlib 2 *palette and a greyscaled version of it! Alternatively, you can control those same fade levels between all the original colors of that very palette and an actual greyscale negative of it!! Perfect for games and graphics demos in FB!!! ;*) !*

PalGreyRangeFadeCtrl.DefaultPal

SUB DESCRIPTION:

```
    Sub PalGreyRangeFadeCtrl.DefaultPal (StartColor, EndColor,
     FadeToGrey.Grade, NegaPalSwitch)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToGrey.Grade` = the custom fade-in level between the selected colors of the default *GFXlib 2* palette and a greyscale version of that very same palette.
(63 = fully faded in to the greyscale of *GFXlib 2* palette; 0 = fully faded out to normal *GFXlib 2* palette)

`NegaPalSwitch` = the operation of whether or not to set the custom fade-in level to *rather* fade between the default *GFXlib 2* palette and a negative greyscale version of it. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the selected order of colors will become like that, otherwise this command just sets the custom fade-in level to fade between that same order of colors of the default *GFXlib 2* palette *and* a normal greyscale version of them.

NOTES ON THIS COMMAND:

Same entire drill as the previous command PalGreyFadeCtrl.DefaultPal*, except that you can actually do* ***any*** *part of the default 256 color* GFXlib 2 *palette rather instead of just all of it alone!!!* ^_-=b !!

NOTE: THESE NEXT ROUTINES HERE SUPPORT THE "768-BYTE"-BASED PALETTES ONLY!!

When these following commands are applied in your FB programs, the colors can **automatically** *change to the shades of *any* custom 768-byte palette that you select, so please be *very* careful if you are using any 256-color palette(s) at all, alright? ;*) !*

LoadUp768Pal

SUB DESCRIPTION:

```
    Sub LoadUp768Pal  (PalFile$)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

NOTES ON THIS COMMAND:

This command just does what it says: it lets you automatically load a custom 768-byte palette file of your choice and simply places it up as your new 256-color palette. It is just like using the "PALETTE" command in this sense here, you know? ;D

FadeIn.768Pal

SUB DESCRIPTION:

```
   Sub FadeIn.768Pal (PalFile$, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading in from black to the custom palette specified.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade from black all the way to your own custom 768-byte palette, all in ONE single pass!! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

FadeOut.768Pal

SUB DESCRIPTION:

```
    Sub FadeOut.768Pal (PalFile$, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to black.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade from your own custom 768-byte palette all the way out to pitch blackness, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

FadeInX.768Pal

Sub Description:

```
Sub FadeInX.768Pal (PalFile$, R.from, G.from, B.from, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`R.from` = the **red** shade level (0 to 63) to fade from.

`G.from` = the **green** shade level (0 to 63) to fade from.

`B.from` = the **blue** shade level (0 to 63) to fade from.

`millisec` = the amount of milliseconds determining the speed of fading in from the selected color to the custom palette specified.

Notes on this Command:

On this command, you can actually determine just how fast or slow you want the entire screen to fade from the color of your choice all the way to your custom 768-byte palette, all in ONE single pass! Higher milliseconds = slower fades; while lower milliseconds = faster fades.

FadeOutX.768Pal

SUB DESCRIPTION:

```
   Sub FadeOutX.768Pal (PalFile$, R.to, G.to, B.to, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`R.to` = the **red** shade level (0 to 63) to fade to.

`G.to` = the **green** shade level (0 to 63) to fade to.

`B.to` = the **blue** shade level (0 to 63) to fade to.

`millisec` = the amount of milliseconds determining the speed of fading out the specified custom palette to the selected color.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade your custom 768-byte palette right into the color of your choice, all in ONE single pass! Higher milliseconds = slower fades; while lower milliseconds = faster fades.

FadeInRange.768Pal

SUB DESCRIPTION:

```
   Sub FadeInRange.768Pal (PalFile$, StartColor, EndColor, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading in from black to the custom palette specified.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from black all the way to your own custom 768-byte palette, all in ONE single pass! Useful for fading certain parts of the screen in, too!! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeOutRange.768Pal

SUB DESCRIPTION:

```
   Sub FadeOutRange.768Pal (PalFile$, StartColor, EndColor, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to black.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from your custom 768-byte palette all the way out to pitch blackness, all in ONE single pass! Useful for fading out certain parts of the screen, too!! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeInRangeX.768Pal

SUB DESCRIPTION:

```
    Sub FadeInRangeX.768Pal (PalFile$, StartColor, EndColor, R.from, G.from,
     B.from, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to fade from.

`G.from` = the **green** shade level (0 to 63) to fade from.

`B.from` = the **blue** shade level (0 to 63) to fade from.

`millisec` = the amount of milliseconds determining the speed of fading in from the selected color to the custom palette specified.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from your chosen color all the way to the your own custom 768-byte palette, all in ONE single pass! Useful for fading certain parts of the screen in, and for doing some real cool palette-lighting effects, too!! Remember, for this command to work the best, the "EndColor" number ***must*** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeOutRangeX.768Pal

SUB DESCRIPTION:

```
   Sub FadeOutRangeX.768Pal (PalFile$, StartColor, EndColor, R.to, G.to, B.to,
    millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.to` = the **red** shade level (0 to 63) to fade to.

`G.to` = the **green** shade level (0 to 63) to fade to.

`B.to` = the **blue** shade level (0 to 63) to fade to.

`millisec` = the amount of milliseconds determining the speed of fading out the specified custom palette to the selected color.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade your own custom 768-byte palette right into the color of your choice, all in ONE single pass! Useful for fading certain parts of the screen in, and for doing some real cool palette-lighting effects, too!! Remember, for this command to work the best, the "EndColor" number ***must*** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

Please turn to the very next page for an FB program example of this using the commands LoadUp768Pal, FadeInRangeX.768Pal and FadeOutRangeX.768Pal!!

PROGRAM EXAMPLE #5:

(This example uses LoadUp768Pal, FadeInRangeX.768Pal *and* FadeOutRangeX.768Pal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom 768-byte palette.
'--------------------------------------------------------------------------
Pal$ = "RelPal.pal" '<--- This is Relsoft's 768-byte custom palette that
                    '     we are gonna be using here for this test!

LoadUp768Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 8)-(DrawPalette, 200), DrawPalette
Next

'Now, let's work some color fading magic on this, shall we?  ^_^ !
'------------------------------------------------------------------


Color 15, 1
Locate 1, 1: ? "Working some 768-byte palette magic!!!"


'--- We fade color entries 20-100 of the custom 768-byte palette right to
'    yellow in a single pass, measuring 30 millisecs. per fade-step, and
'    then we fade it back again using that same color and speed!

FadeOutRangeX.768Pal Pal$, 20, 100, 63, 63, 0, 30
FadeInRangeX.768Pal Pal$, 20, 100, 63, 63, 0, 30


'--- Let's try it the same way with purple, except this time, we use color
'    entries 110-255 and with a faster fade speed of 10 millisecs. per
'    fade-step!

FadeOutRangeX.768Pal Pal$, 110, 255, 63, 0, 63, 10
FadeInRangeX.768Pal Pal$, 110, 255, 63, 0, 63, 10


'--- How 'bout some chillin' this time with a dash of blue using color
'    entries 0-158 and with a *much* slower fade speed of 70 milliseconds
'    per fade-step, hmmm?  ;)

FadeOutRangeX.768Pal Pal$, 0, 158, 0, 0, 63, 70
FadeInRangeX.768Pal Pal$, 0, 158, 0, 0, 63, 70


'--- Or get some red in and out with color entries 129-203 using a MUCH
'    faster fade speed of only 4 milliseconds/fade-step!

FadeOutRangeX.768Pal Pal$, 129, 203, 63, 0, 0, 4
FadeInRangeX.768Pal Pal$, 129, 203, 63, 0, 0, 4


'--- Finally, let's wrap this up now by fading the ENTIRE palette slowly
'    but *all the way* out in only one pass, using just 63 milliseconds
'    per fade-step!  ;*)

FadeOut.768Pal Pal$, 63
```

FadeCtrl.768Pal

SUB DESCRIPTION:

```
   Sub FadeCtrl.768Pal (PalFile$, FadeIn.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`FadeIn.Grade` = the custom fade-in level between a pitch-black palette and the specified custom palette.
(63 = fully faded in to palette; 0 = fully faded out to black)

NOTES ON THIS COMMAND:

With this command, you have FULL and free control of all 64 of the fade levels between a black screen and your own custom 768-byte palette! Perfect for doing fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;) !*

FadeCtrlX.768Pal

SUB DESCRIPTION:

```
Sub FadeCtrlX.768Pal (PalFile$, R.from, G.from, B.from, FadeIn.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`R.from` = the **red** shade level (0 to 63) to use as your “fade-out”-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your “fade-out”-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your “fade-out”-based color shade.

`FadeIn.Grade` = the custom fade-in level between an entire palette of a resulting color here and the specified custom palette. (63 = fully faded in to palette; 0 = fully faded out to that chosen color)

NOTES ON THIS COMMAND:

With this command, you have FULL and free control of all 64 of the fade levels between a whole palette of the color of your choice and your own custom 768-byte palette!! Perfect for doing such “color-to-custom-palette”-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!!! ;) !*

FadeCtrlRangeX.768Pal

SUB DESCRIPTION:

```
Sub FadeCtrlRangeX.768Pal (PalFile$, StartColor, EndColor, R.from, G.from,
 B.from, FadeIn.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to use as your "fade-out"-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your "fade-out"-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your "fade-out"-based color shade.

`FadeIn.Grade` = the custom fade-in level between a palette of a resulting color here and the specified custom palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to that chosen color)

NOTES ON THIS COMMAND:

*With this command, you again have FULL and free control of all 64 of the fade levels between the color of your choice and your custom 768-byte palette, except this time, it is for **any part of the palette**!!! Especially an awesome thing for doing such "color-to-custom-palette"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!!! d=^_^=b ! Keep in mind though that the "EndColor" number **must** be higher than the "StartColor" one in order for this command to work the best!*

PalRotate.768Pal

Sub Description:

```
   Sub PalRotate.768Pal (PalFile$, StartColor, EndColor, Rotate.Level)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the entire specified custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

Notes on this Command:

In this command, you can truly custom-rotate the many colors of your own 768-byte palette forwards or backwards, whether you do part of the palette, or even all of it! It is up to you!! :D Remember now that the "EndColor" number ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalRotateFC.768Pal

SUB DESCRIPTION:

```
   Sub PalRotateFC.768Pal (PalFile$, StartColor, EndColor, Rotate.Level,
    FadeIn.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **<u>only 768 bytes</u>**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`FadeIn.Grade` = the custom fade-in level between a pitch-black palette <u>and</u> the specified custom palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to black)

NOTES ON THIS COMMAND:

In this command, not only can you truly custom-rotate the many colors of your own 768-byte palette forwards or backwards — whether you do part of the palette or even all of it — but also, <u>you have FULL and free control of all 64 of the fade levels as well within your color range</u>, too!! An awesomely great recommendation for working on your games and graphics demos, I must say!!! ^-^=b ! Remember, in order for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one!*

PalRotateFCX.768Pal

SUB DESCRIPTION:

```
   Sub PalRotateFCX.768Pal (PalFile$, StartColor, EndColor, R.from, G.from,
    B.from, Rotate.Level, FadeIn.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to use as your "fade-out"-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your "fade-out"-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your "fade-out"-based color shade.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`FadeIn.Grade` = the custom fade-in level between a palette of a resulting color here and the specified custom palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to that selected color)

NOTES ON THIS COMMAND:

*Very much the same as PalRotateFC.768Pal, except that you are now allowed *as well* custom fades from any single color you wish to your entire 768-byte custom palette, back again, and somewhere in-between, too!!! Now, be sure to try that in your games and graphics demos, as it will do you rather good here!! ^-^ !! Remember, in order for this command to work the best, the "EndColor" number* ***must*** *be higher than the "StartColor" one!*

Please turn to the very next page for an FB program example of this using the commands PalRotate.768Pal, PalRotateFC.768Pal, and PalRotateFCX.768Pal!!

PROGRAM EXAMPLE #6:

(This example uses PalRotate.768Pal, PalRotateFC.768Pal, *and* PalRotateFCX.768Pal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-----------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom 768-byte palette.
'------------------------------------------------------------------------
Pal$ = "CustomPalette_01.pal" '<--- This is a 768-byte custom palette that
                              '     we are gonna be using for this test!

LoadUp768Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, we will rotate the palettes around and around!!  ;)
'--------------------------------------------------------


Color 15, 4
Locate 1, 1: ? "Let's spin that 768-byte palette wheel!!"

Color 15, 0
Locate 2, 1: ? "- Now using PalRotate.768Pal... -"


'--- Using the color entries 20-235 of the custom 768-byte palette, we
'    rotate the colors of the entire palette forward for a short moment,
'    and then backwards for that very same moment.

Rota = 0
DO
  PalRotate.768Pal Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotate.768Pal Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "- Now using PalRotateFC.768Pal... -"


'--- Next, we use the same color entries of the custom 768-byte palette
'    to once again rotate the colors of the entire palette forward for a
'    short moment, and then backwards for that same moment.  BUT, we do
'    it now with fade-ins/fade-outs as you will see right here!  ;*)
```

Continues on next page.......

“Program Example #6” continued from last page........

```
Rota = 0
DO
  PalRotateFC.768Pal Pal$, 20, 235, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFC.768Pal Pal$, 20, 235, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "- Now using PalRotateFCX.768Pal... -"


'--- And now, we do the same thing as the second one here, only this time,
'    with color-based fade-ins/fade-outs that will interest you real
'    good now, so do get psyched-up for this one!!  :D

Rota = 0
DO
  PalRotateFCX.768Pal Pal$, 20, 235, 54, 10, 32, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFCX.768Pal Pal$, 20, 235, 10, 59, 27, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Rota = 0
DO
  PalRotateFCX.768Pal Pal$, 20, 235, 5, 6, 63, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFCX.768Pal Pal$, 20, 235, 63, 63, 63, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "Thank you for watching this.  Peace!!"


'--- We now end this test indeed, as always, by fading out the palette
'    to nothing and exiting this program.  ^_^=b !

FadeOut.768Pal Pal$, 63

'--- Peace!!!  ;*)
```

PalNeg.768Pal

SUB DESCRIPTION:

```
    Sub PalNeg.768Pal (PalFile$, Switch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`Switch` = the operation of whether or not to switch the screen to a negative (or inverted-colors) version of the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts the palette back to that specified custom one automatically.

NOTES ON THIS COMMAND:

With this command, you can actually switch the screen to a "negative"-based effect of your own custom 768-byte palette INSTANTANEOUSLY!

PalNegRange.768Pal

SUB DESCRIPTION:

```
Sub PalNegRange.768Pal (PalFile$, StartColor, EndColor, Switch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Switch` = the operation of whether or not to switch the selected order of colors to a negative (or inverted-colors) version of original colors of the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the order indeed becomes switched like that, otherwise this command reverts the specified color-order range back to the original colors of that custom palette automatically.

NOTES ON THIS COMMAND:

*Same workings as PalNeg.768Pal, except that you can do *any* part of your custom 768-byte palette you wish at anytime!! :D ! Keep in mind now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalNegRotate.768Pal

SUB DESCRIPTION:

```
Sub PalNegRotate.768Pal (PalFile$, StartColor, EndColor, Rotate.Level)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole inverted version of the specified custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

NOTES ON THIS COMMAND:

In this command here, you can truly transform the many colors of your own custom 768-byte palette into an inverted-colors version of it, and at the same time custom-rotate it forwards or backwards, whether you do part of the palette, or even all of it!! Rather so awesome!!! ^_^=b *Remember now that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalFadeToNega.768Pal

SUB DESCRIPTION:

```
   Sub PalFadeToNega.768Pal (PalFile$, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to a negative (or inverted-colors) version of it.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from your whole custom 768-byte palette all the way to an inverted version of it, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

PalFadeFromNega.768Pal

SUB DESCRIPTION:

```
Sub PalFadeFromNega.768Pal (PalFile$, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from a negative (or inverted-colors) version of the specified custom palette all the way back to a normal version of it.

NOTES ON THIS COMMAND:

Does the exact opposite of PalFadeToNega.768Pal*, in that it lets you fade from an inverted version of the entire custom 768-byte palette of your choice right back into a normal version of it once more. Again here, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

PalFadeRangeToNega.768Pal

SUB DESCRIPTION:

```
Sub PalFadeRangeToNega.768Pal (PalFile$, StartColor, EndColor, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the selected order of colors from the specified custom palette all the way to a negative (or inverted-colors) version of the colors from that very same palette.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want the screen to fade from *any* part of your custom 768-byte palette all the way to an inverted version of that same palette, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades. Also, for this command to work the best, the "EndColor" number* ***must*** *be higher than the "StartColor" one.*

PalFadeRangeFromNega.768Pal

Sub Description:

```
Sub PalFadeRangeFromNega.768Pal (PalFile$, StartColor, EndColor, millisec)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the selected order of colors from the negative (or inverted-colors) version of the specified custom palette all the way back to the normal colors from that very same palette.

NOTES ON THIS COMMAND:

*Does the exact opposite of PalFadeRangeToNega.768Pal, in that it lets you fade from *any* part of the inverted version of your custom 768-byte palette all the way back to the normal version of that same palette! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades. Also, for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one.*

Please turn to the very next page for an FB program example of this using the commands PalNeg.768Pal, PalNegRange.768Pal, PalNegRotate.768Pal, PalFadeToNega.768Pal, PalFadeFromNega.768Pal, PalFadeRangeToNega.768Pal, and PalFadeRangeFromNega.768Pal!!

PROGRAM EXAMPLE #7:

(This example uses PalNeg.768Pal, PalNegRange.768Pal, PalNegRotate.768Pal, PalFadeToNega.768Pal, PalFadeFromNega.768Pal, PalFadeRangeToNega.768Pal, *and* PalFadeRangeFromNega.768Pal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom 768-byte palette.
'-------------------------------------------------------------------------
Pal$ = "CustomPalette_01.pal" '<--- This is a 768-byte custom palette that
                              '     we are gonna be using for this test!

LoadUp768Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, let's test out some "negative color"-based palette routines
'on our 768-byte palette that we just loaded, shall we?  ^_^
'------------------------------------------------------------------

Line (0, 0)-(319, 15), 5, BF
Color 15, 5
Locate 1, 1: ? "We inverse the 768-byte custom palette"
Locate 2, 1: ? "and back again in *many* exciting ways!!"
sleep 4500


'--- First off, we switch the full entire custom 768-byte palette to a
'    negative of it *instantly*.  Then, let's wait just five (5) seconds
'    before shutting that palette right back to its normal state again and
'    waiting two (2) more seconds.

PalNeg.768Pal Pal$, Yes
sleep 5000
PalNeg.768Pal Pal$, No
sleep 2000


'--- Let's do the exact same thing here using ONLY color entries 16-173.

PalNegRange.768Pal Pal$, 16, 173, Yes
sleep 5000
PalNegRange.768Pal Pal$, 16, 173, No
sleep 2000
```

Continues on next page.......

“Program Example #7” continued from last page........

```
'--- Now, we actually fade our custom 768-byte palette into a full
'    negative of it in a single pass, measuring just 50 milliseconds per
'    fade-step, and then we fade it right back to normal again using that
'    same exact speed!  ;*)

PalFadeToNega.768Pal Pal$, 50
PalFadeFromNega.768Pal Pal$, 50


'--- Let's do the exact same thing here once again using ONLY color
'    entries 64-214, measuring only 22 milliseconds per fade-step!

PalFadeRangeToNega.768Pal Pal$, 64, 214, 22
sleep 5000
PalFadeRangeFromNega.768Pal Pal$, 64, 214, 22
sleep 2000

Line (0, 0)-(319, 15), 5, BF
Color 15, 5
Locate 1, 1: ? "Now, let's rotate around and around the"
Locate 2, 1: ? "inversed-color version of this palette!!"


'--- Using the color entries 20-235 now, we rotate the inversed colors of
'    the entire custom 768-byte palette forward for a moment, and then
'    backwards for that very same moment.  Amazing, huh?  ^_^

Rota = 0
DO
  PalNegRotate.768Pal Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalNegRotate.768Pal Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Catch you again later, and thank you so"
Locate 2, 1: ? "much for your fine viewing of this!!!"


'--- Finally, let's wrap this up now by fading the ENTIRE custom 768-byte
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.768Pal Pal$, 63


'--- Bye for now!!  ^_-=b !
```

PalNegaFadeCtrl.768Pal

SUB DESCRIPTION:

```
   Sub PalNegaFadeCtrl.768Pal (PalFile$, FadeToNega.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`FadeToNega.Grade` = the custom fade-in level between the specified custom palette and a negative (or inverted-colors) version of that very same palette.
(63 = fully faded in to the negative of 768-byte palette; 0 = fully faded out to normal 768-byte palette)

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between your own custom 768-byte palette and an inverted version of it! Perfect for doing "negative color"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;) !*

PalNegaRangeFadeCtrl.768Pal

SUB DESCRIPTION:

```
Sub PalNegaRangeFadeCtrl.768Pal (PalFile$, StartColor, EndColor,
 FadeToNega.Grade)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToNega.Grade` = the custom fade-in level between the selected order of colors from the specified custom palette and a negative (or inverted-colors) version of those colors from that very same palette.
(63 = fully faded in to the negative of 768-byte palette; 0 = fully faded out to normal 768-byte palette)

NOTES ON THIS COMMAND:

*Same drill as the command PalNegaFadeCtrl.768Pal, only it lets you do ANY part of your own 768-byte palette right as you please, too, whenever you like!!! ^-^ !! A *MUST* for doing "negative color"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is, definitely!!! Do not forget, the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best now.*

PalGreySclﾠ.768Pal

SUB DESCRIPTION:

```
Sub PalGreyScl.768Pal (PalFile$, Switch, NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **<u>only 768 bytes</u>**, please!

`Switch` = the operation of whether or not to switch the screen to a greyscale version of the specified custom palette. Pass **<u>1</u>** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command simply reverses the greyscale effect of that specified palette automatically.

`NegaPalSwitch` = the operation of whether or not to switch the screen *also* to either a greyscale negative version of the specified custom palette or just a regular color negative of that very same palette, depending on the "*Switch*" setting that you have just specified using this command. Pass **<u>1</u>** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that specified palette <u>away</u> from a negative one automatically.

NOTES ON THIS COMMAND:

*With this command, you can actually switch the screen to a "black-and-white"-based effect of your own custom 768-byte palette <u>INSTANTANEOUSLY</u>! At the same time, you can even make a negative of the very same palette *in addition* to that, as well!!* ^-^ !!

PalGreySclRange.768Pal

SUB DESCRIPTION:

```
Sub PalGreySclRange.768Pal (PalFile$, StartColor, EndColor, Switch,
 NegaPalSwitch)
```

PalFile$ = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

StartColor = the **starting** color (0 to 255) within the color-order range.

EndColor = the **ending** color (0 to 255) within the color-order range.

Switch = the operation of whether or not to switch the selected order of colors to a greyscale version of the colors from the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command simply reverses the greyscale effect of those selected colors from that specified palette automatically.

NegaPalSwitch = the operation of whether or not to switch the selected order of colors *also* to either a greyscale negative version of the specified custom palette or just a regular color negative of those original colors from that very same palette, depending on the "*Switch*" setting that you have just specified using this command. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that same order of colors of that specified palette away from a negative one automatically.

NOTES ON THIS COMMAND:

*Same deal as PalGreyScl.768Pal, except that you can use *any* part of your 768-byte custom palette you wish at anytime! :D Keep in mind now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalGreySclRotate.768Pal

SUB DESCRIPTION:

```
   Sub PalGreySclRotate.768Pal (PalFile$, StartColor, EndColor, Rotate.Level,
    NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole greyscale version of the custom specified palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`NegaPalSwitch` = the operation of whether or not to switch that same order of colors *also* to a negative (or inverted-colors) of the greyscale of the original colors from the specified custom palette that is being rotated around. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that same selected order of colors of that specified palette away from a negative one automatically, leaving those selected colors just regular greyscale-based ones of that palette indeed.

NOTES ON THIS COMMAND:

In this command here, you can truly transform the many colors of your very own custom 768-byte palette into a greyscale version of it, and at the same time custom-rotate it forwards or backwards, whether you do part of the palette, or even all of it!! In addition, you can even make a negative of a greyscale of that exact same palette as well ***while rotating it****!!! Rather white-hot stuff, wouldn't you say? :*D !! Remember now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalFadeToGreyScl.768Pal

SUB DESCRIPTION:

```
   Sub PalFadeToGreyScl.768Pal (PalFile$, millisec, NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to a greyscale version of it.

`NegaPalSwitch` = the operation of whether or not to fade the screen to *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen will become like that, otherwise this command just lets you fade that chosen palette into a normal greyscale version of it automatically.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from the whole custom 768-byte palette of yours all the way to a greyscale version of it, all in ONE single pass! Alternatively, on that same pass, you can even fade to a negative of a greyscale of that very palette, too!! ^_^=b ! *Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeFromGreyScl.768Pal

SUB DESCRIPTION:

```
   Sub PalFadeFromGreyScl.768Pal (PalFile$, millisec, NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the greyscale version of the specified custom palette all the way to a normal version of it.

`NegaPalSwitch` = the operation of whether or not to fade the screen from *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette straight to the original "normal colors"-based version of that same palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen will become like that, otherwise this command just lets you fade the normal greyscale version of that chosen palette back to its original colors automatically.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from the greyscale version of your custom 768-byte palette all the way back to its original color-based state, all in ONE single pass! Alternatively, on that exact same pass, you can even fade from a greyscale negative of that very palette over to its normal (not negative) and original colors, too!! ^_^=b ! *Remember, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

PalFadeRangeToGreyScl.768Pal

SUB DESCRIPTION:

```
    Sub PalFadeRangeToGreyScl.768Pal (PalFile$, StartColor, EndColor, millisec,
     NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the specified custom palette all the way to a greyscale version of it.

`NegaPalSwitch` = the operation of whether or not to fade the selected order of colors to *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and that color order indeed will become like that, otherwise this command just lets you fade those same selected colors from that specified palette into a normal greyscale version of it automatically.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want the screen to fade from*any* part of your custom 768-byte palette all the way to a greyscale version of it, all in ONE single pass! Alternatively, on that same pass, you can even fade any part of that very same palette to a greyscale negative of it, too!! ^_^=b ! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeRangeFromGreyScl.768Pal

SUB DESCRIPTION:

```
    Sub PalFadeRangeFromGreyScl.768Pal (PalFile$, StartColor, EndColor,
     millisec, NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the greyscale version of the specified custom palette all the way to a normal version of it.

`NegaPalSwitch` = the operation of whether or not to fade the selected order of colors from *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette straight to the original "normal colors"-based version of that same palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the color order here will become like that, otherwise this command just lets you fade those exact same selected colors from the normal greyscale version of that specified palette back to its original colors automatically.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want to fade from *any* part of the greyscale version of your very own 768-byte custom palette all the way back to its original color-based state, all in ONE single pass! Alternatively, on* ***any*** *portion of that same palette on a single pass, you can even fade from a greyscale negative of it over to its normal (not negative) and original colors, too!! ^_^=b ! And do not forget here, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

Please turn to the very next page for an FB program example of this using the commands PalGreyScl.768Pal, PalGreySclRange.768Pal, PalGreySclRotate.768Pal, PalFadeToGreyScl.768Pal, PalFadeFromGreyScl.768Pal, PalFadeRangeToGreyScl.768Pal, and PalFadeRangeFromGreyScl.768Pal!!

PROGRAM EXAMPLE #8:

(This example uses PalGreySc l.768Pal, PalGreySclRange.768Pal, PalGreySclRotate.768Pal, PalFadeToGreyScl.768Pal, PalFadeFromGreyScl.768Pal, PalFadeRangeToGreyScl.768Pal, *and* PalFadeRangeFromGreyScl.768Pal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom 768-byte palette.
'-------------------------------------------------------------------------
Pal$ = "CustomPalette_01.pal" '<--- This is a 768-byte custom palette that
                              '     we are gonna be using for this test!

LoadUp768Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, do some exciting greyscale palette routines here for our
'custom 768-byte palette!!!  ;*) !
'-------------------------------------------------------------

Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Let's make the 768-byte custom palette"
Locate 2, 1: ? "greyscale and back again and more!!!"
Sleep 4500


'--- First off, we switch the full entire custom 768-byte palette to a
'    greyscale of it *instantly*.  Then between brief moments, we inverse
'    it into a greyscale-based negative, then a color negative, and then
'    back to a normal version of it again.

PalGreyScl.768Pal Pal$, Yes, No
Sleep 3000
PalGreyScl.768Pal Pal$, Yes, Yes
Sleep 3000
PalGreyScl.768Pal Pal$, No, Yes
Sleep 3000
PalGreyScl.768Pal Pal$, No, No
Sleep 3000



'--- Let's do the exact same thing here using ONLY color entries 16-200.

PalGreySclRange.768Pal Pal$, 16, 200, Yes, No
Sleep 3000
PalGreySclRange.768Pal Pal$, 16, 200, Yes, Yes
Sleep 3000
PalGreySclRange.768Pal Pal$, 16, 200, No, Yes
Sleep 3000
PalGreySclRange.768Pal Pal$, 16, 200, No, No
Sleep 3000
```

Continues on next page.......

“Program Example #8” continued from last page........

```
'--- Now, we actually fade our custom 768-byte palette into a full
'    greyscale of it in a single pass, measuring just 100 milliseconds per
'    fade-step, wait two seconds, and then we fade it right back to normal
'    again using that same exact speed!  ;*)

PalFadeToGreyScl.768Pal Pal$, 100, No
sleep 2000
PalFadeFromGreyScl.768Pal Pal$, 100, No


'--- Let's do it again, but this time, we fade to an entire *negative
'    greyscale* of that custom 768-byte palette and then back to normal
'    again in just the next two passes, measuring at the same exact speed
'    per fade-step.

PalFadeToGreyScl.768Pal Pal$, 100, Yes
sleep 2000
PalFadeFromGreyScl.768Pal Pal$, 100, Yes


'--- We repeat the "greyscale/negative greyscale"-based fades once more
'    here, only this time, we use color entries 1-166 of our custom
'    768-byte palette now, measuring only 10 milliseconds per fade-step.

PalFadeRangeToGreyScl.768Pal Pal$, 1, 166, 10, No
sleep 2000
PalFadeRangeFromGreyScl.768Pal Pal$, 1, 166, 10, No

PalFadeRangeToGreyScl.768Pal Pal$, 1, 166, 10, Yes
sleep 2000
PalFadeRangeFromGreyScl.768Pal Pal$, 1, 166, 10, Yes



Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Now rotating around and around the"
Locate 2, 1: ? "greyscale version of this palette!!"


'--- Using the color entries 20-235 now, we rotate the greyscaled colors of
'    that exact 768-byte palette forward for a moment, and then backwards
'    for that very same moment.  Rather amazing, huh?  ^_^

Rota = 0
DO
  PalGreySclRotate.768Pal Pal$, 20, 235, Rota, No
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalGreySclRotate.768Pal Pal$, 20, 235, Rota, No
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Rota = 0
Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Let's also do the same for the negative"
Locate 2, 1: ? "greyscale version of this palette!!!"
```

Continues on next page.......

*"Program Example #8" *still* continued from last page........*

```
'--- Same thing again, but this time rotating around and around the
'    negative greyscale of our same 768-byte custom palette!!

DO
  PalGreySclRotate.768Pal Pal$, 20, 235, Rota, Yes
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalGreySclRotate.768Pal Pal$, 20, 235, Rota, Yes
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Check you out later, and God so richly"
Locate 2, 1: ? "bless 'ya!!  Thank you so much now!!!"


'--- Finally, let's close this now by fading our custom 768-byte
'    palette slowly but *all the way* out in only one pass, using
'    just 63 milliseconds per fade-step!  ;*)

FadeOut.768Pal Pal$, 63


'--- Signing off, over and out!!  ^-^=b !
```

PalGreyFadeCtrl.768Pal

SUB DESCRIPTION:

```
Sub PalGreyFadeCtrl.768Pal (PalFile$, FadeToGrey.Grade, NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`FadeToGrey.Grade` = the custom fade-in level between the specified custom palette and a greyscale version of that very same palette.
(63 = fully faded in to the greyscale of 768-byte palette; 0 = fully faded out to normal 768-byte palette)

`NegaPalSwitch` = the operation of whether or not to set the custom fade-in level to *rather* fade between the specified custom palette and a negative greyscale version of it. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the entire palette will become like that, otherwise this command just sets the custom fade-in level to fade between that same specified palette and a normal greyscale version of it.

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between your own custom 768-byte palette and a greyscaled version of it! Alternatively, you can control those same fade levels between all the original colors of that very palette and an actual greyscale negative of it!! Perfect for games and graphics demos in FB!!! ;) !*

PalGreyRangeFadeCtrl.768Pal

SUB DESCRIPTION:

```
    Sub PalGreyRangeFadeCtrl.768Pal (PalFile$, StartColor, EndColor,
     FadeToGrey.Grade, NegaPalSwitch)
```

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToGrey.Grade` = the custom fade-in level between the selected colors of the specified custom palette and a greyscale version of that very same palette.
(63 = fully faded in to the greyscale of 768-byte palette; 0 = fully faded out to normal 768-byte palette)

`NegaPalSwitch` = the operation of whether or not to set the custom fade-in level to *rather* fade between the specified custom palette and a negative greyscale version of it. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the selected order of colors will become like that, otherwise this command just sets the custom fade-in level to fade between that same order of colors of that exact specified palette *and* a normal greyscale version of them.

NOTES ON THIS COMMAND:

*Same entire drill as the previous command PalGreyFadeCtrl.768Pal, except that you can actually do **any** part of your custom 768-byte palette rather instead of just all of it alone!!! ^_-=b !!*

NOTE: THESE NEXT ROUTINES HERE SUPPORT THE PP256-BASED PALETTES ONLY!!

When these following commands are applied in your FB programs, the colors can **automatically** *change to the shades of *any* custom PixelPlus 256 (or PP256) palette that you select, so please be *very* careful if you are using any 256-color palette(s) at all, alright? ;*) !*

LoadUpPP256Pal

SUB DESCRIPTION:

```
    Sub LoadUpPP256Pal (PP256PalFile$)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

This command just does what it says: it lets you automatically load a custom PP256-based palette file of your choice and simply places it up as your new 256-color palette. It is just like using the "PALETTE" command in this sense here, you know? ;D

FadeIn.PP256

SUB DESCRIPTION:

```
   Sub FadeIn.PP256 (PP256PalFile$, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading in from black to the custom palette specified.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade from black all the way to your own custom PP256-based palette, all in ONE single pass!! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

FadeOut.PP256

SUB DESCRIPTION:

```
   Sub FadeOut.PP256 (PP256PalFile$, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to black.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade from your own custom PP256-based palette all the way out to pitch blackness, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

FadeInX.PP256

Sub Description:

```
Sub FadeInX.PP256 (PP256PalFile$, R.from, G.from, B.from, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`R.from` = the **red** shade level (0 to 63) to fade from.

`G.from` = the **green** shade level (0 to 63) to fade from.

`B.from` = the **blue** shade level (0 to 63) to fade from.

`millisec` = the amount of milliseconds determining the speed of fading in from the selected color to the custom palette specified.

Notes on this Command:

On this command, you can actually determine just how fast or slow you want the entire screen to fade from the color of your choice all the way to your custom PP256-based palette, all in ONE single pass! Higher milliseconds = slower fades; while lower milliseconds = faster fades.

FadeOutX.PP256

SUB DESCRIPTION:

```
   Sub FadeOutX.PP256 (PP256PalFile$, R.to, G.to, B.to, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`R.to` = the **red** shade level (0 to 63) to fade to.

`G.to` = the **green** shade level (0 to 63) to fade to.

`B.to` = the **blue** shade level (0 to 63) to fade to.

`millisec` = the amount of milliseconds determining the speed of fading out the specified custom palette to the selected color.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the entire screen to fade your custom PP256-based palette right into the color of your choice, all in ONE single pass! Higher milliseconds = slower fades; while lower milliseconds = faster fades.

FadeInRange.PP256

SUB DESCRIPTION:

```
   Sub FadeInRange.PP256 (PP256PalFile$, StartColor, EndColor, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading in from black to the custom palette specified.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from black all the way to your own custom PP256-based palette, all in ONE single pass! Useful for fading certain parts of the screen in, too!! Remember, for this command to work the best, the "EndColor" number ***must*** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeOutRange.PP256

SUB DESCRIPTION:

```
Sub FadeOutRange.PP256 (PP256PalFile$, StartColor, EndColor, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to black.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from your custom PP256-based palette all the way out to pitch blackness, all in ONE single pass! Useful for fading out certain parts of the screen, too!! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeInRangeX.PP256

SUB DESCRIPTION:

```
    Sub FadeInRangeX.PP256 (PP256PalFile$, StartColor, EndColor, R.from, G.from,
     B.from, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to fade from.

`G.from` = the **green** shade level (0 to 63) to fade from.

`B.from` = the **blue** shade level (0 to 63) to fade from.

`millisec` = the amount of milliseconds determining the speed of fading in from the selected color to the custom palette specified.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade from your chosen color all the way to the your own custom PP256-based palette, all in ONE single pass! Useful for fading certain parts of the screen in, and for doing some real cool palette-lighting effects, too!! Remember, for this command to work the best, the "EndColor" number ***must*** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

FadeOutRangeX.PP256

SUB DESCRIPTION:

```
    Sub FadeOutRangeX.PP256 (PP256PalFile$, StartColor, EndColor, R.to, G.to,
     B.to, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.to` = the **red** shade level (0 to 63) to fade to.

`G.to` = the **green** shade level (0 to 63) to fade to.

`B.to` = the **blue** shade level (0 to 63) to fade to.

`millisec` = the amount of milliseconds determining the speed of fading out the specified custom palette to the selected color.

NOTES ON THIS COMMAND:

When using this command, you can actually determine just how fast or slow you want the color-order range within your screen to fade your own custom PP256-based palette right into the color of your choice, all in ONE single pass! Useful for fading certain parts of the screen in, and for doing some real cool palette-lighting effects, too!! Remember, for this command to work the best, the "EndColor" number **must** *be higher than the "StartColor" one. Also here, higher milliseconds will determine slower fades, while lower milliseconds constitute more and more faster fades.*

Please turn to the very next page for an FB program example of this using the commands LoadUpPP256Pal, FadeInRangeX.PP256 and FadeOutRangeX.PP256!!

PROGRAM EXAMPLE #9:

(This example uses LoadUpPP256Pal, FadeInRangeX.PP256 *and* FadeOutRangeX.PP256.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom PP256-based palette!
'--------------------------------------------------------------------------
Pal$ = "PP256Pal1.pal" '<--- This is our custom palette fresh outta PP256
                       '     that we are gonna be using here for this test!

LoadUpPP256Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 8)-(DrawPalette, 200), DrawPalette
Next

'Now, let's work some color fading magic again on this, shall we?  ^_^ !
'--------------------------------------------------------------------------

Color 15, 1
Locate 1, 1: ? "Working some PP256-based palette jazz!!!"


'--- We fade color entries 20-100 of our custom PP256 palette right to
'    yellow in a single pass, measuring 30 millisecs. per fade-step, and
'    then we fade it back again using that same color and speed!

FadeOutRangeX.PP256 Pal$, 20, 100, 63, 63, 0, 30
FadeInRangeX.PP256 Pal$, 20, 100, 63, 63, 0, 30


'--- Let's try it the same way with purple, except this time, we use color
'    entries 110-255 and with a faster fade speed of 10 millisecs. per
'    fade-step!

FadeOutRangeX.PP256 Pal$, 110, 255, 63, 0, 63, 10
FadeInRangeX.PP256 Pal$, 110, 255, 63, 0, 63, 10


'--- How 'bout some chillin' this time with a dash of blue using color
'    entries 0-158 and with a *much* slower fade speed of 70 milliseconds
'    per fade-step, hmmm?  ;)

FadeOutRangeX.PP256 Pal$, 0, 158, 0, 0, 63, 70
FadeInRangeX.PP256 Pal$, 0, 158, 0, 0, 63, 70


'--- Or get some red in and out with color entries 129-203 using a MUCH
'    faster fade speed of only 4 milliseconds/fade-step!

FadeOutRangeX.PP256 Pal$, 129, 203, 63, 0, 0, 4
FadeInRangeX.PP256 Pal$, 129, 203, 63, 0, 0, 4


'--- Finally, let's wrap this up now by fading the ENTIRE palette slowly
'    but *all the way* out in only one pass, using just 63 milliseconds
'    per fade-step!  ;*)

FadeOut.PP256 Pal$, 63
```

FadeCtrl.PP256

SUB DESCRIPTION:

```
   Sub FadeCtrl.PP256 (PP256PalFile$, FadeIn.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`FadeIn.Grade` = the custom fade-in level between a pitch-black palette and the specified custom palette.
(63 = fully faded in to palette; 0 = fully faded out to black)

NOTES ON THIS COMMAND:

With this command, you have FULL and free control of all 64 of the fade levels between a black screen and your own custom PP256-based palette! Perfect for doing fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;) !*

FadeCtrlX.PP256

SUB DESCRIPTION:

```
Sub FadeCtrlX.PP256 (PP256PalFile$, R.from, G.from, B.from, FadeIn.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`R.from` = the **red** shade level (0 to 63) to use as your "fade-out"-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your "fade-out"-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your "fade-out"-based color shade.

`FadeIn.Grade` = the custom fade-in level between an entire palette of a resulting color here and the specified custom palette. (63 = fully faded in to palette; 0 = fully faded out to that chosen color)

NOTES ON THIS COMMAND:

With this command, you have FULL and free control of all 64 of the fade levels between a whole palette of the color of your choice and your own custom PP256-based palette!! Perfect for doing such "color-to-custom-palette"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!!! ;) !*

FadeCtrlRangeX.PP256

Sub Description:

```
    Sub FadeCtrlRangeX.PP256 (PP256PalFile$, StartColor, EndColor, R.from,
     G.from, B.from, FadeIn.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to use as your "fade-out"-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your "fade-out"-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your "fade-out"-based color shade.

`FadeIn.Grade` = the custom fade-in level between a palette of a resulting color here and the specified custom palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to that chosen color)

Notes on this Command:

*With this command, you again have FULL and free control of all 64 of the fade levels between the color of your choice and your own custom PP256-based palette, except this time, it is for **any part of the palette**!!! Especially an awesome thing for doing such "color-to-custom-palette"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!!!* d=^_^=b ! *Keep in mind though that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalRotate.PP256

SUB DESCRIPTION:

```
Sub PalRotate.PP256 (PP256PalFile$, StartColor, EndColor, Rotate.Level)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the entire specified custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

NOTES ON THIS COMMAND:

In this command, you can truly custom-rotate the many colors of your own PP256-based palette forwards or backwards, whether you do part of the palette, or even all of it! It is up to you!! :D Remember now that the “EndColor” number ***must*** *be higher than the “StartColor” one in order for this command to work the best!*

PalRotateFC.PP256

SUB DESCRIPTION:

```
Sub PalRotateFC.PP256 (PP256PalFile$, StartColor, EndColor, Rotate.Level,
  FadeIn.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`FadeIn.Grade` = the custom fade-in level between a pitch-black palette and the specified custom palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to black)

NOTES ON THIS COMMAND:

In this command, not only can you truly custom-rotate the many colors of your own PP256-based palette forwards or backwards — whether you do part of the palette or even all of it — but also, you have FULL and free control of all 64 of the fade levels as well within your color range, too!! An awesomely great recommendation for working on your games and graphics demos, I must say!!! ^-^=b ! *Remember, in order for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one!*

PalRotateFCX.PP256

Sub Description:

```
Sub PalRotateFCX.PP256 (PP256PalFile$, StartColor, EndColor, R.from, G.from,
 B.from, Rotate.Level, FadeIn.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`R.from` = the **red** shade level (0 to 63) to use as your "fade-out"-based color shade.

`G.from` = the **green** shade level (0 to 63) to use as your "fade-out"-based color shade.

`B.from` = the **blue** shade level (0 to 63) to use as your "fade-out"-based color shade.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`FadeIn.Grade` = the custom fade-in level between a palette of a resulting color here and the specified custom palette, according *only* to the color-order range that you specify using this command.
(63 = fully faded in to palette; 0 = fully faded out to that selected color)

Notes on this Command:

*Very much the same as PalRotateFC.PP256, except that you are now allowed *as well* custom fades from any single color you wish to your entire PP256-based custom palette, back again, and somewhere in-between, too!!! Now, be sure to try that in your games and graphics demos, as it will do you rather good here!! ^-^ !! Remember, in order for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one!*

Please turn to the very next page for an FB program example of this using the commands PalRotate.PP256, PalRotateFC.PP256, and PalRotateFCX.PP256!!

PROGRAM EXAMPLE # 10:

(This example uses PalRotate.PP256, PalRotateFC.PP256, *and* PalRotateFCX.PP256.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom PP256-based palette!
'--------------------------------------------------------------------------
Pal$ = "PP256Pal1.pal" '<--- This is our custom palette fresh outta PP256
                       '     that we are gonna be using here for this test!

LoadUpPP256Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, we will rotate the palettes around and around!!  ;)
'--------------------------------------------------------

Color 15, 4
Locate 1, 1: ? "Let's spin that PP256 palette wheel!!!"

Color 15, 0
Locate 2, 1: ? "- Now using PalRotate.PP256... -"


'--- Using the color entries 20-235 of our custom PP256-based palette,
'    we rotate the colors of the entire palette forward for a short
'    moment, and then backwards for that very same moment.

Rota = 0
DO
  PalRotate.PP256 Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotate.PP256 Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "- Now using PalRotateFC.PP256... -"


'--- Next, we use the same color entries of that custom PP256 palette
'    to once again rotate the colors of the entire palette forward for a
'    short moment, and then backwards for that same moment.  BUT, we do
'    it now with fade-ins/fade-outs as you will see right here!  ;*)
```

Continues on next page.......

“Program Example #10” continued from last page........

```
Rota = 0
DO
  PalRotateFC.PP256 Pal$, 20, 235, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFC.PP256 Pal$, 20, 235, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "- Now using PalRotateFCX.PP256... -"


'--- And now, we do the same thing as the second one here, only this time,
'    with color-based fade-ins/fade-outs that will interest you real
'    good now, so do get psyched-up for this one!!  :D

Rota = 0
DO
  PalRotateFCX.PP256 Pal$, 20, 235, 54, 10, 32, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFCX.PP256 Pal$, 20, 235, 10, 59, 27, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Rota = 0
DO
  PalRotateFCX.PP256 Pal$, 20, 235, 5, 6, 63, Rota, Rota mod 63
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 600

Rota = 600
DO
  PalRotateFCX.PP256 Pal$, 20, 235, 63, 63, 63, Rota, Rota mod 63
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Color 15, 0
Locate 2, 1: ? "Thanks so much for viewing.  See 'ya!!"


'--- We now end this test indeed, as always, by fading out the palette
'    to nothing and exiting this program.  ^_^=b !

FadeOut.PP256 Pal$, 63

'--- Word up, and check you later!!!  ;*)
```

PalNeg.PP256

SUB DESCRIPTION:

```
Sub PalNeg.PP256 (PP256PalFile$, Switch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`Switch` = the operation of whether or not to switch the screen to a negative (or inverted-colors) version of the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts the palette back to that specified custom one automatically.

NOTES ON THIS COMMAND:

With this command, you can actually switch the screen to a "negative"-based effect of your own custom PP256 palette INSTANTANEOUSLY!

PalNegRange.PP256

SUB DESCRIPTION:

```
Sub PalNegRange.PP256 (PP256PalFile$, StartColor, EndColor, Switch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Switch` = the operation of whether or not to switch the selected order of colors to a negative (or inverted-colors) version of original colors of the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the order indeed becomes switched like that, otherwise this command reverts the specified color-order range back to the original colors of that custom palette automatically.

NOTES ON THIS COMMAND:

*Same workings as PalNeg.PP256, except that you can do *any* part of your custom PP256-based palette you wish at anytime!! :D ! Keep in mind now that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalNegRotate.PP256

SUB DESCRIPTION:

```
Sub PalNegRotate.PP256 (PP256PalFile$, StartColor, EndColor, Rotate.Level)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole inverted version of the specified custom palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

NOTES ON THIS COMMAND:

In this command here, you can truly transform the many colors of your own custom PP256 palette into an inverted-colors version of it, and at the same time custom-rotate it forwards or backwards, whether you do part of the palette, or even all of it!! Rather so awesome!!! ^_^=b *Remember now that the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best!*

PalFadeToNega.PP256

SUB DESCRIPTION:

```
    Sub PalFadeToNega.PP256 (PP256PalFile$, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to a negative (or inverted-colors) version of it.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from your whole custom PP256-based palette all the way to an inverted version of it, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.

PalFadeFromNega.PP256

SUB DESCRIPTION:

```
   Sub PalFadeFromNega.PP256 (PP256PalFile$, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from a negative (or inverted-colors) version of the specified custom palette all the way back to a normal version of it.

NOTES ON THIS COMMAND:

Does the exact opposite of PalFadeToNega.PP256, in that it lets you fade from an inverted version of the entire custom PP256-based palette of your choice right back into a normal version of it once more. Cool stuff! Again right here, higher milliseconds = slower fades; while lower milliseconds = faster fades.

PalFadeRangeToNega.PP256

SUB DESCRIPTION:

```
Sub PalFadeRangeToNega.PP256 (PP256PalFile$, StartColor, EndColor, millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the selected order of colors from the specified custom palette all the way to a negative (or inverted-colors) version of the colors from that very same palette.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want the screen to fade from *any* part of your custom PP256-based palette all the way to an inverted version of that same palette, all in ONE single pass! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades. Also, for this command to work the best, the "EndColor" number* **must** *be higher than the "StartColor" one.*

PalFadeRangeFromNega.PP256

Sub Description:

```
   Sub PalFadeRangeFromNega.PP256 (PP256PalFile$, StartColor, EndColor,
    millisec)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading from the selected order of colors from the negative (or inverted-colors) version of the specified custom palette all the way back to the normal colors from that very same palette.

NOTES ON THIS COMMAND:

*Does the exact opposite of PalFadeRangeToNega.PP256, in that it lets you fade from *any* part of the inverted version of your custom PP256-based palette all the way back to the normal version of that same palette! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades. Also, for this command to work the best, the "EndColor" number* ***must*** *be higher than the "StartColor" one.*

Please turn to the very next page for an FB program example of this using the commands PalNeg.PP256, PalNegRange.PP256, PalNegRotate.PP256, PalFadeToNega.PP256, PalFadeFromNega.PP256, PalFadeRangeToNega.PP256, and PalFadeRangeFromNega.PP256!!

PROGRAM EXAMPLE #11 :

(This example uses PalNeg.PP256, PalNegRange.PP256, PalNegRotate.PP256, PalFadeToNega.PP256, PalFadeFromNega.PP256, PalFadeRangeToNega.PP256, *and* PalFadeRangeFromNega.PP256.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom PP256-based palette!
'--------------------------------------------------------------------------
Pal$ = "PP256Pal1.pal" '<--- This is our custom palette fresh outta PP256
                       '     that we are gonna be using here for this test!

LoadUpPP256Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, let's test out some "negative color"-based palette routines
'on our PP256-based palette that we just loaded, shall we?  ^_^
'------------------------------------------------------------------

Line (0, 0)-(319, 15), 5, BF
Color 15, 5
Locate 1, 1: ? "We inverse our PP256-based palette and"
Locate 2, 1: ? "back again in *many* exciting ways!!!"
sleep 4500


'--- First off, we switch the full entire custom PP256-based palette to a
'    negative of it *instantly*.  Then, let's wait just five (5) seconds
'    before shutting that palette right back to its normal state again and
'    waiting two (2) more seconds.

PalNeg.PP256 Pal$, Yes
sleep 5000
PalNeg.PP256 Pal$, No
sleep 2000


'--- Let's do the exact same thing here using ONLY color entries 16-173.

PalNegRange.PP256 Pal$, 16, 173, Yes
sleep 5000
PalNegRange.PP256 Pal$, 16, 173, No
sleep 2000
```

Continues on next page.......

"Program Example #11" continued from last page........

```
'--- Now, we actually fade our custom PP256-based palette into a full
'    negative of it in a single pass, measuring just 67 milliseconds per
'    fade-step, and then we fade it right back to normal again using that
'    same exact speed!  ;*)

PalFadeToNega.PP256 Pal$, 67
PalFadeFromNega.PP256 Pal$, 67


'--- Let's do the exact same thing here once again using ONLY color
'    entries 64-214, measuring only 36 milliseconds per fade-step!

PalFadeRangeToNega.PP256 Pal$, 64, 214, 36
sleep 5000
PalFadeRangeFromNega.PP256 Pal$, 64, 214, 36
sleep 2000

Line (0, 0)-(319, 15), 5, BF
Color 15, 5
Locate 1, 1: ? "Now, let's rotate around and around the"
Locate 2, 1: ? "inversed-color version of this palette!!"


'--- Using the color entries 20-235 now, we rotate the inversed colors of
'    that entire custom PP256 palette forward for a moment, and then
'    backwards for that very same moment.  Amazing, huh?  ^_^

Rota = 0
DO
  PalNegRotate.PP256 Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalNegRotate.PP256 Pal$, 20, 235, Rota
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "See you once more, and I hope you have"
Locate 2, 1: ? "enjoyed this one!!  Thank you!!!"


'--- Finally, let's wrap this up now by fading the ENTIRE custom PP256
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.PP256 Pal$, 63


'--- Catch you later, and thanks again!!  ^_-=b !
```

PalNegaFadeCtrl.PP256

SUB DESCRIPTION:

```
   Sub PalNegaFadeCtrl.PP256 (PP256PalFile$, FadeToNega.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`FadeToNega.Grade` = the custom fade-in level between the specified custom palette and a negative (or inverted-colors) version of that very same palette.
(63 = fully faded in to the negative of PP256 palette; 0 = fully faded out to normal PP256 palette)

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between your own custom PP256 palette and an inverted version of it! Perfect for doing "negative color"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is!! ;) !*

PalNegaRangeFadeCtrl.PP256

SUB DESCRIPTION:

```
    Sub PalNegaRangeFadeCtrl.PP256 (PP256PalFile$, StartColor, EndColor,
     FadeToNega.Grade)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToNega.Grade` = the custom fade-in level between the selected order of colors from the specified custom palette and a negative (or inverted-colors) version of those colors from that very same palette.
(63 = fully faded in to the negative of PP256 palette; 0 = fully faded out to normal PP256 palette)

NOTES ON THIS COMMAND:

*Same drill as the command PalNegaFadeCtrl.PP256, only it lets you do ANY part of your own PP256-based palette right as you please, too, whenever you like!!! ^-^ !! A *MUST* for doing "negative color"-based fade-ins/fade-outs while the on-screen action of the run-time of your FB project is still going, be it a game or graphics demo or whatever it is, definitely!!! Do not forget, the "EndColor" number* **must** *be higher than the "StartColor" one in order for this command to work the best now.*

PalGreyScl.PP256

SUB DESCRIPTION:

```
   Sub PalGreyScl.PP256 (PP256PalFile$, Switch, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`Switch` = the operation of whether or not to switch the screen to a greyscale version of the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command simply reverses the greyscale effect of that specified palette automatically.

`NegaPalSwitch` = the operation of whether or not to switch the screen *also* to either a greyscale negative version of the specified custom palette or just a regular color negative of that very same palette, depending on the "*Switch*" setting that you have just specified using this command. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that specified palette away from a negative one automatically.

NOTES ON THIS COMMAND:

*With this command, you can actually switch the screen to a "black-and-white"-based effect of your own custom PP256-based palette INSTANTANEOUSLY! At the same time, you can even make a negative of the very same palette *in addition* to that, as well!! ^-^ !!*

PalGreySclRange.PP256

SUB DESCRIPTION:

```
Sub PalGreySclRange.PP256 (PP256PalFile$, StartColor, EndColor, Switch,
 NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Switch` = the operation of whether or not to switch the selected order of colors to a greyscale version of the colors from the specified custom palette. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command simply reverses the greyscale effect of those selected colors from that specified palette automatically.

`NegaPalSwitch` = the operation of whether or not to switch the selected order of colors *also* to either a greyscale negative version of the specified custom palette or just a regular color negative of those original colors from that very same palette, depending on the "*Switch*" setting that you have just specified using this command. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that same order of colors of that specified palette away from a negative one automatically.

NOTES ON THIS COMMAND:

*Same deal as PalGreyScl.PP256, except that you can use *any* part of your PP256-based custom palette you wish at anytime! :D Keep in mind now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalGreySclRotate.PP256

Sub Description:

```
Sub PalGreySclRotate.PP256 (PP256PalFile$, StartColor, EndColor,
 Rotate.Level, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`Rotate.Level` = the number position in which the selected order of colors will rotate around and around the whole greyscale version of the custom specified palette. Increasing and increasing numbers will move the colors forward, while decreasing and decreasing numbers move the colors backward.

`NegaPalSwitch` = the operation of whether or not to switch that same order of colors *also* to a negative (or inverted-colors) of the greyscale of the original colors from the specified custom palette that is being rotated around. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen becomes switched like that, otherwise this command reverts that same selected order of colors of that specified palette away from a negative one automatically, leaving those selected colors just regular greyscale-based ones of that palette indeed.

Notes on this Command:

In this command here, you can truly transform the many colors of your very own custom PP256 palette into a greyscale version of it, and at the same time custom-rotate it forwards or backwards, whether you do part of the palette, or even all of it!! In addition, you can even make a negative of a greyscale of that exact same palette as well ***while rotating it****!!! Rather white-hot stuff, wouldn't you say? :*D !! Remember now that the "EndColor" number* ***must*** *be higher than the "StartColor" one in order for this command to work the best!*

PalFadeToGreyScl.PP256

Sub Description:

```
   Sub PalFadeToGreyScl.PP256 (PP256PalFile$, millisec, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way to a greyscale version of it.

`NegaPalSwitch` = the operation of whether or not to fade the screen to *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen will become like that, otherwise this command just lets you fade that chosen palette into a normal greyscale version of it automatically.

Notes on this Command:

This command lets you actually determine just how fast or slow you want the screen to fade from the whole custom PP256 palette of yours all the way to a greyscale version of it, all in ONE single pass! Alternatively, on that same pass, you can even fade to a negative of a greyscale of that very palette, too!! ^_^=b ! *Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeFromGreyScl.PP256

SUB DESCRIPTION:

```
Sub PalFadeFromGreyScl.PP256 (PP256PalFile$, millisec, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the greyscale version of the specified custom palette all the way to a normal version of it.

`NegaPalSwitch` = the operation of whether or not to fade the screen from *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette straight to the original "normal colors"-based version of that same palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the screen will become like that, otherwise this command just lets you fade the normal greyscale version of that chosen palette back to its original colors automatically.

NOTES ON THIS COMMAND:

This command lets you actually determine just how fast or slow you want the screen to fade from the greyscale version of your custom PP256 palette all the way back to its original color-based state, all in ONE single pass! Alternatively, on that exact same pass, you can even fade from a greyscale negative of that very palette over to its normal (not negative) and original colors, too!! ^_^=b ! Remember, higher milliseconds = slower fades; while lower milliseconds = faster fades.

PalFadeRangeToGreyScl.PP256

SUB DESCRIPTION:

```
    Sub PalFadeRangeToGreyScl.PP256 (PP256PalFile$, StartColor, EndColor,
     millisec, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the specified custom palette all the way to a greyscale version of it.

`NegaPalSwitch` = the operation of whether or not to fade the selected order of colors to *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette, in any speed that you have just specified in the "*millisec*" setting. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and that color order indeed will become like that, otherwise this command just lets you fade those same selected colors from that specified palette into a normal greyscale version of it automatically.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want the screen to fade from*any* part of your custom PP256-based palette all the way to a greyscale version of it, all in ONE single pass! Alternatively, on that same pass, you can even fade any part of that very same palette to a greyscale negative of it, too!! ^_^=b ! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

PalFadeRangeFromGreyScl.PP256

SUB DESCRIPTION:

```
Sub PalFadeRangeFromGreyScl.PP256 (PP256PalFile$, StartColor, EndColor,
 millisec, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the greyscale version of the specified custom palette all the way to a normal version of it.

`NegaPalSwitch` = the operation of whether or not to fade the selected order of colors from *rather* a negative (or inverted-colors) of the greyscale version of the specified custom palette straight to the original “normal colors”-based version of that same palette, in any speed that you have just specified in the “*millisec*” setting. Pass **1** or higher (or either “*FBPM.True*” or “*Yes*”) here, and the color order here will become like that, otherwise this command just lets you fade those exact same selected colors from the normal greyscale version of that specified palette back to its original colors automatically.

NOTES ON THIS COMMAND:

*This command lets you actually determine just how fast or slow you want to fade from *any* part of the greyscale version of your very own PP256 custom palette all the way back to its original color-based state, all in ONE single pass! Alternatively, on **any** portion of that same palette on a single pass, you can even fade from a greyscale negative of it over to its normal (not negative) and original colors, too!! ^_^=b ! And do not forget here, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

Please turn to the very next page for an FB program example of this using the commands PalGreyScl.PP256, PalGreySclRange.PP256, PalGreySclRotate.PP256, PalFadeToGreyScl.PP256, PalFadeFromGreyScl.PP256, PalFadeRangeToGreyScl.PP256, and PalFadeRangeFromGreyScl.PP256!!

PROGRAM EXAMPLE # 12:

(This example uses PalGreySclPP256, PalGreySclRange.PP256, PalGreySclRotate.PP256, PalFadeToGreyScl.PP256, PalFadeFromGreyScl.PP256, PalFadeRangeToGreyScl.PP256, *and* PalFadeRangeFromGreyScl.PP256.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries from a custom PP256-based palette!
'---------------------------------------------------------------------------
Pal$ = "PP256Pal1.pal" '<--- This is our custom palette fresh outta PP256
                       '     that we are gonna be using here for this test!

LoadUpPP256Pal Pal$  '<--- *VERY* important that we load this baby in here!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Now, do some exciting greyscale palette routines here for our
'custom PP256-based palette!!!  ;*) !
'--------------------------------------------------------------

Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Let's make the PP256 custom palette"
Locate 2, 1: ? "greyscale and back again and more!!!"
Sleep 4500


'--- First off, we switch the full entire custom PP256-based palette to a
'    greyscale of it *instantly*.  Then between brief moments, we inverse
'    it into a greyscale-based negative, then a color negative, and then
'    back to a normal version of it again.

PalGreyScl.PP256 Pal$, Yes, No
Sleep 3000
PalGreyScl.PP256 Pal$, Yes, Yes
Sleep 3000
PalGreyScl.PP256 Pal$, No, Yes
Sleep 3000
PalGreyScl.PP256 Pal$, No, No
Sleep 3000



'--- Let's do the exact same thing here using ONLY color entries 16-200.

PalGreySclRange.PP256 Pal$, 16, 200, Yes, No
Sleep 3000
PalGreySclRange.PP256 Pal$, 16, 200, Yes, Yes
Sleep 3000
PalGreySclRange.PP256 Pal$, 16, 200, No, Yes
Sleep 3000
PalGreySclRange.PP256 Pal$, 16, 200, No, No
Sleep 3000
```

Continues on next page.......

“Program Example #12” continued from last page........

```
'--- Now, we actually fade our custom PP256 palette into a full
'    greyscale of it in a single pass, measuring just 72 milliseconds per
'    fade-step, wait two seconds, and then we fade it right back to normal
'    again using that same exact speed!  ;*)

PalFadeToGreyScl.PP256 Pal$, 72, No
sleep 2000
PalFadeFromGreyScl.PP256 Pal$, 72, No


'--- Let's do it again, but this time, we fade to an entire *negative
'    greyscale* of that custom PP256 palette and then back to normal
'    again in just the next two passes, measuring at the same exact speed
'    per fade-step.

PalFadeToGreyScl.PP256 Pal$, 72, Yes
sleep 2000
PalFadeFromGreyScl.PP256 Pal$, 72, Yes


'--- We repeat the "greyscale/negative greyscale"-based fades once more
'    here, only this time, we use color entries 1-166 of our custom
'    PP256 palette now, measuring only 17 milliseconds per fade-step.

PalFadeRangeToGreyScl.PP256 Pal$, 1, 166, 17, No
sleep 2000
PalFadeRangeFromGreyScl.PP256 Pal$, 1, 166, 17, No

PalFadeRangeToGreyScl.PP256 Pal$, 1, 166, 17, Yes
sleep 2000
PalFadeRangeFromGreyScl.PP256 Pal$, 1, 166, 17, Yes



Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Now rotating around and around the"
Locate 2, 1: ? "greyscale version of this palette!!"


'--- Using the color entries 20-235 now, we rotate the greyscaled colors of
'    that exact PP256 palette forward for a moment, and then backwards
'    for that very same moment.  Rather amazing, huh?  ^_^

Rota = 0
DO
  PalGreySclRotate.PP256 Pal$, 20, 235, Rota, No
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalGreySclRotate.PP256 Pal$, 20, 235, Rota, No
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Rota = 0
Line (0, 0)-(319, 15), 8, BF
Color 15, 8
Locate 1, 1: ? "Let's also do the same for the negative"
Locate 2, 1: ? "greyscale version of this palette!!!"
```

Continues on next page.......

“Program Example #12” *still* continued from last page........

```
'--- Same thing again, but this time rotating around and around the
'    negative greyscale of our same PP256-based custom palette!!

DO
  PalGreySclRotate.PP256 Pal$, 20, 235, Rota, Yes
  sleep 5
  Rota = Rota + 1
LOOP until Rota >= 1200

Rota = 1200
DO
  PalGreySclRotate.PP256 Pal$, 20, 235, Rota, Yes
  sleep 5
  Rota = Rota - 1
LOOP until Rota <= 0

Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Be seeing you again, and thank 'ya once"
Locate 2, 1: ? "more!!  Splendid fortune to you all!!!"


'--- Finally, let's close this now by fading our custom PP256
'    palette slowly but *all the way* out in only one pass, using
'    just 63 milliseconds per fade-step!  ;*)

FadeOut.PP256 Pal$, 63


'--- And now, all finished!!  d=^-^=b !
```

PalGreyFadeCtrl.PP256

Sub Description:

```
Sub PalGreyFadeCtrl.PP256 (PP256PalFile$, FadeToGrey.Grade, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`FadeToGrey.Grade` = the custom fade-in level between the specified custom palette and a greyscale version of that very same palette.
(63 = fully faded in to the greyscale of PP256 palette; 0 = fully faded out to normal PP256 palette)

`NegaPalSwitch` = the operation of whether or not to set the custom fade-in level to *rather* fade between the specified custom palette and a negative greyscale version of it. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the entire palette will become like that, otherwise this command just sets the custom fade-in level to fade between that same specified palette and a normal greyscale version of it.

Notes on this Command:

With this command, you have FULL and free reign over all 64 of the fade levels between your own custom PP256-based palette and a greyscaled version of it! Alternatively, you can control those same fade levels between all the original colors of that very palette and an actual greyscale negative of it!! Perfect for games and graphics demos in FB!!! ;) !*

PalGreyRangeFadeCtrl.PP256

SUB DESCRIPTION:

```
    Sub PalGreyRangeFadeCtrl.PP256 (PP256PalFile$, StartColor, EndColor,
     FadeToGrey.Grade, NegaPalSwitch)
```

`PP256PalFile$` = the filename for the custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToGrey.Grade` = the custom fade-in level between the selected colors of the specified custom palette and a greyscale version of that very same palette.
(63 = fully faded in to the greyscale of PP256 palette; 0 = fully faded out to normal PP256 palette)

`NegaPalSwitch` = the operation of whether or not to set the custom fade-in level to *rather* fade between the specified custom palette and a negative greyscale version of it. Pass **1** or higher (or either "*FBPM.True*" or "*Yes*") here, and the selected order of colors will become like that, otherwise this command just sets the custom fade-in level to fade between that same order of colors of that exact specified palette *and* a normal greyscale version of them.

NOTES ON THIS COMMAND:

Same entire drill as the previous command PalGreyFadeCtrl.PP256, except that you can actually do ***any*** *part of your custom PP256 palette rather instead of just all of it alone!!!* ^_-=b !!

—— PALETTE CROSSFADING ROUTINES

Or, routines to cause your palettes to fade between each other, leaving you drag-out speechless!!

NOTE: THESE NEXT ROUTINES SUPPORT CROSSFADING <u>ONLY</u> BETWEEN THE FOLLOWING:

A 256-COLOR GFXLIB 2 PALETTE —— 768-BYTE PALETTE.

When these following commands are applied in your FB programs, the colors can **automatically** *change to the crossfading-based shades of the default 256-color GFXlib 2 palette and/or a custom 768-byte palette, so please be *very* careful if you are using any custom 256-color palette(s) of your own choice! ;*) !*

CrossFade.Default_to_768Pal

SUB DESCRIPTION:

```
    Sub CrossFade.Default_to_768Pal (PalFile$, millisec)
```

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade to. Must be **<u>only 768 bytes</u>**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the default *GFXlib 2* palette all the way to the specified custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will <u>actually</u> let you fade the entire screen from the default 256-color GFXlib 2 palette all the way to your own custom 768-byte palette of your choice, all in *any* speed that you want, all in ONE single pass!!! ^_^=b Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFade.768Pal_to_Default

SUB DESCRIPTION:

```
Sub CrossFade.768Pal_to_Default (PalFile$, millisec)
```

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way back to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

Same exact features as CrossFade.Default_to_768Pal*, *except* that it rather lets you fade the entire screen from your own custom 768-byte palette right back to the default 256-color* GFXlib 2 *palette in its entirety, all in ONE single pass!!! ;*) Again folks, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.Default_to_768Pal

SUB DESCRIPTION:

```
    Sub CrossFadeRange.Default_to_768Pal (StartColor, EndColor, PalFile$,
     millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade to. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the default *GFXlib 2* palette all the way to the specified custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the color-order range within your screen from the default 256-color GFXlib 2 palette all the way to your own custom 768-byte palette of your choice, all in *any* speed that you want, all in ONE single pass!!! ^_-=b ! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.768Pal_to_Default

SUB DESCRIPTION:

```
    Sub CrossFadeRange.768Pal_to_Default (StartColor, EndColor, PalFile$,
     millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the specified custom palette all the way back to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

*Same exact features as CrossFadeRange.Default_to_768Pal, *except* that it rather lets you fade the color-order range within your screen from your own custom 768-byte palette right back to the default 256-color GFXlib 2 palette in its entirety, all in ONE single pass!!! ;*) Once more, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

CrossFadeCtrl.Default_to_768Pal

SUB DESCRIPTION:

```
Sub CrossFadeCtrl.Default_to_768Pal (FadeTo768Pal.Grade, PalFile$)
```

`FadeTo768Pal.Grade` = the custom fade-in level between the default *GFXlib 2* palette and the custom 768-byte palette that you choose from the "*PalFile$*" setting within this very command.
(63 = fully faded in to 768-byte palette; 0 = fully faded out to default *GFXlib 2* palette)

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between the default GFXlib 2 palette and your own custom 768-byte palette!!! Perfect for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! d=^_^=b !!

CrossFadeRangeCtrl.Default_to_768Pal

SUB DESCRIPTION:

```
   Sub CrossFadeRangeCtrl.Default_to_768Pal (StartColor, EndColor,
    FadeTo768Pal.Grade, PalFile$)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeTo768Pal.Grade` = the custom fade-in level between the selected order of colors from the default *GFXlib 2* palette and that very same color order from the custom 768-byte palette that you choose using the "*PalFile$*" setting within this very command.
(63 = fully faded in to 768-byte palette; 0 = fully faded out to default *GFXlib 2* palette)

`PalFile$` = the filename for the custom 256-color palette (.pal) of your choice. Must be **only 768 bytes**, please!

NOTES ON THIS COMMAND:

Same great features as CrossFadeCtrl.Default_to_768Pal, except that you can REALLY fade any color range you want between the default GFXlib 2 palette and your own custom 768-byte palette!!! Grandly awesome stuff for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! d=^_^=b !!

Please turn to the very next page for an FB program example of this using the commands CrossFade.Default_to_768Pal, CrossFade.768Pal_to_Default, CrossFadeRange.Default_to_768Pal, CrossFadeRange.768Pal_to_Default, CrossFadeCtrl.Default_to_768Pal, and CrossFadeRangeCtrl.Default_to_768Pal!!

PROGRAM EXAMPLE # 13:

(This example uses CrossFade.Default_to_768Pal, CrossFade.768Pal_to_Default, CrossFadeRange.Default_to_768Pal, CrossFadeRange.768Pal_to_Default, CrossFadeCtrl.Default_to_768Pal, *and* CrossFadeRangeCtrl.Default_to_768Pal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries for the default GFXlib 2 palette.
'------------------------------------------------------------------------
For DrawPalette = 0 to 255
  Line (DrawPalette, 8)-(DrawPalette, 200), DrawPalette
Next


'Then, we add our own custom 256 color palette that has only 768 bytes in
'it!  ;*)
'------------------------------------------------------------------------
Pal$ = "CustomPalette_01.pal" '<--- This is a 768-byte custom palette that
                              '     we are gonna be using for this test!


'Now, let's AMAZE some people with some brain-boggling palette crossfades
'between the default GFXlib 2 palette and our custom 768-byte one right
'here!!!  ^_- !
'------------------------------------------------------------

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now crossfading between the GFXlib 2"
Locate 2, 1: ? "palette and a custom 768-byte palette!!"
sleep 3000


'--- First off, let's fade the default GFXlib 2 palette all the way into
'    that custom 768-byte palette in one pass, using only 32 milliseconds
'    per fade-step!  And then, we wait two seconds before we fade the
'    palette back to normal again at the same exact speed.  Then we wait
'    two more seconds.  ;*)

CrossFade.Default_to_768Pal Pal$, 32
sleep 2000
CrossFade.768Pal_to_Default Pal$, 32
sleep 2000


'--- Now let's do it all again, but this time, using color entries 40-156
'    and just 15 milliseconds per fade-step!

CrossFadeRange.Default_to_768Pal 40, 156, Pal$, 15
sleep 2000
CrossFadeRange.768Pal_to_Default 40, 156, Pal$, 15
sleep 2000
```

Continues on next page.......

“Program Example #13” continued from last page........

```
'--- Here comes the fun part: we crossfade the same palettes to and fro
'    *while* we draw the lines on the screen, using the command
'    "CrossFadeCtrl.Default_to_768Pal"!!!  :D !

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now let's draw some lines while we"
Locate 2, 1: ? "crossfade between the same palettes!!"

For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeCtrl.Default_to_768Pal Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeCtrl.Default_to_768Pal 63 - Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next


'--- Let us do that again, but this time, using the command
'    "CrossFadeRangeCtrl.Default_to_768Pal" to fade color entries 100-250
'    between the GFXlib 2 default palette and our 768-byte palette!!!
'    ^_-=b !

For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.Default_to_768Pal 100, 250, Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.Default_to_768Pal 100, 250, 63 - Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Be seeing you again now, and I thank"
Locate 2, 1: ? "you so much here for watching!!!"


'--- Finally, let's wrap it up as always by fading the ENTIRE default
'    GFXlib 2 palette slowly but *all the way* out in only one pass,
'    using just 63 milliseconds per fade-step!  ;*)

FadeOut.DefaultPal 63


'--- Chao!!!  ^-^ !
```

NOTE: THESE NEXT ROUTINES SUPPORT CROSSFADING ONLY BETWEEN THE FOLLOWING:

A 768-BYTE PALETTE → ANOTHER 768-BYTE PALETTE.

When these following commands are applied in your FB programs, the colors can **automatically** *change to the crossfading-based shades of one or even TWO custom 768-byte palette(s) that you choose, so please be *very* careful if you are using any custom 256-color palette(s) of your own choice! ;*) !*

CrossFade.768Pal_to_768Pal

SUB DESCRIPTION:

```
Sub CrossFade.768Pal_to_768Pal (PalFile.from$, PalFile.to$, millisec)
```

`PalFile.from$` = the filename for your **first** custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PalFile.to$` = the filename for your **second** custom 256-color palette (.pal) to fade to. Must also be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the *first* specified custom palette all the way to the *second* specified custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the entire screen from the first 768-byte custom palette right into the second 768-byte custom palette, all in *any* speed that you want, all in ONE single pass!!! ^_^ =b Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.768Pal_to_768Pal

SUB DESCRIPTION:

```
    Sub CrossFadeRange.768Pal_to_768Pal (StartColor, EndColor, PalFile.from$,
     PalFile.to$, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PalFile.from$` = the filename for your **first** custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PalFile.to$` = the filename for your **second** custom 256-color palette (.pal) to fade to. Must also be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the *first* specified custom palette all the way to the *second* specified custom palette.

NOTES ON THIS COMMAND:

*Same thing as CrossFade.768Pal_to_768Pal, *except* that it rather lets you fade the color-order range within your screen from your first custom 768-byte palette to your second custom 768-byte palette in its entirety, all in ONE single pass!!! ;*) Again, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

CrossFadeCtrl.768Pal_to_768Pal

SUB DESCRIPTION:

```
    Sub CrossFadeCtrl.768Pal_to_768Pal (FadeTo2nd768Pal.Grade, PalFile.from$,
     PalFile.to$)
```

`FadeTo2nd768Pal.Grade` = the custom fade-in level between your *first* selectable custom 768-byte palette and your *second* selectable custom 768-byte palette.
(63 = fully faded in to *second* 768-byte palette; 0 = fully faded out to *first* 768-byte palette)

`PalFile.from$` = the filename for your **first** custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PalFile.to$` = the filename for your **second** custom 256-color palette (.pal) to fade to. Must also be **only 768 bytes**, please!

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between your first selectable custom 768-byte palette and your second selectable custom 768-byte palette!!! Wow!! It is such wildly excellent stuff for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! :D !

CrossFadeRangeCtrl.768Pal_to_768Pal

SUB DESCRIPTION:

```
   Sub CrossFadeRangeCtrl.768Pal_to_768Pal (StartColor, EndColor,
    FadeTo2nd768Pal.Grade, PalFile.from$, PalFile.to$)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeTo2nd768Pal.Grade` = the custom fade-in level between the selected order of colors from your *first* selectable custom 768-byte palette and that very same color order from your *second* selectable custom 768-byte palette.
(63 = fully faded in to *second* 768-byte palette; 0 = fully faded out to *first* 768-byte palette)

`PalFile.from$` = the filename for your **first** custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PalFile.to$` = the filename for your **second** custom 256-color palette (.pal) to fade to. Must also be **only 768 bytes**, please!

NOTES ON THIS COMMAND:

Same great features as CrossFadeCtrl.768Pal_to_768Pal*, except that you can REALLY fade any color range you want between your* first *selectable custom 768-byte palette and your* second *selectable custom 768-byte palette!!!* ***Richly spectactular stuff*** *for your games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!!* d=^_^=b !!

Please turn to the very next page for an FB program example of this using the commands CrossFade.768Pal_to_768Pal, CrossFadeRange.768Pal_to_768Pal, CrossFadeCtrl.768Pal_to_768Pal, *and* CrossFadeRangeCtrl.768Pal_to_768Pal*!!*

PROGRAM EXAMPLE #14:

(This example uses CrossFade.768Pal_to_768Pal, CrossFadeRange.768Pal_to_768Pal, CrossFadeCtrl.768Pal_to_768Pal, *and* CrossFadeRangeCtrl.768Pal_to_768Pal.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we get in our TWO custom 768-byte palettes for use at once, and then
'draw up the 256 color entries for this dynamic duo!!  ^_^=b !
'---------------------------------------------------------------------------
Pal.1$ = "CustomPalette_01.pal" '<--- This is our FIRST custom 768 palette
                                '     that we will bring out here!!

Pal.2$ = "CustomPalette_02.pal" '<--- This is our SECOND custom 768 palette
                                '     that we will bring out here, also!!

LoadUp768Pal Pal.1$  '<--- Now loading custom palette #1!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Right here, let's blow 'em away in such WONDROUS AWE with some wildly
'awesome palette crossfades between our two custom 768-byte palettes,
'starting now!!!  ^_- !
'-------------------------------------------------------------

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now, let's crossfade between our TWO"
Locate 2, 1: ? "custom 768-byte palettes at once!!!"
sleep 3000


'--- First off, let's fade the first custom 768-byte palette all the way
'    into our second one here in just one pass, using only 41 milliseconds
'    per fade-step!  And then, we wait two seconds before we fade the
'    palette back our first again at the same exact speed.  Then we wait
'    two more seconds.  ;*)

CrossFade.768Pal_to_768Pal Pal.1$, Pal.2$, 41
sleep 2000
CrossFade.768Pal_to_768Pal Pal.2$, Pal.1$, 41
sleep 2000


'--- Now let's do it all again, but this time, using color entries 40-156
'    and just 22 milliseconds per fade-step!

CrossFadeRange.768Pal_to_768Pal 40, 156, Pal.1$, Pal.2$, 22
sleep 2000
CrossFadeRange.768Pal_to_768Pal 40, 156, Pal.2$, Pal.1$, 22
sleep 2000
```

Continues on next page.......

“Program Example #14” continued from last page........

```
'--- Here comes the rockin' part: we crossfade the same palettes to and
'    fro *while* we draw the lines on the screen, using the command
'    "CrossFadeCtrl.768Pal_to_768Pal"!!!  :D !

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now let's draw some lines while we"
Locate 2, 1: ? "crossfade between the same palettes!!"

For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeCtrl.768Pal_to_768Pal Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeCtrl.768Pal_to_768Pal 63 - Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next


'--- Let us do that again, but this time, using the command
'    "CrossFadeRangeCtrl.768Pal_to_768Pal" to fade color entries 100-250
'    between our first 768-byte palette and our second!!!  ^_-=b !

For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.768Pal_to_768Pal 100, 250, Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.768Pal_to_768Pal 100, 250, 63 - Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Thanks so rather much for tuning in."
Locate 2, 1: ? "God bless you all and good night!!!"


'--- Finally, let's wrap it up as always by fading our first 768-byte
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.768Pal Pal.1$, 63


'--- Sayanora, ladies and gentlemen!!!  ^-^ !
```

NOTE: THESE NEXT ROUTINES SUPPORT CROSSFADING ONLY BETWEEN THE FOLLOWING:
A PP256 PALETTE — ANOTHER PP256 PALETTE.

When these following commands are applied in your FB programs, the colors can **automatically** *change to the crossfading-based shades of one or even TWO custom PixelPlus 256 (or PP256) palette(s) that you choose, so please be *very* careful if you are using any custom 256-color palette(s) of your own choice! ;*) !*

CrossFade.PP256_to_PP256

SUB DESCRIPTION:

```
    Sub CrossFade.PP256_to_PP256 (PP256PalFile.from$, PP256PalFile.to$,
     millisec)
```

`PP256PalFile.from$` = the filename for your **first** custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`PP256PalFile.to$` = the filename for your **second** custom PP256-based palette (.pal) to fade to. Must also be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the *first* specified custom palette all the way to the *second* specified custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the entire screen from the first PP256-based custom palette right into the second PP256-based custom palette, all in *any* speed that you want, all in ONE single pass!!! ^_^=b Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.PP256_to_PP256

SUB DESCRIPTION:

```
    Sub CrossFadeRange.PP256_to_PP256 (StartColor, EndColor,
     PP256PalFile.from$, PP256PalFile.to$, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PP256PalFile.from$` = the filename for your **first** custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`PP256PalFile.to$` = the filename for your **second** custom PP256-based palette (.pal) to fade to. Must also be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the *first* specified custom palette all the way to the *second* specified custom palette.

NOTES ON THIS COMMAND:

*Same thing as CrossFade.PP256_to_PP256, *except* that it rather lets you fade the color-order range within your screen from your first custom PP256 palette to your second custom PP256 palette in its entirety, all in ONE single pass!!! ;*) Again, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

CrossFadeCtrl.PP256_to_PP256

SUB DESCRIPTION:

```
   Sub CrossFadeCtrl.PP256_to_PP256 (FadeTo2ndPP256Pal.Grade,
    PP256PalFile.from$, PP256PalFile.to$)
```

`FadeTo2ndPP256Pal.Grade` = the custom fade-in level between your *first* selectable custom PP256 palette and your *second* selectable custom PP256 palette.
(63 = fully faded in to *second* PP256 palette; 0 = fully faded out to *first* PP256 palette)

`PP256PalFile.from$` = the filename for your **first** custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`PP256PalFile.to$` = the filename for your **second** custom PP256-based palette (.pal) to fade to. Must also be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between your first selectable custom PP256-based palette and your second selectable custom PP256-based palette!!! Wow!! It is such wildly excellent stuff for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! :D !

CrossFadeRangeCtrl.PP256_to_PP256

SUB DESCRIPTION:

```
    Sub CrossFadeRangeCtrl.PP256_to_PP256 (StartColor, EndColor,
     FadeTo2ndPP256Pal.Grade, PP256PalFile.from$, PP256PalFile.to$)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeTo2ndPP256Pal.Grade` = the custom fade-in level between the selected order of colors from your *first* selectable custom PP256 palette and that very same color order from your *second* selectable custom PP256 palette.
(63 = fully faded in to *second* PP256 palette; 0 = fully faded out to *first* PP256 palette)

`PP256PalFile.from$` = the filename for your **first** custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`PP256PalFile.to$` = the filename for your **second** custom PP256-based palette (.pal) to fade to. Must also be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

Same great features as CrossFadeCtrl.PP256_to_PP256*, except that you can REALLY fade any color range you want between your* first *selectable custom PP256 palette and your* second *selectable custom PP256 palette!!!* ***Richly spectactular stuff*** *for your games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!!* d=^_^=b !!

Please turn to the very next page for an FB program example of this using the commands CrossFade.PP256_to_PP256*,* CrossFadeRange.PP256_to_PP256*,* CrossFadeCtrl.PP256_to_PP256*, and* CrossFadeRangeCtrl.PP256_to_PP256*!!*

PROGRAM EXAMPLE # 15:

(This example uses CrossFade.PP256_to_PP256, CrossFadeRange.PP256_to_PP256, CrossFadeCtrl.PP256_to_PP256, *and* CrossFadeRangeCtrl.PP256_to_PP256.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we get in our TWO custom PP256 palettes for use at once, and then
'draw up the 256 color entries for this couple of them!!  ^_^=b !
'--------------------------------------------------------------------------
Pal.1$ = "PP256Pal1.pal" '<--- This is our FIRST custom PP256 palette that
                         '     we will bring out here!!

Pal.2$ = "PP256Pal2.pal" '<--- This is our SECOND custom PP256 palette that
                         '     we will bring out here, also!!

LoadUpPP256Pal Pal.1$  '<--- Now loading custom palette #1!

For DrawPalette = 0 to 255
  Line (DrawPalette, 16)-(DrawPalette, 200), DrawPalette
Next


'Right here, let's intensely wow 'em like wildfire with some truly
'awesome palette crossfades between our two custom PP256-based palettes,
'starting now!!!  ^_- !
'------------------------------------------------------------

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now, let's crossfade between our TWO"
Locate 2, 1: ? "custom PP256-based palettes at once!!!"
sleep 3000


'--- First off, let's fade the first custom PP256 palette all the way
'    into our second one here in just one pass, using only 36 milliseconds
'    per fade-step!  And then, we wait two seconds before we fade the
'    palette back our first again at the same exact speed.  Then we wait
'    two more seconds.  ;*)

CrossFade.PP256_to_PP256 Pal.1$, Pal.2$, 36
sleep 2000
CrossFade.PP256_to_PP256 Pal.2$, Pal.1$, 36
sleep 2000


'--- Now let's do it all again, but this time, using color entries 40-156
'    and just 18 milliseconds per fade-step!

CrossFadeRange.PP256_to_PP256 40, 156, Pal.1$, Pal.2$, 18
sleep 2000
CrossFadeRange.PP256_to_PP256 40, 156, Pal.2$, Pal.1$, 18
sleep 2000
```

Continues on next page.......

“Program Example #15” continued from last page........

```
'--- Here comes the best part: we crossfade the same palettes to and
'    fro *while* we draw some pixels on the screen, using the command
'    "CrossFadeCtrl.PP256_to_PP256"!!!  :D !

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now let's draw some circles while we"
Locate 2, 1: ? "crossfade between the same palettes!!"

For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeCtrl.PP256_to_PP256 Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeCtrl.PP256_to_PP256 63 - Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next


'--- Let us do that again, but this time, using the command
'    "CrossFadeRangeCtrl.PP256_to_PP256" to fade color entries 100-250
'    between our first PP256-based palette and our second!!!  ^_-=b !

For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.PP256_to_PP256 100, 250, Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.PP256_to_PP256 100, 250, 63 - Fade, Pal.1$, Pal.2$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next
Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Be seeing you all real soon, and I thank"
Locate 2, 1: ? "you for such a wonderful time!!  Chao!!!"


'--- Finally, let's wrap it up as always by fading our first PP256-based
'    palette slowly but *all the way* out in only one pass, using just 63
'    milliseconds per fade-step!  ;*)

FadeOut.PP256 Pal.1$, 63


'--- Later!!!  ^^ !
```

NOTE: THESE NEXT ROUTINES SUPPORT CROSSFADING ONLY BETWEEN THE FOLLOWING:

A 256-COLOR GFXLIB 2 PALETTE ⟷ PP256 PALETTE.

When these following commands are applied in your FB programs, the colors can **automatically** *change to the crossfading-based shades of the default 256-color GFXlib 2 palette and/or a custom PixelPlus 256 (or PP256) palette, so please be *very* careful if you are using any custom 256-color palette(s) of your own choice! ;*) !*

CrossFade.Default_to_PP256

SUB DESCRIPTION:

```
Sub CrossFade.Default_to_PP256 (PP256PalFile$, millisec)
```

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade to. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the default *GFXlib 2* palette all the way to the specified custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the entire screen from the default 256-color GFXlib 2 palette all the way to your own custom PP256-based palette of your choice, all in *any* speed that you want, all in ONE single pass!!! ^_^=b Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFade.PP256_to_Default

SUB DESCRIPTION:

```
    Sub CrossFade.PP256_to_Default (PP256PalFile$, millisec)
```

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom palette all the way back to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

*Same exact features as CrossFade.Default_to_PP256, *except* that it rather lets you fade the entire screen from your own custom PP256 palette right back to the default 256-color GFXlib 2 palette in its entirety, all in ONE single pass!!! ;*) Again folks, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.Default_to_PP256

SUB DESCRIPTION:

```
Sub CrossFadeRange.Default_to_PP256 (StartColor, EndColor, PP256PalFile$,
 millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade to. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the default *GFXlib 2* palette all the way to the specified custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the color-order range within your screen from the default 256-color GFXlib 2 palette all the way to your own custom PP256 palette of your choice, all in *any* speed that you want, all in ONE single pass!!! ^_-=b ! Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.PP256_to_Default

SUB DESCRIPTION:

```
Sub CrossFadeRange.PP256_to_Default (StartColor, EndColor, PP256PalFile$,
 millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the specified custom palette all the way back to the default *GFXlib 2* palette.

NOTES ON THIS COMMAND:

*Same exact features as CrossFadeRange.Default_to_PP256, *except* that it rather lets you fade the color-order range within your screen from your own custom PP256 palette right back to the default 256-color GFXlib 2 palette in its entirety, all in ONE single pass!!! ;*) Once more, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

CrossFadeCtrl.Default_to_PP256

SUB DESCRIPTION:

```
Sub CrossFadeCtrl.Default_to_PP256 (FadeToPP256Pal.Grade, PP256PalFile$)
```

`FadeToPP256Pal.Grade` = the custom fade-in level between the default *GFXlib 2* palette and the custom PP256 palette that you choose from the "*PP256PalFile$*" setting within this very command.
(63 = fully faded in to PP256 palette; 0 = fully faded out to default *GFXlib 2* palette)

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between the default GFXlib 2 *palette and your own PP256-based custom palette!!! Perfect for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!!*
d=^_^=b !!

CrossFadeRangeCtrl.Default_to_PP256

SUB DESCRIPTION:

```
   Sub CrossFadeRangeCtrl.Default_to_PP256 (StartColor, EndColor,
    FadeToPP256Pal.Grade, PP256PalFile$)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToPP256Pal.Grade` = the custom fade-in level between the selected order of colors from the default *GFXlib 2* palette and that very same color order from the custom PP256 palette that you choose from the "*PP256PalFile$*" setting within this very command.
(63 = fully faded in to PP256 palette; 0 = fully faded out to default *GFXlib 2* palette)

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) of your choice. Must be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

Same great features as CrossFadeCtrl.Default_to_PP256, except that you can REALLY fade any color range you want between the default GFXlib 2 palette and your own custom PP256 palette!!! Grandly awesome stuff for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! d=^_^=b !!

Please turn to the very next page for an FB program example of this using the commands CrossFade.Default_to_PP256, CrossFade.PP256_to_Default, CrossFadeRange.Default_to_PP256, CrossFadeRange.PP256_to_Default, CrossFadeCtrl.Default_to_PP256, and CrossFadeRangeCtrl.Default_to_PP256!!

PROGRAM EXAMPLE #16:

(This example uses CrossFade.Default_to_PP256, CrossFade.PP256_to_Default, CrossFadeRange.Default_to_PP256, CrossFadeRange.PP256_to_Default, CrossFadeCtrl.Default_to_PP256, *and* CrossFadeRangeCtrl.Default_to_PP256.*)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'------------------------------------------------------------------------
Screen 13, 8, 0, 1


'Next, we draw up the 256 color entries for the default GFXlib 2 palette.
'------------------------------------------------------------------------
For DrawPalette = 0 to 255
  Line (DrawPalette, 8)-(DrawPalette, 200), DrawPalette
Next


'Then, we add our own custom PP256-based palette that has only 1,024
'bytes in it!  ;*)
'------------------------------------------------------------------------
Pal$ = "PP256Pal1.pal" '<--- This is a PP256-based custom palette that we
                       '     are gonna be using for this test!


'Now, prepare to be dazzled with some brain-boggling palette crossfades
'between the default GFXlib 2 palette and our custom PP256-based one right
'here!!!  ^_- !
'------------------------------------------------------------

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now crossfading between the GFXlib 2"
Locate 2, 1: ? "palette and our custom PP256 palette!!"
sleep 3000


'--- First off, let's fade the default GFXlib 2 palette all the way into
'    that custom PP256 palette in one pass, using only 48 milliseconds per
'    fade-step!  And then, we wait two seconds before we fade the palette
'    back to normal again at the same exact speed.  Then we wait two more
'    seconds.  ;*)

CrossFade.Default_to_PP256 Pal$, 48
sleep 2000
CrossFade.PP256_to_Default Pal$, 48
sleep 2000


'--- Now let's do it all again, but this time, using color entries 57-181
'    and just 25 milliseconds per fade-step!

CrossFadeRange.Default_to_PP256 57, 181, Pal$, 25
sleep 2000
CrossFadeRange.PP256_to_Default 57, 181, Pal$, 25
sleep 2000
```

Continues on next page.......

“Program Example #16” continued from last page........

```
'--- Here comes the exciting part: we crossfade the same palettes to and
'    fro *while* we draw the circles on the screen, using the command
'    "CrossFadeCtrl.Default_to_PP256"!!!  :D !

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now let's draw some circles while we"
Locate 2, 1: ? "crossfade between the same palettes!!"

For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeCtrl.Default_to_PP256 Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeCtrl.Default_to_PP256 63 - Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next


'--- Let us do that again, but this time, using the command
'    "CrossFadeRangeCtrl.Default_to_PP256" to fade color entries 80-234
'    between the GFXlib 2 default palette and our PP256-based palette!!!
'    ^_-=b !

For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.Default_to_PP256 80, 234, Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.Default_to_PP256 80, 234, 63 - Fade, Pal$
Next
For ExtraLines = 0 to 80
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
Next
Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "It is such a pleasure of you to check"
Locate 2, 1: ? "this one out!!  Thank you, and chao!!!"


'--- Finally, let's end this as always by fading the ENTIRE default
'    GFXlib 2 palette slowly but *all the way* out in only one pass,
'    using just 63 milliseconds per fade-step!  ;*)

FadeOut.DefaultPal 63


'--- See 'ya now!!!  ^_- !
```

NOTE: THESE NEXT ROUTINES SUPPORT CROSSFADING ONLY BETWEEN THE FOLLOWING:

A 768-BYTE PALETTE —— PP256 PALETTE.

When these following commands are applied in your FB programs, the colors can ***automatically*** *change to the crossfading-based shades of a 768-byte palette and/or a custom PixelPlus 256 (or PP256) palette, so please be *very* careful if you are using any custom 256-color palette(s) of your own choice! ;*) !*

CrossFade.768Pal_to_PP256

SUB DESCRIPTION:

```
Sub CrossFade.768Pal_to_PP256 (PalFile$, PP256PalFile$, millisec)
```

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade to. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified custom 768-byte palette all the way to the specified PP256 custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the entire screen from your custom 768-byte palette all the way to your own custom PP256-based palette of your choice, all in *any* speed that you want, all in ONE single pass!!! ^_^=b Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFade.PP256_to_768Pal

SUB DESCRIPTION:

```
   Sub CrossFade.PP256_to_768Pal (PP256PalFile$, PalFile$, millisec)
```

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade to. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading from the specified PP256 custom palette all the way to the specified custom 768-byte palette.

NOTES ON THIS COMMAND:

*Same exact features as CrossFade.768Pal_to_PP256, *except* that it rather lets you fade the entire screen from your own custom PP256 palette right back to your custom 768-byte palette in its entirety, all in ONE single pass!!! ;*) Again folks, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.768Pal_to_PP256

SUB DESCRIPTION:

```
    Sub CrossFadeRange.768Pal_to_PP256 (StartColor, EndColor, PalFile$,
     PP256PalFile$, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade to. Must be **only 1,024 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the specified custom 768-byte palette all the way to the specified PP256 custom palette.

NOTES ON THIS COMMAND:

*This command — once it is used properly — will actually let you fade the color-order range within your screen from your custom 768-byte palette all the way to your own custom PP256 palette of your choice, all in *any* speed that you want, all in ONE single pass!!!* ^_-=b ! *Remember, higher milliseconds determine slower fades, while lower milliseconds constitute more and more faster fades.*

CrossFadeRange.PP256_to_768Pal

SUB DESCRIPTION:

```
    Sub CrossFadeRange.PP256_to_768Pal (StartColor, EndColor, PP256PalFile$,
     PalFile$, millisec)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade from. Must be **only 1,024 bytes**, please!

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade to. Must be **only 768 bytes**, please!

`millisec` = the amount of milliseconds determining the speed of fading the selected order of colors from the specified PP256 custom palette all the way to the specified custom 768-byte palette.

NOTES ON THIS COMMAND:

*Same exact features as CrossFadeRange.768Pal_to_PP256, *except* that it rather lets you fade the color-order range within your screen from your own custom PP256 palette right back to your custom 768-byte palette in its entirety, all in ONE single pass!!! ;*) Once more, higher milliseconds = slower fades; while lower milliseconds = faster fades.*

CrossFadeCtrl.768Pal_to_PP256

SUB DESCRIPTION:

```
Sub CrossFadeCtrl.768Pal_to_PP256 (FadeToPP256Pal.Grade, PalFile$,
 PP256PalFile$)
```

`FadeToPP256Pal.Grade` = the custom fade-in level between your selectable custom 768-byte palette and your selectable custom PP256 palette.
(63 = fully faded in to PP256 palette; 0 = fully faded out to 768-byte palette)

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade to. Must be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

With this command, you have FULL and free reign over all 64 of the fade levels between your 768-byte palette and your own PP256-based custom palette!!! Perfect for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! d=^_^=b !!

CrossFadeRangeCtrl.768Pal_to_PP256

SUB DESCRIPTION:

```
    Sub CrossFadeRangeCtrl.768Pal_to_PP256 (StartColor, EndColor,
     FadeToPP256Pal.Grade, PalFile$, PP256PalFile$)
```

`StartColor` = the **starting** color (0 to 255) within the color-order range.

`EndColor` = the **ending** color (0 to 255) within the color-order range.

`FadeToPP256Pal.Grade` = the custom fade-in level between the selected order of colors from your selectable 768-byte custom palette and that very same color order from your selectable custom PP256 palette. (63 = fully faded in to PP256 palette; 0 = fully faded out to 768-byte palette)

`PalFile$` = the filename for your custom 256-color palette (.pal) to fade from. Must be **only 768 bytes**, please!

`PP256PalFile$` = the filename for your custom PP256-based palette (.pal) to fade to. Must be **only 1,024 bytes**, please!

NOTES ON THIS COMMAND:

Same great features as CrossFadeCtrl.768Pal_to_PP256, except that you can REALLY fade any color range you want between your custom 768-byte palette and your own custom PP256 palette!!! Grandly awesome stuff for games and graphics demos in FB, as well as for creating some real mind-bending palette effects, too!!! d=^_^=b !!

Please turn to the very next page for an FB program example of this using the commands CrossFade.768Pal_to_PP256, CrossFade.PP256_to_768Pal, CrossFadeRange.768Pal_to_PP256, CrossFadeRange.PP256_to_768Pal, CrossFadeCtrl.768Pal_to_PP256, and CrossFadeRangeCtrl.768Pal_to_PP256!!

PROGRAM EXAMPLE # 17:

(This example uses CrossFade.768Pal_to_PP256, CrossFade.PP256_to_768Pal, CrossFadeRange.768Pal_to_PP256, CrossFadeRange.PP256_to_768Pal, CrossFadeCtrl.768Pal_to_PP256, *and* CrossFadeRangeCtrl.768Pal_to_PP256*.)*

```
'$include: "FBnewpal.bi"

'Let's set up the good 'ol fullscreen 256-color 320x200 graphics mode
'with no pages.  :D
'-----------------------------------------------------------------------
Screen 13, 8, 0, 1


'Then, we add both palettes for this test: our own custom 768-byte
'palette and our own PP256-based palette!  And after that, the 256
'color entries will then be drawn for that 768-byte baby.  ;*)
'-----------------------------------------------------------------------
Pal.768$ = "CustomPalette_01.pal" '<--- This is our custom 768 palette
                                  '     that we will be using!

Pal.PP256$ = "PP256Pal1.pal"  '<--- This is a PP256-based custom palette
                              '     we are *also* are gonna be using here!

LoadUp768Pal Pal.768$  '<--- Revving up our 768-byte custom palette!!

For DrawPalette = 0 to 255
  Line (DrawPalette, 8)-(DrawPalette, 200), DrawPalette
Next


'Now, watch as we rock the 256-color house with some palette crossfades
'between our own 768-byte palette and our custom PP256-based one right
'here!!!  ^_- !
'-----------------------------------------------------------

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now crossfading between our 768-byte"
Locate 2, 1: ? "palette and our custom PP256 palette!!"
sleep 3000


'--- First off, let's fade our custom 768-byte palette all the way into
'    that custom PP256 palette in one pass, using only 52 milliseconds per
'    fade-step!  And then, we wait two seconds before we fade the palette
'    back to normal again at the same exact speed.  Then we wait two more
'    seconds.  ;*)

CrossFade.768Pal_to_PP256 Pal.768$, Pal.PP256$, 52
sleep 2000
CrossFade.PP256_to_768Pal Pal.PP256$, Pal.768$, 52
sleep 2000


'--- Now let's do it all again, but this time, using color entries 78-164
'    and just 37 milliseconds per fade-step!

CrossFadeRange.768Pal_to_PP256 78, 164, Pal.768$, Pal.PP256$, 37
sleep 2000
CrossFadeRange.PP256_to_768Pal 78, 164, Pal.PP256$, Pal.768$, 37
sleep 2000
```

Continues on next page.......

"Program Example #17" continued from last page........

```
'--- Here comes the AWESOME part: we crossfade the same palettes to and
'    fro *while* we draw the circles, lines, and boxes on the screen,
'    using the command "CrossFadeCtrl.768Pal_to_PP256"!!!  :D !

Line (0, 0)-(319, 15), 6, BF
Color 15, 6
Locate 1, 1: ? "Now let's draw some stuff while we"
Locate 2, 1: ? "crossfade between the same palettes!!"

For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeCtrl.768Pal_to_PP256 Fade, Pal.768$, Pal.PP256$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeCtrl.768Pal_to_PP256 63 - Fade, Pal.768$, Pal.PP256$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1, B
Next


'--- Let us do that again, but this time, using the command
'    "CrossFadeRangeCtrl.768Pal_to_PP256" to fade color entries 80-234
'    between our own 768-byte palette and our PP256-based palette!!!
'    ^_-=b !

For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.768Pal_to_PP256 80, 234, Fade, Pal.768$, Pal.PP256$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1
Next
For Fade = 0 to 63
  sleep 35
  Circle (Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 66), Int (rnd(1) * 50) + 1, Int (rnd(1) * 255) + 1
  CrossFadeRangeCtrl.768Pal_to_PP256 80, 234, 63 - Fade, Pal.768$, Pal.PP256$
Next
For ExtraLines = 0 to 80
  sleep 35
  Line -(Int(rnd (1) * 319) + 1, Int (rnd(1) * 183) + 16), Int (rnd(1) * 255) + 1, BF
Next
Line (0, 0)-(319, 15), 0, BF
Color 15, 0
Locate 1, 1: ? "Check 'ya out later, and have such an"
Locate 2, 1: ? "*excellent* day now!  Peace!!!"


'--- Finally, time to conclude by fading the ENTIRE custom 768-byte
'    palette slowly but *all the way* out in only one pass, using just
'    63 milliseconds per fade-step!  ;*)

FadeOut.768Pal Pal.768$, 63


'--- And we outta here!!!  ^_- !
```

—— COLOR ADJUST ROUTINE

Or, a little lesson in spicing up your palette with just a wee bit here and a pinch there!

WriteColor

SUB DESCRIPTION:

```
Sub WriteColor (Col, R, G, B)
```

`Col` = the color number (0 to 255) to change into any RGB color combination (within the "*R*", "*G*", and "*B*" variables).

`R` = the **red** shade level (0 to 63).

`G` = the **green** shade level (0 to 63).

`B` = the **blue** shade level (0 to 63).

NOTES ON THIS COMMAND:

*When this command is called, it will change any color number you want to *any* RGB color combination you want, easily and INSTANTANEOUSLY! Useful for graphics demos and even your games, too!!*

EXAMPLE USAGES OF THIS COMMAND:

```
WriteColor 6, 0, 0, 63
```

Changes the RGB values of color #6 to make it ***blue***.

```
WriteColor 172, 63, 32, 0
```

Changes the RGB values of color #172 to make it ***orange***.

```
WriteColor 15, 63, 0, 63
```

Changes the RGB values of color #15 to make it ***purple***.

APPENDIX A: 768-BYTE PALETTES

Or, let's dissect in some awesome truths about how this type of palette first took shape!

*Originally, from way back in the 'ye good olde Microsoft DOS (or MS-DOS) days, some of you may remember something about a type of palette file that was portable yet *extremely* so powerful, in that 1) it enabled a growing and vast amount of programmers everywhere to just break on outta the all-too-familiar default 256-color palette to create custom 256-color ones that are both truly imaginative and even breathtakingly original; and 2) it was no more than* ***only 768 bytes in size****!!! :o ! Imagine that, huh? This type of file is simply called a .PAL file.*

For a 768-byte palette file (.PAL), it begins with a little something like this:

	1-bit color	2-bit color	3-bit color	4-bit color	5-bit color	6-bit color	7-bit color	8-bit color
Math order:	2 ^ 1	2 ^ 2	2 ^ 3	2 ^ 4	2 ^ 5	2 ^ 6	2 ^ 7	2 ^ 8
Total colors:	2	4	8	16	32	64	128	**256**

........whereas for example, if you were to do this mathematical problem of.....

`2 ^ 3 = 8`

*........then that would be the *exact* same as.....*

`2 * 2 * 2 = 8   (2 * 2 = 4; and then 4 * 2 = 8)`

So then, if you can multiply just eight 2's in a row, then that would actually give you a grand total of ***256 displayable colors****!!! ;*) !*

For the 256 displayable colors indeed, let's think this through, shall we?

Each single color here consists ENTIRELY of only three (3) channels:

Red (R)**, **Green (G)**, and **Blue (B)

*........with just *one* (1) single byte per channel. The three color channels, then, count simply as only 3 bytes to store per color. So, if we just add it like this for **all 256 colors**:*

Red (**R**) channel	***256 bytes***
Green (**G**) channel	***256 bytes***
Blue (**B**) channel	***256 bytes***

256 + 256 = 512
....then 512 + 256 = **768 bytes total!!**

*........**BOOM!!** There is your custom 768-byte palette file, because that is what it's no less than really all about!!! Not too complicated now, was it?* ^_-=b !

APPENDIX B: CONSTANTS

Or, here are some litte yet *very* important things for you to remember in this lib!

Here are the constants that you will be using as little shortcuts to help make things a little easier in your FreeBASIC coding during the use of this first-ever FB palette library:

```
CONST FBPM.True = 1, FBPM.False = 0
CONST Yes = 1, No = 0
```

In other words, do feel absolutely free to use those four substitutes to speed you right along a little bit to avoid confusion. Eventually, they will help make it more practical enough to do whatever you want it to in this palette machine!!! ^_^ !

Appendix C: Program Example Index

Or, a little refresher course on what we've learned today!

Program Example #:	**Purpose of the Example:**	**Page:**
- 1 -	*Shows you how to fade in and out the selectable colors from the* GFXlib 2 *default palette.*	18
- 2 -	*Shows you how to rotate colors around and around the* GFXlib 2 *default palette.*	25
- 3 -	*Shows you the MANY cool tricks that you can achieve and more with a negative of the* GFXlib 2 *default palette!*	33
- 4 -	*Shows you the* <u>*amazingly awesome*</u> *greyscale techniques and loads more within them for the* GFXlib 2 *default palette!*	44
- 5 -	*Shows you how to load a custom 768-byte palette, and then to fade in and out the selectable colors from it!*	58
- 6 -	*Shows you how to rotate the colors around and around your own custom 768-byte palette!*	65
- 7 -	*Shows you the MANY cool tricks that you can achieve and more with a negative of a custom 768-byte palette!*	74
- 8 -	*Shows you the* <u>*amazingly awesome*</u> *greyscale techniques and loads more within them for your custom 768-byte palette!*	85
- 9 -	*Shows you how to load a custom PixelPlus 256 (PP256) palette, and then to fade in and out the selectable colors from it!*	99
- 10 -	*Shows you how to rotate colors around and around your own PP256-based palette.*	106
- 11 -	*Shows you the MANY cool tricks that you can achieve and more with a negative of a PP256-based palette!*	115
- 12 -	*Shows you the* <u>*amazingly awesome*</u> *greyscale techniques and loads more within them for your own PP256-based palette!*	126
- 13 -	*Shows you how to do spectactularly good crossfades between the* GFXlib 2 *default palette and your 768-byte palette!*	137
- 14 -	*Shows you how to do spectactularly good crossfades between* ***<u>TWO</u>*** *custom 768-byte palettes!*	143
- 15 -	*Shows you how to do spectactularly good crossfades between* ***<u>TWO</u>*** *custom PP256-based palettes!!*	149
- 16 -	*Shows you how to do spectactularly good crossfades between the* GFXlib 2 *default palette and your PP256-based palette!!*	157
- 17 -	*Shows you how to do spectactularly good crossfades between your own 768-byte palette and your PP256-based palette!!*	165

—— CHANGELOG

Or, I will show you a little history on what happened since the very first coming of this FB lib!

version 1.1 – sunday, march 13, 2005

- *Updated and improved the speed on my custom-emulated version of the default 256-color GFXlib 2 palette, thereby causing the supporting fading/rotation/negative/crossfade/greyscale routines for that palette to have at least a bit of a noticable speed increase (thank you, VonGodric!!* ^_-=b *)!!*

version 1.0 – saturday, march 12, 2005

- ***Initial Release****, FIRST presented on QBasicNews.com forums on that very morning EST!!* ^-^ !

—— FINAL WORDS

Or, let me kindly say some last (but NOT least!) things to you in close!

*Whew!! Now THAT was one *long* documentation to say the least!! But also, I hope you enjoyed your wonderful read of this as much as I did creating it. Remember, I did not write this documentation for myself. No. Rather, I wrote it clearly* ***TO BE A RATHER VERY BLESSING TO YOU AND THE ENTIRE QB45/QB71/FREEBASIC COMMUNITY****, period, total end. ;*) !! Not only that, but that is the whole entire purpose of this wonderful first-ever palette library for FB that will guarantee that you will have an intense blast using, I promise you that. I CAN PROMISE IT, with all of my very heart. d=^_-=b !*

Also, one final thing: for ***NEWER*** *versions of FreeBASIC, please follow the same exact install instructions as clearly described on* **Page** *7. That way, you can be sure that the functionality on this whole library will *fully* work in its entirety on that newer version indeed.* ***Be sure to remember that!!*** *:D*

Let's give some credit to where it is due, shall we now?

First of all, ***EXTRA*** *Special Thanks *definitely* go out to Almighty Jehovah God, who has no doubt enabled me and empowered me and encouraged me to do both this whole original FB library* ***and*** *all of the above as fully told in this whole documentation, beginning to end!!! Praise and maximum glory be to Him forever!!! d=^_-=b !!*

Also, Special Thanks to the following wonderful people for getting me so inspired to create "The New FreeBASIC 8-Bit Palette Machine":

- *Andre Victor T. Vicentini (aka v1ctor)*
- *Angelo Mottola (for the* ***MOST AWESOME LIB*** GFXlib 2 *for FB!!!* ***d=^_^=b !!*** *)*
- *Richard Eric M. Lope (aka Relsoft)*
- *VonGodric (for helping me to improve the speed on my routines supporting the default* GFXlib 2 *palette!!)*
- *Sterling Christensen*
- *Chris Chadwick*
- *Steve Nunnally of Acid Works Software*
- *P. Bindels (for his *awesome* inspirations for me to just code my own routines for this lib!!* ***;*)*** *)*

And finally, such SPLENDIDLY AWESOME greets to the following great people (and fellow FB users, too!!!):

- *Wildcard (aka Brendan Urquhart)*
- *Fling-master (aka Gered King)*
- *Dav*
- *Oracle*
- *Pete Berg*
- *Dark_prevail*
- *Nekrophidius (aka Necros Ihsan Nodtveidt)*
- *TheBigBasicQ*
- *Nemesis*
- *Na Than Assh Antti (aka Na_th_an)*
- *Neo Deus Ex Machina*
- *Aetherfox*
- *Dr_Davenstein*
- *.........and to* ***ALL OTHER MEMBERS*** *of the QBasicNews.com forums!!!* ***^_^=b***

*I most sincerely hope with all of my heart that you so richly enjoy this library as the truly definitive 8-bit palette manipulation tool for FreeBASIC, as I am *sure* you likely will once you experience it for your precious self. I guarantee it!! ;D !*

God so utterly bless all of you, and a most grandly pleasant "Bon Voyage"!!! ^-^ !!

From the words of Adigun Azikiwe Polack, the official creator of "The New FreeBASIC 8-Bit Palette Machine".

Presented to you by Adigun Azikiwe Polack.