# TUTORIAL USING NASM PE MACROS

In this tutorial we are going to use NASM PE MACROS ( NPM for short) on a sample application. This tutorial is about a dialogbox and a bitmap inside it. This will be a perfect example for us to see how NPM works.

First thing we do is to define our dialog in terms of NPM. There is only one control inside dialog which will hold bitmap resource. Decleration of dialog in NPM is very similar to a resource script.

|  |
| --- |
| DIALOG dialog, 10, 10, 200, 200  STYLE DS\_CENTER  CAPTION 'NASM PE MACROS'  FONT 8, "Tahoma"    CONTROL '', ID\_STATIC, 'STATIC', 0, 0, 200, 200, SS\_BITMAP  ENDDIALOG |

DIALOG macro has default styles applied to it. But additionaly we want DS\_CENTER style applied to it. Below sytles are deault dialog styles of NPM.

|  |
| --- |
| STYLE DS\_MODALFRAME | DS\_3DLOOK | DS\_CENTER | DS\_CENTERMOUSE | WS\_CAPTION |

We will add caption to our dialog.

|  |
| --- |
| CAPTION "NASM PE MACROS" |

We will add a FONT declaration to our the dialog. If you don't add a font decleration default is 8, "MS Sans Serif".

|  |
| --- |
| FONT 8, "Tahoma" |

Now it is time for controls. We are only going to add a static conrol to hold our bitmap.

|  |
| --- |
| CONTROL '', ID\_STATIC, 'STATIC', 0, 0, 200, 200, SS\_BITMAP |

Dialog is finished. We need to include bitmap to the resources with the help of BITMAP macro.

|  |
| --- |
| BITMAP bitmap, 'bitmap.bmp' |

Both bitmap and dialog resources has to be added to the resource tree. Each resource has to be put on resource tree otherwise Windows can't see our resources.

|  |
| --- |
| RESOURCE  TYPE RT\_BITMAP  ID ID\_BITMAP  LANG  LEAF RVA(bitmap), SIZEOF(bitmap)  ENDLANG  ENDID  ENDTYPE  TYPE RT\_DIALOG  ID ID\_DIALOG  LANG  LEAF RVA(dialog), SIZEOF(dialog)  ENDLANG  ENDID  ENDTYPE  ENDRESOURCE |

This application as other win32 applications heavily depends on Windows API. We have to import API functions we use. Below is how this is done in NPM.

|  |
| --- |
| IMPORT  LIB user32.dll  FUNC DialogBoxParamA  FUNC EndDialog  FUNC SendMessageA  FUNC LoadBitmapA  FUNC GetDlgItem  FUNC GetDC  FUNC ReleaseDC  FUNC GetClientRect  ENDLIB  LIB kernel32.dll  FUNC GetModuleHandleA  ENDLIB  LIB gdi32.dll  FUNC DeleteObject  FUNC SelectObject  FUNC GetObjectA  FUNC CreateCompatibleDC  FUNC CreateCompatibleBitmap  FUNC StretchBlt  FUNC BitBlt  FUNC DeleteDC  ENDLIB  ENDIMPORT |

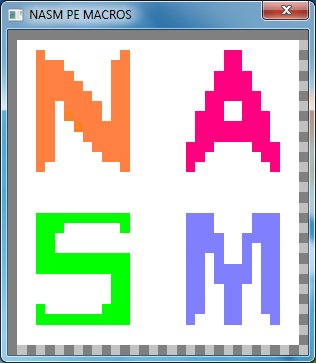
Now it is time to turn our attention to actual instructions. First thing is to create bare skeleton PE file in NPM. Which has following form.

|  |
| --- |
| %include 'pe.inc' %include 'windows.inc'  ; Resource constanst here  PE32  ; Data declarations here  START  ; instructions here  ; data directories here  END |

START is entry point of our PE. In other words it is our WinMain. We should write down instructions after START. We put constant declarations made by %define directives above PE32 and we put data variables below PE32. Below is an excerpt from tutorial.asm file.

|  |
| --- |
| ...  START  push ebp  mov ebp, esp  ; Get module handle  push NULL  call [VA(GetModuleHandleA)]  mov [VA(hIns)], eax  ; DialogBox  push 0  push VA(DlgProc)  push 0  push ID\_DIALOG  push dword [VA(hIns)]  call [VA(DialogBoxParamA)]  ; return  mov esp, ebp  pop ebp  ret 16  ... |

When you assemble tutorial.asm with nasm -f bin -o tutorial.exe you get your 32-bit executable. Both source and executable is given in project. A snapshot of the running executable is given below.



For latest NPM and documentation visit https://github.com/nalsakas/pe project site.