Introduction

SKELETON Version 2.0

32-Bit Assembly Language Programming

for the Microsoft® Windows® 95 operating system

by Wayne J. Radburn

I have seen many books on Windows® programming and assembly language programming. Rarely have I found information that covers assembly language programming for Windows®. I offer my SKELETON as an example of how such programming can be done. As you can see, it goes beyond the bare minimum required to get a window up on the screen. In this help file I briefly describe the Features used in SKELETON indicating where in the files they are implemented. I also give short File Descriptions and make a few suggestions for Building Projects based on the SKELETON files.

I hope that you will find this freeware SKELETON example helpful for learning a little more about assembly language programming for Windows® and that you may also find some use for it as a framework to start building your own applications.

Icon

The Icon source file Icon.ico gets used in Resource.rc. The constant IDI_ICON in Common.inc identifies the Icon and gets used during Initialization in WinMain.asm and also in the definition of the About Box.

Menu Bar

The Menu Bar defined in Resource.rc provides the user with some commands available for the application. The constant IDM_MENU in Common.inc identifies the Menu Bar and gets used during Initialization in WinMain.asm. Other constants identify the commands in the File and Help pop-up menus and get sent with the WM_COMMAND message to WndProc.asm.

The New, Open, Save, and Save As commands get handled in WndProc.asm by a call to the appropriate procedure in CmdFile.asm. Some of these commands can get initiated by keystrokes called Accelerators or by Toolbar buttons.

The Exit command gets handled in WndProc.asm by a call to the procedure MsgWM_CLOSE in Msg.asm. Applications that do not need to do anything before exiting could simply jump to caseWM_DESTROY in WndProc.asm.

The Help Topics command gets handled by a jump to caseIDM_HELPTOPICS in WndProc.asm which displays Help.

The About Skeleton command gets handled by a jump to caseIDM_ABOUT in WndProc.asm which displays the About Box.

Accelerators

The Accelerators defined in Resource.rc allow the user to initiate some Menu Bar commands with a simple keystroke. The constant IDA_ACCEL in Common.inc identifies the accelerator table and gets used just before the MessageLoop in WinMain.asm.

The message loop translates the accelerator keystroke into the appropriate command which gets sent with the WM_COMMAND message to WndProc.asm and gets handled as described in the Menu Bar feature.

Toolbar

The Toolbar displays buttons containing bitmap images related to the Menu Bar commands which provide shortcuts to those commands.

In response to the WM_CREATE message, WndProc.asm calls MsgWM_CREATE in Msg.asm which calls the CreateTBar procedure in ToolBar.asm. The Toolbar buttons created for SKELETON support Tooltips.

In response to the WM_SIZE message, WndProc.asm calls MsgWM_SIZE in Msg.asm which sends the message to the Toolbar so that it can adjust to the new window size.

Tooltips

The Tooltips used in SKELETON display descriptive text about the Toolbar buttons from the STRINGTABLE in Resource.rc.

The WM_NOTIFY message gets handled by a jump to caseWM_NOTIFY in WndProc.asm which checks for the TTN_NEEDTEXT notification which gets handled by a call to NtfTTN_NEEDTEXT in ToolBar.asm.

Status Bar

The Status Bar used in SKELETON displays descriptive text about the Menu Bar and System Menu commands from the STRINGTABLE in Resource.rc.

In response to the WM_CREATE message, WndProc.asm calls MsgWM_CREATE in Msg.asm which calls the CreateSBar procedure in StatBar.asm.

The WM_MENUSELECT message gets handled by a jump to caseWM_MENUSELECT in WndProc.asm which calls MsgWM_MENUSELECT in StatBar.asm.

In response to the WM_SIZE message, WndProc.asm calls MsgWM_SIZE in Msg.asm which sends the message to the Status Bar so that it can adjust to the new window size.

Help

The Help Topics command from the Help menu of the Menu Bar gets handled in WndProc.asm by a jump to caseIDM_HELPTOPICS which gets WinHelp to open the Skeleton.cnt and Skeleton.hlp files which contain helpful information about the SKELETON files.

WinHelp gets sent the HELP_QUIT message while processing the WM_DESTROY message in caseWM_DESTROY in WndProc.asm.

About Box

The About dialog box displays the Icon, program information, version number and copyright. It gets defined in Resource.rc and identified by the constant IDD_ABOUT in Common.inc.

The About Skeleton command from the Help menu of the Menu Bar gets handled in WndProc.asm by a jump to caseIDM_ABOUT which creates the dialog box. Further initialization and message processing takes place in the About dialog procedure in About.asm.

Version Information

The Version Information defined in Resource.rc and can be viewed by clicking the right mouse button on SKELETON and selecting Properties from the pop-up menu.

MakeFile

The NMAKE utility searches for the default file MakeFile which contains the Assembler, Linker, Resource Compiler, and Help Compiler commands necessary to build from the source files an updated SKELETON.

The command nmake clean erases the intermediate *.OBJ *.RES files allowing one to generate an updated release version with the command nmake or a debug version with the command nmake debug=y.

Note that changes to WindowsA.inc or Common.inc may have global effects requiring a clean rebuild.

Common.inc

Common.inc contains the EQU, PROTO, EXTERNDEF directives for the constants, procedures, and variables that require sharing between two or more of the Assembly Language Files and gets included near the beginning of each of these files. WindowsA.inc gets included near the beginning of Common.inc. Many of the constants must also get defined with the same value in Resource.rc.

WindowsA.inc

WindowsA.inc contains a very small ASCII (non-UNICODE) subset of the Win32 API Constants, Type Definitions, Structures, and Function Prototypes and gets included at the beginning of Common.inc.

Note that adding more Function Prototypes to WindowsA.inc may require the addition of other Libraries to the linking section of MakeFile.

Resource.rc

Resource.rc contains the SKELETON resource definitions or sources for the Icon, Menu Bar, Accelerators, About Box, and Version Information. The Resource Compiler uses this file to generate the Resource.res file which the Linker uses when creating SKELETON.

Icon.ico

I used Image Editor to create the SKELETON 32x32 Skull Icon which has a transparent background.

WinMain.asm

WinMain.asm contains SKELETON's starting point at the label Start which also gets placed in the linking section of MakeFile. The WinMain parameters get stored in global variables before the call to WinMain. The SKELETON application exits after WinMain returns.

The WinMain procedure first makes a call to OnlyOneInstance and Initialization before loading the Accelerators. It then stays in the MessageLoop sending the queued messages to WndProc.asm until the user chooses to exit SKELETON which causes WinMain to return.

The OnlyOneInstance procedure tries to create a semaphore to prevent another instance of SKELETON from starting. If the semphore already existed, the first SKELETON gets brought to the foreground and the second attempt exits. Remove this procedure and the call to it in WinMain to allow more than one instance of SKELETON to run at a time.

The Initialization procedure fills in the WNDCLASSEX structure in order to register the class, create the window, and then finally get it displayed. Note that further initialization takes place in response to the WM_CREATE message handled in WndProc.asm by a call to MsgWM_CREATE in Msg.asm.

Note that according to the STDCALL calling convention, the called procedure should preserve the EBX, ESI, EDI, EBP registers (also, the direction flag is clear on entry and should be returned clear). The INVOKE directive takes care of preserving EBP for procedures defined with parameters or local variables. Using USES EBX, ESI, EDI in the PROC directive when defining procedures will preserve these other registers. I have not done this for WinMain and WndProc since these procedures do not change EBX, ESI, EDI. Note that the procedures that they call should now preserve these registers if they get used (see NewWindowName in CmdFile.asm for an example).

WndProc.asm

The WndProc procedure checks the message parameter and jumps to the appropriate section for messages that require further processing. All other messages get sent to the DefWindowProc for default processing. Note that for a large number of messages it becomes more space-time efficient to search through a table of message IDs (using repne scasd for example) and calling into another table which contains the corresponding address of the procedure which handles the message.

Some messages and commands get handled completely within WndProc:

caseWM_DESTROY first sends the HELP_QUIT message to WinHelp and then sends the WM_QUIT message which breaks out of the MessageLoop in WinMain.asm returning the exit code before SKELETON terminates.

caseWM_NOTIFY handles messages or events sent by controls by a jump to the appropriate section. The TTN_NEEDTEXT message for Tooltips gets handled in this section.

caseWM_COMMAND handles commands from the Menu Bar, Accelerators, and Toolbar by a jump to the appropriate section.

caseIDM_HELPTOPICS sends the HELP_FINDER message to WinHelp which opens the help file named in the .CONST section of WndProc.

caseIDM_ABOUT creates the About Box.

The other messages and commands get handled by a call to a procedure in Msg.asm or CmdFile.asm.

Msg.asm

Msg.asm contains additional procedures which handle messages from WndProc.asm.

The caseWM_PAINT procedure redraws regions of the client area that have changed or require updating. SKELETON does not display anything.

The caseWM_CREATE procedure centers the SKELETON window on the desktop. The call to InitCommonControls must be made before creating the Toolbar and Status Bar. See CmdFile.asm for details about CmdIDM_NEW. Note that at this point the call to CreateWindowEx in WinMain has not yet returned so the hWnd parameter of WndProc gets used instead of hMainWnd which does not yet have a valid handle value.

The caseWM_CLOSE procedure first calls SaveChanges in CmdFile.asm which returns FALSE if SKELETON should not exit.

The caseWM_SIZE procedure makes adjustments required by a change in window size. Both the Status Bar and the Toolbar get updated.

Misc.asm

The MiscCenterWnd procedure centers the position of a child window based on the position of the parent window, but will keep the child window visible on the desktop. It gets used in Msg.asm to center the main window on the desktop and in About.asm to center the About Box.

About.asm

The About dialog procedure handles the About Box dialog message processing. The caseWM_INITDIALOG section centers the window while the caseWM_COMMAND section checks to see if the user sent a command to close the dialog window which gets handled by a jump to caseEND.

StatBar.asm

The CreateSBar procedure creates a default one part Status Bar.

The MsgWM_MENUSELECT procedure displays descriptive text in a simple mode status bar based on whether the System Menu, File Menu, Help Menu, or commands in these menus, have been activated by the mouse or keyboard. The simple mode status bar text does not effect the items in the other parts set up for the status bar.

ToolBar.asm

The CreateTBar procedure creates a Tool Bar with the three buttons defined in the tbButton table which uses the default small color bitmaps.

The NtfTTN_NEEDTEXT procedure points to a string filled with the required text for the Tooltips control.

CmdFile.asm

CmdFile.asm contains procedures which can provide a framework for supporting files with commands from the Menu Bar. The named or changed status of the file gets stored in fFileStatus. The file name gets placed on the title bar of the SKELETON window.

The CmdIDM_NEW procedure first calls SaveChanges. If it returns TRUE then the file name gets set to the default name Untitled and the fFileStatus NAMEDbit and CHANGEDbit both get cleared before a call to NewWindowName. SKELETON does not have any other data structures to initialize.

The CmdIDM_OPEN procedure first calls SaveChanges. If it returns TRUE then the Open common dialog box prompts the user for a file to open. If this returns with a name then the fFileStatus NAMEDbit gets set and the CHANGEDbit gets cleared before a call to NewWindowName. SKELETON does not open any files.

The CmdIDM_SAVE procedure first checks the fFileStatus NAMEDbit and the Save As common dialog prompts the user for a file name if required. The fFileStatus NAMEDbit gets set and the CHANGEDbit gets cleared before a call to NewWindowName. SKELETON does not save anything to any files.

The SaveChanges procedure checks the fFileStatus CHANGEDbit and if changes have been made displays a Yes/No/Cancel message box for saving the changes. It returns FALSE only if the user chooses to cancel.

The NewWindowName procedure changes the title bar text to the file name first without the file extension followed by an asterisk if the file has changed and then the application name.

Skeleton.hlp

Skeleton.hlp and Skeleton.cnt get used by WinHelp to display the help about the SKELETON files. I found that the help files of the Help Compiler contained the best source of information about building help files. You now also have the source files of SKELETON help as an example.

Skeleton.cnt

Skeleton.cnt contains the contents page of Skeleton.hlp which I created using the Help Compiler.

HelpFile.hpj

The Help Compiler controls the content of this file based on selected options, files, and settings, which it uses when building Skeleton.hlp.

HelpFile.rtf

I actually used Notepad with the Word Wrap option to create this file. A text editor that handles Rich-Text Format (RTF) files would have made building the help files much easier and quicker. More information about RTF files can be found in the help files of the Help Compiler.

Before you Start

If you plan on using the SKELETON files for application development, you should do the following now so that you have less to do when you copy the files into a new project folder:

- Edit MakeFile so that the paths to the Building Tools correctly point to the tools that you use.
- Edit Common.inc so that the path correctly points to where you keep WindowsA.inc.
- Edit Resource.rc and put your name and/or company name in the definition of the About Box and Version Information.

Building Tools

The following lists the additional software I used in building the SKELETON project:

Microsoft® Macro Assembler (MASM) Version 6.11d

ML	Assembler

Microsoft® Visual C++® Version 4.0

LINK	32-Bit Linker
NMAKE	Program Maintenance Utility

Microsoft® Win32® Software Development Kit (SDK)

*.LIB, *.H	Library Files and Headers
RC	Resource Compiler
IMAGEDIT	Image Editor
HCW	Help Workshop/Compiler

Microsoft® Windows® 95

NOTEPAD	Text Editor

Getting Started

As an exercise to test your familiarity with the SKELETON files, you could create several folders edited to contain different versions of SKELETON with fewer features. Simple applications may not require the toolbar, status bar, file and/or help features. Now when you come up with a great idea for an application you can:

- Copy a SKELETON folder to a new folder.
- Create a new Icon for your application.
- Rename the PROJECT in MakeFile.
- Rename the szClassName in WinMain.asm.
- Change the About Box and Version Information text in Resource.rc.
- Change some of the STRINGTABLE and MENU text in Resource.rc.
- Rename the szHelpFile in WndProc.asm.
- Remove $(PROJECT).hlp from ALL: in MakeFile and put it back in when you are ready to start building your help files.
- Erase Skeleton.hlp, Skeleton.cnt, HelpFile.hpj, and HelpFile.rtf since your help files will contain information specific to your application.

- The rest really depends on what you have envisioned for your application. SKELETON can be used for getting started. It is now up to you to finish the rest of the work on that great idea...