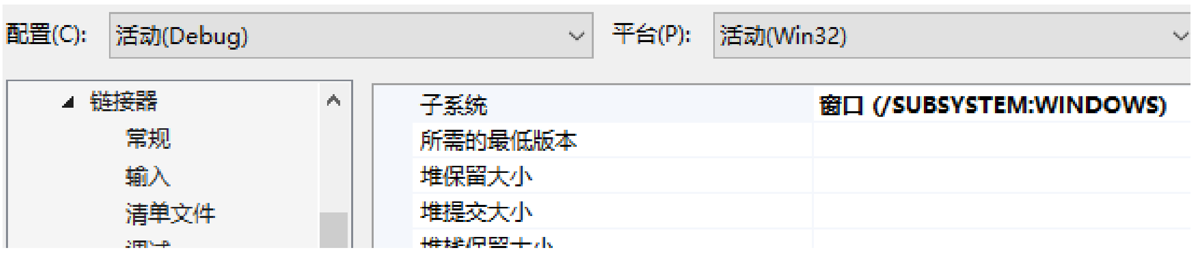
Win32窗口程序运行说明

该文档源文件在GitHub，大家可以直接在GitHub上提PR来改进该文档。

环境配置

首先需要配置好命令行版本VS+汇编环境，还需要修改子系统为“窗口”，具体操作是：

项目上右键——属性——配置属性——链接器——系统——子系统，选择**窗口**：



配置(C): 活动(Debug)　平台(P): 活动(Win32)

子系统	窗口 (/SUBSYSTEM:WINDOWS)
所需的最低版本	
堆保留大小	
堆提交大小	

相关说明

1. 请首先阅读课本第七版第11章《MS-Windows编程》，特别是：
   - 表11-1：MS-Windows和MASM的类型转换；
2. 程序中会使用Win32 API（官方教程在：https://learn.microsoft.com/zh-cn/windows/win32/api/ ），可以参考该官方教程及第七版课本第11章《MS-Windows编程》的内容，在汇编中通过函数调用的方法调用这些API。
   - 需要找某些功能时，请在搜索引擎中搜索【Win32API 功能名】。比如，要调用画图功能，请搜索【Win32API 画图】。
3. 在Win32窗口程序实现某些功能时，一般**不需要使用中断。**
   - 例如：**不要**使用`int 10h`在Win32窗口程序中画图（一个例子是 https://blog.csdn.net/qq_40298054/article/details/84496944 ，请务必注意这里的程序是**运行在DOSBox中的**，不是Win32窗口程序！！）
4. 如果子系统选择控制台，会导致运行时多出一个黑框，但不会有别的影响。
5. 很多结构和函数的定义在`windows.inc`中都有，可以直接用。**但是**`windows.inc`**及一些其他的masm32中自带的inc文件与**`Irvine32.inc`**和**`GraphWin.inc`**不兼容，大家可以参考下文中**`WinApp_v2.asm`**中include文件的方式。**

示例程序

程序1-WinApp.asm

汇编程序使用`WinApp.asm`（可以到云盘下载，也已经传到网络学堂上）。其内容对应于教材（第7版）第11.2.6节代码，可以打开窗口，显示提示框，并在退出程序时有所提示。其中用到了`Irvine32.inc`和`GraphWin.inc`，这两个文件都在`Irvine`文件夹中。

```
TITLE Windows Application                   (WinApp.asm)

; This program displays a resizable application window and
; several popup message boxes.
; Thanks to Tom Joyce for creating a prototype
; from which this program was derived.
; Last update: 9/24/01

INCLUDE Irvine32.inc
INCLUDE GraphWin.inc

;==================== DATA =======================
.data

AppLoadMsgTitle BYTE "Application Loaded",0
AppLoadMsgText  BYTE "This window displays when the WM_CREATE "
                BYTE "message is received",0

PopupTitle BYTE "Popup Window",0
PopupText  BYTE "This window was activated by a "
           BYTE "WM_LBUTTONDOWN message",0

GreetTitle BYTE "Main Window Active",0
GreetText  BYTE "This window is shown immediately after "
           BYTE "CreateWindow and UpdateWindow are called.",0

CloseMsg   BYTE "WM_CLOSE message received",0

ErrorTitle  BYTE "Error",0
WindowName  BYTE "ASM Windows App",0
className   BYTE "ASMWin",0

; Define the Application's Window class structure.
MainWin WNDCLASS <NULL,WinProc,NULL,NULL,NULL,NULL,NULL, \
    COLOR_WINDOW,NULL,className>

msg       MSGStruct <>
winRect   RECT <>
hMainWnd  DWORD ?
hInstance DWORD ?

;=================== CODE =========================
.code
WinMain PROC
; Get a handle to the current process.
    INVOKE GetModuleHandle, NULL
    mov hInstance, eax
    mov MainWin.hInstance, eax
```

```
; Load the program's icon and cursor.
    INVOKE LoadIcon, NULL, IDI_APPLICATION
    mov MainWin.hIcon, eax
    INVOKE LoadCursor, NULL, IDC_ARROW
    mov MainWin.hCursor, eax

; Register the window class.
    INVOKE RegisterClass, ADDR MainWin
    .IF eax == 0
      call ErrorHandler
      jmp Exit_Program
    .ENDIF

; Create the application's main window.
; Returns a handle to the main window in EAX.
    INVOKE CreateWindowEx, 0, ADDR className,
      ADDR WindowName,MAIN_WINDOW_STYLE,
      CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,
      CW_USEDEFAULT,NULL,NULL,hInstance,NULL
    mov hMainWnd,eax

; If CreateWindowEx failed, display a message & exit.
    .IF eax == 0
      call ErrorHandler
      jmp  Exit_Program
    .ENDIF

; Show and draw the window.
    INVOKE ShowWindow, hMainWnd, SW_SHOW
    INVOKE UpdateWindow, hMainWnd

; Display a greeting message.
    INVOKE MessageBox, hMainWnd, ADDR GreetText,
      ADDR GreetTitle, MB_OK

; Begin the program's message-handling loop.
Message_Loop:
    ; Get next message from the queue.
    INVOKE GetMessage, ADDR msg, NULL,NULL,NULL

    ; Quit if no more messages.
    .IF eax == 0
      jmp Exit_Program
    .ENDIF

    ; Relay the message to the program's WinProc.
    INVOKE DispatchMessage, ADDR msg
    jmp Message_Loop

Exit_Program:
      INVOKE ExitProcess,0
WinMain ENDP

;-----------------------------------------------------
WinProc PROC,
    hWnd:DWORD, localMsg:DWORD, wParam:DWORD, lParam:DWORD
; The application's message handler, which handles
; application-specific messages. All other messages
; are forwarded to the default Windows message
; handler.
;-----------------------------------------------------
    mov eax, localMsg

    .IF eax == WM_LBUTTONDOWN       ; mouse button?
      INVOKE MessageBox, hWnd, ADDR PopupText,
        ADDR PopupTitle, MB_OK
      jmp WinProcExit
    .ELSEIF eax == WM_CREATE        ; create window?
      INVOKE MessageBox, hWnd, ADDR AppLoadMsgText,
        ADDR AppLoadMsgTitle, MB_OK
      jmp WinProcExit
    .ELSEIF eax == WM_CLOSE     ; close window?
      INVOKE MessageBox, hWnd, ADDR CloseMsg,
        ADDR WindowName, MB_OK
      INVOKE PostQuitMessage,0
      jmp WinProcExit
    .ELSE       ; other message?
      INVOKE DefWindowProc, hWnd, localMsg, wParam, lParam
      jmp WinProcExit
    .ENDIF

WinProcExit:
    ret
WinProc ENDP

;---------------------------------------------------
ErrorHandler PROC
; Display the appropriate system error message.
;---------------------------------------------------
.data
pErrorMsg  DWORD ?      ; ptr to error message
messageID  DWORD ?
.code
    INVOKE GetLastError ; Returns message ID in EAX
    mov messageID,eax

    ; Get the corresponding message string.
    INVOKE FormatMessage, FORMAT_MESSAGE_ALLOCATE_BUFFER + \
      FORMAT_MESSAGE_FROM_SYSTEM,NULL,messageID,NULL,
      ADDR pErrorMsg,NULL,NULL

    ; Display the error message.
    INVOKE MessageBox,NULL, pErrorMsg, ADDR ErrorTitle,
      MB_ICONERROR+MB_OK

    ; Free the error message string.
    INVOKE LocalFree, pErrorMsg
    ret
ErrorHandler ENDP

END WinMain
```

程序2-WinApp_v2.asm

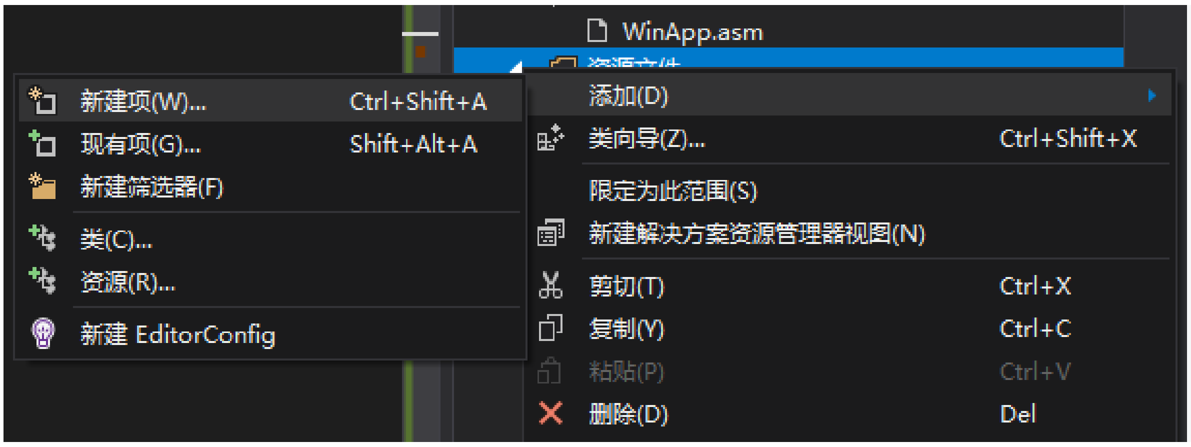
基于`WinApp.asm`修改得到的程序`WinApp_v2.asm`（可以到云盘下载，也已经传到网络学堂上）。使用`windows.inc`而非`Irvine32`和`GraphWin`。

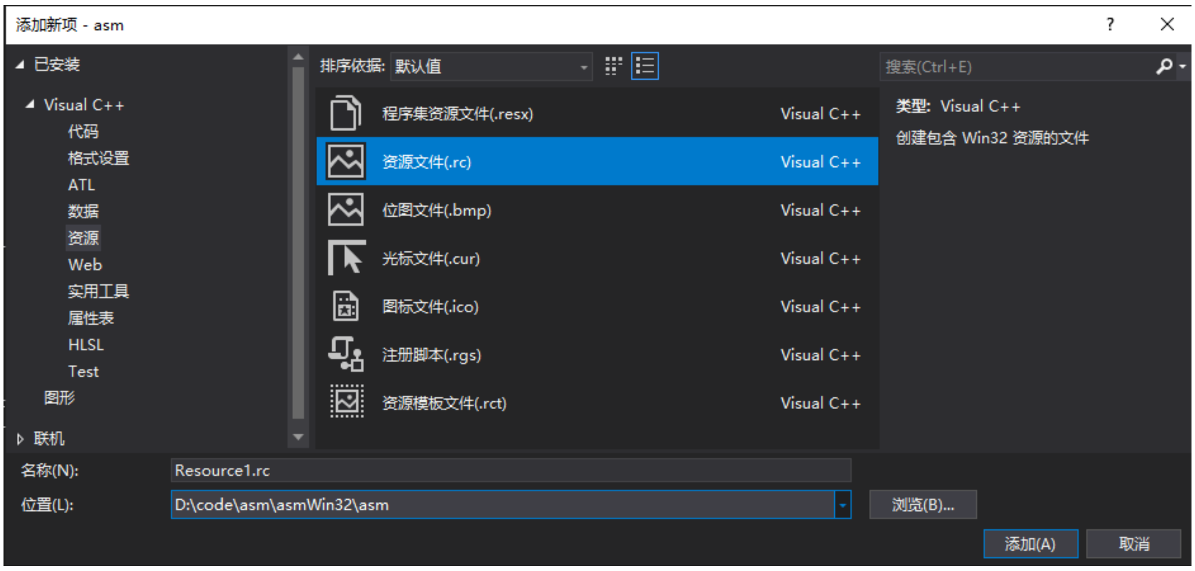
```
TITLE Windows Application                   (WinApp_v2.asm)

; Another version of WinApp.asm
; Modified by: HenryFox
; Last update: 10/13/21
; Original version uses Irvine32 and GraphWin, this version uses windows.inc

; This program displays a resizable application window and
; several popup message boxes.
; Thanks to Tom Joyce for creating a prototype
; from which this program was derived.

.386
.model flat, stdcall
option casemap: none

include         windows.inc
include         gdi32.inc
includelib      gdi32.lib
include         user32.inc
includelib      user32.lib
include         kernel32.inc
includelib      kernel32.lib
include         masm32.inc
includelib      masm32.lib
include         msvcrt.inc
includelib      msvcrt.lib
include         shell32.inc
includelib      shell32.lib

;------------------ Structures ----------------

WNDCLASS STRUC
  style           DWORD ?
  lpfnWndProc     DWORD ?
  cbClsExtra      DWORD ?
  cbWndExtra      DWORD ?
  hInstance       DWORD ?
  hIcon           DWORD ?
  hCursor         DWORD ?
  hbrBackground   DWORD ?
  lpszMenuName    DWORD ?
  lpszClassName   DWORD ?
WNDCLASS ENDS

MSGStruct STRUCT
  msgWnd        DWORD ?
  msgMessage    DWORD ?
  msgWparam     DWORD ?
  msgLparam     DWORD ?
  msgTime       DWORD ?
  msgPt         POINT <>
```

```
MSGStruct ENDS

MAIN_WINDOW_STYLE = WS_VISIBLE+WS_DLGFRAME+WS_CAPTION+WS_BORDER+WS_SYSMENU \
    +WS_MAXIMIZEBOX+WS_MINIMIZEBOX+WS_THICKFRAME

;==================== DATA =======================
.data

;后续部分与WinApp.asm一致
```

资源文件加载

资源文件上面右键，新建项，选择**资源——资源文件(.rc)**



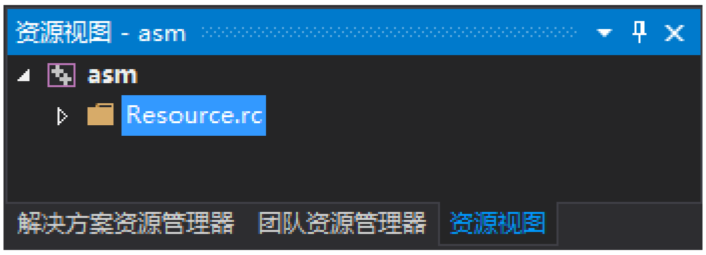


添加新项 - asm
已安装
Visual C++
代码
格式设置
ATL
数据
资源
Web
实用工具
属性表
HLSL
Test
图形
联机
排序依据: 默认值
搜索(Ctrl+E)
程序集资源文件(.resx) Visual C++
资源文件(.rc) Visual C++
位图文件(.bmp) Visual C++
光标文件(.cur) Visual C++
图标文件(.ico) Visual C++
注册脚本(.rgs) Visual C++
资源模板文件(.rct) Visual C++
浏览(B)...
添加(A)
取消

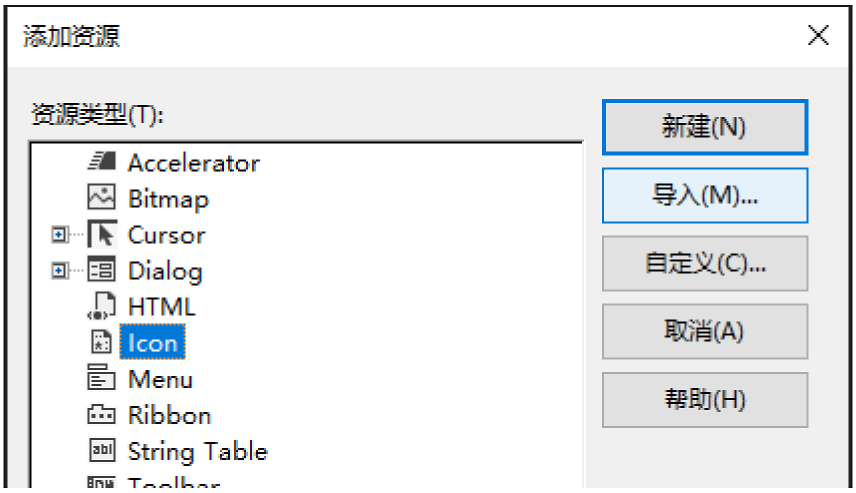
类型: Visual C++
创建包含 Win32 资源的文件

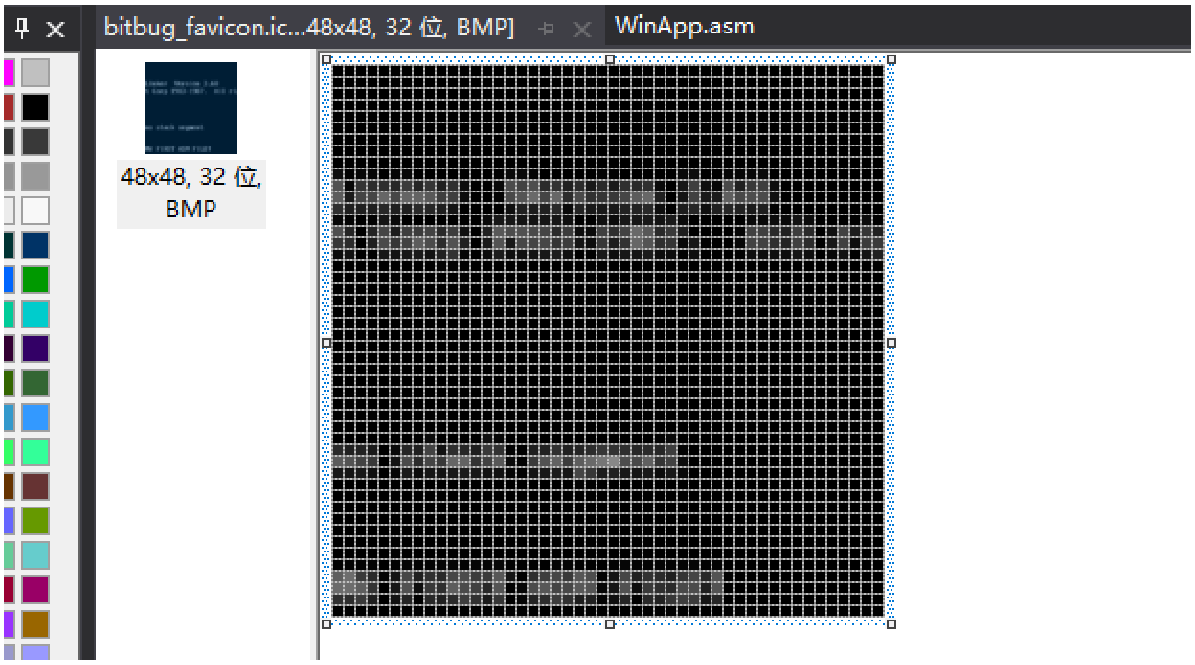
名称(N): Resource1.rc
位置(L): D:\code\asm\asmWin32\asm

之后双击rc文件，会打开资源视图：

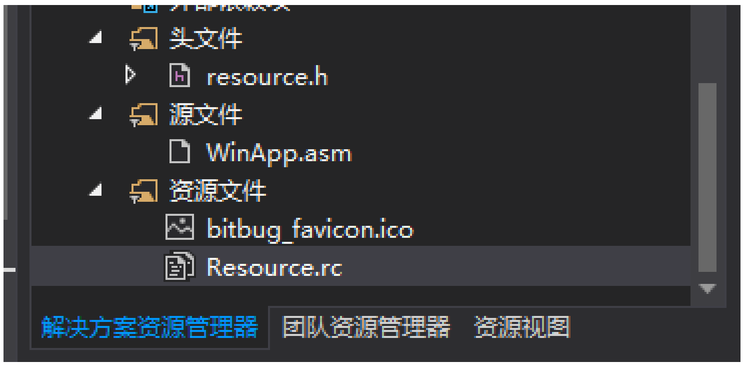


rc文件上右键——添加资源，选择对应的类型，选择导入（这里以ico文件为例），选择自己想要导入的文件即可：

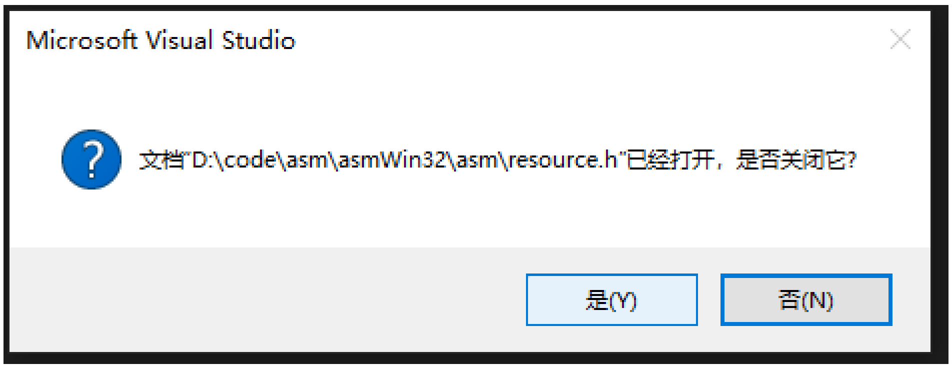




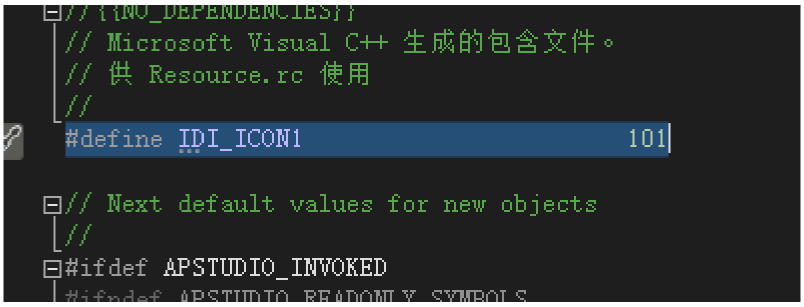
再次切换到解决方案资源管理器标签页，可以看到：



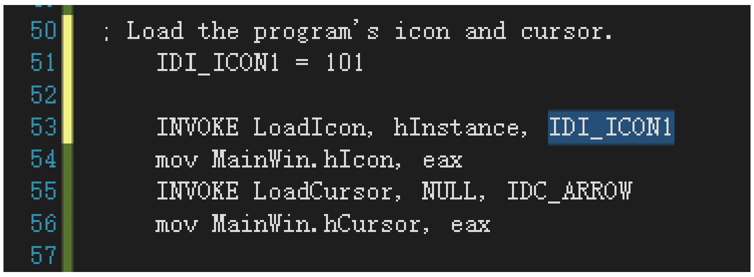
刚才添加的ico文件会出现在这里，并且会有一个`resource.h`文件，双击打开（如果提示已经打开，则点“是”）可以看到一个宏定义，我们需要在`asm`文件中使用同样的数字来指代对应的资源（如图中的101）：


Microsoft Visual Studio

文档"D:\code\asm\asmWin32\asm\resource.h"已经打开，是否关闭它?


```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ 生成的包含文件。
// 供 Resource.rc 使用
//
#define IDI_ICON1                       101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
```

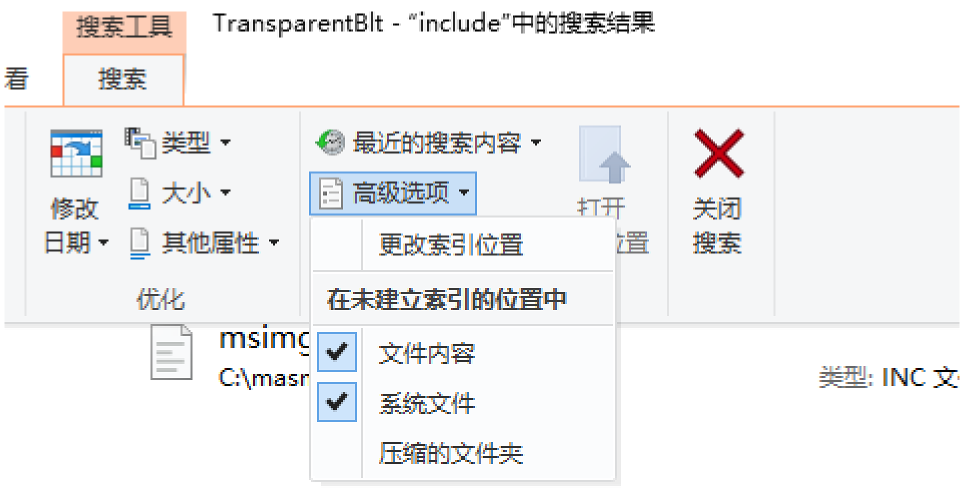
使用时可以在`asm`文件中添加`IDI_ICON1 = 101`一行，并且在合适位置调用（如加载icon，作为程序的图标）：


```
    ; Load the program's icon and cursor.
        IDI_ICON1 = 101

        INVOKE LoadIcon, hInstance, IDI_ICON1
        mov MainWin.hIcon, eax
        INVOKE LoadCursor, NULL, IDC_ARROW
        mov MainWin.hCursor, eax
```

汇编中相关函数查找示例

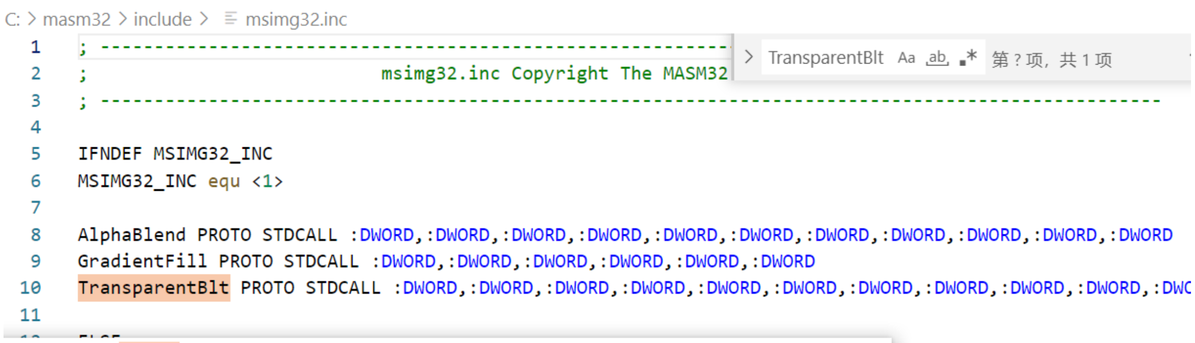
先确认Win32 API中的对应函数名，比如需要用到`TransparentBlt`，可以这样查找对应的`inc`和`lib`文件：

在你的masm32安装路径中的`include`文件夹（如：`C:\masm32\include`）中搜索该内容，记得搜索中高级选项选中“文件内容”：

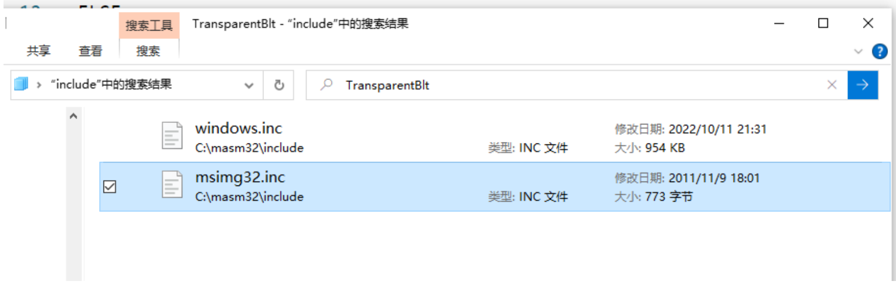


[✓ 文件内容]
[✓ 系统文件]

在搜出的`inc`文件中，确认有这个查找的函数：


C: > masm32 > include > msimg32.inc
TransparentBlt Aa ab .* 第 ? 项，共 1 项

```
; -----------------------------------------------------------------
;                    msimg32.inc Copyright The MASM32
; -----------------------------------------------------------------------------------------------

IFNDEF MSIMG32_INC
MSIMG32_INC equ <1>

AlphaBlend PROTO STDCALL :DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD
GradientFill PROTO STDCALL :DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD
TransparentBlt PROTO STDCALL :DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWORD,:DWC
```

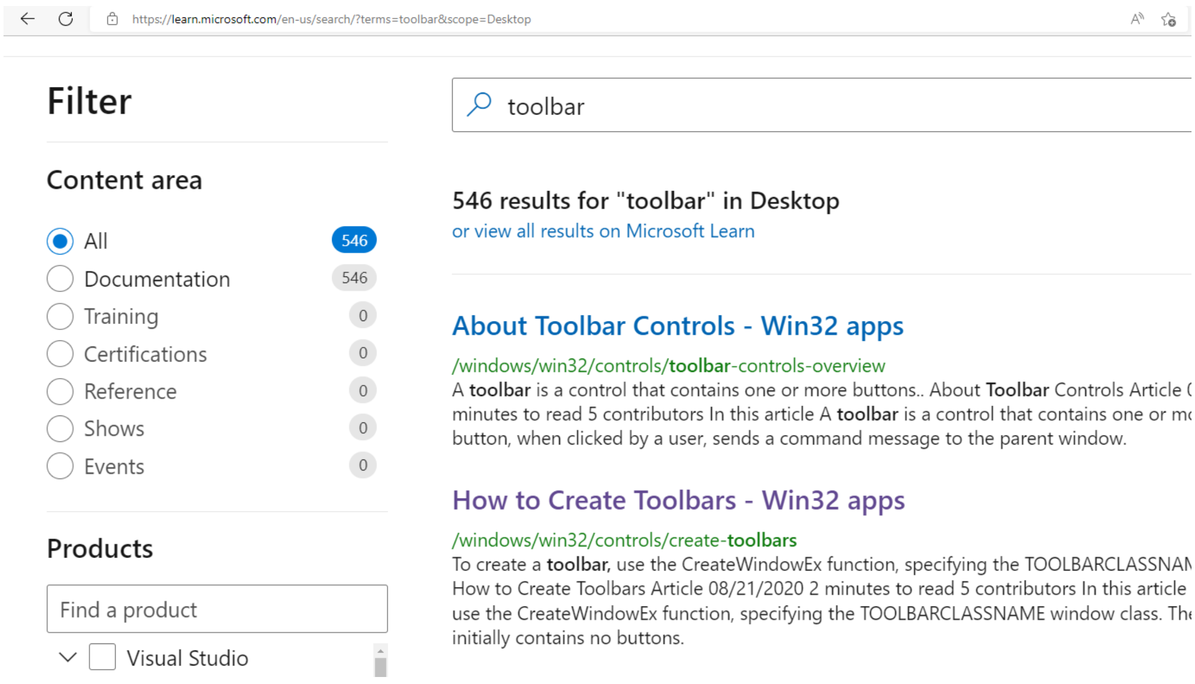


windows.inc C:\masm32\include	类型: INC 文件	修改日期: 2022/10/11 21:31 大小: 954 KB
msimg32.inc C:\masm32\include	类型: INC 文件	修改日期: 2011/11/9 18:01 大小: 773 字节

因为这个文件名是`msimg32`，因此在代码中`include msimg32.inc`并`includelib msimg32.lib`即可。

Win32API相关功能查找示例

建议在 https://learn.microsoft.com/en-us/windows/win32 搜索相关功能，比如如果想创建ToolBar，直接搜索ToolBar：



https://learn.microsoft.com/en-us/search/?terms=toolbar&scope=Desktop

Filter

toolbar

Content area

- All 546
- Documentation 546
- Training 0
- Certifications 0
- Reference 0
- Shows 0
- Events 0

546 results for "toolbar" in Desktop
or view all results on Microsoft Learn

About Toolbar Controls - Win32 apps
/windows/win32/controls/toolbar-controls-overview
A **toolbar** is a control that contains one or more buttons.. About **Toolbar** Controls Article (
minutes to read 5 contributors In this article A **toolbar** is a control that contains one or mo
button, when clicked by a user, sends a command message to the parent window.

Products

Find a product

Visual Studio

How to Create Toolbars - Win32 apps
/windows/win32/controls/create-**toolbars**
To create a **toolbar**, use the CreateWindowEx function, specifying the TOOLBARCLASSNAN
How to Create Toolbars Article 08/21/2020 2 minutes to read 5 contributors In this article
use the CreateWindowEx function, specifying the TOOLBARCLASSNAME window class. The
initially contains no buttons.

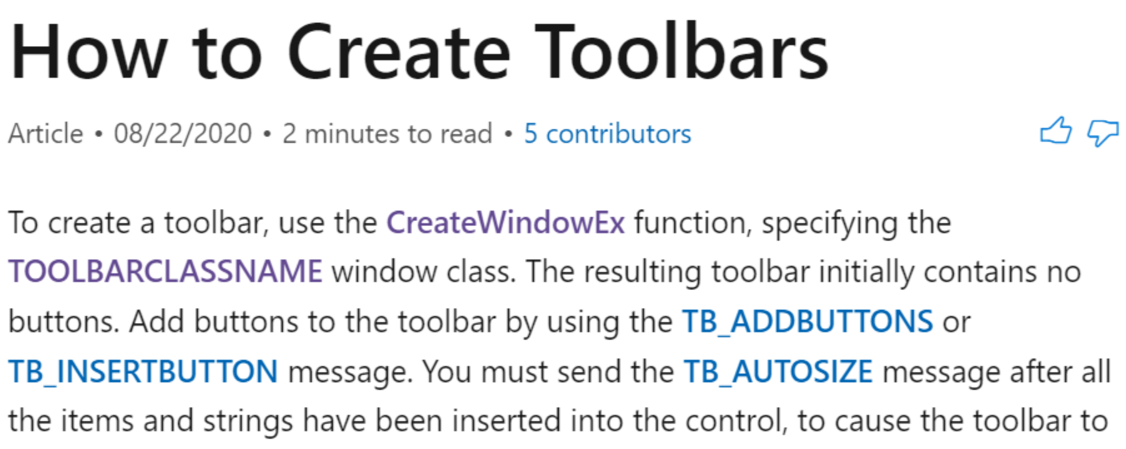
这个How to Create Toolbars - Win32 apps | Microsoft Learn就是你需要的。可以将里面给的代码示例改为汇编形式，如：

```
HWND hwndToolbar = CreateWindowEx(0, TOOLBARCLASSNAME, NULL,
                                  WS_CHILD | TBSTYLE_WRAPABLE, 0, 0, 0, 0,
                                  hwndParent, NULL, g_hInst, NULL);
```

改成：

```
invoke CreateWindowEx, 0, addr TOOLBARCLASSNAME, NULL, \
                       WS_CHILD or TBSTYLE_WRAPABLE, 0, 0, 0, 0, \
                       hwnd, NULL, hInstance, NULL
mov hWinToolBar,eax
```

这里的TOOLBARCLASSNAME可以根据说明

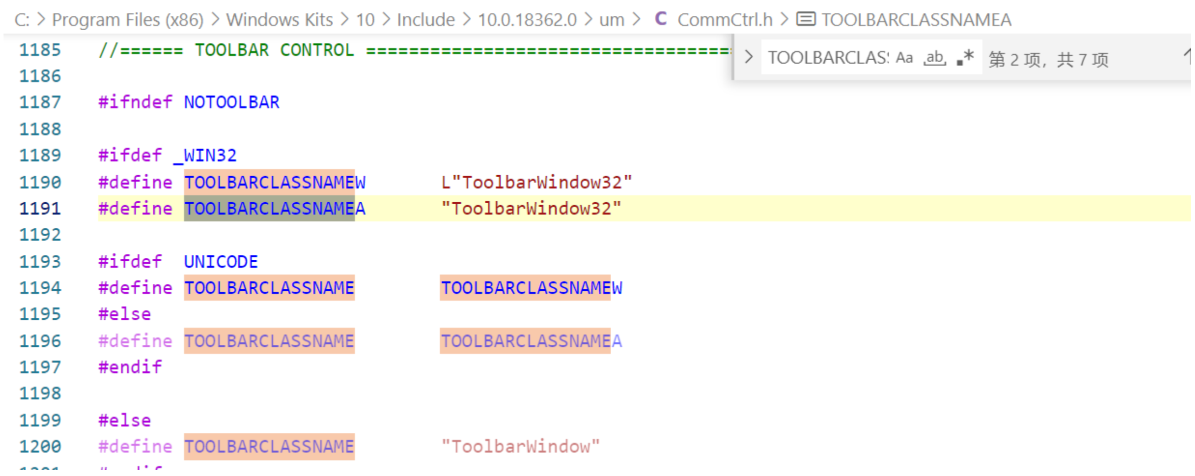


How to Create Toolbars

Article • 08/22/2020 • 2 minutes to read • 5 contributors

To create a toolbar, use the CreateWindowEx function, specifying the TOOLBARCLASSNAME window class. The resulting toolbar initially contains no buttons. Add buttons to the toolbar by using the TB_ADDBUTTONS or TB_INSERTBUTTON message. You must send the TB_AUTOSIZE message after all the items and strings have been inserted into the control, to cause the toolbar to

找到[Window Classes (CommCtrl.h) - Win32 apps | Microsoft Learn]，然后打开本机的相应文件（如：`C:\Program Files (x86)\Windows Kits\10\Include\10.0.18362.0\um\CommCtrl.h`），找到定义：


C: > Program Files (x86) > Windows Kits > 10 > Include > 10.0.18362.0 > um > CommCtrl.h > TOOLBARCLASSNAMEA

```
//====== TOOLBAR CONTROL ==================================

#ifndef NOTOOLBAR

#ifdef _WIN32
#define TOOLBARCLASSNAMEW       L"ToolbarWindow32"
#define TOOLBARCLASSNAMEA       "ToolbarWindow32"

#ifdef  UNICODE
#define TOOLBARCLASSNAME        TOOLBARCLASSNAMEW
#else
#define TOOLBARCLASSNAME        TOOLBARCLASSNAMEA
#endif

#else
#define TOOLBARCLASSNAME        "ToolbarWindow"
```

因此，需要在代码里面提前写：

```
TOOLBARCLASSNAME BYTE "ToolbarWindow32",0
```