Win32 Cheatsheet

C++/Win32 References

Win32.chm
Microsoft Docs (MSDN)
cppreference

Basic Types

```
typedef unsigned int UINT; // u
typedef INT BOOL; // TRUE or FALSE (or -1?)
typedef char CHAR; typedef uint8_t BYTE;
typedef uint16_t WORD, USHORT; // w
typedef int32_t LONG; // 4-byte
typedef uint32_t ULONG, DWORD; // dw
typedef uint64_t ULONGLONG; // 8-byte
typedef ptrdiff_t LONG_PTR;
typedef size_t ULONG_PTR; // 4- or 8-byte
typedef wchar_t WCHAR; // 2-byte
typedef WCHAR TCHAR; // #ifdef UNICODE
typedef CHAR TCHAR; // Ansi
typedef TCHAR *LPTSTR; // psz, for text
typedef const TCHAR *LPCTSTR; // for text
```

P... and LP... are pointer types.
LPC... is const pointer.

Handle Types

HWND --- Window. // hWnd
HMODULE --- Instance of executable.
HINSTANCE --- Same as HMODULE. // hInst
HICON --- Icon. // hIcon
HCURSOR --- Cursor // hCursor
HBITMAP --- Bitmap. // hBitmap
HFONT --- Font. // hFont
HMENU --- Menu. // hMenu
HPEN --- Pen. // hPen
HBRUSH --- Brush. // hBrush
HRGN --- Region. // hRgn
HDC --- Device Context. // hDC
HKEY --- Registry key. // hKey
HACCEL --- Keyboard Accelerators.

Messaging Types

MSG --- Structure for messaging.
WPARAM --- Used in procedure.
LPARAM --- Used in procedure.

WinMain (The main function for Windows app)

```
// For Ansi or (MBCS)
INT WINAPI
WinMain(hinst, hinstPrev, lpCmdLine, nCmdShow);

// For Unicode:
INT WINAPI
wWinMain(hinst, hinstPrev, lpCmdLine, nCmdShow);
```

Message Box (as a simple dialog)

```
INT id = MessageBox(HWND, text, title, mb_flags);
id: IDABORT, IDCANCEL, IDIGNORE, IDNO, IDOK,
    IDRETRY, IDYES.
mb_flags: MB_{OK,OKCANCEL,YESNO,YESNOCANCEL,...}
          MB_ICON{INFORMATION,ERROR,WARNING}.
```

Procedures (callback functions)

```
// Window Procedure
// (Register by RegisterClass[Ex] with class name).
// Message loop is needed.
LRESULT CALLBACK
WindowProc(hwnd, uMsg, wParam, lParam);
default: return DefWindowProc(hwnd, uMsg, wp, lp);
// Message loop:
MSG msg; while (GetMessage(&msg, NULL, 0, 0))
{ TranslateMessage(&msg); DispatchMessage(&msg); }

// Dialog Procedure.
INT_PTR CALLBACK
DialogProc(hwnd, uMsg, wp, lp); default: return 0;
```

Creating Dialogs

```
// Modal dialog.
id = DialogBox(hInst, name, hwndParent, DialogProc);
   // name is the resource name
   // or MAKEINTRESOURCE(res_id).
// Modal dialog with parameter.
id = DialogBox(hInst, name, hwndParent, DialogProc,
               pData);
// Modeless dialog. Use IsDialogMessage in msg loop.
hwnd = CreateDialog(hInst, name, hwndParent,
                    DialogProc);
// Modeless dialog with parameter
hwnd = CreateDialogParam(hInst, name, hwndParent,
                         DialogProc, pData);
```

Dialog Manipulation

```
INT id = GetDlgCtrlID(hwndCtrl);
HWND hwndCtrl = GetDlgItem(hwnd, ctrl_id);
i = GetDlgItemInt(hwnd, ctrl_id, &bTranslated, bSigned);
GetDlgItemText(hwnd, ctrl_id, pszText, cchText);
SendDlgItemMessage(hwnd, ctrl_id,
                   uMsg, wParam, lParam);
SetDlgItemInt(hwnd, ctrl_id, iValue, bSigned);
SetDlgItemText(hwnd, ctrl_id, text);
EndDialog(hwnd, id); //  for modal dialogs
```

Window Manipulation

```
hwnd = CreateWindow(class_name, text, dwStyle,
         x, y, cx, cy, hwndParent, hMenu, hInst, pData);
hwnd = CreateWindowEx(dwExStyle, (...samely...));
b = IsWindow(hwnd);
DestroyWindow(hwnd); // for window or modeless

bEnabled = IsWindowEnabled(hwnd);
EnableWindow(hwnd, bEnable);
bVisible = IsWindowVisible(hwnd);
ShowWindow(hwnd, SW_SHOW or SW_HIDE);

b = IsChild(hwndParent, hwnd);
SetParent(hwndChild, hwndNewParent);
bMaximized = IsZoomed(hwnd);
ShowWindow(hwnd, SW_MAXIMIZE);
bMinimized = IsIconic(hwnd);
ShowWindow(hwnd, SW_MINIMIZE);

hwnd = FindWindow(class, text);
hwnd = FindWindowEx(parent, child_after, class, text);
```

Window Positioning

```
struct POINT { LONG x, y; }; // pt
struct SIZE { LONG cx, cy; }; // size
struct RECT { LONG left, top, right, bottom; }; // rc
GetWindowRect(hwnd, &rc);
GetClientRect(hwnd, &rc);
hwnd = WindowFromPoint(pt);
hwnd = ChildWindowFromPoint(hwndParent, pt);
MapWindowPoints(hwndFrom, hwndTo, ppt, cpt);
MoveWindow(hwnd, x, y, cx, cy, bRedraw);
SetWindowPos(hwnd, child_after, x, y, cx, cy,
             swp_flags);
```