消息映射值集合

[cpp] view plain copy

print?

```cpp
enum AfxSig
{
    AfxSig_end = 0,     // [marks end of message map]

    AfxSig_bD,      // BOOL (CDC*)
    AfxSig_bb,      // BOOL (BOOL)
    AfxSig_bWww,    // BOOL (CWnd*, UINT, UINT)
    AfxSig_hDWw,    // HBRUSH (CDC*, CWnd*, UINT)
    AfxSig_hDw,     // HBRUSH (CDC*, UINT)
    AfxSig_iwWw,    // int (UINT, CWnd*, UINT)
    AfxSig_iww,     // int (UINT, UINT)
    AfxSig_iWww,    // int (CWnd*, UINT, UINT)
    AfxSig_is,      // int (LPTSTR)
    AfxSig_lwl,     // LRESULT (WPARAM, LPARAM)
    AfxSig_lwwM,    // LRESULT (UINT, UINT, CMenu*)
    AfxSig_vv,      // void (void)

    AfxSig_vw,      // void (UINT)
    AfxSig_vww,     // void (UINT, UINT)
    AfxSig_vvii,    // void (int, int) // wParam is ignored
    AfxSig_vwww,    // void (UINT, UINT, UINT)
    AfxSig_vwii,    // void (UINT, int, int)
    AfxSig_vwl,     // void (UINT, LPARAM)
    AfxSig_vbWW,    // void (BOOL, CWnd*, CWnd*)
    AfxSig_vD,      // void (CDC*)
    AfxSig_vM,      // void (CMenu*)
    AfxSig_vMwb,    // void (CMenu*, UINT, BOOL)

    AfxSig_vW,      // void (CWnd*)
    AfxSig_vWww,    // void (CWnd*, UINT, UINT)
    AfxSig_vWp,     // void (CWnd*, CPoint)
    AfxSig_vWh,     // void (CWnd*, HANDLE)
    AfxSig_vwW,     // void (UINT, CWnd*)
    AfxSig_vwWb,    // void (UINT, CWnd*, BOOL)
    AfxSig_vwwW,    // void (UINT, UINT, CWnd*)
    AfxSig_vwwx,    // void (UINT, UINT)
    AfxSig_vs,      // void (LPTSTR)
    AfxSig_vOWNER,  // void (int, LPTSTR), force return TRUE
    AfxSig_iis,     // int (int, LPTSTR)
    AfxSig_wp,      // UINT (CPoint)
    AfxSig_wv,      // UINT (void)
    AfxSig_vPOS,    // void (WINDOWPOS*)
    AfxSig_vCALC,   // void (BOOL, NCCALCSIZE_PARAMS*)
    AfxSig_vNMHDRpl,    // void (NMHDR*, LRESULT*)
    AfxSig_bNMHDRpl,    // BOOL (NMHDR*, LRESULT*)
    AfxSig_vwNMHDRpl,   // void (UINT, NMHDR*, LRESULT*)
    AfxSig_bwNMHDRpl,   // BOOL (UINT, NMHDR*, LRESULT*)
    AfxSig_bHELPINFO,   // BOOL (HELPINFO*)
    AfxSig_vwSIZING,    // void (UINT, LPRECT) -- return TRUE

    // signatures specific to CCmdTarget
    AfxSig_cmdui,   // void (CCmdUI*)
    AfxSig_cmduiw,  // void (CCmdUI*, UINT)
    AfxSig_vpv,     // void (void*)
    AfxSig_bpv,     // BOOL (void*)

    // Other aliases (based on implementation)
    AfxSig_vwwh,                // void (UINT, UINT, HANDLE)
    AfxSig_vwp,                 // void (UINT, CPoint)
    AfxSig_bw = AfxSig_bb,      // BOOL (UINT)
    AfxSig_bh = AfxSig_bb,      // BOOL (HANDLE)
    AfxSig_iw = AfxSig_bb,      // int (UINT)
    AfxSig_ww = AfxSig_bb,      // UINT (UINT)
    AfxSig_bv = AfxSig_wv,      // BOOL (void)
    AfxSig_hv = AfxSig_wv,      // HANDLE (void)
    AfxSig_vb = AfxSig_vw,      // void (BOOL)
    AfxSig_vbh = AfxSig_vww,    // void (BOOL, HANDLE)
    AfxSig_vbw = AfxSig_vww,    // void (BOOL, UINT)
    AfxSig_vhh = AfxSig_vww,    // void (HANDLE, HANDLE)
    AfxSig_vh = AfxSig_vw,      // void (HANDLE)
    AfxSig_viSS = AfxSig_vwl,   // void (int, STYLESTRUCT*)
    AfxSig_bwl = AfxSig_lwl,
    AfxSig_vwMOVING = AfxSig_vwSIZING,  // void (UINT, LPRECT) -- return TRUE

    AfxSig_vW2,                 // void (CWnd*) (CWnd* comes from lParam)
    AfxSig_bWCDS,               // BOOL (CWnd*, COPYDATASTRUCT*)
    AfxSig_bwsp,                // BOOL (UINT, short, CPoint)
    AfxSig_vws,
};
```

收藏到代码笔记

上面的这些值会在 CMDTARG.CPP 文件中的_AfxDispatchCmdMsg（）函数中用到

[cpp] view plain copy

print?

```cpp
AFX_STATIC BOOL AFXAPI _AfxDispatchCmdMsg(CCmdTarget* pTarget, UINT nID, int nCode, AFX_PMSG pfn, void* pExtra, UINT nSig, AFX_CMDHANDLERINFO* pHandlerInfo)
        // return TRUE to stop routing
{
    ASSERT_VALID(pTarget);
    UNUSED(nCode);   // unused in release builds

    union MessageMapFunctions mmf;
    mmf.pfn = pfn;
    BOOL bResult = TRUE; // default is ok

    if (pHandlerInfo != NULL)
    {
        // just fill in the information, don't do it
        pHandlerInfo->pTarget = pTarget;
        pHandlerInfo->pmf = mmf.pfn;
        return TRUE;
    }

    switch (nSig)
    {
    case AfxSig_vv:
        // normal command or control notification
        ASSERT(CN_COMMAND == 0);        // CN_COMMAND same as BN_CLICKED
        ASSERT(pExtra == NULL);
        (pTarget->*mmf.pfn_COMMAND)();
        break;

    case AfxSig_bv:
        // normal command or control notification
        ASSERT(CN_COMMAND == 0);        // CN_COMMAND same as BN_CLICKED
        ASSERT(pExtra == NULL);
        bResult = (pTarget->*mmf.pfn_bCOMMAND)();
        break;

    case AfxSig_vw:
        // normal command or control notification in a range
        ASSERT(CN_COMMAND == 0);        // CN_COMMAND same as BN_CLICKED
        ASSERT(pExtra == NULL);
        (pTarget->*mmf.pfn_COMMAND_RANGE)(nID);
        break;

    case AfxSig_bw:
        // extended command (passed ID, returns bContinue)
        ASSERT(pExtra == NULL);
        bResult = (pTarget->*mmf.pfn_COMMAND_EX)(nID);
        break;

    case AfxSig_vNMHDRpl:
        {
            AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
            ASSERT(pNotify != NULL);
            ASSERT(pNotify->pResult != NULL);
            ASSERT(pNotify->pNMHDR != NULL);
            (pTarget->*mmf.pfn_NOTIFY)(pNotify->pNMHDR, pNotify->pResult);
        }
        break;
    case AfxSig_bNMHDRpl:
        {
            AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
            ASSERT(pNotify != NULL);
            ASSERT(pNotify->pResult != NULL);
            ASSERT(pNotify->pNMHDR != NULL);
            bResult = (pTarget->*mmf.pfn_bNOTIFY)(pNotify->pNMHDR, pNotify->pResult);
        }
        break;
    case AfxSig_vwNMHDRpl:
        {
            AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
            ASSERT(pNotify != NULL);
            ASSERT(pNotify->pResult != NULL);
            ASSERT(pNotify->pNMHDR != NULL);
            (pTarget->*mmf.pfn_NOTIFY_RANGE)(nID, pNotify->pNMHDR,
                pNotify->pResult);
        }
        break;
    case AfxSig_bwNMHDRpl:
        {
            AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
            ASSERT(pNotify != NULL);
            ASSERT(pNotify->pResult != NULL);
            ASSERT(pNotify->pNMHDR != NULL);
            bResult = (pTarget->*mmf.pfn_NOTIFY_EX)(nID, pNotify->pNMHDR,
                pNotify->pResult);
        }
        break;
    case AfxSig_cmdui:
        {
            // ON_UPDATE_COMMAND_UI or ON_UPDATE_COMMAND_UI_REFLECT case
            ASSERT(CN_UPDATE_COMMAND_UI == (UINT)-1);
            ASSERT(nCode == CN_UPDATE_COMMAND_UI || nCode == 0xFFFF);
            ASSERT(pExtra != NULL);
            CCmdUI* pCmdUI = (CCmdUI*)pExtra;
            ASSERT(!pCmdUI->m_bContinueRouting);    // idle - not set
            (pTarget->*mmf.pfn_UPDATE_COMMAND_UI)(pCmdUI);
            bResult = !pCmdUI->m_bContinueRouting;
            pCmdUI->m_bContinueRouting = FALSE;     // go back to idle
        }
        break;

    case AfxSig_cmduiw:
        {
            // ON_UPDATE_COMMAND_UI case
            ASSERT(nCode == CN_UPDATE_COMMAND_UI);
            ASSERT(pExtra != NULL);
            CCmdUI* pCmdUI = (CCmdUI*)pExtra;
            ASSERT(pCmdUI->m_nID == nID);           // sanity assert
            ASSERT(!pCmdUI->m_bContinueRouting);    // idle - not set
            (pTarget->*mmf.pfn_UPDATE_COMMAND_UI_RANGE)(pCmdUI, nID);
            bResult = !pCmdUI->m_bContinueRouting;
            pCmdUI->m_bContinueRouting = FALSE;     // go back to idle
        }
        break;

    // general extensibility hooks
    case AfxSig_vpv:
        (pTarget->*mmf.pfn_OTHER)(pExtra);
        break;
    case AfxSig_bpv:
        bResult = (pTarget->*mmf.pfn_OTHER_EX)(pExtra);
        break;

    default:    // illegal
        ASSERT(FALSE);
        return 0;
    }
```

```cpp
    return bResult;
}
```

但是注意：上面的函数中会出现 MessageMapFunctions 这么一个枚举类型的结构体型结构，一般 F12 是定位不到他的定义的，其实他是在 AFXIMPL.H 文件中定义的

MessageMapFunctions 结构体中实际上定义的是一些函数的指针

[cpp] view plain copy

print?

```cpp
union MessageMapFunctions
{
    AFX_PMSG pfn;   // generic member function pointer

    // specific type safe variants for WM_COMMAND and WM_NOTIFY messages
    void (AFX_MSG_CALL CCmdTarget::*pfn_COMMAND)();
    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_bCOMMAND)();
    void (AFX_MSG_CALL CCmdTarget::*pfn_COMMAND_RANGE)(UINT);
    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_COMMAND_EX)(UINT);

    void (AFX_MSG_CALL CCmdTarget::*pfn_UPDATE_COMMAND_UI)(CCmdUI*);
    void (AFX_MSG_CALL CCmdTarget::*pfn_UPDATE_COMMAND_UI_RANGE)(CCmdUI*, UINT);
    void (AFX_MSG_CALL CCmdTarget::*pfn_OTHER)(void*);
    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_OTHER_EX)(void*);

    void (AFX_MSG_CALL CCmdTarget::*pfn_NOTIFY)(NMHDR*, LRESULT*);
    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_bNOTIFY)(NMHDR*, LRESULT*);
    void (AFX_MSG_CALL CCmdTarget::*pfn_NOTIFY_RANGE)(UINT, NMHDR*, LRESULT*);
    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_NOTIFY_EX)(UINT, NMHDR*, LRESULT*);

    // type safe variant for thread messages

    void (AFX_MSG_CALL CWinThread::*pfn_THREAD)(WPARAM, LPARAM);

    // specific type safe variants for WM-style messages
    BOOL    (AFX_MSG_CALL CWnd::*pfn_bD)(CDC*);
    BOOL    (AFX_MSG_CALL CWnd::*pfn_bb)(BOOL);
    BOOL    (AFX_MSG_CALL CWnd::*pfn_bWww)(CWnd*, UINT, UINT);
    BOOL    (AFX_MSG_CALL CWnd::*pfn_bHELPINFO)(HELPINFO*);
    BOOL    (AFX_MSG_CALL CWnd::*pfn_bWCDS)(CWnd*, COPYDATASTRUCT*);
    HBRUSH  (AFX_MSG_CALL CWnd::*pfn_hDWw)(CDC*, CWnd*, UINT);
    HBRUSH  (AFX_MSG_CALL CWnd::*pfn_hDw)(CDC*, UINT);
    int     (AFX_MSG_CALL CWnd::*pfn_iwWw)(UINT, CWnd*, UINT);
    int     (AFX_MSG_CALL CWnd::*pfn_iww)(UINT, UINT);
    int     (AFX_MSG_CALL CWnd::*pfn_iWww)(CWnd*, UINT, UINT);
    int     (AFX_MSG_CALL CWnd::*pfn_is)(LPTSTR);
    LRESULT (AFX_MSG_CALL CWnd::*pfn_lwl)(WPARAM, LPARAM);
    LRESULT (AFX_MSG_CALL CWnd::*pfn_lwwM)(UINT, UINT, CMenu*);
    void    (AFX_MSG_CALL CWnd::*pfn_vv)(void);

    void    (AFX_MSG_CALL CWnd::*pfn_vw)(UINT);
    void    (AFX_MSG_CALL CWnd::*pfn_vww)(UINT, UINT);
    void    (AFX_MSG_CALL CWnd::*pfn_vvii)(int, int);
    void    (AFX_MSG_CALL CWnd::*pfn_vwww)(UINT, UINT, UINT);
    void    (AFX_MSG_CALL CWnd::*pfn_vwii)(UINT, int, int);
    void    (AFX_MSG_CALL CWnd::*pfn_vwl)(WPARAM, LPARAM);
    void    (AFX_MSG_CALL CWnd::*pfn_vbWW)(BOOL, CWnd*, CWnd*);
    void    (AFX_MSG_CALL CWnd::*pfn_vD)(CDC*);
    void    (AFX_MSG_CALL CWnd::*pfn_vM)(CMenu*);
    void    (AFX_MSG_CALL CWnd::*pfn_vMwb)(CMenu*, UINT, BOOL);

    void    (AFX_MSG_CALL CWnd::*pfn_vW)(CWnd*);
    void    (AFX_MSG_CALL CWnd::*pfn_vWww)(CWnd*, UINT, UINT);
    void    (AFX_MSG_CALL CWnd::*pfn_vWp)(CWnd*, CPoint);
    void    (AFX_MSG_CALL CWnd::*pfn_vWh)(CWnd*, HANDLE);
    void    (AFX_MSG_CALL CWnd::*pfn_vwW)(UINT, CWnd*);
    void    (AFX_MSG_CALL CWnd::*pfn_vwWb)(UINT, CWnd*, BOOL);
    void    (AFX_MSG_CALL CWnd::*pfn_vwwW)(UINT, UINT, CWnd*);
    void    (AFX_MSG_CALL CWnd::*pfn_vwwx)(UINT, UINT);
    void    (AFX_MSG_CALL CWnd::*pfn_vs)(LPTSTR);
    void    (AFX_MSG_CALL CWnd::*pfn_vOWNER)(int, LPTSTR);   // force return
 TRUE
    int     (AFX_MSG_CALL CWnd::*pfn_iis)(int, LPTSTR);
    UINT    (AFX_MSG_CALL CWnd::*pfn_wp)(CPoint);
    UINT    (AFX_MSG_CALL CWnd::*pfn_wv)(void);
    void    (AFX_MSG_CALL CWnd::*pfn_vPOS)(WINDOWPOS*);
    void    (AFX_MSG_CALL CWnd::*pfn_vCALC)(BOOL, NCCALCSIZE_PARAMS*);
    void    (AFX_MSG_CALL CWnd::*pfn_vwp)(UINT, CPoint);
    void    (AFX_MSG_CALL CWnd::*pfn_vwwh)(UINT, UINT, HANDLE);
    BOOL    (AFX_MSG_CALL CWnd::*pfn_bwsp)(UINT, short, CPoint);
    void    (AFX_MSG_CALL CWnd::*pfn_vws)(UINT, LPCTSTR);
};
```

其中 AFX_PMSG 这个结构的定义是在 AFXWIN.H 文件中定义的

[cpp] view plain copy

print?

```
typedef void (AFX_MSG_CALL CCmdTarget::*AFX_PMSG)(void);
```

通过以上的这些数据和函数 MFC 把其内部凌乱的流程函数全部都进行了整理、汇总；但是请注意：_AfxDispatchCmdMsg 虽然是一个 MFC 的全局函数但是 MessageMapFunctions 结构中的函数指针所指向的大多数都是 CCmdTarget 类中的函数，可见 CCmdTarget 类对前期流程控制的重要性