

消息映射值集合

[[cpp](#)] [view plain copy](#)

print?

```
1. enum AfxSig
2. {
3.     AfxSig_end = 0,    // [marks end of message map]
4.
5.     AfxSig_bD,        // BOOL (CDC*)
6.     AfxSig_bb,        // BOOL (BOOL)
7.     AfxSig_bWww,      // BOOL (CWnd*, UINT, UINT)
8.     AfxSig_hDww,      // HBRUSH (CDC*, CWnd*, UINT)
9.     AfxSig_hDw,       // HBRUSH (CDC*, UINT)
10.    AfxSig_iWww,       // int (UINT, CWnd*, UINT)
11.    AfxSig_iww,        // int (UINT, UINT)
12.    AfxSig_iWww,       // int (CWnd*, UINT, UINT)
13.    AfxSig_is,         // int (LPTSTR)
14.    AfxSig_lwl,        // LRESULT (WPARAM, LPARAM)
15.    AfxSig_lwM,        // LRESULT (UINT, UINT, CMenu*)
16.    AfxSig_vv,         // void (void)
17.
18.    AfxSig_vw,         // void (UINT)
19.    AfxSig_vww,        // void (UINT, UINT)
20.    AfxSig_vvii,       // void (int, int) // wParam is ignored
21.    AfxSig_vwww,       // void (UINT, UINT, UINT)
22.    AfxSig_vwii,       // void (UINT, int, int)
23.    AfxSig_vwl,        // void (UINT, LPARAM)
24.    AfxSig_vbWw,       // void (BOOL, CWnd*, CWnd*)
25.    AfxSig_vD,         // void (CDC*)
26.    AfxSig_vM,         // void (CMenu*)
27.    AfxSig_vMwb,       // void (CMenu*, UINT, BOOL)
28.
29.    AfxSig_vW,         // void (CWnd*)
30.    AfxSig_vWww,       // void (CWnd*, UINT, UINT)
31.    AfxSig_vWp,        // void (CWnd*, CPoint)
32.    AfxSig_vWh,        // void (CWnd*, HANDLE)
33.    AfxSig_vwW,        // void (UINT, CWnd*)
34.    AfxSig_vwWb,       // void (UINT, CWnd*, BOOL)
35.    AfxSig_vwwW,       // void (UINT, UINT, CWnd*)
36.    AfxSig_vwwx,       // void (UINT, UINT)
37.    AfxSig_vs,         // void (LPTSTR)
38.    AfxSig_vOWNER,     // void (int, LPTSTR), force return TRUE
39.    AfxSig_iis,        // int (int, LPTSTR)
40.    AfxSig_wp,         // UINT (CPoint)
```

```

41.  AfxSig_wv,      // UINT (void)
42.  AfxSig_vPOS,   // void (WINDOWPOS*)
43.  AfxSig_vCALC,  // void (BOOL, NCCALCSIZE_PARAMS*)
44.  AfxSig_vNMHDRp1, // void (NMHDR*, LRESULT*)
45.  AfxSig_bNMHDRp1, // BOOL (NMHDR*, LRESULT*)
46.  AfxSig_vwNMHDRp1, // void (UINT, NMHDR*, LRESULT*)
47.  AfxSig_bwNMHDRp1, // BOOL (UINT, NMHDR*, LRESULT*)
48.  AfxSig_bHELPINFO, // BOOL (HELPINFO*)
49.  AfxSig_vwSIZING, // void (UINT, LPRECT) -- return TRUE
50.
51.  // signatures specific to CCmdTarget
52.  AfxSig_cmdui,   // void (CCmdUI*)
53.  AfxSig_cmduiw, // void (CCmdUI*, UINT)
54.  AfxSig_vpv,    // void (void*)
55.  AfxSig_bpv,    // BOOL (void*)
56.
57.  // Other aliases (based on implementation)
58.  AfxSig_vvwh,   // void (UINT, UINT, HANDLE)
59.  AfxSig_vvp,    // void (UINT, CPoint)
60.  AfxSig_bw = AfxSig_bb, // BOOL (UINT)
61.  AfxSig_bh = AfxSig_bb, // BOOL (HANDLE)
62.  AfxSig_iw = AfxSig_bb, // int (UINT)
63.  AfxSig_ww = AfxSig_bb, // UINT (UINT)
64.  AfxSig_bv = AfxSig_wv, // BOOL (void)
65.  AfxSig_hv = AfxSig_wv, // HANDLE (void)
66.  AfxSig_vb = AfxSig_vw, // void (BOOL)
67.  AfxSig_vbh = AfxSig_vvw, // void (BOOL, HANDLE)
68.  AfxSig_vbw = AfxSig_vvw, // void (BOOL, UINT)
69.  AfxSig_vhh = AfxSig_vvw, // void (HANDLE, HANDLE)
70.  AfxSig_vh = AfxSig_vw, // void (HANDLE)
71.  AfxSig_viSS = AfxSig_vwl, // void (int, STYLESTRUCT*)
72.  AfxSig_bwl = AfxSig_lwl,
73.  AfxSig_vwMOVING = AfxSig_vwSIZING, // void (UINT, LPRECT) -- return TRUE
    E
74.
75.  AfxSig_vW2,      // void (CWnd*) (CWnd* comes from lParam)
76.  AfxSig_bWCDS,   // BOOL (CWnd*, COPYDATASTRUCT*)
77.  AfxSig_bwsp,    // BOOL (UINT, short, CPoint)
78.  AfxSig_vws,
79. };

```



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?

```
1. AFX_STATIC BOOL AFXAPI _AfxDispatchCmdMsg(CCmdTarget* pTarget, UINT nID, int
   nCode, AFX_PMSG pfn, void* pExtra, UINT nSig, AFX_CMDHANDLERINFO* pHandlerI
   nfo)
2.     // return TRUE to stop routing
3. {
4.     ASSERT_VALID(pTarget);
5.     UNUSED(nCode); // unused in release builds
6.
7.     union MessageMapFunctions mmf;
8.     mmf.pfn = pfn;
9.     BOOL bResult = TRUE; // default is ok
10.
11.    if (pHandlerInfo != NULL)
12.    {
13.        // just fill in the information, don't do it
14.        pHandlerInfo->pTarget = pTarget;
15.        pHandlerInfo->pmf = mmf.pfn;
16.        return TRUE;
17.    }
18.
19.    switch (nSig)
20.    {
21.    case AfxSig_vv:
22.        // normal command or control notification
23.        ASSERT(CN_COMMAND == 0); // CN_COMMAND same as BN_CLICKED
24.        ASSERT(pExtra == NULL);
25.        (pTarget->*mmf.pfn_COMMAND)();
26.        break;
27.
28.    case AfxSig_bv:
29.        // normal command or control notification
30.        ASSERT(CN_COMMAND == 0); // CN_COMMAND same as BN_CLICKED
31.        ASSERT(pExtra == NULL);
32.        bResult = (pTarget->*mmf.pfn_bCOMMAND)();
33.        break;
34.
35.    case AfxSig_vw:
36.        // normal command or control notification in a range
37.        ASSERT(CN_COMMAND == 0); // CN_COMMAND same as BN_CLICKED
38.        ASSERT(pExtra == NULL);
```

```

39.         (pTarget->*mmf.pfn_COMMAND_RANGE)(nID);
40.         break;
41.
42.     case AfxSig_bw:
43.         // extended command (passed ID, returns bContinue)
44.         ASSERT(pExtra == NULL);
45.         bResult = (pTarget->*mmf.pfn_COMMAND_EX)(nID);
46.         break;
47.
48.     case AfxSig_vNMHDRpl:
49.         {
50.             AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
51.             ASSERT(pNotify != NULL);
52.             ASSERT(pNotify->pResult != NULL);
53.             ASSERT(pNotify->pNMHDR != NULL);
54.             (pTarget->*mmf.pfn_NOTIFY)(pNotify->pNMHDR, pNotify->pResult);
55.         }
56.         break;
57.     case AfxSig_bNMHDRpl:
58.         {
59.             AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
60.             ASSERT(pNotify != NULL);
61.             ASSERT(pNotify->pResult != NULL);
62.             ASSERT(pNotify->pNMHDR != NULL);
63.             bResult = (pTarget->*mmf.pfn_bNOTIFY)(pNotify->pNMHDR, pNotify->
pResult);
64.         }
65.         break;
66.     case AfxSig_vwNMHDRpl:
67.         {
68.             AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
69.             ASSERT(pNotify != NULL);
70.             ASSERT(pNotify->pResult != NULL);
71.             ASSERT(pNotify->pNMHDR != NULL);
72.             (pTarget->*mmf.pfn_NOTIFY_RANGE)(nID, pNotify->pNMHDR,
73.                 pNotify->pResult);
74.         }
75.         break;
76.     case AfxSig_bwNMHDRpl:
77.         {
78.             AFX_NOTIFY* pNotify = (AFX_NOTIFY*)pExtra;
79.             ASSERT(pNotify != NULL);
80.             ASSERT(pNotify->pResult != NULL);
81.             ASSERT(pNotify->pNMHDR != NULL);

```

```

82.         bResult = (pTarget->*mmf.pfn_NOTIFY_EX)(nID, pNotify->pNMHDR,
83.             pNotify->pResult);
84.     }
85.     break;
86.     case AfxSig_cmdui:
87.     {
88.         // ON_UPDATE_COMMAND_UI or ON_UPDATE_COMMAND_UI_REFLECT case
89.         ASSERT(CN_UPDATE_COMMAND_UI == (UINT)-1);
90.         ASSERT(nCode == CN_UPDATE_COMMAND_UI || nCode == 0xFFFF);
91.         ASSERT(pExtra != NULL);
92.         CCmdUI* pCmdUI = (CCmdUI*)pExtra;
93.         ASSERT(!pCmdUI->m_bContinueRouting); // idle - not set
94.         (pTarget->*mmf.pfn_UPDATE_COMMAND_UI)(pCmdUI);
95.         bResult = !pCmdUI->m_bContinueRouting;
96.         pCmdUI->m_bContinueRouting = FALSE; // go back to idle
97.     }
98.     break;
99.
100.    case AfxSig_cmduiw:
101.    {
102.        // ON_UPDATE_COMMAND_UI case
103.        ASSERT(nCode == CN_UPDATE_COMMAND_UI);
104.        ASSERT(pExtra != NULL);
105.        CCmdUI* pCmdUI = (CCmdUI*)pExtra;
106.        ASSERT(pCmdUI->m_nID == nID); // sanity assert
107.        ASSERT(!pCmdUI->m_bContinueRouting); // idle - not set
108.        (pTarget->*mmf.pfn_UPDATE_COMMAND_UI_RANGE)(pCmdUI, nID);
109.        bResult = !pCmdUI->m_bContinueRouting;
110.        pCmdUI->m_bContinueRouting = FALSE; // go back to idle
111.    }
112.    break;
113.
114.    // general extensibility hooks
115.    case AfxSig_vpv:
116.        (pTarget->*mmf.pfn_OTHER)(pExtra);
117.        break;
118.    case AfxSig_bpv:
119.        bResult = (pTarget->*mmf.pfn_OTHER_EX)(pExtra);
120.        break;
121.
122.    default: // illegal
123.        ASSERT(FALSE);
124.        return 0;
125.    }

```

```
126.     return bResult;
127. }
```

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?

```
1. union MessageMapFunctions
2. {
3.     AFX_PMSG pfn;    // generic member function pointer
4.
5.     // specific type safe variants for WM_COMMAND and WM_NOTIFY messages
6.     void (AFX_MSG_CALL CCmdTarget::*pfn_COMMAND)();
7.     BOOL (AFX_MSG_CALL CCmdTarget::*pfn_bCOMMAND)();
8.     void (AFX_MSG_CALL CCmdTarget::*pfn_COMMAND_RANGE)(UINT);
9.     BOOL (AFX_MSG_CALL CCmdTarget::*pfn_COMMAND_EX)(UINT);
10.
11.    void (AFX_MSG_CALL CCmdTarget::*pfn_UPDATE_COMMAND_UI)(CCmdUI*);
12.    void (AFX_MSG_CALL CCmdTarget::*pfn_UPDATE_COMMAND_UI_RANGE)(CCmdUI*, UI
    NT);
13.    void (AFX_MSG_CALL CCmdTarget::*pfn_OTHER)(void*);
14.    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_OTHER_EX)(void*);
15.
16.    void (AFX_MSG_CALL CCmdTarget::*pfn_NOTIFY)(NMHDR*, LRESULT*);
17.    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_bNOTIFY)(NMHDR*, LRESULT*);
18.    void (AFX_MSG_CALL CCmdTarget::*pfn_NOTIFY_RANGE)(UINT, NMHDR*, LRESULT*
    );
19.    BOOL (AFX_MSG_CALL CCmdTarget::*pfn_NOTIFY_EX)(UINT, NMHDR*, LRESULT*);
20.
21.    // type safe variant for thread messages
22.
23.    void (AFX_MSG_CALL CWinThread::*pfn_THREAD)(WPARAM, LPARAM);
24.
25.    // specific type safe variants for WM-style messages
26.    BOOL (AFX_MSG_CALL CWnd::*pfn_bd)(CDC*);
27.    BOOL (AFX_MSG_CALL CWnd::*pfn_bb)(BOOL);
28.    BOOL (AFX_MSG_CALL CWnd::*pfn_bWw)(CWnd*, UINT, UINT);
29.    BOOL (AFX_MSG_CALL CWnd::*pfn_bHELPINFO)(HELPINFO*);
30.    BOOL (AFX_MSG_CALL CWnd::*pfn_bWCDS)(CWnd*, COPYDATASTRUCT*);
31.    HBRUSH (AFX_MSG_CALL CWnd::*pfn_hDw)(CDC*, CWnd*, UINT);
```

```

32.  HBRUSH  (AFX_MSG_CALL CWnd::*pfn_hDw)(CDC*, UINT);
33.  int     (AFX_MSG_CALL CWnd::*pfn_iwWw)(UINT, CWnd*, UINT);
34.  int     (AFX_MSG_CALL CWnd::*pfn_iwW)(UINT, UINT);
35.  int     (AFX_MSG_CALL CWnd::*pfn_iwWw)(CWnd*, UINT, UINT);
36.  int     (AFX_MSG_CALL CWnd::*pfn_is)(LPTSTR);
37.  LRESULT (AFX_MSG_CALL CWnd::*pfn_lwL)(WPARAM, LPARAM);
38.  LRESULT (AFX_MSG_CALL CWnd::*pfn_lwWm)(UINT, UINT, CMenu*);
39.  void    (AFX_MSG_CALL CWnd::*pfn_vv)(void);
40.
41.  void    (AFX_MSG_CALL CWnd::*pfn_vw)(UINT);
42.  void    (AFX_MSG_CALL CWnd::*pfn_vwW)(UINT, UINT);
43.  void    (AFX_MSG_CALL CWnd::*pfn_vvii)(int, int);
44.  void    (AFX_MSG_CALL CWnd::*pfn_vwWw)(UINT, UINT, UINT);
45.  void    (AFX_MSG_CALL CWnd::*pfn_vwii)(UINT, int, int);
46.  void    (AFX_MSG_CALL CWnd::*pfn_vwL)(WPARAM, LPARAM);
47.  void    (AFX_MSG_CALL CWnd::*pfn_vbWw)(BOOL, CWnd*, CWnd*);
48.  void    (AFX_MSG_CALL CWnd::*pfn_vD)(CDC*);
49.  void    (AFX_MSG_CALL CWnd::*pfn_vM)(CMenu*);
50.  void    (AFX_MSG_CALL CWnd::*pfn_vMwb)(CMenu*, UINT, BOOL);
51.
52.  void    (AFX_MSG_CALL CWnd::*pfn_vW)(CWnd*);
53.  void    (AFX_MSG_CALL CWnd::*pfn_vWwW)(CWnd*, UINT, UINT);
54.  void    (AFX_MSG_CALL CWnd::*pfn_vWp)(CWnd*, CPoint);
55.  void    (AFX_MSG_CALL CWnd::*pfn_vWh)(CWnd*, HANDLE);
56.  void    (AFX_MSG_CALL CWnd::*pfn_vwW)(UINT, CWnd*);
57.  void    (AFX_MSG_CALL CWnd::*pfn_vwWb)(UINT, CWnd*, BOOL);
58.  void    (AFX_MSG_CALL CWnd::*pfn_vwWw)(UINT, UINT, CWnd*);
59.  void    (AFX_MSG_CALL CWnd::*pfn_vwWx)(UINT, UINT);
60.  void    (AFX_MSG_CALL CWnd::*pfn_vs)(LPTSTR);
61.  void    (AFX_MSG_CALL CWnd::*pfn_vOWNER)(int, LPTSTR); // force return
        TRUE
62.  int     (AFX_MSG_CALL CWnd::*pfn_iis)(int, LPTSTR);
63.  UINT    (AFX_MSG_CALL CWnd::*pfn_wp)(CPoint);
64.  UINT    (AFX_MSG_CALL CWnd::*pfn_wv)(void);
65.  void    (AFX_MSG_CALL CWnd::*pfn_vPOS)(WINDOWPOS*);
66.  void    (AFX_MSG_CALL CWnd::*pfn_vCALC)(BOOL, NCCALCSIZE_PARAMS*);
67.  void    (AFX_MSG_CALL CWnd::*pfn_vwp)(UINT, CPoint);
68.  void    (AFX_MSG_CALL CWnd::*pfn_vwWh)(UINT, UINT, HANDLE);
69.  BOOL    (AFX_MSG_CALL CWnd::*pfn_bwsp)(UINT, short, CPoint);
70.  void    (AFX_MSG_CALL CWnd::*pfn_vws)(UINT, LPCTSTR);
71. };

```

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?

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

通过以上的这些数据 and 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