



Calling C++ Developed DLL and XLL by Excel (2)

2007/08/13 羅紹玫

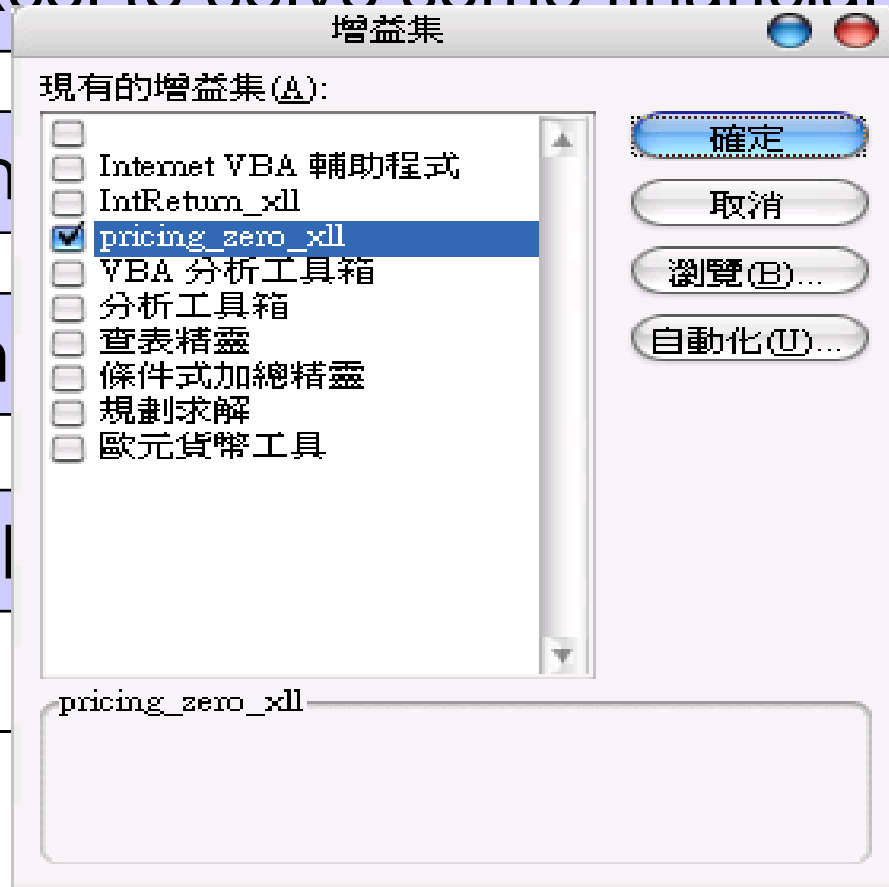
Review

Using Excel to solve some financial problems

function

Dynam

Extensil



Microsoft Excel 97 Developer's Kit

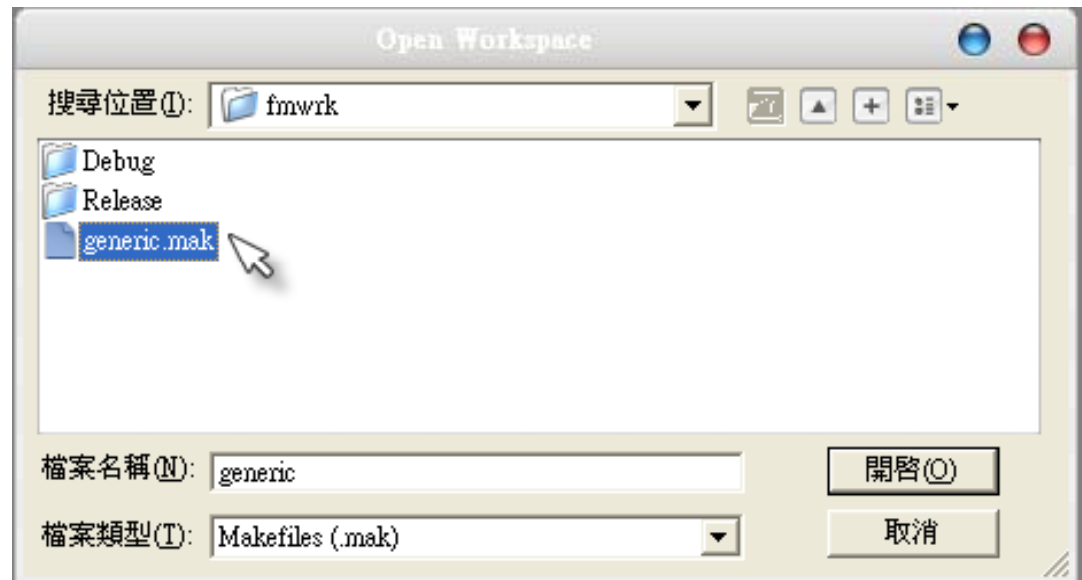
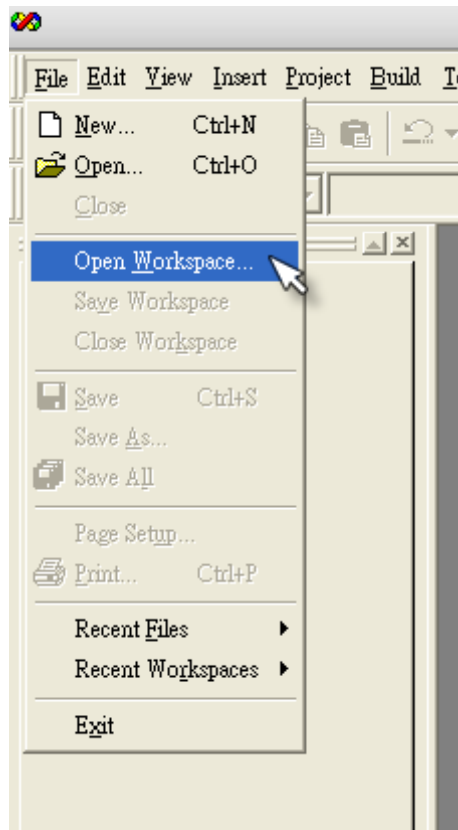
Turning DLLs into XLLs



- Creating XLL using Visual C++ 6.0
 - Download Excel 97 SDK
 - Framework (a template of XLL)
- Accessing XLL by Excel Add-in

Creating XLL using VC6

- Open a template project



Creating XLL using VC6 (Cont.)

- The **xlAuto** Interface function

- xlAutoOpen

- Called when Excel starts up or adds-in loaded

- xlAutoClose

- Called when Excel close down or adds-in unloaded

- xlAutoAdd

- called when the Add-In Manager adds an XLL

- xlAutoRemove

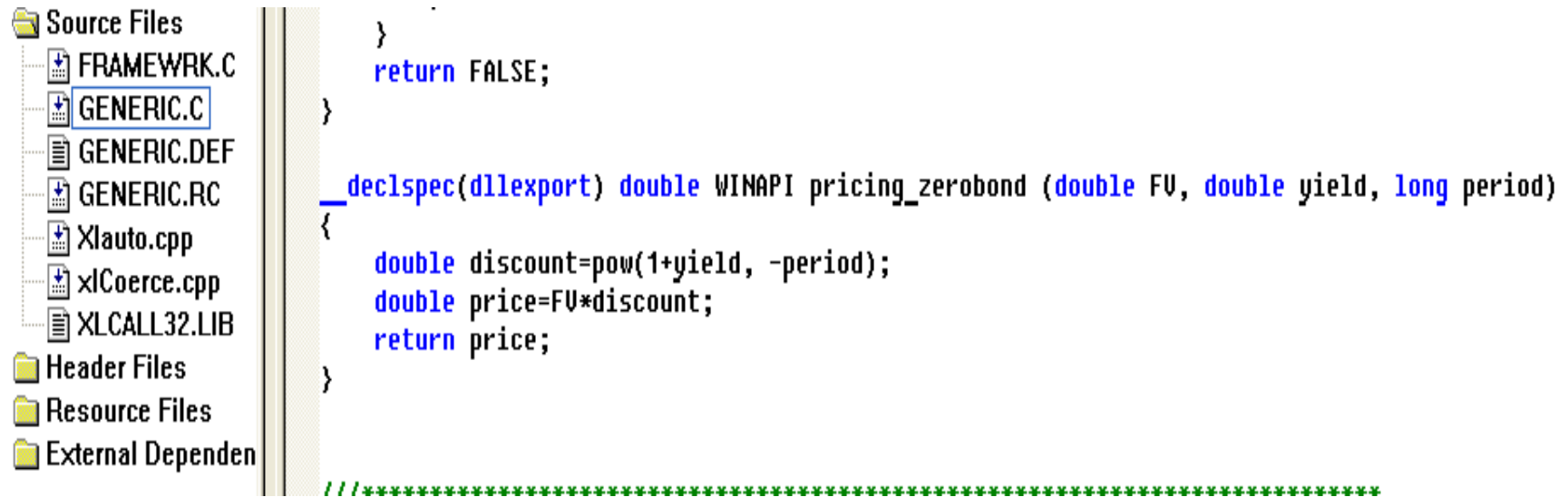
- called when the Add-In Manager removes an XLL

Creating XLL using VC6 (Cont.)

- The **xIAuto** Interface function (cont.)
 - xlAddInManagerInfo
 - Called when the first time the Add-In Manager is invoked
 - xlAutoRegister
 - called if a macro sheet tries to register a function without specifying the argument and return value types

Creating XLL using VC6 (Cont.)

- Adding code (pricing a zero-coupon bond)
-in generic.c

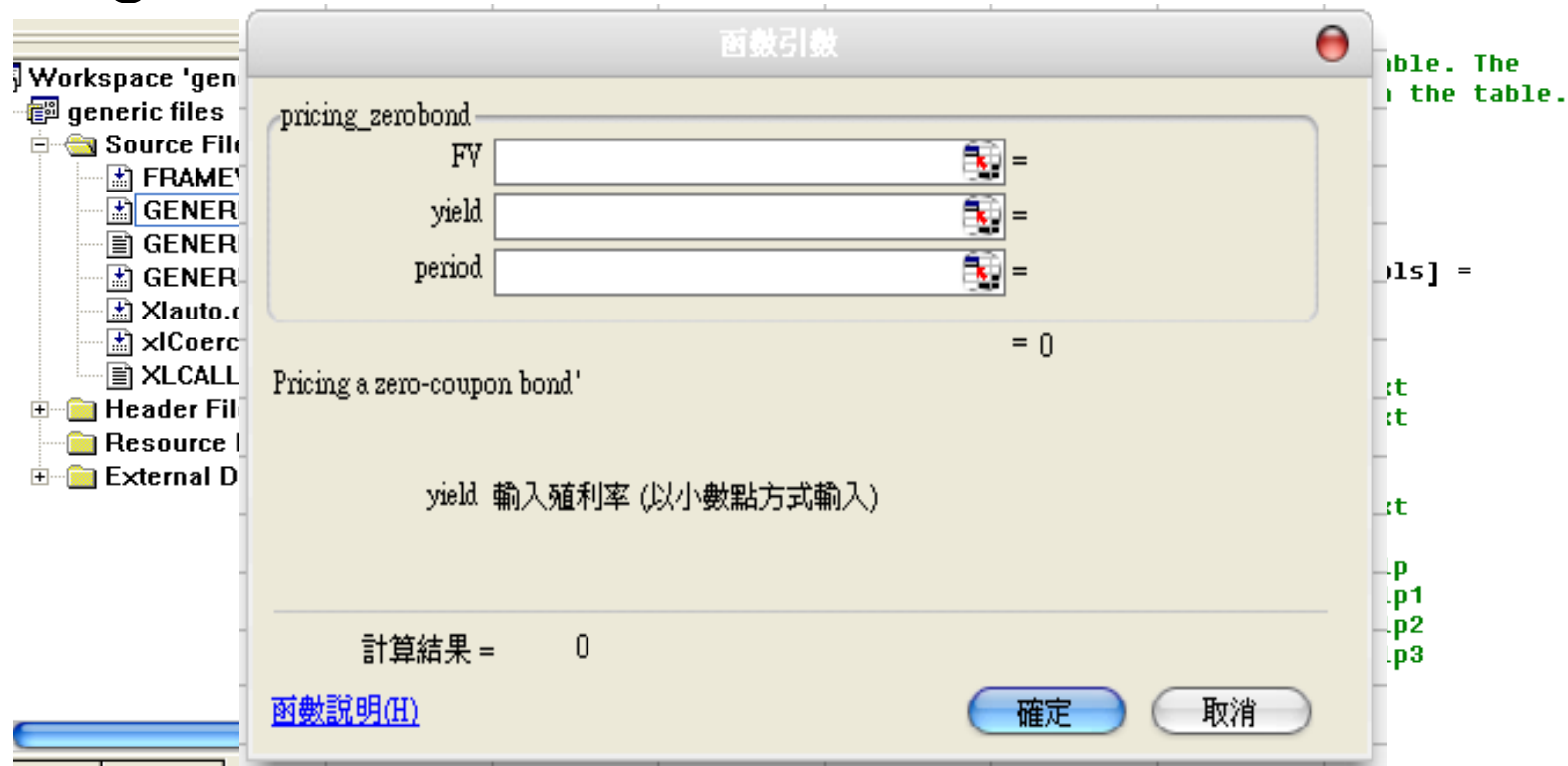


```
    }  
    return FALSE;  
}  
  
_declspec(dllexport) double WINAPI pricing_zero bond (double FV, double yield, long period)  
{  
    double discount=pow(1+yield, -period);  
    double price=FV*discount;  
    return price;  
}  
  
//*****
```

使用 **_declspec(dllexport)** 關鍵字匯出
DLL的資料、函式、類別或類別成員函式

Creating XLL using VC6 (Cont.)

- Adding code
-in generic.c



Creating XLL using VC6 (Cont.)

Code	Description	Pass by	C Declaration
	Logical	Value	checkbox int


```
Source Files
  FRAMEWRK.C
  GENERIC.C
  GENERIC.DEF
  GENERIC.RC
  Xlauto.cpp

Workspace 'generic': 1 p
  generic files
    Source Files
      FRAMEWRK.C
      GENERIC.C
      GENERIC.DEF
      GENERIC.RC
      Xlauto.cpp
      xlCoerce.cpp
      XLCALL32.LIB
    Header Files
    Resource Files
    External Depend

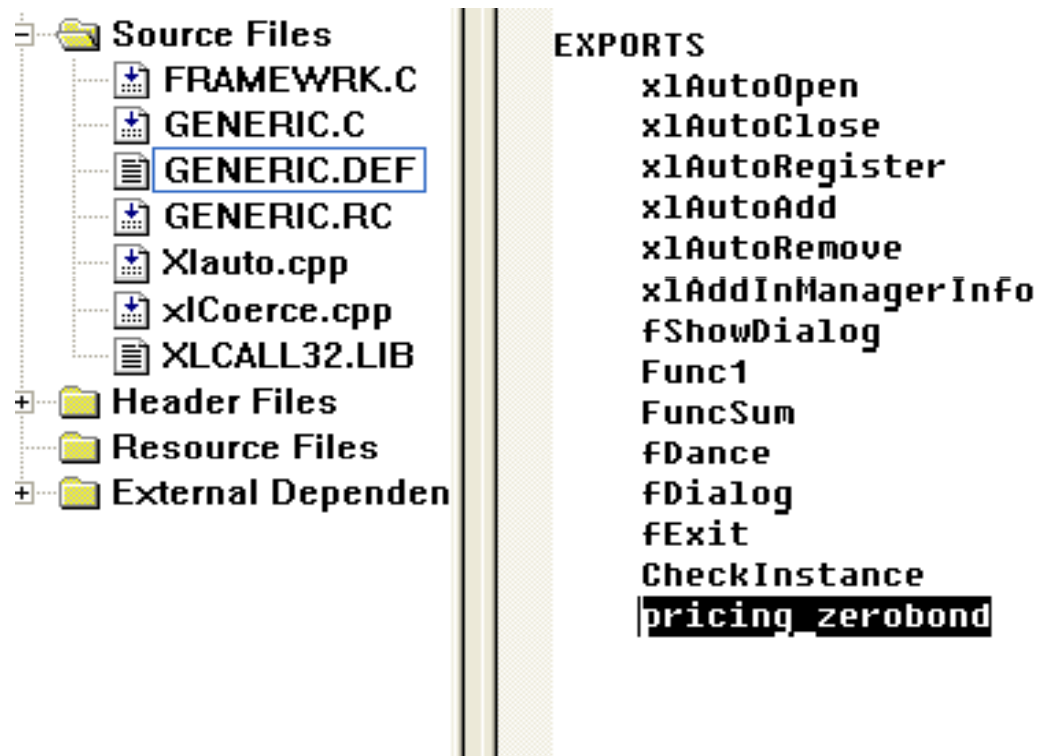
}
return FALSE;
}

_declspec(dllexport) double INAPI pricing_zerobond(double FV, double yield, long period)
{
// arguments two through eleven of the REGISTER function.
// g_rgWorksheetFuncsRows define the number of rows in the table. The
// g_rgWorksheetFuncsCols represents the number of columns in the table.
//
#define g_rgWorksheetFuncsRows 3
#define g_rgWorksheetFuncsCols 12

static LPSTR g_rgWorksheetFuncs
    [g_rgWorksheetFuncsRows][g_rgWorksheetFuncsCols] =
{
    { "pricing_zerobond", // Procedure
      "BBBJ", // type_text
      "pricing_zerobond", // function_text
      "FV, yield, period", // argument_text
      "1", // macro_type
      "Generic Add-In", // category
      "", // shortcut_text
      "", // help_topic
      "Pricing a zero-coupon bond", // function_help
      "輸入零息債券之票面價值", // argument_help1
      "輸入殖利率 (以小數點方式輸入)", // argument_help2
      "輸入零息債券總期間" // argument_help3
    },
    { " Func1", // Procedure
      " BB" // type_text
    }
};
```

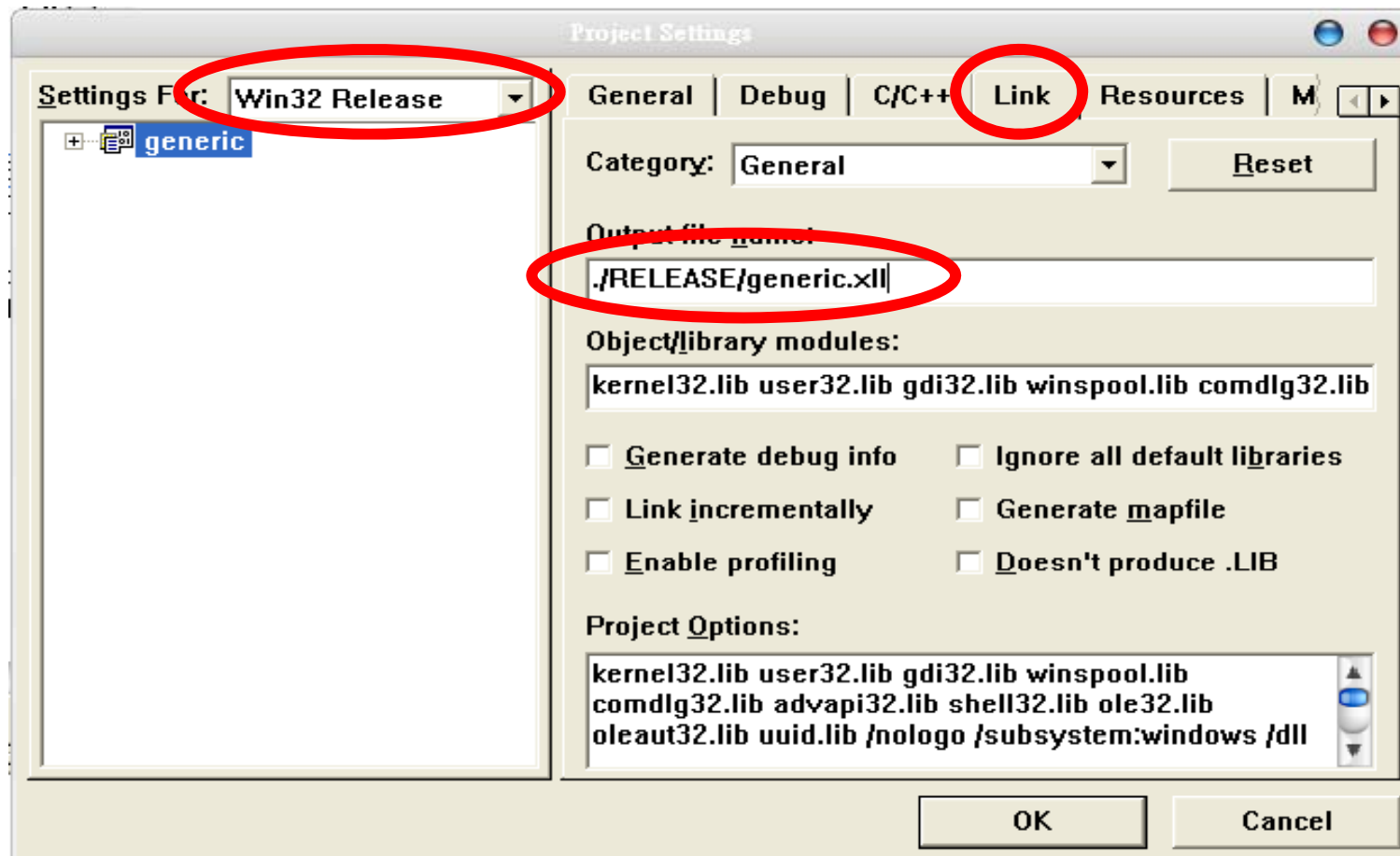
Creating XLL using VC6 (Cont.)

- Adding code
-in generic.def



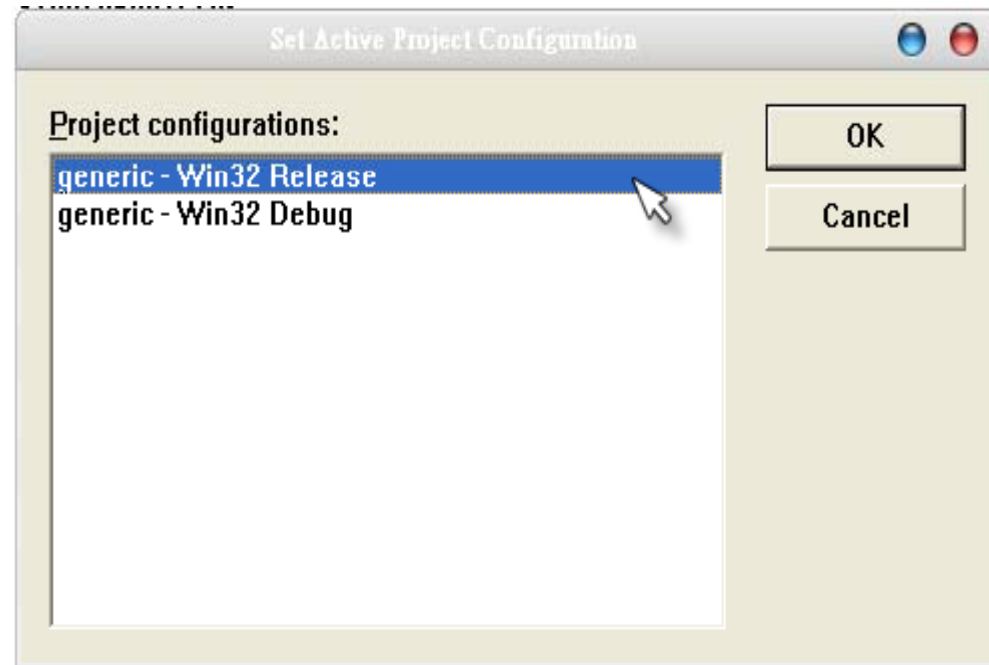
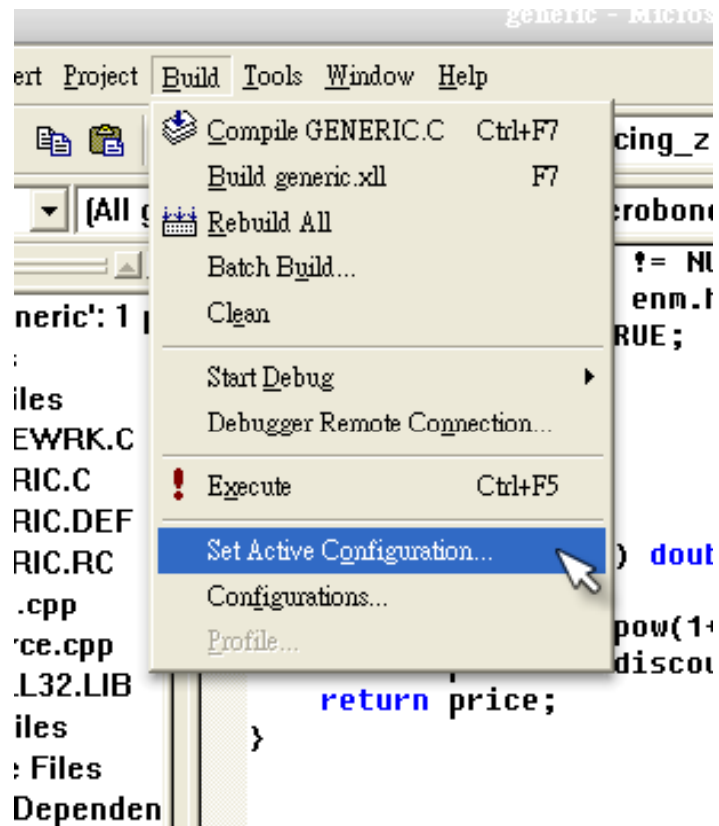
Creating XLL using VC6 (Cont.)

- Project / Settings (Alt+F7)



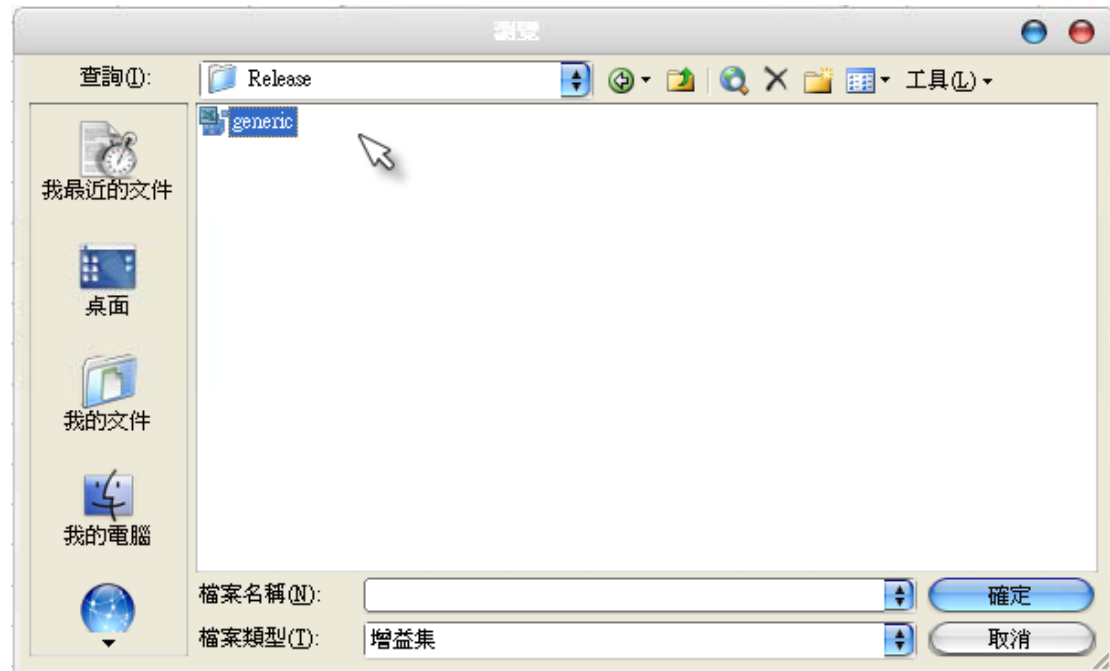
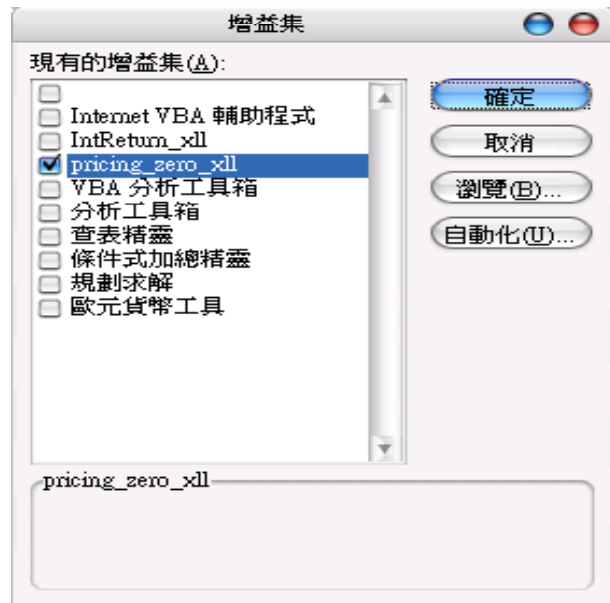
Creating XLL using VC6 (Cont.)

- Build the xll file



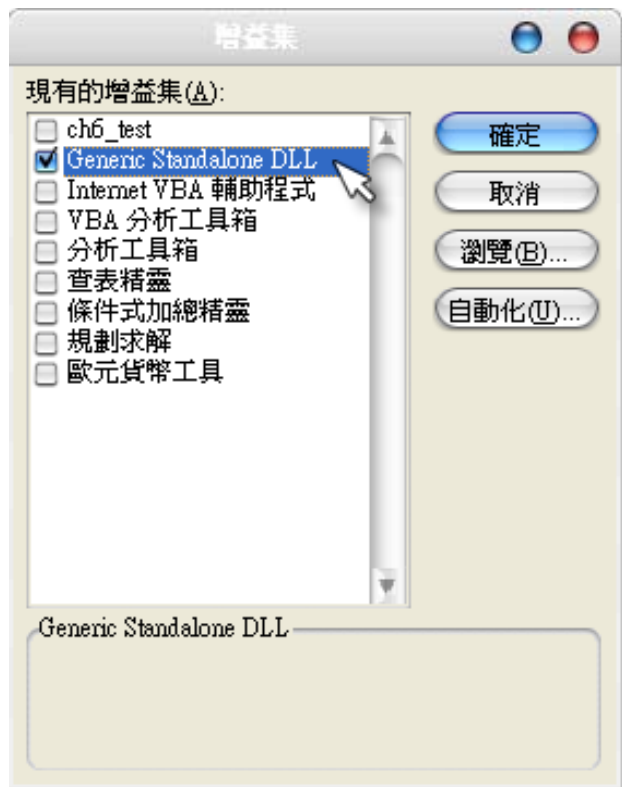
Accessing XLL by Excel

- In Excel



Accessing XLL by Excel (Cont.)

● Result



Pass Data between Excel & DLL

- Four ways to communicate

- Via native C/C++ data types, converted automatically by Excel
- Via a structure that describes and contains 2-D arrays of 8-byte doubles (xl_array)
- Via a structure that can not only represent the contents of any cell, but also ranges and a few other things (xloper)
- Via a structure that can represent the contents of any cell (oper)

Native C/C++ data types

- [signed] short [int] (16-bit), [signed] short [int] * (16-bit), [signed] [long] int, double, double *,.....
- Other type, eg., bool, char are not directly supported.
- If Excel can't convert input value, it will **not** call the function but will instead return **#VALUE!** error.

xl_array



- xl_array structure's advantage
 - Memory management is easy
 - Accessing the data is simple
- xl_array structure's disadvantage
 - xl_array can only contain numbers
 - Difficulties with the freeing of dynamic allocated memory. (Using static)
 - This data type can't be used for optional arguments

xloper--structure

- The xloper can represent
 - Cell value
 - Arrays
 - Single cells, single block, multiple cells...
- The xloper structure contains two part :
 - A 2-byte WORD – data type of xloper
 - An 8-byte C union interpreted accroding to the type of xloper

xloper--structure (Cont.)

```
typedef struct xloper
{
    union
    {
        double num;           /* xtypeNum */
        LPSTR str;            /* xtypeStr */
        WORD bool;           /* xtypeBool */
        WORD err;            /* xtypeErr */
        short int w;         /* xtypeInt */
        struct
        {
            WORD count;      /* always = 1 */
            XLREF ref;
        } sref;              /* xtypeSRef */
        struct
        {
            XLMREF far *lpmref;
            DWORD idSheet;
        } mref;              /* xtypeRef */
        struct
        {
            struct xloper far *lparray;
            WORD rows;
            WORD columns;
        } array;            /* xtypeMulti */
    }
};
```

```
struct
{
    union
    {
        short int level;     /* xlfowRestart */
        short int tbctrl;   /* xlfowPause */
        DWORD idSheet;      /* xlfowGoto */
    } valflow;
    WORD rw;                /* xlfowGoto */
    BYTE col;               /* xlfowGoto */
    BYTE xlfow;             /* xtypeFlow */
} flow;
struct
{
    union
    {
        BYTE far *lpbData;  /* data passed to XL */
        HANDLE hdata;      /* data returned from XL */
    } h;
    long cbData;           /* xtypeBigData */
} bigdata;
} val;
WORD xtype;
} XLOPER, FAR *LPXLOPER;
```

xloper--structure (Cont.)

- xtype types table

```
int __stdcall xloper_type(xloper *p_op)
{
    if(p_op->xltype & xltypeStr)
        return xltypeStr;
    return 0;
}
```

Constant as defined in xlcall.h	Hexadecimal representation
xltypeNum	0x0001
xltypeStr	0x0002
xltypeBool	0x0004
xltypeRef	0x0008
xltypeErr	0x0010
xltypeMulti	0x0040
xltypeMissing	0x0080
xltypeNil	0x0100
xltypeSref	0x0400
xltypeInt	0x0800
xltypeBigData	0x0802