

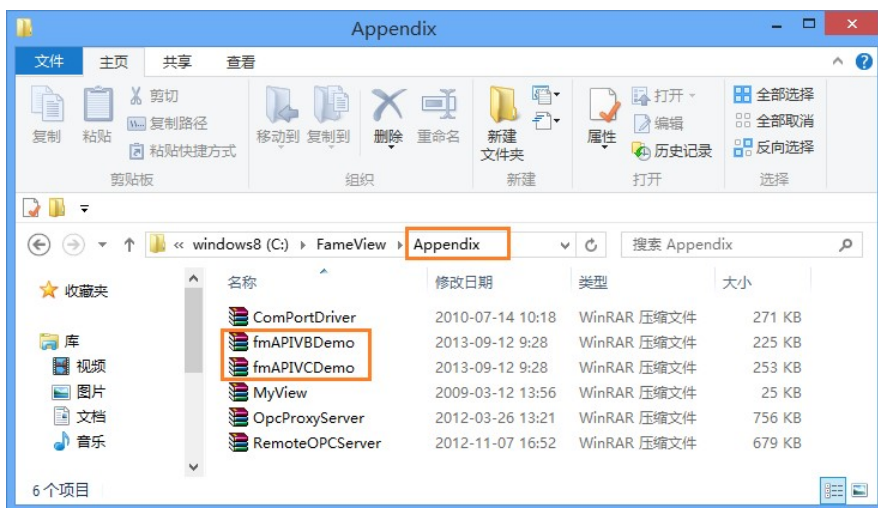
## 17. 用户编程

编号	内容	页码
17.1	API 访问组态系统	17-2
17.2	COM 组件访问本地组态系统	17-11
17.3	COM 组件访问远程组态系统	17-15
17.4	COM 组件访问本地组态项目	17-18
17.5	通过命令行方式操作组态系统	17-23
17.6	串口驱动编程模板	17-28
17.7	编写脚本对象	17-32
17.8	苹果移动设备(iOS)连接网络服务器	17-35

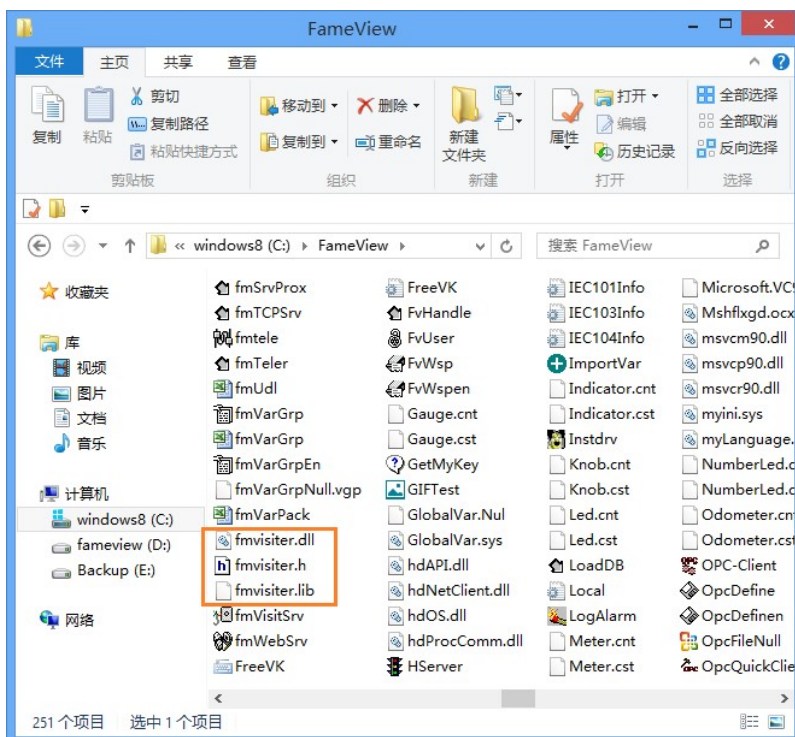


## 17.1 使用 API 访问组态系统

- API 函数支持多进程访问, 缺省不支持多线程访问;  
如果使用多线程并发调用, 建议使用互斥对象避免 API 函数并发执行;  
多线程应用的任一线程调用函数 SetMultiThreadFlag() 后, API 函数自行互斥, 但影响执行效率;
- 系统 Appendix 目录下, 有两个分别使用 VC60 和 VB60 编写的访问系统的例子程序, 供学习参考, 分别为 fmAPIVCDemo、fmAPIVBDemo:



- 系统安装目录下提供 3 个 API 访问库相关文件, 需把文件拷贝到编写代码目录:  
fmvisitor.h、fmvisitor.lib、fmvisitor.dll;



## □ API 提供接口函数:

//设置多线程使用标志, 只需在多线程的某个线程调用即可, API 函数自行互斥, 但影响效率

```
void SetMultiThreadFlag();
```

//[1]打开与关闭运行数据库, 建议修改多个变量时使用, 明显提高效率;

//打开或关闭运行数据库成功返回 1, 否则返回 0;

```
BOOL OpenRunDB();
```

```
BOOL CloseRunDB();
```

//[2]通过变量索引访问变量, 建议使用, 提高访问变量速度;

//VarType - 变量类型, 取值 1-8, 对应 AI/AO/AR/DI/DO/DR/VA/VD

//VarName - 变量名称, 最大长度 20 字符, 支持通配符格式“xxxxx\_[%]”, 例如“3001\_[%]”

//VarIndex - 变量索引 (0-500000)

```
#define AI 1
```

```
#define AO 2
```

```
#define AR 3
```

```
#define DI 4
```

```
#define DO 5
```

```
#define DR 6
```

```
#define VA 7
```

```
#define VD 8
```

```
#define VT 9
```

```
BOOL GetVarIndex(BYTE VarType, char* VarName, int* VarIndex);
```

```
BOOL GetVarDbIValue(BYTE VarType, int VarIndex, double* dblValue);
```

```
BOOL SetVarDbIValue(BYTE VarType, int VarIndex, double dblValue);
```

```
BOOL GetVTVarText(int VarIndex, char* retText);
```

```
BOOL SetVTVarText(int VarIndex, char* setText);
```

//[3]通过变量名称访问变量, 建议对访问速度无要求时使用;

```
BOOL GetAIDbIValue(char* VarName, double* retValue);
```

```
BOOL GetAODbIValue(char* VarName, double* retValue);
```

```
BOOL SetAODbIValue(char* VarName, double setValue);
```

```
BOOL GetARDbIValue(char* VarName, double* retValue);
```

```
BOOL SetARDbIValue(char* VarName, double setValue);
```

```
BOOL GetDIStatus(char* VarName, BYTE* retStatus);
```

```
BOOL GetDOSStatus(char* VarName, BYTE* retStatus);
```

```
BOOL SetDOSStatus(char* VarName, BYTE setStatus);
```

```
BOOL GetDRStatus(char* VarName, BYTE* retStatus);
```

```
BOOL SetDRStatus(char* VarName, BYTE setStatus);
```

```

BOOL  GetVADblValue(char* VarName, double* retValue);
BOOL  SetVADblValue(char* VarName, double setValue);
BOOL  GetVDSStatus(char* VarName, BYTE* retStatus);
BOOL  SetVDSStatus(char* VarName, BYTE  setStatus);
BOOL  GetVTText(char* VarName, char* retText);
BOOL  SetVTText(char* VarName, char* setText);

```

#### //[4]读写变量包

```

BOOL  GetVarPackage(char* VarNames, char* VarValues, char* Separator);
BOOL  SetVarPackage(char* VarNames, char* VarValues, char* Separator, char* retCode);

```

#### //[5]读写设备数据表

```

// DevNo      - 设备号(1-2000)
// StartByte  - 开始字节单元(0-1023)
// ByteLength - 字节长度(1-1024)
// pBuffer    - 设备号数据缓存区
BOOL  GetDevTable(int DevNo, int StartByte, int ByteLength, PBYTE pBuffer);
BOOL  SetDevTable(int DevNo, int StartByte, int ByteLength, PBYTE pBuffer);    //异步
BOOL  SetDevTableSyn(int DevNo, int StartByte, int ByteLength, PBYTE pBuffer); //同步
BOOL  SetBatchDevTableSyn(PBYTE setBuffer, PBYTE retBuffer); //同步批量修改多个设备号
BOOL  SetDevnoClass(int DevNo, int newClass); //设置某设备号动态扫描级别

```

#### //[6]用户管理

```

int  SetUserManagement(char* UserInfo);
int  SetOperateLevel(char* LevelInfo);
int  UserLogin(char* UserName, char* Password);
int  UserLogoff();

```

□ VC++编程举例:

[1]. 把文件 fmvisiter.h、fmvisiter.lib 拷贝到项目目录下, 并加入到项目:

[2]. 使 fmvisiter.dll 文件与编译后可执行文件在相同目录:

[3]. 通过变量名直接访问变量:

```
//读取 AI. VAR1 变量
double dblValue=0;
BOOL bValue=GetAIVarDblValue("VAR1",&dblValue);
//读取 DI. VAR1 变量
BYTE byteValue=0;
BOOL bValue=GetDIVarValue("VAR1",&byteValue);
//修改 AR. VAR1 变量=123. 45
BOOL bValue=SetARVarDblValue("VAR1",123.45);
//修改 DR. VAR1 变量=1
BOOL bValue=SetDRVarValue("VAR1",1);
```

[4]. 读取设备数据表

```
//设备表 D2 的 B0 单元开始, 读取 100 字节数据
BYTE Buffer[100];
GetDevTable(2,0,100,Buffer);
```

[5]. 获取变量索引, 读取变量值

```
int VarIndex=-1;
double VarValue=0;
if(GetVarIndex(1,"AI1",&VarIndex)==TRUE){
    if(VarIndex>=0){
        GetVarDblValue(1,VarIndex,&VarValue);
    }
}
```

[6]. 修改多个 VT 变量(VT0001-VT0300):

```
char varName[21];
char varText[100];
CString ts;
CTime ct=CTime::GetCurrentTime();
if(OpenRunDB()==TRUE){
    for(int i=1;i<=300;i++){
        memset(varName,0,21);
        ts.Format("VT%.4i",i);
```

```

        memcpy(varName, ts, ts.GetLength());
        ts=ct.Format("%Y-%m-%d %H:%M:%S");
        memset(varText, 0, 100);
        memcpy(varText, ts, ts.GetLength());
        SetVTText(varName, varText);
    }
    CloseRunDB();
}

```

[7]. 访问成批变量(VA.VA0001-VA10000), 打开运行数据库, 并通过索引访问;

```

int varIndex[10000];
CString ts;
char ch[21];
for(int i=0;i<10000;i++){
    ts.Format("VA%.4i", i+1);
    memset(ch, 0, 21);
    memcpy(ch, ts, ts.GetLength());
    if(GetVarIndex(VA, ch, &varIndex[i])==FALSE) varIndex[i]=-1;
}
if(OpenRunDB()==TRUE){
    double dblValue=0;
    for(int i=0;i<10000;i++){
        if(varIndex[i]>=0){
            GetVarDblValue(VA, varIndex[i], &dblValue);
            dblValue++;
            SetVarDblValue(VA, varIndex[i], dblValue);
        }
    }
    CloseRunDB();
}

```

## □ 读取变量包

```
BOOL GetVarPackage(char* VarNames, char* VarValues, char* Separator);
```

参数	说明	举例
VarNames	多变量名称, 变量格式 Type. Name	"VA. VA1, VD. VD1, VT. VT1"
	参数内容指定文件, 变量名称源自文件 缺省组态安装路径 MyFile 目录	"file:varnames.txt" "file:c:\\temp\\varnames.txt"
	变量名以数组方式表达	"array:vartype. varname[start, end]"
VarValues	返回多变量值, 须保证缓冲区足够	"12, 0, text"
	参数内容指定文件, 变量值被写入文件 缺省组态安装路径 Temp 目录	"file:varvalues.txt" "file:c:\\temp\\varvalues.txt"
	变量格式错误时, (error)	"1, (error), 3, text"
	变量名称错误时, (null)	"1, (null), 3, text"
	变量通讯无效时, {value}	"1, {0}, 3, text"
Separator	多变量名称或数值间分隔符	", " "; " "   "..."

例 1(直接读取):

```
CString varName="VA. %VA11, VD. %VD1, VT. %VT1";
char retBuffer[100];
memset(retBuffer, 0, 100);
BOOL bValue=GetVarPackage(varName.GetBuffer(0), retBuffer, ", ");
varName.ReleaseBuffer();
```

例 2(通过文件读取):

```
CString varName="file:varnames.txt";
char retBuffer[100];
memset(retBuffer, 0, 100);
CString fileName="file:varpackage.txt";
memcpy(retBuffer, fileName, fileName.GetLength());
BOOL bValue=GetVarPackage(varName.GetBuffer(0), retBuffer, ", ");
varName.ReleaseBuffer();
```

例 3(数组方式, AI 变量中变量名为 VAR1, VAR2, VAR3, ..., VAR10 的 10 个变量, %i 表示通配符):

```
CString varName="array:AI. VAR%i[1, 10]";
```

## □ 修改变量包

```
BOOL SetVarPackage(char* VarNames, char* VarValues, char* Separator, char* retCode);
```

参数	说明	举例
VarNames	多变量名称, 变量格式是 Type.Name	"VA. VA1, VD. VD1, VT. VT1"
	参数内容指定文件, 变量名称源自文件 缺省组态安装路径 MyFile 目录	"file:varnames.txt" "file:c:\\temp\\varnames.txt"
	变量名以数组方式表达	"array:vartype. varname[start, end]"
VarValues	设置多变量值	"1, 2, 3, text"
	参数内容指定文件, 则变量值源自文件 缺省组态安装路径 MyFile 目录	"file:varvalues.txt" "file:c:\\temp\\varvalues.txt"
Separator	多变量名称或数值间分隔符	"," " "; " "   "..."
RetCode	各变量修改返回代码, 1=成功, 0=失败	"1, 0, 1"

例 1(直接修改):

```
CString varName="VA. %VA1, VD. %VD1, VT. %VT1";
CString varValue="11, 0, 1234";
char retBuffer[100];
memset(retBuffer, 0, 100);
BOOL bValue=SetVarPackage(varName.GetBuffer(0), varValue.GetBuffer(0), ",", retBuffer);
varName.ReleaseBuffer();
varValue.ReleaseBuffer();
```

例 2(通过文件修改):

```
CString varName="file:varnames.txt";
CString varValue="file:varvalues.txt";
char retBuffer[100];
memset(retBuffer, 0, 100);
CString fileName="file:retCode.txt";
memcpy(retBuffer, fileName, fileName.GetLength());
BOOL bValue=SetVarPackage(varName.GetBuffer(0), varValue.GetBuffer(0), ",", retBuffer);
varName.ReleaseBuffer();
varValue.ReleaseBuffer();
```

例 3(数组方式, AR 变量中变量名为 VAR1, VAR2, VAR3, ..., VAR10 的 10 个变量, %i 表示通配符):

```
CString varName="array:AR. VAR%i[1, 10]";
```



- 同步批量修改 1-1024 个设备号, 等待完成并返回修改结果

BOOL SetBatchDevTableSyn(PBYTE setBuffer, PBYTE retBuffer);

setBuffer, 修改设备表缓冲区, 格式如下:

[0][1][2]	缓冲区总长度	高字节在前, 低字节在后
[3][4]	设备号 (1-2000)	修改第 1 个设备号
[5][6]	起始字节 (0-1023)	
[7][8]	字节长度 (1-1024)	
[9]..[n]	修改数据	
[n+1][n+2]	设备号 (1-2000)	修改第 2 个设备号
[n+3][n+4]	起始字节 (0-1023)	
[n+5][n+6]	字节长度 (1-1024)	
[n+7]..[m]	修改数据	
...	...	...

retBuffer, 每设备号修改结果:

0=成功, 1=设备号错误, 2=开始字节错误, 3=字节长度错误, 4/5=通讯中断, 6=修改失败, 7=数据无变化;

举例:

```

BYTE Buffer[15*1000];
BYTE devData[10][1000];
WORD devNo=0, startByte=0, byteLength=1000;
//修改设备号缓冲区
DWORD i=3;
for(int j=2; j<=11; j++) {
    devNo=j;
    Buffer[i]=devNo/0x100; i++;
    Buffer[i]=devNo&0xFF; i++;
    Buffer[i]=startByte/0x100; i++;
    Buffer[i]=startByte&0xFF; i++;
    Buffer[i]=byteLength/0x100; i++;
    Buffer[i]=byteLength&0xFF; i++;
    memcpy(&Buffer[i], devData[j-2], byteLength);
    i+=byteLength;
}
//总长度
Buffer[0]=i/0x10000;
Buffer[1]=i/0x100;
Buffer[2]=i&0xFF;
BYTE retBuffer[10];
BOOL b=SetBatchDevTableSyn(Buffer, retBuffer);

```

## □ VB 编程举例:

## [1]. 访问设备数据表:

```

Dim Buffer(1000) As Byte
Dim ret As Long
ret = GetDevTable(1, 0, 1000, Buffer(0))
ret = OpenRunDB()
If ret = 1 Then
    Buffer(0) = Buffer(0) + 1
    SetDevTable 1, 0, 1000, Buffer(0)
    ret = CloseRunDB()
End If

```

## [2]. 通过索引访问变量:

```

Dim ret As Long
ret = OpenRunDB()
If ret = 1 Then
    Dim varIndex As Long
    ret = GetVarIndex(7, "%VA1", varIndex)
    If varIndex >= 0 Then
        Dim varValue As Double
        ret = GetVarDbIValue(7, varIndex, varValue)
        varValue = varValue + 1
        ret = SetVarDbIValue(7, varIndex, varValue)
        Text2.Text = varValue
    End If
    ret = CloseRunDB()
End If

```

## [3]. 通过变量名称直接访问 VA 变量:

```

Dim Value As Double
Dim ret As Long
ret = GetVADbIValue("%VA1", Value)
Value= Value+1
ret = SetVADbIValue("%VA1", Value)

```

## [4]. 通过变量名称直接访问 VT 变量:

```

Dim Buffer() As Byte
Buffer = Space$(200)
GetVTText "%TIME", Buffer(0)
Dim s As String
s = StrConv(Buffer,vbUnicode)

```

## 17.2 使用 COM 组件访问本地组态系统

□ 监控系统提供 COM 组件, 访问本地变量;

组件名称		fmDMO.RunDb	
对应文件		fmDMO.dll	
组 件 方 法	启动监控系统		Startup(Timeout)
	停止监控系统		Stop()
	初始化运行数据库		VARIANT Init()
	打开运行数据库		VARIANT Open()
	关闭运行数据库		Close()
	得到变量索引		VarIndex=GetVarIndex(VarType, VarName)
	通过变量索引读取变量值		VarValue=GetVarValue(VarType, VarIndex)
	通过变量索引修改变量值		SetVarValue(VarType, VarIndex, VarValue)
	通过变量名称读取变量值		VarValue=GetVarValueEx(VarName1)
	通过变量名称修改变量值		SetVarValueEx(VarName1, VarValue)
	批量读取多变量		VarCount=GetMulVarValues(VarNames, ValueArray)
	批量修改多变量		VarCount=SetMulVarValues(VarNames, ValueArray)
	读取设备表数据		GetDevTable(DevNo, StartByte, ByteLength, Buffer)
	修改设备表数据		SetDevTable (DevNo, StartByte, ByteLength, Buffer)
	读取某类过滤变量值		FilterValues=ReadFilterVarValues(VarType1, VarFilter)
参 数	VarType	变量类型	1=AI, 2=AO, 3=AR, 4=DI, 5=DO, 6=DR, 7=VA, 8=VD, 9=VT
	VarName	变量名称	最大长度 20 字符
	VarName1	变量名称 1	格式: "XX.YYYYYYYYYY"
	VarIndex	变量索引	范围: 0-500000
	VarValue	变量值	类型: VT_R8 VT_UI1 VT_BSTR
	VarNames	多个变量名	例如: "AI. AI1 AR. AR1 DI. DI1 DR. DR1 VA. VA1 VD. VD1"
	ValueArray	数值数组	类型: VT_R8 VT_UI1 VT_BSTR
	VarFilter	变量过滤名	"P1_*", 最多支持 3000 变量
	FilrerValues	变量过滤值	"ok(1-3000): P1_AI1, 10  P1_AI2, 20  P1_AI3, {0} "
	DevNo	设备号	1-2000
	StartByte	开始字节	0-1023
	ByteLength	字节长度	1-1024
	Buffer	字节缓冲区	1-1024

□ 例 1 (VC++, 通过变量索引访问变量):

[1]. 函数 InitInstance() 必须添加:

```
if(!AfxOleInit()) return FALSE;
```

[2]. stdafx.h 头文件添加:

```
#import "c:\\fameview\\fmDmo.dll" no_namespace named_guids
```

[3]. 访问变量:

```
{
    IRundbPtr pRundbObj;
    HRESULT hr=pRundbObj.CreateInstance("fmDMO.Rundb");
    if(hr!=0) {
        return;
    }
    _variant_t vt=pRundbObj->Open();
    if(vt.iVal!=1) {
        return;
    }
    _variant_t varType;
    varType.vt=VT_UI1;
    varType.bVal=7;
    _bstr_t varName="%VA1";
    _variant_t varIndex=pRundbObj->GetVarIndex(varType, varName);
    if(varIndex.lVal>=0) {
        _variant_t varValue=pRundbObj->GetVarValue(varType, varIndex);
        varValue.dblVal+=1;
        pRundbObj->SetVarValue(varType, varIndex, varValue);
    }
    pRundbObj->Close();
}
```

□ 例 2 (VBScript, 通过变量索引访问变量):

```
Set obj=CreateObject("fmDMO.Rundb")
```

```
If obj.Open()=1 Then
```

```
    n=obj.GetVarIndex(7, "%VA1")
```

```
    x=obj.GetVarValue(7, n)
```

```
    x=x+1
```

```
    obj.SetVarValue 7, n, x
```

```
    obj.Close
```

```
End If
```

```
Set obj=Nothing
```

- 例3 (VBScript, 通过变量索引访问文本变量);

```
Set obj=CreateObject("fmDMO.Rundb")
If obj.Open()=1 Then
    n=obj.GetVarIndex(9,"VT1")
    s=obj.GetVarValue(9,n)
    obj.SetVarValue 9,n,s&"abc"
    obj.Close
End If
Set obj=Nothing
```

- 例4 (VB, 通过变量索引访问变量)

```
Dim obj As Object
Set obj = CreateObject("fmDMO.Rundb")
If obj.Open() = 1 Then
    Dim n As Long
    n = obj.GetVarIndex(7,"%VA1")
    Dim x As Single
    x = obj.GetVarValue(7, n)
    x = x + 1
    obj.SetVarValue 7, n, x
    obj.Close
End If
Set obj = Nothing
```

- 例5 (VBScript, 通过变量名直接访问变量);

```
Set obj=CreateObject("fmDMO.Rundb")
If obj.Open()=1 Then
    x=obj.GetVarValueEx("VA.VA1")
    x=x+1
    obj.SetVarValueEx "VA.VA1",x
    obj.Close
End If
Set obj=Nothing
```

- 例6 (VB, 批量读取变量):

```
Dim valueArray(3)
nCount=obj.GetMulVarValue("VA.VA1|AI.AI1|DI.DI1|",valueArray)
nCount=obj.GetMulVarValue("varnames.txt",valueArray)
```

## □ 例 7 (VB, 批量修改变量):

```

Dim valueArray(3)
valueArray(0)=10
valueArray(1)=20
valueArray(2)=30
nCount=obj.SetMulVarValue("VA. VA1|AR. AR1|DR. DR1|", valueArray)
nCount=obj.SetMulVarValue("varnames.txt", valueArray)

```

## □ 例 8 (VB, 访问设备数据表):

```

Dim Buffer(1000)
Set obj = CreateObject("fmDM0.Rundb")
If obj.Open() = 1 Then
    obj.GetDevTable 1, 0, 10, Buffer
    obj.SetDevTable 1, 0, 10, Buffer
    obj.Close
End If
Set obj = Nothing

```

## □ 例9 (VBScript, 读取某类全部变量值);

```

Set obj=CreateObject("fmDM0.Rundb")
If obj.Open()=1 Then
    s=obj.ReadFilterVarValues("AI", "P1_*") ' s="ok(3):|P1_AI1,10| P1_AI1,20| P1_AI1,30|"
    obj.Close
End If
Set obj=Nothing

```

### 17.3 使用 COM 组件访问远程组态系统

□ 监控系统提供 COM 组件, 访问远程运行数据库;

适用于少量数据访问, 大批量数据访问建议使用 Socket 编程连接网络服务器;

组件名称		fmdm0.TCPRunDb
组件文件		fmdm0.dll
组件方法	设置本地网卡	SetLocalIp localIp 如果不调用此函数, 则使用缺省的 IP 地址
	打开远程运行数据库	VARIANT Open(srvIp, srvPort, tm)
	关闭远程运行数据库	Close()
	得到变量索引	VarIndex=GetVarIndex(VarType, VarName)
	读取变量值	VarValue=GetVarValue(VarType, VarIndex)
	修改变量值	RetValue=SetVarValue(VarType, VarIndex, VarValue)
	读取某些变量值	VarValues=ReadVarValues(VarNames)
	修改某些变量值	RetCode=WriteVarValues(VarName, VarValues)
	以数组方式读取变量值	VarValues=ReadVarArray(VarArray, start, end)
	以数组方式修改变量值	RetCode=WriteVarArray(VarArray, start, end, VarValues)
	以索引方式读取变量值	VarValues=ReadVarBlock(VarType1, start, end)
	以索引方式修改变量值	RetCode=WriteVarBlock(VarType1, start, end, VarValues)
	读取某设备号字节数值	byteValues=ReadDevnoBytes(devno, start, end)
	修改某设备号字节数值	RetCode=WriteDevnoBytes(devno, start, end, byteValues)
	读取某类全部变量值	FilrerValues=ReadFilterVarValues(VarType1, VarFilter)
参数	VarType	变量类型 1=AI, 2=AO, 3=AR, 4=DI, 5=DO, 6=DR, 7=VA, 8=VD, 9=VT
	VarName	变量名称 最大长度 20 字符
	VarIndex	变量索引 范围: 0-500000
	VarValue	变量值 类型: VT_R8 VT_UI1 VT_BSTR
	VarNames	多变量名 竖线分隔, 例如: "AI. AI1 AR. AR1 VT. VT1"
	VarValues	多变量值 竖线分隔, 例如: "10 20 1. 23 abcd "
	VarArray	变量名组 %i 表示编号, 例如: VA. VA%i, AR. AR%. 2i
	VarType1	变量类型 "AI/AO/AR/DI/DO/DR/VA/VD/VT"
	VarFilter	变量过滤名 "P1_*", 最多支持 3000 变量
	FilrerValues	变量过滤值 "ok(1-3000): P1_AI1, 10  P1_AI2, 20  P1_AI3, 30 "
	Start	开始编号
	End	结束编号

## □ 例 1 (VbScript, 访问变量):

```

Set obj=CreateObject("fmDMO.TCPRundb")
If obj.Open("192.168.1.100",5002,1000)=1 Then
    n=obj.GetVarIndex(7,"%VA1")
    x=obj.GetVarValue(7,n)
    x=x+1
    obj.SetVarValue 7,n,x
    obj.Close
End If
Set obj=Nothing

```

## □ 例 2: ( VbScript, 访问文本变量):

```

Set obj=CreateObject("fmDMO.TCPRundb")
If obj.Open("192.168.1.100",5002,1000)=1 Then
    n=obj.GetVarIndex(9,"%VT1")
    s=obj.GetVarValue(9,n)
    s=s&"abcd"
    obj.SetVarValue 9,n,s
    obj.Close
End If
Set obj=Nothing

```

## □ 例 3: ( VbScript, 读取某些变量):

```

Set obj=CreateObject("fmDMO.TCPRundb")
If obj.Open("192.168.1.100",5002,1000)=1 Then
    s=obj.ReadVarValues("AR.AR1|VA.VA1|VT.VT1")
    obj.Close
End If
Set obj=Nothing

```

## □ 例 4: ( VbScript, 修改某些变量):

```

varName="":varValue=""
For i=1 To 1000
    varName = varName &"VA.VA"&CStr(i)&"|"
    varValue= varValue&CStr(i)&"|"
Next
Set obj=CreateObject("fmDMO.TCPRundb")
If obj.Open("127.0.0.1",5002,1000)=1 Then
    s=obj.WriteVarValues(varName,varValue)
    obj.Close
End If
Set obj=Nothing

```



## □ 例 5: (VBScript, 修改数组变量):

```

Set obj=CreateObject("fmDMO.TCPRundb")
If obj.Open("192.168.1.100",5002,1000)=1 Then
    s=obj.WriteVarArray("AR.AR%i",1,10,"1|2|3|4|5|6|7|8|9|10")
    obj.Close
End If
Set obj=Nothing

```

## □ 例 6: (VC++, 通过变量索引访问变量):

[1]. 函数 InitInstance() 必须添加:

```
if(!AfxOleInit()) return FALSE;
```

[2]. stdafx.h 头文件添加:

```
#import "c:\\fameview\\fmDmo.dll" no_namespace named_guids
```

[3]. 访问变量:

```

ITCPRundbPtr pRundbObj;
HRESULT hr=pRundbObj.CreateInstance("fmDMO.TCPRundb");
if(hr!=0) return;
_variant_t srvPort,tm;
srvPort.vt=VT_I4;srvPort.lVal=5002;
tm.vt=VT_UI4;tm.lVal=1000;
_variant_t vt=pRundbObj->Open(L"127.0.0.1",srvPort,tm);
if(vt.iVal!=1) return;
_variant_t varType;
varType.vt=VT_UI1;
varType.bVal=7;
_bstr_t varName="%VA1";
_variant_t varIndex=pRundbObj->GetVarIndex(varType,varName);
if(varIndex.lVal>=0){
    _variant_t varValue=pRundbObj->GetVarValue(varType,varIndex);
    varValue.dblVal+=1;
    pRundbObj->SetVarValue(varType,varIndex,varValue);
}
pRundbObj->Close();

```

## □ 例 7: (VBScript, 读取某类全部变量值):

```

Set obj=CreateObject("fmDMO.TCPRundb")
If obj.Open("192.168.1.100",5002,1000)=1 Then
    s=obj.ReadFilterVarValues("AI","P1_*") 's="ok(3):|P1_AI1,10| P1_AI1,20| P1_AI1,30|"
    obj.Close
End If
Set obj=Nothing

```

## 17.4 COM 组件访问本地组态项目

组件名称		fmDMO. Project	
组件文件		fmDMO. dll	
1	修改内部变量初始值 retValue=SetVXVarInitialValue (varName, initValue)		
	varName	内部变量, 格式:XX. YY XX 变量类型 (VA/VD/VT) YY 变量名称, 支持通配符 (%), 实现批量设置, 例如:VA. P1_V%%	
	initValue	变量初始值, VA 对应类型 R8, VD 对应类型 UI1, VT 对应类型 BSTR	
	retValue	函数返回值	0 = ok
			1 = 组态数据库打开失败
			2 = 变量长度错误 (1<4)
			3 = 变量格式错误
			4 = 变量类型错误
			5 = 变量值类型错误
			6 = 变量表打开错误
			7 = 查询变量空
	8 = 修改变量值错误		
	举例	Set obj=CreateObject ("fmDMO. Project") n1=obj.SetVXVarInitialValue ("VA. P1_V%", 800) n2=obj.SetVXVarInitialValue ("VD. VD1", 1) n3=obj.SetVXVarInitialValue ("VT. VT1", "abcdef123") Set obj=Nothing	
删除某个或某些变量 retValue=DeleteVariables (varNames)			
varNames	变量名称, 格式:XX. YYYYY XX 变量类型 (AI/AO/AR/DI/DO/DR/VA/VD/VT/DOC/CA/CMP/FB/FG/TM/FX) YY 变量名称, 支持通配符 (%) 批量格式, 例如:AI. P1_V%%		
retValue	函数返回值	0 = ok	
		1 = 组态数据库打开失败	
		2 = 变量格式错误	
		3 = 变量类型错误	
		4 = 变量表打开错误	
		5 = 查询变量空	
		6 = 删除变量错误	
举例	Set obj=CreateObject ("fmDMO. Project") n=obj.DeleteVariables ("VA. P1_V%") Set obj=Nothing		

3	修改设备数据表配置参数			
	retValue=ModifyDeviceTableParameter(driverName, itemName, oldValue, newValue)			
	driverName	驱动名称, 如“S7TCP”, “EtherNet/Logix”, “MC-QTCPIP”, “MB_TCPIP”		
	itemName	配置项名称	“SrvIP”	以太网 IP 地址
			“SrvPort”	以太网 TCP 端口
			“LocalIP”	本地 IP 地址
	oldValue	原始值, 替换修改指定原始值		
	newValue	修改值, 如: “192.168.1.10”, 8001		
	retValue	函数返回值	0 = ok	
			1 = 组态数据库打开失败	
2 = 配置表打开错误				
3 = 查询驱动名称空				
4 = 修改参数错误				
举例	Set obj=CreateObject(“fmDMO. Project”) n1=obj.ModifyDeviceTableParameter(“S7TCP”, “SrvIP”, “0.0.0.0”, “19.168.100”) n2=obj.ModifyDeviceTableParameter(“MC-QTCPIP”, “SrvPort”, 0, 1025) Set obj=Nothing			

4	导入项目文件			
	retValue=ImportFile(sourcePath, sourceFile, projPath, projFile, handleMode)			
	sourcePath	被导入文件路径		
	sourceFile	被导入文件名称		
	projPath	项目路径, 填写子路径名或“. ”当作安装目录, 空则根据文件后缀类型自动处理		
	projFile	导入后文件名称, 空则与被导入文件名称相同		
	handleMode	处理方式	Bit(0)=1 只读	
			Bit(1)=1 隐藏	
			Bit(2)=1 数据连接文件被应用	
	retValue	函数返回值	0 = ok	
			1 = 文件路径错误	
			2 = 文件名称空	
			3 = 复制文件错误	
			4 = 数据库连接文件. 应用. 打开组态数据库失败	
			5 = 数据库连接文件. 应用. 打开组态数据表失败	
6 = 数据库连接文件. 应用. 更新组态数据表失败				
举例	Set obj=CreateObject(“fmDMO. Project”) n1=obj.ImportFile(“c:\Backup\PictureFile”, “PIC1.drw”, “”, “”, 0) n2=obj.ImportFile(“c:\Backup\MyFile”, “cfgFile.txt”, “MyFile”, “”, 1) n3=obj.ImportFile(“c:\Backup\”, “Database1.mdb”, “. ”, “”, 0) Set obj=Nothing			

5

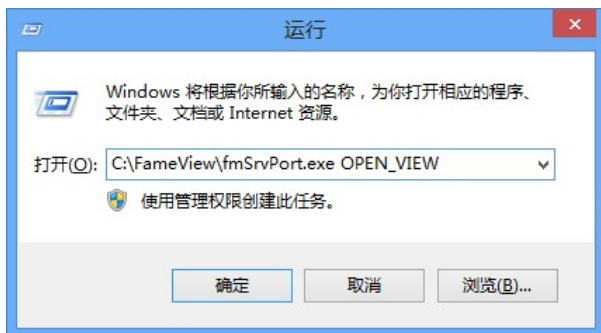
设置启动任务					
retValue=SetStartupTask(taskIndex,taskValue)					
taskIndex	任务编号	01	画面显示	101	随系统启动:屏蔽桌面
		02	变量报警	102	自启动延时
		03	实时报表	103	安静方式启动
		04	配方应用	104	系统退出备份
		05	用户管理	105	禁止休眠
		06	双机冗余	106	启动标识
		07	全局变量	107	启动口令
		08	数据库连接	108	口令错误注销
		09	批量连接	109	退出口令
		10	全局脚本	110	窗口位置
		11	网络服务器	111	附加程序
		12	串口服务器		
		13	BACNETIP 服务		
		14	ADSL 数据服务		
		15	短信数据服务		
		16	远程连接服务		
		17	OPC 服务器		
		18	变量组		
		19	CDT 转发服务		
		20	CEMS 国标服务		
		21	IEC104 转发服务		
		22	数据库转发		
		23	Web 服务器		
		24	SNMP 代理服务		
		25	实时数据库连接		
		26	佳华云		
taskValue	任务值, 如:"Yes","No","Yes:123456","Yes:300,600","appl.exe app2.exe"				
retValue	函数返回值	0=ok, 1=error			
举例	<pre>Set obj=CreateObject("fmDMO.Project") n=obj.SetStartupTask(1,"Yes") n=obj.SetStartupTask(101,"Yes:No") n=obj.SetStartupTask(102,"5") n=obj.SetStartupTask(107,"Yes:123456") n=obj.SetStartupTask(110,"Yes:300,600") n=obj.SetStartupTask(111, "appl.exe app2.exe")  Set obj=Nothing</pre>				

6	设置画面显示属性 retValue=SetDisplayProperty(startPicture, viewAttrib, closePassword, multiScr, memPicture)		
	startPicture	开始画面	"pic1.drw"
	viewAttrib	显示属性	Bit(1) = [0001] 全屏显示 Bit(2) = [0002] 允许 全屏非全屏切换 Bit(3) = [0004] 屏蔽系统菜单 Bit(4) = [0008] 最小化按钮 Bit(5) = [0010] 最大化按钮 Bit(6) = [0020] 伸缩性边框 Bit(7) = [0040] 禁止窗口移动 Bit(8) = [0080] 屏蔽 ALT+Tab 等 Bit(9) = [0100] 屏蔽 Alt+F4 等 Bit(10)= [0200] 屏蔽 Ctrl+Alt+Del Bit(11)= [0400] 屏蔽水平滚动条 Bit(12)= [0800] 屏蔽垂直滚动条 Bit(13)= [1000] 顶层显示 Bit(14)= [2000] 防止画面意外关闭
	exitPassword	关闭口令	"Yes:123456", "No"
	multiScr	多屏显示	"Yes:pic1, 0, 0, 1024, 768 pic2, 1024, 0, 1024, 768", "No"
	memPicture	内存画面	"pic1 pic2 pic3", ""
	retValue	函数返回值	0=ok, 1=error
	举例	<pre>Set obj=CreateObject("fmDM0.Project") n=obj.SetDisplayProperty("pic1",&amp;H0003,"No","No","") Set obj=Nothing</pre>	

7	执行组态命令 retValue=ShellExecute(cmdText)		
	cmdText	命令行文本, 内容参考 17.5 表	
	retValue	函数返回值	0=ok 1=error
	举例	Set obj=CreateObject("fmDMO. Project") n=obj. ShellExecute("Startup_Task") Set obj=Nothing	
8	定义用户操作级别 retValue=DefineOperateLevel(levelInfo)		
	levelInfo	级别信息: "级别20 级别21"	
	retValue	函数返回值	0 = ok 1 = 读配置文件错误 2 = 写配置文件错误
	举例	Set obj=CreateObject("fmDMO. Project") n=obj. DefineOperateLevel("级别20 级别21 级别22") Set obj=Nothing	
9	定义用户名称 retValue=DefineUserName(userInfo)		
	userInfo	用户信息: "用户 1, 口令, 拥有级别, 自动注销时间, 缺省用户 用户 2... 用户 3..."	
	retValue	函数返回值	0 = ok 1 = 读配置文件错误 2 = 写配置文件错误
	举例	Set obj=CreateObject("fmDMO. Project") n=obj. DefineUserName("user1, 1111, 111, 0, N user2, 2222, 001, 5, Y") Set obj=Nothing	

## 17.5 通过命令行方式操作组态系统

- 监控系统提供fmSrvPort.exe程序作为接口, 赋予命令参数执行相应操作;



- 支持参数:

编号	参数	操作
1	Startup_FameView	以默认方式启动监控系统
2	Startup_FameView_Hide	以隐藏方式启动监控系统
3	Startup_FameView_NoDemoWnd	以无演示窗口方式启动监控系统
4	Exit_FameView	退出监控系统
5	New_Project	项目. 新建
6	Load_Project	项目. 调入
7	Backup_Project	项目. 备份
8	Load_Project_xxxxxx	项目. 调入指定项目, xxxxxx为项目名称
9	Backup_Project_xxxxxx	项目. 备份指定项目, xxxxxx为项目名称
10	ProjectRunLimit	项目. 运行期限
11	ProjectInfo	项目. 信息
12	Startup_Task	设置. 启动任务
13	AutoLogin	设置. 设置. 自动登录
14	RelationalDatabase	设置. 关联数据库
15	DogDriver	授权. 加密狗驱动
16	TestDog	授权. 测试加密狗
17	UpgradeDog	授权. 升级加密狗
18	Install_DevDriver	通讯. 安装驱动
19	Delete_DevDriver	通讯. 删除驱动
20	Startup_DevDriver	通讯. 启动驱动
21	Edit_DevTable	通讯. 设备数据表
22	Open_DevTable	通讯. 监视设备数据表
23	OPEN_COMMDRV_XXXXXX	通讯. 监视驱动程序, XXXXXX为驱动程序名称
24	RUNDBParameter	运行数据库. 规模参数
25	RUNDBCrossReference	运行数据库. 交叉参考
26	RUNDBMonitorObject	运行数据库. 监控对象

27	RUNDBToExcel	运行数据库->Excel
28	RUNDBFromExcel	运行数据库<-Excel
29	EDITVAR_XX	编辑变量:XX=AI/AO/AR/DI/DO/DR/VA/VD/VT
30	Open_RunDB	运行数据库, 监视
31	PictureManager	画面, 文件管理器
32	DisplayProperty	画面, 显示属性
33	OPEN_DRAW	画面, 制作
34	OPEN_VIEW	画面, 显示
35	DefineVariableAlarm	报警, 定义变量报警
36	SetAlarmParam	报警, 设定报警参数
37	SetAlarmTable	报警, 定义报警表格
38	MonitorAlarm	报警, 监视报警
39	DOCProperty	历史数据库, 属性
40	DOCQueryScheme	历史数据库, 查询方案
41	DOCCurveParam	历史数据库, 缺省曲线参数
42	VariableFile	变量文件
43	DefineClass	用户管理, 定义级别
44	DefineUser	用户管理, 定义用户
45	SetNetworkServer	网络服务器, 配置
46	DefineVariablePackage	网络服务器, 变量包
47	DefineRecordPackage	网络服务器, 记录包
48	MonitorNetworkServer	网络服务器, 监视
49	CfgCOMPortServer	串口服务器, 配置
50	DefineVariablePackage	串口服务器, 变量包
51	MonitorCOMPortServer	串口服务器, 监视
52	CfgHServer	双机冗余, 配置
53	MonitorHServer	双机冗余, 监视
54	GlobalVariable	全局变量
55	ADSLDataServer	ADSL 数据服务器, 配置 监视
56	DefineVariablePackage	ADSL 数据服务器, 变量包
57	DefineRecordPackage	ADSL 数据服务器, 记录包
58	SetSMSModule	短信服务, 设置短信模块
59	EditSMSContent	短信服务, 编辑短信内容
60	SendSMSContent	短信服务, 发送短信内容
61	SendAlarmSMS	短信服务, 发送报警短信
62	MonitorSmsServer	短信服务, 监视短信服务
63	SetCDTService	CDT 转发, 设置
64	MonitorCDTService	CDT 转发, 监视



65	SetOPCServer	OPC 服务器. 设置
66	OPCClientTool	OPC 服务器. OPC 客户端工具
67	SetIEC104Service	IEC104 转发服务. 设置
68	MonitorIEC104Service	IEC104 转发服务. 监视
69	SetBACNETIPService	BACNETIP 服务. 设置
70	MonitorBACNETIPService	BACNETIP 服务. 监视
71	SetSNMPProx	SNMP 代理服务. 设置
72	DefineSNMPProxVariable	SNMP 代理服务. 定义变量
73	MonitorSNMPProx	SNMP 代理服务. 监视
74	SetRLinkService	远程连接服务. 设置
75	MonitorRLinkService	远程连接服务. 监视
76	SetWebServer	Web 服务器. 设置
77	DefineWebDBTable	Web 服务器. 定义数据库表格查询
78	DefineWebDBCure	Web 服务器. 定义数据库曲线查询
79	MonitorWebServer	Web 服务器监视
80	EditRecipe	配方. 编辑
81	RecipeNo	配方. 编号
82	TableRecipe	配方. 表格
83	GlobalScriptFile	全局脚本. 脚本文件
84	GlobalScriptScheme	全局脚本. 运行策略
85	EditDBLink	数据库连接. 编辑
86	ApplyDBLink	数据库连接. 应用
87	DBReportFile	数据库连接. 报表文件
88	EditDBBatchLink	批量数据库连接. 编辑
89	ApplyDBBatchLink	批量数据库连接. 应用
90	DBReportFile	批量数据库连接. 报表文件
91	EditREALDBLink	实时数据库连接. 编辑
92	ApplyREALDBLink	实时数据库连接. 应用
93	CfgFileREALDBLink	实时数据库连接. 输出配置文件
94	/REALDBTOOL	实时数据库连接. 客户端工具
95	Database2Database	数据库转发
96	RegisterControl	实用工具. 注册控件
97	SystemEvent	实用工具. 系统事件
98	CommunicationEvent	实用工具. 通讯事件
99	COMPortDebugger	实用工具. 串口调试器
100	TCPDebugger	实用工具. TCP 调试器
101	UDPDebugger	实用工具. UDP 调试器
102	/GETTCPSTATUS	实用工具. 网络端口状态

103	/GETROUTESTATUS	实用工具, 网络路由表
104	/GETARPTABLE	实用工具, 地址解析表
105	MODBUSSimulation	实用工具, MODBUS 仿真
106	MBRTUSimulation	实用工具, MBRTU 仿真
107	MODBUSDebugger	实用工具, MODBUS 调试器
108	FindIP	实用工具, IP 查找
109	DeviceSimulation	实用工具, 设备仿真
110	Ascii2Hex	实用工具, ASCII 转 16 进制
111	VNCClient	实用工具, VNC 远程桌面
112	/FindNetSrv	实用工具, 查找网络服务器
113	/MANAGE_SQLDB	实用工具, 管理 SQL Server 数据库
114	DTUSimulation	实用工具, DTU仿真
115	/SNMPCNTTOOL	实用工具, SNMP客户端
116	/TOADDBATCHRECORD /TXTFILENAME=	实用工具, 增补数据库
117	REPLACE_VARDESC	实用工具, 替换变量描述
118	COMPACT_DATABASE	实用工具, 压缩运行数据库
119	/REALDBTOOL	实用工具, 实时数据库客户端
120	/ATTACH_SQLDB	实用工具, 附加SQL Server数据库
121	/DETACH_SQLDB	实用工具, 分离SQL Server数据库
122	SystemCustomize	系统定制
123	/SelectSysInfo	选择系统信息
124	Open_CFGMGR	启动组态管理器
125	OPEN_SCRKEY	打开屏幕键盘
126	OPEN_DBLINK	监视数据库连接
127	OPEN_DATETIME	打开时间修改窗口
128	OPEN_ODBCCFG	打开ODBC配置窗口
129	OPEN_NETLINK	打开网络连接窗口
130	OPEN_RUNTASK	显示程序运行窗口
131	CLOSE_CTRL+ALT+DEL	屏蔽CTRL+ALT+DEL
132	OPEN_CTRL+ALT+DEL	开放CTRL+ALT+DEL
133	EXIT_WINDOWS	启动组态时关闭Windows
134	LOGOFF_WINDOWS	启动组态时注销Windows
135	REBOOT_WINDOWS	启动组态时重启Windows
136	LOCK_WINDOWS	锁定Windows
138	SHOW_DESKTOP	显示桌面
139	HIDE_DESKTOP	隐藏桌面
140	DISABLE_TASKKEY	屏蔽任务键
141	ENABLE_TASKKEY	使能任务键

142	TERMI_PROCESS_	强制终止程序
143	SCREENSAVE	进入屏幕保护
144	OPEN_网络服务器	监视网络服务器
145	OPEN_ADSL服务器	监视ADSL服务器
146	OPEN_远程连接服务器	监视远程连接服务器
147	Define_VarFile	定义变量文件
148	EDIT_WEBDBTABLE	编辑Web数据库表格查询方案
149	EDIT_WEBDBCURVE	编辑Web数据库曲线查询方案
150	OPEN_WEBSERVER	监视Web服务器
151	SET_WEBSERVER	设置Web服务器
152	/CREATE_UDL_FILE /cfgFile=xxx.txt	根据配置文件自动生产数据库连接文件

□ 例(VC++, 执行 fmSrvPort.exe 文件):

```
CString path=getenv("FameView_Path");
CString fn =path+"fmSrvPort.exe Load_Project";
::WinExec(fn, SW_SHOW);
```

英文界面显示, 需以英文方式安装组态软件, 并附加参数/langtype=1

```
CString path=getenv("FameView_Path");
CString fn = path+"fmSrvPort.exe Load_Project /langtype=1";
::WinExec(fn, SW_SHOW);
```

## 17.6 串口驱动编程模板

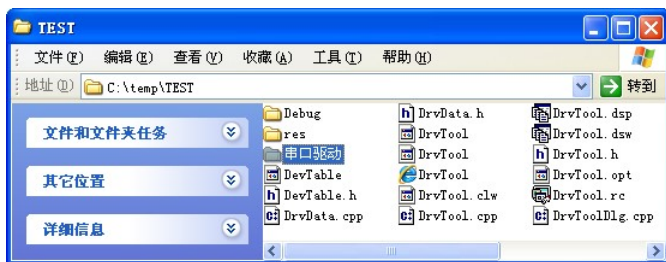
- 系统安装目录下的<附件>子目录下, 提供了<串口驱动.rar>驱动模板;
- 通过串口驱动模板, 高级用户可以自行开发串口驱动程序;
- 把驱动模板解压并拷贝到相应目录:



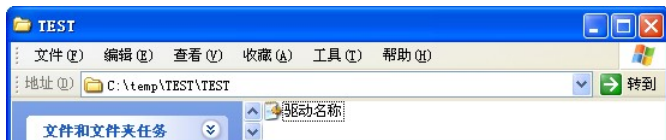
- 把模板的目录改为驱动程序的名称, 如“TEST”;



- 把其中的子目录“串口驱动”也修改为新的驱动名称“TEST”:



- 打开 TEST 目录, 其中包含 1 个“驱动名称.ini”文件:



- 把“驱动名称.ini”文件修改为“TEST.ini”:

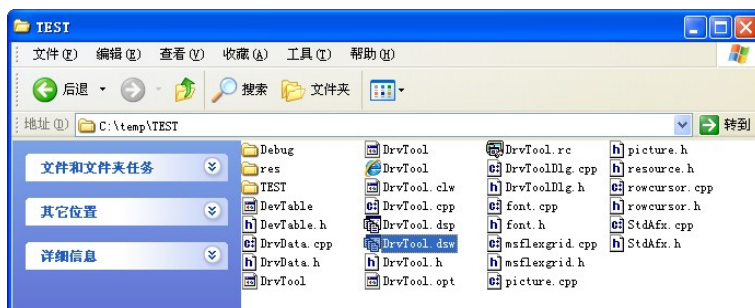


- 打开 TEST.ini 文件, 设置驱动的内容并保存:

参数	描述	举例
[设备驱动]		
DrvName	驱动程序名称	TEST
DrvDesc	驱动程序描述	测试驱动程序
DrvFileName	驱动程序对应的可执行程序文件名称	TEST.exe
[数据类型]		
共支持 24 种数据类型, 不使用的数据类型不需要定义;		
DataType1	驱动程序支持的第 1 种数据类型	AI - 模拟量输入
DataType2	驱动程序支持的第 2 种数据类型	DI - 开关量输入
...	...	...
DataType24	驱动程序支持的第 24 种数据类型	
[数据格式]		
某数据类型支持的数据格式: 1=BYTE+WORD+DWORD, 2=WORD, 4=DWORD, 5=WORD+DWORD, 6=BYTE, 0=不使用		
DataType1_fmt	第 1 种数据类型的数据格式	4
DataType2_fmt	第 2 种数据类型的数据格式	1
...	...	...
DataType24_fmt	第 24 种数据类型的数据格式	0
[访问方式]		
某数据类型支持的访问方式: 1=RW+R, 2=R, 3=W, 4=RW+R+W		
AccessMode1	第 1 种数据类型支持的访问方式	2
AccessMode2	第 2 种数据类型支持的访问方式	2
AccessMode3	第 3 种数据类型支持的访问方式	2
AccessMode4	第 4 种数据类型支持的访问方式	2
AccessMode5	第 5 种数据类型支持的访问方式	2
AccessMode6	第 6 种数据类型支持的访问方式	2
...	...	...
AccessMode24	第 24 种数据类型支持的访问方式	0

[通讯参数]			
定义设备号时, 驱动程序的参数描述			
S1Desc	第 1 个参数描述	m_PlcNo	站号
S2Desc	第 2 个参数描述, 串口驱动不使用	m_ConnectName	
S3Desc	第 3 个参数描述, 串口驱动不使用	m_VFDName	
S4Desc	第 4 个参数描述, 串口驱动不使用	m_DevName	
S5Desc	第 5 个参数描述	m_ScanClass	扫描级别:
S6Desc	第 6 个参数描述	m_DataType	数据类型:
S7Desc	第 7 个参数描述	m_AccessMode	访问方式:
S8Desc	第 8 个参数描述	m_DataFormat	数据格式:
S9Desc	第 9 个参数描述, 数据辅助参数	m_AreaNo	数据块号:
S10Desc	第 10 个参数描述	m_DataStart	开始地址:
S11Desc	第 11 个参数描述	m_DataLen	数据长度:
S12Desc	第 12 个参数描述, 串口确省参数		COM:009600, 8, 1, N
各参数对应的确省值			
S1Value	第 1 个参数确省值		1
S2Value	第 2 个参数确省值		
S3Value	第 3 个参数确省值		
S4Value	第 4 个参数确省值		
S5Value	第 5 个参数确省值		
S6Value	第 6 个参数确省值		
S7Value	第 7 个参数确省值		
S8Value	第 8 个参数确省值		
S9Value	第 9 个参数确省值		
S10Value	第 10 个参数确省值		0
S11Value	第 11 个参数确省值		10

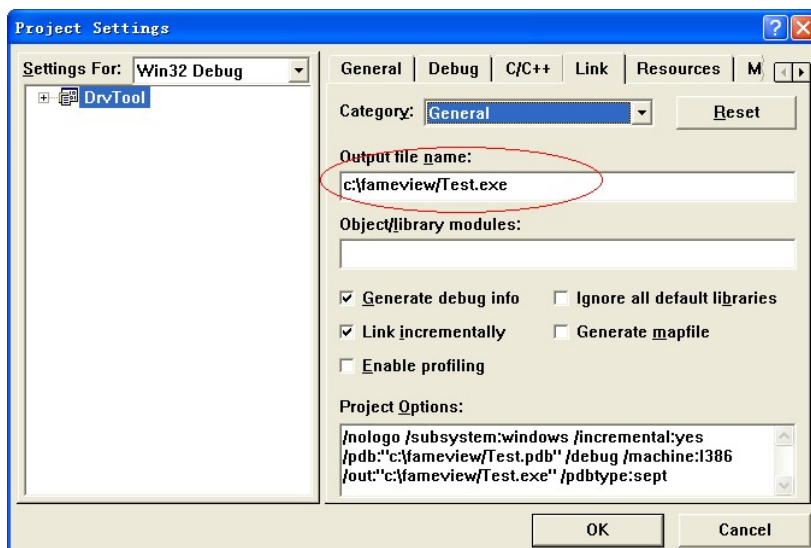
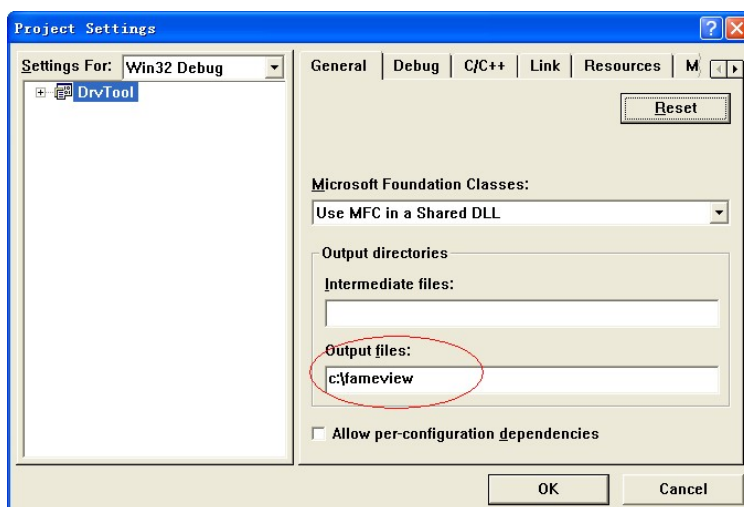
- 用 VC++ V6.0 打开驱动模板项目 (DrvTool.dsw):



- 打开 DrvTool.h 文件, 在代码中设置驱动程序的名称“TEST”:

```
#define DRV_NAME "TEST"
```

- 执行菜单 Project 下的 Settings 命令, 设置驱动程序编译后的路径和文件名称:

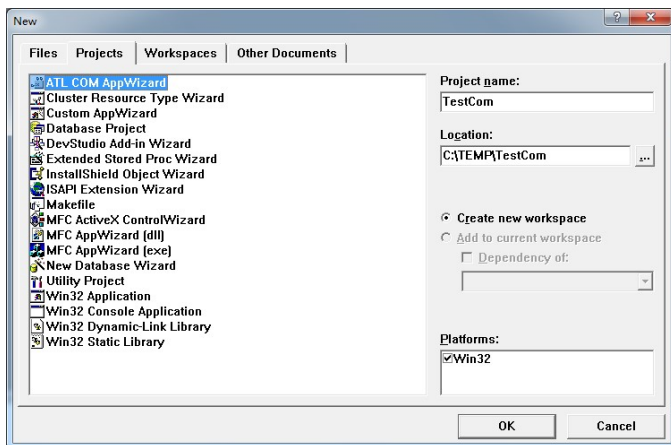


- 打开文件 DrvToolDlg.cpp 文件, 进行代码编写;
- 大多数代码不需要修改, 主要修改[7. 8. 6]和[7. 9. 5]部分, 分别进行读取和写入操作;
- 把 TEST.ini 文件和最后生成的 exe 文件, 拷贝到组态软件目录下, 进行安装并调试运行;

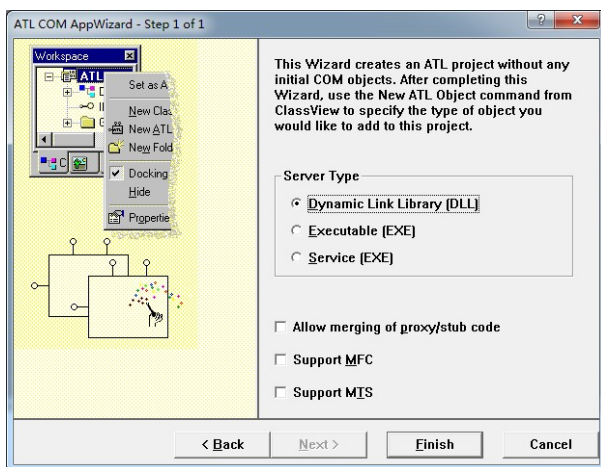
## 17.7 编写脚本对象

- 使用VC、VB、C#等编程语言, 编写支持双接口COM组件, 被组态软件VBScript脚本调用;
- 以VC++为例, 制作双接口的COM组件, 名称: TestCOM. TestObj

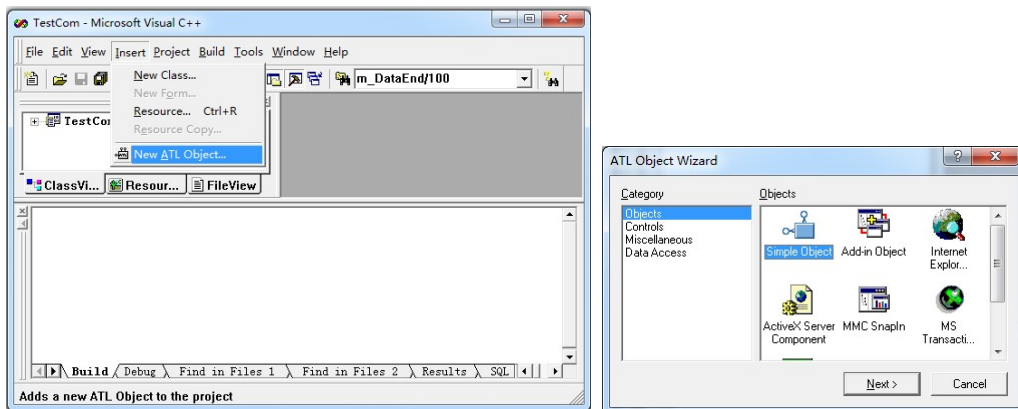
[1]. 建立COM组件项目:



[3]. 编译为动态连接库:



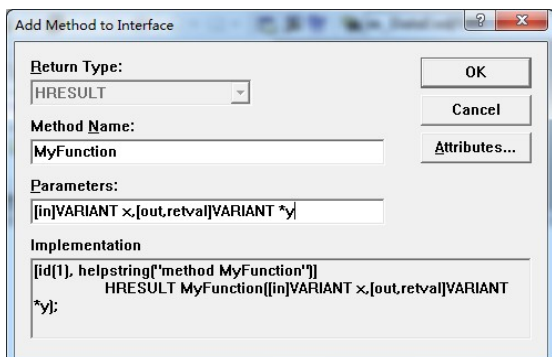
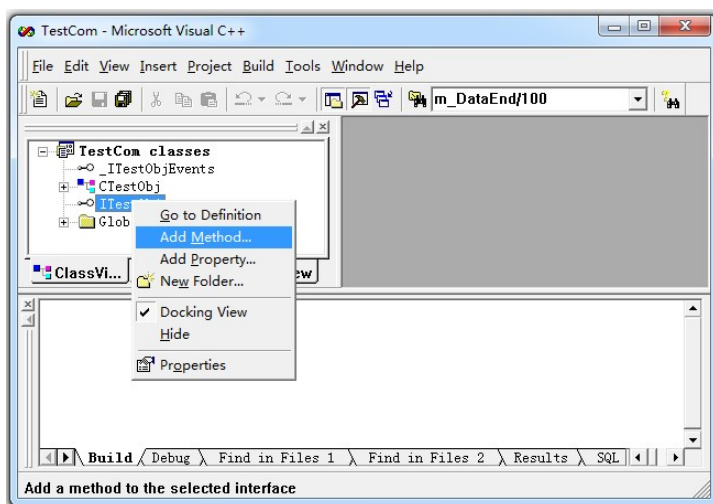
[4]. 添加COM对象, 对象名称: TestObj

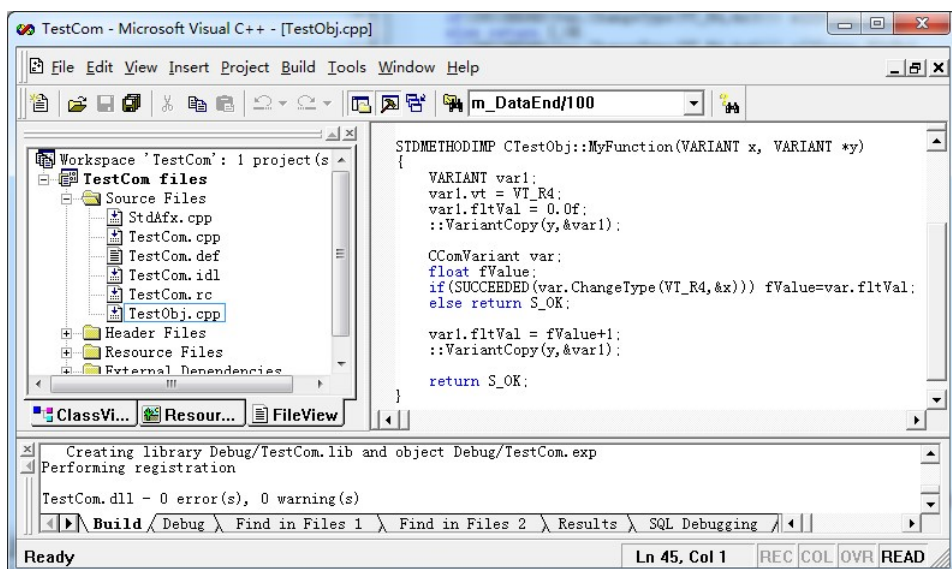






[5]. 生成对象添加接口函数, 如 $y = \text{MyFunction}(x)$ :





- [6]. 组件需要注册, 建议把编译生成TestCom.dll文件拷贝到Setup.exe所在安装目录下,  
Setup.ini文件填写文件名称, 安装时自动注册:



- [7]. 画面或全局脚本中使用COM组件:

```
x=RunSys.GetVarValue(VA,"%VA1",-1)
Set testObj = CreateObject("TestCom.TestObj")
y=testObj.MyFunction(x)
RunSys.SetVarValue VA,"%VA2",-1,y
Set testObj=Nothing
```

## 17.8 苹果移动设备 (iOS) 连接网络服务器

使用苹果移动设备 iPhone/iPad 连接网络服务器：



Xcode 5.0 程序例子代码, 并参考Appendix目录下的演示项目 iOSReadNetSrv.rar:

```
// ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
{
    CFSocketRef cntSocket;
    float      fValue1;
    float      fValue2;
    BOOL       readVarStatus;
}
@end

// ViewController.m
#import "ViewController.h"
#import <sys/socket.h>
#import <netinet/in.h>
#import <arpa/inet.h>
#import <unistd.h>

@interface ViewController ()
@property (weak, nonatomic) IBOutlet UITextField *SrvIp;
@property (weak, nonatomic) IBOutlet UITextField *SrvPort;
@property (weak, nonatomic) IBOutlet UITextField *VarValue1;
@property (weak, nonatomic) IBOutlet UITextField *VarValue2;
- (IBAction)ConnectNetServer:(id)sender;
```

```

- (IBAction)StopConnectNetServer:(id)sender;
@property (weak, nonatomic) IBOutlet UIButton *StartConnectButton;
@property (weak, nonatomic) IBOutlet UIButton *StopConnectButton;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.SrvIp.text=@"192.168.2.101";
    self.SrvPort.text=@"5002";
    cntSocket=NULL;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

//执行连接按钮
- (IBAction)ConnectNetServer:(id)sender
{
    CFSocketContext socketContext={0, (__bridge void *) (self), NULL, NULL, NULL};
    cntSocket=CFSocketCreate(kCFAllocatorDefault, PF_INET, SOCK_STREAM, IPPROTO_TCP,
                            kCFSocketConnectCallBack, ClientConnectCallBack, &socketContext);
    if(cntSocket==NULL) return;

    struct sockaddr_in addr;
    memset(&addr, 0, sizeof(addr));
    addr.sin_len = sizeof(addr);
    addr.sin_family = AF_INET;
    addr.sin_port = htons(self.SrvPort.text.intValue);
    addr.sin_addr.s_addr = inet_addr([self.SrvIp.text UTF8String]);
    CFDataRef address=CFDataCreate(kCFAllocatorDefault, (UInt8 *)&addr, sizeof(addr));
    CFSocketConnectToAddress(cntSocket, address, -1);
    CFRunLoopSourceRef sourceRef=CFSocketCreateRunLoopSource(kCFAllocatorDefault, cntSocket, 0);
    CFRunLoopAddSource(CFRunLoopGetCurrent(), sourceRef, kCFRunLoopCommonModes);
    CFRelease(sourceRef);
    self.StartConnectButton.enabled=FALSE;
}

```

```

//停止服务器连接
- (IBAction)StopConnectNetServer:(id)sender
{
    readVarStatus=FALSE;
    self.StartConnectButton.enabled=TRUE;
    self.StopConnectButton.enabled=FALSE;
}

//连接服务器回调函数
static void ClientConnectCallBack(CFSocketRef socket, CFSocketCallBackType type,
                                   CFDataRef address, const void *data, void *info)
{
    ViewController *myView=(__bridge ViewController *)info;
    if(myView==NULL) return;
    if(data==NULL) {
        myView.StopConnectButton.enabled=TRUE;
        [myView StartupThread];
    }else{
        myView.StartConnectButton.enabled=TRUE;
    }
}

//启动通讯线程
- (void)StartupThread
{
    readVarStatus=TRUE;
    NSThread *readThread=[[NSThread alloc] initWithTarget:self
                                                         selector:@selector(ReadVarPackThread) object:self];
    [readThread start];
}

//通讯线程, 读取变量包中的两个变量
- (void)ReadVarPackThread
{
    UInt8 Buffer[100];
    for(;;) {
        if(readVarStatus==FALSE) break;
        if(cntSocket==NULL) break;
        Buffer[0]=0x3E;Buffer[1]=0x2A;
        Buffer[2]=0x07;Buffer[3]=0xD1;
        Buffer[4]=0x00;Buffer[5]=0x00;
        Buffer[6]=0x00;Buffer[7]=0x02;
    }
}

```

```

        if (send(CFSocketGetNative(cntSocket), Buffer, 8, 0) <= 0) {
            [self performSelectorOnMainThread:@selector(ReadVarPackError) withObject:nil waitUntilDone:YES];
            Break;
        }

        if (recv(CFSocketGetNative(cntSocket), Buffer, 14, 0) <= 0) {
            [self performSelectorOnMainThread:@selector(ReadVarPackError) withObject:nil waitUntilDone:YES];
            Break;
        }

        memcpy(&fValue1, &Buffer[6], 4);
        memcpy(&fValue2, &Buffer[10], 4);

        [self performSelectorOnMainThread:@selector(DisplayVarValue) withObject:nil waitUntilDone:YES];
        sleep(1);
    }

    if (cntSocket != NULL) {
        CFSocketInvalidate(cntSocket);
        CFRelease(cntSocket);
        cntSocket = NULL;
    }
}

//显示变量值到屏幕
- (void)DisplayVarValue
{
    self.VarValue1.text = [NSString stringWithFormat:@"%%.2f", fValue1];
    self.VarValue2.text = [NSString stringWithFormat:@"%%.2f", fValue2];
}

//通讯发生错误
- (void)ReadVarPackError
{
    self.StartConnectButton.enabled = TRUE;
    self.StopConnectButton.enabled = FALSE;
}

@end

```