Contents

1. Copyright 2
2. General description 2
3. Kit 3.0 modifying explanation 2
4. How to use the toolkit 2
5. Project convert tool 7
6. Driver information structures 8
7. Interface functions 15
    7.1 Interface description..... 15
    7.2 How to call interface functions..... 21
        7.2.1. Development environment 21
        7.2.2. Running environment 25
8 Program Structure 27
    8.1 The project class (CPro)..... 28
    8.2 The device class (CDev)..... 29
    8.3 The communication class (CCom)..... 37
        8.3.1 CSerialCom 37
    8.4 The debug message class (CDebug)..... 39
    We provide the following message flags now:..... 40
    8.5 The helper function class (CHelper)..... 43
    8.6 The data check functions class (CCheck)..... 52
9. Error status codes 55
10.How to use “devlst.dat” file 59
11.The last submit files 61

1. Copyright

Beijing Asiacontrol technology development CO.,LTD (Asiacontrol for short) is the copyright holder of the KingView Driver development Toolkit. No part of this kit shall be reproduced, or transmitted by any means without the prior written permission of AsiaControl Company. Organizations, groups or individual who have bought the kit are only allowed using it and not offering third party.

2. General description

The kit uses Microsoft standard COM component technology. So you can create independent objects having own variables when building the interface.

When develop finished, we get just a DLL file. All functions in it are called by the two KingView applications - TouchExplorer.exe and TouchVew.exe. If the two applications don't start, all functions will never be executed.

KingView driver is the bridge of KingView and devices. The kit is used to develop KingView driver. As long as having the communication protocol of devices, you may develop KingView driver by the kit.

When developing, Only allowed to use Visual Studio C++ .

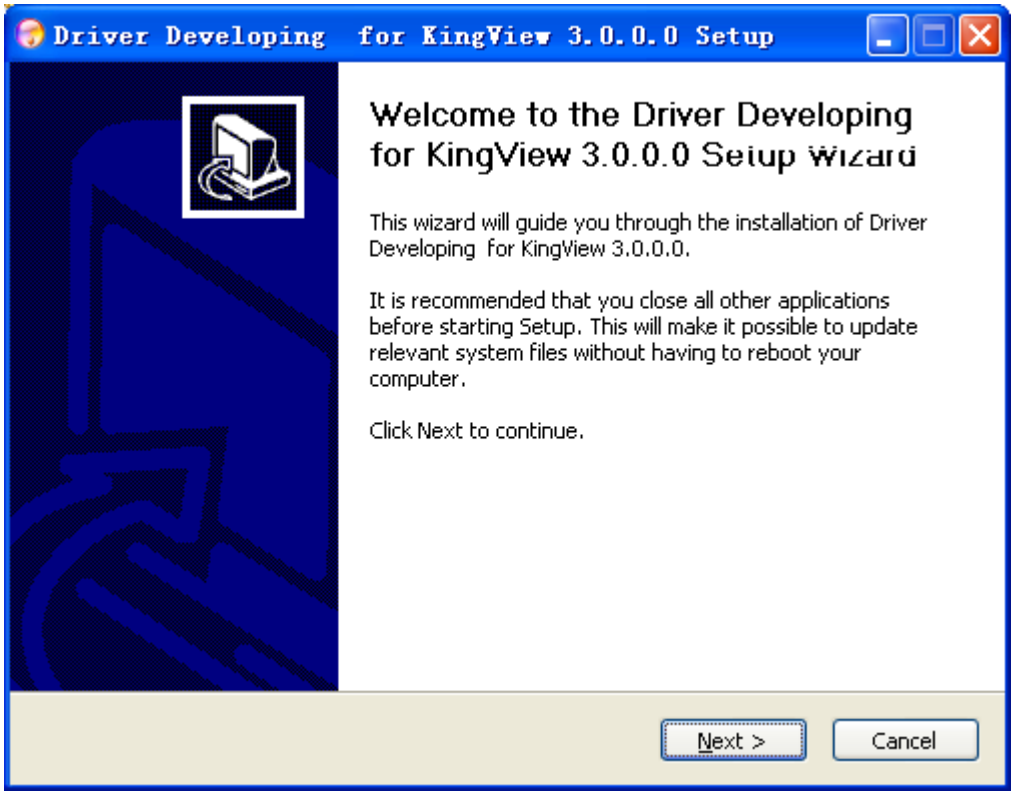
3. Kit 3.0 modifying explanation

a) General and embedded versions are united to a set of codes for synchronous update.
b) Generate the device class to provide the solution of drivers of different types devices.
c) Detailed, strong and stable code of basic class.
d) Perfect function and Wizard.
e) Standard help functions.
f) Standard code formant and uniform English versions.
g) Modify some realize methods.

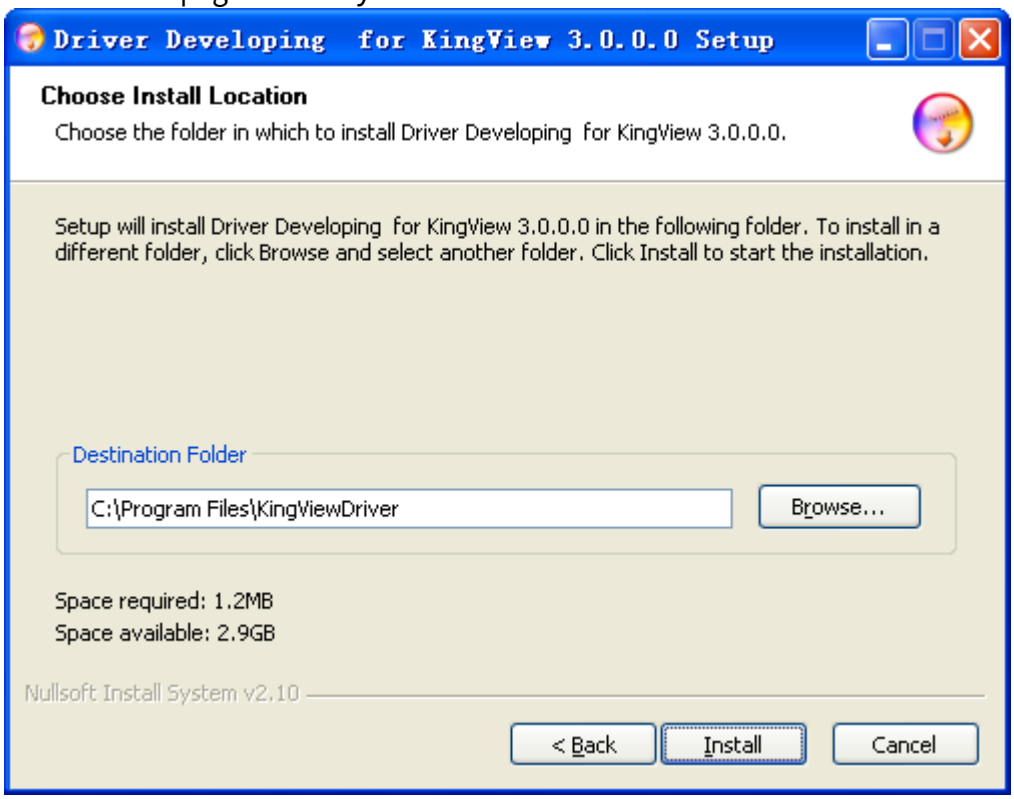
4. How to use the toolkit

We provide a wizard tool "DriverToolKit3.0.0.0.exe" in which you can create a driver framework rapidly.

When you first using the kit wizard, you should install the tool as following,



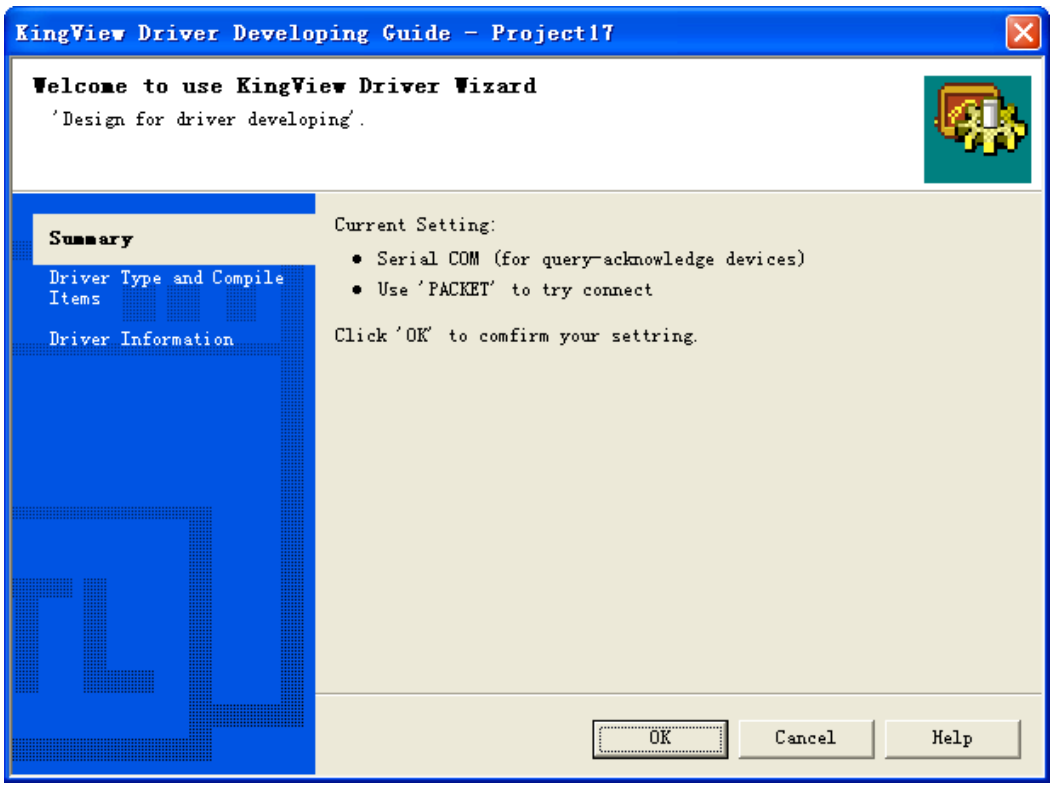
the second page allow you to choose the folder in which to install,



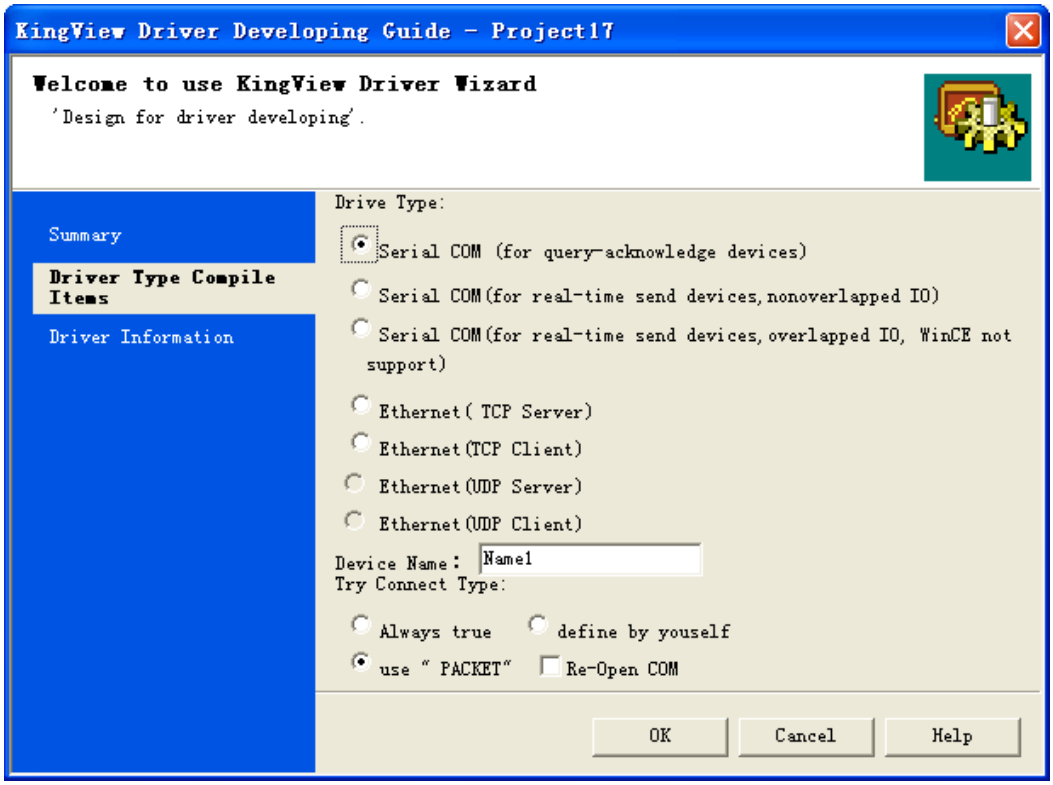
Click “Install” you will Successfully finished the install.

The following steps will help you create a driver project using the kit wizard.

(1) Start Visual Studio.NET 2003
(2) Create a new project. Select VisualC++ project and “KingView Wizard” then input a valid project name, click “OK”.
(3) The first page of the Wizard is “summary” containing some default items. If you don’t want to change it, click “OK” to create project directly.



(4) The second page is “Driver Type and Compile Items” which has the following:



Driver type:

- **Serial COM (for query-acknowledge devices)**
  It is query-type driver for general devices

- **Serial COM(for real-time send devices,nonoverlapped IO)**
  It is special driver for real-time send devices. The driver creates a thread for monitoring serial port event, that is non-overlapped I/O.

- **Serial COM(for real-time send devices,overlapped IO, WinCE not support)**
  It is also special for real-time send devices. Different from the above type, it uses overlapped I/O. Don't recommend it because embedded version doesn't support overlapped I/O and there is little difference between overlapped IO and non-overlapped IO in drivers.

- **Ethernet(TCP Server)**
  It is Ethernet server driver realized via query-acknowledge type. When needing apply to real-time send devices, you can directly use functions in communication class to monitor network events.

- **Ethernet(TCP Client)**
  It is Ethernet client driver realized via query-acknowledge type. When needing apply to auto-send devices, you can directly use functions in communication class to monitor network events.

- **Ethernet(UDP Server)**
  It is not realized now. Please refer to TCP.

- **Ethernet(UDP Client)**
  It is not realized now. Please refer to TCP.

Device name:

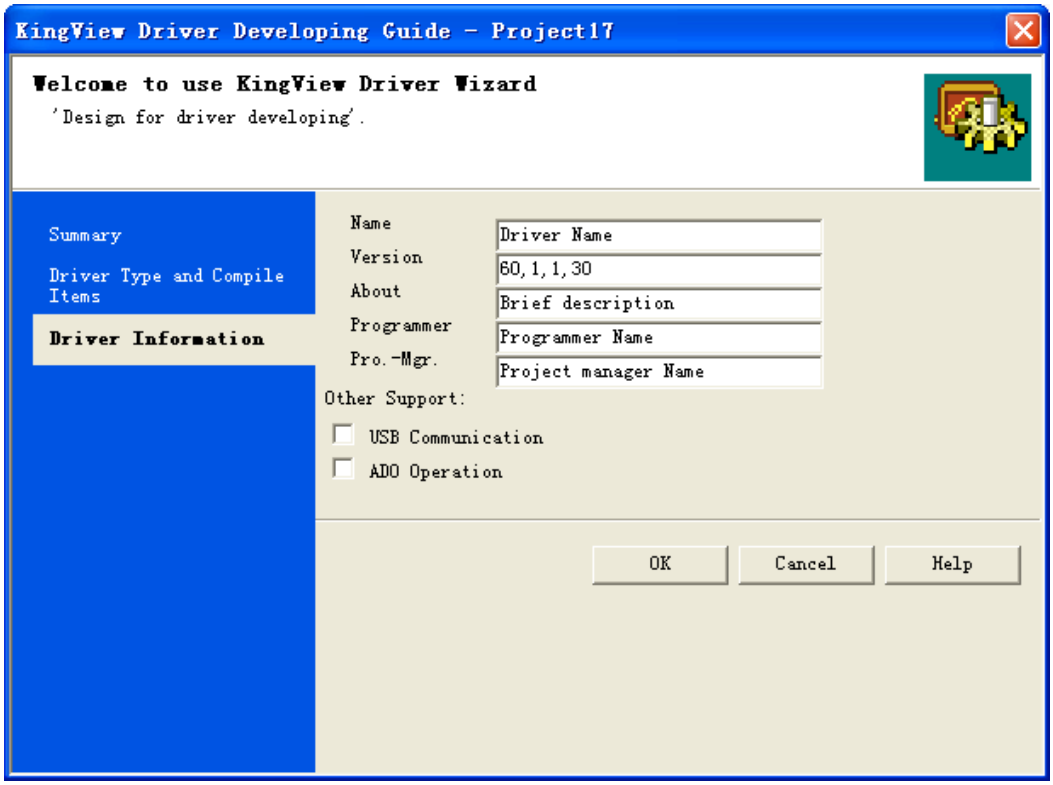
The default is "Name1". You can change it, but must keep it the same as in devlst.dat.

Try to connect type:

When communication failed, the driver tries one of the following ways to connect again:

- Always true, usually for communication card
- Define by yourself, usually for net communication or for real-time send devices
- use " PACKET", default option, for common drivers

(5) The third page is "Driver Information", which contains the following:



Driver Basic Information

- Name
- Version
- About
- Programmer
- Pro.-Mgr.

They will be written automatically to the History.txt file under the Submit folder and the readme file of the project.

Other Support:

- USB Communication
- ADO Operation

(6) Click "ok" to build your project successfully.

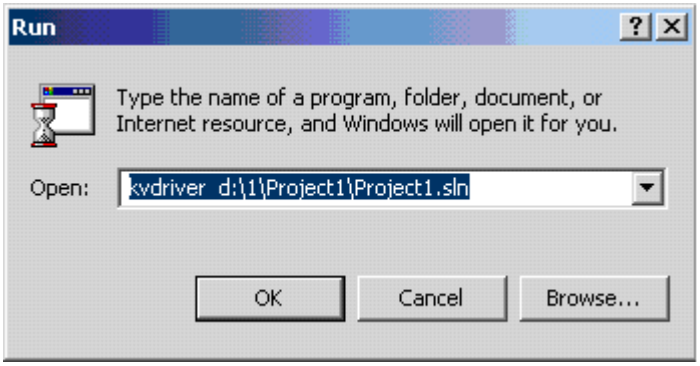
5. Project convert tool

As developing driver for KingView, we also need compile embedded driver for embedded KingView. The project convert tool can convert your Visual Studio 2003 project to Visual Studio 98 Project and embedded Visual C++ Project. So your can edit and compile your program in Visual Studio 2003, Visual Studio 98 or embedded Visual C++ environment freely.

The tool is saved in the C:\Program Files\KingViewDriver directory after install the kit.

You can save the tool to the C:\WINDOWS\system32 directory before using it.

When using the tool, you type "kvdriver" in the "Run" dialog then drag a Visual Studio 2003 project file to the dialog as below.



6. Driver information structures

All the Structures defined in "DataType.h"

_DevAddr

This structure defines the device address format.

```
Typedef struct _DevAddr
{
      UINT  nDevAddr;
      TCHAR sDevAddr[MAX_DEV_LEN];
}DEVADDR, * PDEVADDR;
```

Members:

sDevAddr:

The raw address string which user input in the KingView dialog.

nDevAddr:

The disposed address No.

ComDevice

This structure defines the control setting for a serial communications device.

```
struct ComDevice
```

```
{
      short nFlag;
      WORD  deviceKind;
      TCHAR  deviceName[32];
      int   baudRate;
      short dataBits;
      short stopBits;
      short parity;
      short timeOut;
      short failTimes;
      short retryInterval;
      short isRs232;
};
```

Members :

nFlag :

Serial port flag, not used now.

DeviceKind :

Device kind, '1' representative serial com device

DeviceName :

Logical name ={COM1, COM2...}

BaudRate :

Specifies the baud rate at which the communication device operates, actual baud rate value allowed from 110 to 256000.

DataBits :

Specifies the number of bits in the bytes transmitted and received

StopBits :

Specifies the number of stop bits to be used. It is one of the following values.

Value	Description
ONESTOPBIT	1 stop bit
ONE5STOPBITS	1.5 stop bits
TWOSTOPBITS	2 stop bits

Parity :

Specifies the parity scheme to be used. It is one of the following values.

Value	Description
EVENPARITY	Even

MARKPARITY	Mark
NOPARITY	No parity
ODDPARITY	Odd
SPACEPARITY	Space

TimeOut:
Specifies the communication timeout values, in ms.

FailTimes:
Specifies the fail Times, it is not used now.

RetryInterval:
Specifies the try connect interval in the serial port communication.

isRs232:
Specifies the communication mode, '1' representative RS232, '2' representative RS422, '3' representative RS485.

PLCVAR
This structure defines the user's variable attributes.

```
Typedef struct var
{
      WORD wVarID;
      WORD wVarType;
      TCHAR szVarName[32];
      short  nDeviceIndex;
      short  nUnitNo;
      DEVADDR *pDevAddr;
      TCHAR   *pszRegName;
      short  nRegType;
      short  nSubType;
      short  nSubType1;
      short  nSubType2;
      short  nNo;
      short  nDataType;
      short  nAccessMode;
      long   nFrequency;
      long  nTimerCount;
      CWinThread* pComThread;
      DDEValue   maxRaw;
      DDEValue   minRaw;
      short      bConvertion;
      short      isBad;
      short      isUnvalid;
```

```
}PLCVAR, * PPLCVAR;
```

Members:

wvarID:
Specifies the variable ID number.

wVar*Type*:
Specifies the variable type, It is one of the following types
Bit, byte, int, uint, BCD, long, float, longbcd, string

szVarName:
The variable name

nDeviceIndex:
The devices index No.

nUnitNo:
The communication devices address No.

pDevAddr:
Pointer to device address structure.

pszRegName:
Register name

nReg*Type*;
Register Type

nSubType:
nSubType1:
nSubType2:
Spare variables.

nNo:
The variable channels No.

nDataType:
The data *Type.*

nAccessMode:
I/O Mode.

nFrequency:

Sampling frequency.

nTimerCount:
The timer count number.

pComThread:
The serial port communication thread.

maxRaw:
The maximum raw value.

minRaw:
The minimum raw value.

PACKET
This structure defines communication packet's attributes

```
Typedef struct Packet
{
      short  nDeviceIndex;
      short  nUnitNo;
      DEVADDR * pDevAddr;
      TCHAR    * pszDevName;
      TCHAR   *pszRegName;
      short  nRegType;
      short  nSubType;
      short  nSubType1;
      short  nSubType2;
      int    nStartNo;
      int    nEndNo;
      short  nPacketType;
      CPtrList varList;
}PACKET,* PPACKET;
```

Members:
nDeviceIndex:
Devices index No.

nUnitNo:
The devices unit No.

pDevAddr:
Pointer to devices structure.

pszDevName:

Device name

nRegType:
Register Type

nSubType:
nSubType1:
nSubType2:
Spare variables

nStartNo:
Beginning channel offset

nEndNo:
End channel offset

nPacketType:
Specifies the packet type, It is one of the following types
Read or write

VarList:
Store 'struct id_no'or 'struct ID_NO2' variables.

ID_NO
This structure defines variable id No. attributes for the old interface.

```
Typedef struct IdNo
{
      WORD wVarId;
      WORD wNo;
      WORD wData Type;
      BYTE Special[4];
      PlcValue  plcValue;
}ID_NO;
```

Members:
wVarId:
Variable ID number.

wNo:
Register channel offset

wDataType:
Variable data *type.*

Special:
Spare use Variable

PlcValue:
Pointer to the PLCVAR structure.

ID_NO2
This structure defines variable id No. attributes for the new interface.

```
Typedef struct IdNo2
{
      WORD wVarId;
      WORD wNo;
      WORD wData Type;
      PlcValue  plcValue;
      BYTE Special[4];
      WORD wQualities;
      FILETIME ftTimeStamps;

}ID_NO2;
```

Members:
wVarId:
Variable ID

wNo:
Register channel offset

wDataType:
Variable data Type

Special:
Spare use variable.

plcValue:
Pointer to the PLCVAR structure

wQualities:
Quality stamps variable, 0xc0 representative GOOD, 0 representative BAD.

ftTimeStamps:
Time stamps variable, to set the current time.

DEVICE_INFO

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/018135134006006104