WDT-01

Windows Software User Manual

Warranty

All products manufactured by Acquire Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

Acquire Inc. assume no liability for damages consequent to the use of this product. Acquire Inc. reserves the right to change this manual at any time without notice. The information furnished by Acquire Inc. is believed to be accurate and reliable. However, no responsibility is assumed by Acquire Inc. for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software **on a single machine.** The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Table of Contents

I. Introduction

The DOS software driver of WDT-01 is given in “WDT-01 User Manual”. The function & control details of WDT-01 are also given in that manual. The user should refer to that manual for details first.

This software & manual is designed for **general purpose DIO&RS232 Windows applications.** Therefore the user should refer to “WDT-01 User Manual” first, then convert these DOS program into Windows program. This software can be used in Windows 95 & Windows NT.

The USRT.H, UART.LIB & UART.DLL are designed for RS-232 applications.
The DIO.H, DIO.LIB & DIO.DLL are designed for DIO applications.

A. Disk Contents

It is recommended to install this software driver to your hard disk and backup the companion floppy disk. The contents of the companion floppy disk are given as following:

\W95\VB*.*	symbol 224 \f "Wingdings" \s 12→“ VB demo
\W95\Delphi*.*	symbol 224 \f "Wingdings" \s 12→“ Delphi demo
\W95\Vbdemo*.*	symbol 224 \f "Wingdings" \s 12→“ for all DIO cards
\W95\DIO*.*	symbol 224 \f "Wingdings" \s 12→“ for all DIO cards
\W95\RS232*.*	symbol 224 \f "Wingdings" \s 12→“ for all RS-232 application
\WNT\SYS*.*	symbol 224 \f "Wingdings" \s 12→“ for driver registration
\WNT\Vbdemo*.*	symbol 224 \f "Wingdings" \s 12→“ for all DIO cards
\WNT\DIO*.*	symbol 224 \f "Wingdings" \s 12→“ for all DIO cards
\WNT\RS232*.*	symbol 224 \f "Wingdings" \s 12→“ for all RS-232 application

Refer to chapter 2,3,4 for Win 95 applications

Refer to chapter 5,6,7,8 for Win NT applications

I. Windows 95 Applications

A. C Call DLLs

All the demo program given in \W95 are designed with C language. **It is testing OK under Windows 95 and Visual C++ 4.0 compiler**. The key points are given as following:

1. Enter the DOS command prompt under Windows.
2. Make sure the PATH include the Visual C++ compiler
3. Execute the \MSDEV\BIN\VCVARS32.BAT one time to setup the environment. The VCVARS32.BAT is provided by Visual C++.
4. The source program must include "****.H"
5. Copy the ****.LIB, import library, to the same directory with source program
6. Copy the ****.DLL, to the same directory with source program
7. Edit the source program (refer to \W95???\???.C)
8. Edit the NMAKE file (refer to \W95\???\???.MAK)
9. Edit the BATCH file (refer to \W95\???\???.BAT)
10. Execute the batch file (???.BAT)
11. Execute the execution file (???.exe)

NOTE : The **???.lib is used in linking time** and the **???.DLL is used in execution time.**

A. MFC Call DLLs

The usage of UART.DLL for MFC user is very similar to that for C user. **It is testing OK under Windows 95 and Visual C++ 4.0**. The key points are given as following:

1. Use MFC wizard to create source code
2. The source program must include "****.H"
3. Copy the ****.LIB, import library, to the same directory with source program
4. Copy the ****.DLL, to the same directory with source program
5. Select **Build/Settings/Link** and key "UART.LIB" in the **object/library modules** field

A. BC++ Call DLLs

The DLLs are created by Visual C++ 4.0. The ****.H and ****.lib are also designed for Visual C/C++. The BC++ can not use this two file. The modification part is given below:

```
#include <conio.h>
#include <windows.h>
HINSTANCE hDLLLib;
// Step 1 : declare a functionpointer
float CALLBACK (*FLOAT_SUB_2)(flaot fA, float fB);
main()
{
// Step 2 : load dll
hDllLib=LoadLibrary("****.dll");
if (hDLLLib)
    {
// Step 3 : get the function address
    FLOAT_SUB_2=(FARPROC)GetProcAddress(hDLLLib,"FLOAT_SUB_2");
    if (FLOAT_SUB_2)
        {
// Step 4 : call function
        printf("1.2-3.4=%f",FLOAT_SUB_2(1.2,3.4));
        }
    else printf("get FLOAT_SUB_2 function address error");
// Step 5 : free library
    FreeLibrary(hDLLib);
    }
else printf("load ****.dll error");
getch();
}
```

This usage can be divided into **5 steps** listing above. **Using this modification**

and **.DLL, the user can use BC++ to call DLLs..**

A. VB Call DLLs

\W95\VB\UART.DLL symbol 224 \f "Wingdings" \s 12→“ DLLs

\W95\VB\FROM1.FRM symbol 224 \f "Wingdings" \s 12→“ form and source file

\W95\VB\MODULE1.BAS symbol 224 \f "Wingdings" \s 12→“ declare file

\W95\VB\PROJECT1.VBP symbol 224 \f "Wingdings" \s 12→“ project file

NOTE : 1. Testing under **Windows 95 and VB 4.0 (32 bits)**

2. The MODULE1.BAS is designed for demo purpose and the MODULE1.BAS now only support “SHORT_SUB_2(A,B)”. The user can modify this file to support all DLLs.

Module1.BAS

```
Attribute VB_Name = "Module1"
Declare Function SHORT_SUB_2 Lib "a:\w95\vb\uart.dll" (ByVal a As Integer, ByVal b As Integer) As Integer
```

FORM1.FRM (partial)

```
Private Sub Command1_Click()
    Dim a As Integer, b As Integer, c As Integer
    a = Val(Text1.Text)
    b = Val(Text2.Text)
    c = SHORT_SUB_2(a, b)
    Text3.Text = c
```

A. Delphi Call DLLs

\W95\DELPHI\UART.PAS symbol 224 \f "Wingdings" \s 12→“ unit file
\W95\DELPHI\UART.DLL symbol 224 \f "Wingdings" \s 12→“ DLLs

\W95\DELPHI\UNIT1.PAS symbol 224 \f "Wingdings" \s 12→“ demo source file
\W95\DELPHI\UNIT1.DFM symbol 224 \f "Wingdings" \s 12→“ form file
\W95\DELPHI\PROJECT1.DPR symbol 224 \f "Wingdings" \s 12→“ project file

NOTE : 1. testing under **Windows 95 and Delphi 2.0 (32 bits)**
2. The UART.PAS is designed for demo purpose and the UART.PAS now only support “SHORT_SUB_2(A,B)”. The user can modify this file to support all DLLs.

```
unit UART;
interface
function SHORT_SUB_2(a: smallint; b: smallint): smallint;  StdCall;
implementation
function SHORT_SUB_2; external 'UART.DLL' name 'SHORT_SUB_2';
end.
```

```
procedure TForm1.Button1Click(Sender: TObject) ;
var
  a,b,c   : smallint;
begin
  a := StrToInt(Edit1.text);
```

```
  b := StrToInt(Edit2.text);
  c := SHORT_SUB_2(a,b);
  Edit3.text := IntToStr(c);
end;
end.
```

A. Demo Program

We use a common demo program for all ICP DAS C-software demo program. This demo program will accept the **I/O Base Address** and **DMA Channel Number** and **IRQ Channel Number** and call the different DLLs for usage demonstration.

Note 1: Some demo program does not use any of these three value

Note 2: Some demo program use these values in different name, format or functions.

Note 3: The demo program given in this section is used as reference. The user should refer to ????.C given in floppy disk for details.

```
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include "****.h"

void READ_CMD(char *);
short       ASCII_TO_HEX(char);
void TEST_CMD(HWND, int, int, int, int);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

short       nDMA=-1, nIRQ=-1;
WORD    wBase=0x220;
int    iLine;

/* ----------------------------------------------------------------- */

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                PSTR szCmdLine, int iCmdShow)
    {
    static char szAppName[] = "TestA822";
    HWND       hwnd ;
    MSG    msg ;
```

```
WNDCLASSEX  wndclass ;
```

```
wndclass.cbSize     = sizeof(wndclass);
wndclass.style      = CS_HREDRAW|CS_VREDRAW;
wndclass.lpfnWndProc   = WndProc;
wndclass.cbClsExtra    = 0;
wndclass.cbWndExtra    = 0;
wndclass.hInstance     = hInstance;
wndclass.hIcon      = LoadIcon(NULL, IDI_APPLICATION);
wndclass.hCursor          = LoadCursor(NULL, IDC_ARROW);
wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
wndclass.lpszMenuName  = NULL;
wndclass.lpszClassName = szAppName;
wndclass.hIconSm        = LoadIcon(NULL, IDI_APPLICATION);

RegisterClassEx(&wndclass) ;
hwnd=CreateWindow(szAppName,"TESTA822 = [BASE:3] [DMA:1] [IRQ:2]",
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT,
                CW_USEDEFAULT, CW_USEDEFAULT,
                NULL, NULL, hInstance, NULL) ;
ShowWindow(hwnd,iCmdShow);
UpdateWindow(hwnd);

while (GetMessage(&msg, NULL, 0, 0))
    {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
    }
return msg.wParam;
}
```

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg, WPARAM wParam,
LPARAM lParam)
{
static int  cxChar, cyChar, cxClient, cyClient, cxBuffer, cyBuffer,
          xCaret, yCaret;
static char cBuf[80];
HDC         hdc;
TEXTMETRIC  tm;
PAINTSTRUCT ps;
int       i;

switch (iMsg)
  {
  case WM_CREATE :                // window initial
      hdc=GetDC(hwnd);
      SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));
      GetTextMetrics(hdc, &tm);
      cxChar=tm.tmAveCharWidth;
      cyChar=tm.tmHeight;
      ReleaseDC(hwnd, hdc);
      return 0;
  case WM_SIZE :
      cxClient=LOWORD(lParam);        // window size in pixels
      cyClient=HIWORD(lParam);
      cxBuffer=max(1,cxClient/cxChar); // window size in characters
      cyBuffer=max(1,cyClient/cyChar);
      return 0;
  case WM_SETFOCUS :
      CreateCaret(hwnd, NULL, cxChar, cyChar);
      SetCaretPos(xCaret * cxChar, yCaret * cyChar);
      ShowCaret(hwnd);
      return 0;
  case WM_KILLFOCUS :
      HideCaret(hwnd);
      DestroyCaret();
      return 0;
```

```
case WM_CHAR :                    // user press KEYBOARD
    for (i = 0 ; i < (int) LOWORD(lParam) ; i++)
    {
    switch (wParam)
     {
     case '\b' :                  // backspace pressed
       if (xCaret > 0)
       {
       xCaret-- ;
       cBuf[xCaret]=' ' ;
       HideCaret(hwnd);
       hdc=GetDC(hwnd);
       SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));
       TextOut(hdc, xCaret * cxChar, yCaret * cyChar,cBuf+xCaret,1);
       ShowCaret(hwnd);
       ReleaseDC(hwnd, hdc);
       }
       break ;
     case '\r' :                  // carriage return  pressed
       cBuf[xCaret]=0 ;
       if (xCaret!=0) {xCaret=0; yCaret++;}
       READ_CMD(cBuf);
       TEST_CMD(hwnd,xCaret, cxChar, yCaret,cyChar);
       xCaret=0; yCaret+=iLine;
       if (yCaret >= cyBuffer) InvalidateRect(hwnd, NULL, TRUE);
       break ;
     case '\x1B' :                // escape pressed
       InvalidateRect (hwnd, NULL, TRUE) ;
       xCaret=yCaret=0;
       break ;
```

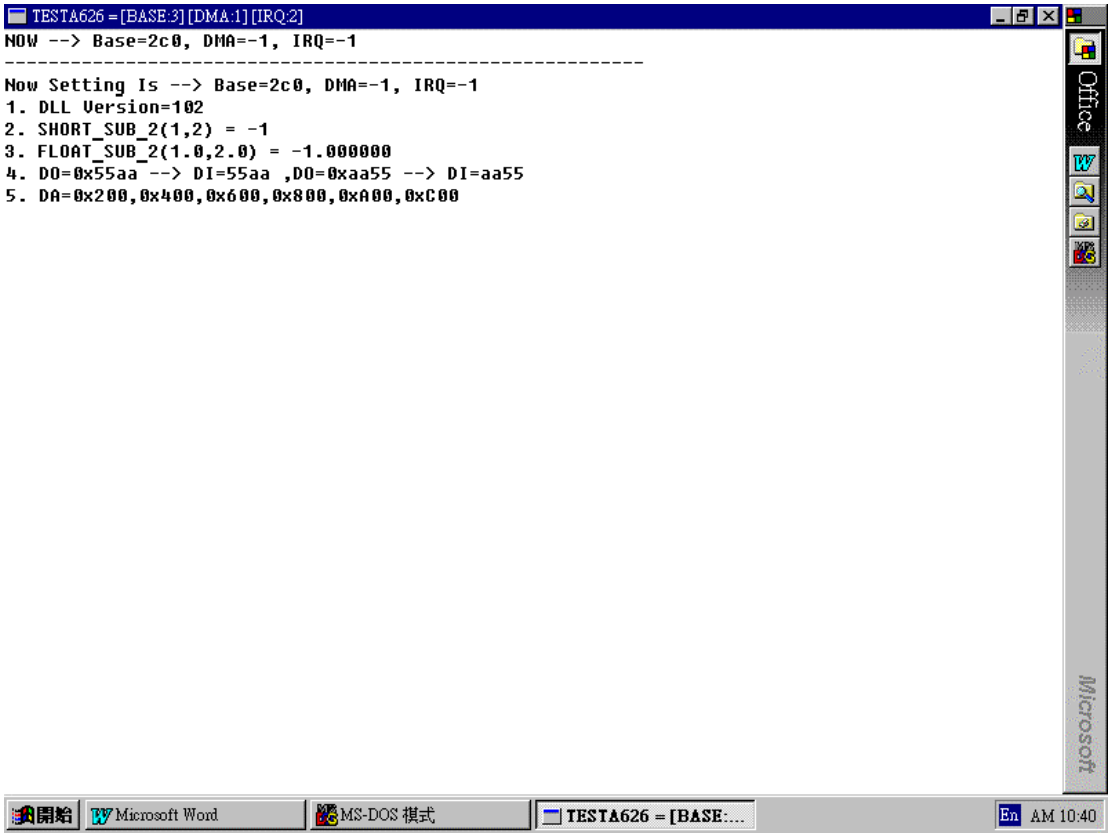
```
        default :               // other KEY pressed
          cBuf[xCaret]=(char) wParam;
          HideCaret(hwnd) ;
          hdc=GetDC (hwnd);
          SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));
          TextOut(hdc,xCaret*cxChar,yCaret*cyChar,cBuf+xCaret,1);
          ShowCaret(hwnd);
          ReleaseDC(hwnd, hdc);
          xCaret++;
          break ;
        }
      }
      SetCaretPos(xCaret*cxChar, yCaret*cyChar);
      return 0;
  case WM_PAINT :                   // clr and show HELP
      InvalidateRect(hwnd, NULL, TRUE) ;
      hdc=BeginPaint(hwnd, &ps);
      SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));

      sprintf(cBuf,"NOW --> Base=%x, DMA=%d, IRQ=%d",wBase,nDMA,nIRQ);
      TextOut(hdc,0,0,cBuf,strlen(cBuf));
      xCaret = 0 ; yCaret=1;
      SetCaretPos(0,yCaret*cyChar);

      EndPaint(hwnd, &ps);
      return 0;
  case WM_DESTROY :
      PostQuitMessage(0);
      return 0 ;
  }
return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```

```
/* [0][1][2]=base_address, [4]=DMA, [6][7]=IRQ                          */
void READ_CMD(char szCmd[])
{
short nT1,nT2,nT3;

if (szCmd[0]==0) return;        // only press [Enter]

nT1=ASCII_TO_HEX(szCmd[0]);       // HEX format
nT2=ASCII_TO_HEX(szCmd[1]);
nT3=ASCII_TO_HEX(szCmd[2]);
wBase=nT1*256+nT2*16+nT3 ;

nDMA=ASCII_TO_HEX(szCmd[4]);   // DECIMAL format
if (nDMA==0) nDMA=-1 ;

nT1=ASCII_TO_HEX(szCmd[6]);       // DECIMAL format
nT2=ASCII_TO_HEX(szCmd[7]);
nIRQ=nT1*10+nT2;
if (nIRQ==0) nIRQ=-1 ;
}

short ASCII_TO_HEX(char cChar)
{
if (cChar<='9') return(cChar-'0');
else if (cChar<='F') return(cChar-'A'+10);
else return(cChar-'a'+10);
}

/* ---------------------------------------------------------------------- */

void TEST_CMD(HWND hwnd, int x, int dx, int y, int dy)
{
```

```
}
```

The READ_COM only accept **fix format** command. The command format is given as below:

- if **nDMA**=0 symbol 224 \f "Wingdings" \s 12→“ no DMA is used symbol 224 \f "Wingdings" \s 12→“ **nDMA** will set to -1
- if **nIRQ**=0 symbol 224 \f "Wingdings" \s 12→“ no IRQ is used symbol 224 \f "Wingdings" \s 12→“ **nIRQ** will set to -1
- if = symbol 224 \f "Wingdings" \s 12→“ accept current setting of **wBase** and **nDMA** and **nIRQ**.



I. DIO

This driver is design for **DIO Cards & WDT-01 Card**. The user must refer to the original "???? User Manual" for details. The chapter gives an demo program for DIO-48. This example is very useful for the other DIO board. It is recommended to refer to chapter 4 for all DIO user.

A. NAPDIO.VxD Installation

If the user need to use interrupt, the NAPDIO.VxD must install in the system before software execution. The installation steps are given as following:

step 1 : place the companion floppy disk into A
step 2: enter DOS prompt under Windows
step 3 : a:
step 4 : cd w95
step 5 : cd dio
step 6 : cd vxd
step 7 : copy napdio.vxd c:\windows\system*.*
step 8 : power off the computer then power on

The DIO.DLL will call NAPDIO.VxD automatically. **If the user try to execute the demo program given in the companion floppy disk, the NAPDIO.VxD must be installed first.**

A. DIO.H

```
#define EXPORTS extern "C" __declspec (dllexport)

#define NoError                 0
#define VxdOpenError            1
#define VxdNoOpen               2
#define GetVxdVersionError      3
#define InstallIrqError         4
#define ClearIntCountError      5
#define GetIntCountError        6

EXPORTS short CALLBACK DIO_ShortSub2(short nA, short nB);
EXPORTS float CALLBACK DIO_FloatSub2(float fA, float fB);
EXPORTS WORD  CALLBACK DIO_GetDllVersion(void);
EXPORTS void CALLBACK DIO_OutputByte(WORD wPortAddr, WORD
                    wOutputVal);
EXPORTS WORD  CALLBACK DIO_InputByte(WORD wPortAddr);
EXPORTS WORD  CALLBACK DIO_OpenVxd(void);
EXPORTS void  CALLBACK DIO_CloseVxd(void);
EXPORTS WORD  CALLBACK DIO_GetVxdVersion(WORD *wVxdVersion);
EXPORTS WORD  CALLBACK DIO_InstallIrq(WORD wBase, WORD wIrq);
EXPORTS WORD  CALLBACK DIO_ResetIntCount(void);
EXPORTS WORD  CALLBACK DIO_GetIntCount(WORD *wIntCount);
```

A. DIO_ShortSub2, DIO_FloatSub2

test function

A. DIO_GetDllVersion

test function

A. DIO_OutputByte

- **Description :**
  This subroutine will send the 8 bits data to the desired I/O port.

- **Syntax :**
  void DIO_OutputByte(WORD wPortAddr, WORD wOutputVal);

- **Input Parameter :**
  wPortAddr : I/O port address, for example, 0x220
  wOutputVal : 8 bits data send to I/O port

- **Return Value :** void

- **Demo Program :** Refer to Sec. 3.13

A. DIO_InputByte

- **Description :**
  This subroutine will input the 8 bits data from the desired I/O port.

- **Syntax :**
  WORD DIO_InputByte(WORD wPortAddr);

- **Input Parameter :**
  wPortAddr : I/O port address, for example, 0x220

- **Return Value :**
  16 bits data with the leading 8 bits are all 0

- **Demo Program :** Refer to Sec. 3.13

A. DIO_OpenVxd

- **Description :** This subroutine will open the NAPDIO.VxD. **If the user use Interrupt, this function must be called once before the related functions are called.**

- **Syntax :** WORD DIO_OpenVxd();

- **Input Parameter :** void

- **Return Value :**
  NoError : OK
  VxdOpenError : open NAPDIO.VxD error
  (copy NAPDIO.VxD into c:\windows\system*.vxd)

- **Demo Program :** Refer to Sec. 3.13

A. DIO_CloseVxd

- **Description :**
  This subroutine will close the NAPDIO.VxD.

- **Syntax :** void DIO_CloseVxd();

- **Input Parameter :** void

- **Return Value :** void

- **Demo Program :** Refer to Sec. 3.13

A. DIO_GetVxdVersion

- **Description :** This subroutine will read the version number of NAPDIO.VxD.
- **Syntax :** WORD DIO_GetVxdVersion(WORD *wVxdVersion);
- **Input Parameter :** *wVxdVersion : address of wVxdVersion
- **Return Value :**
  NoError : OK
  VxdNoOpen : the NAPDIO.VxD no open
  GetVxdVersionError : read VxD version error
- **Demo Program :** Refer to Sec. 3.13

A. DIO_InstallIrq

- **Description :** This subroutine will install the IRQ service routine.
- **Syntax :** WORD DIO_InstallIrq(WORD wBase, WORD wIrq);
- **Input Parameter :**
  wBase : I/O base address of board
  wIrq : IRQ channel number
- **Return Value :**
  NoError : OK
  VxdNoOpen : the NAPDIO.VxD no open
  InstallIrqError : IRQ installation error
- **Demo Program :** Refer to Sec. 3.13

A. DIO_ResetIntCount

- **Description :** This subroutine will reset the **wIntCount** defined in NAPDIO.VxD.

- **Syntax :** WORD DIO_ResetIntCount();

- **Input Parameter :** void

- **Return Value :**
  NoError : OK
  VxdNoOpen : the NAPDIO.VxD no open
  ClearIntCountError : **wIntCount** clear error

- **Demo Program :** Refer to Sec. 3.13

A. DIO_GetIntCount

- **Description :** This subroutine will read the **wIntCount** defined in NAPDIO.VxD.

- **Syntax :** WORD DIO_ReadIntCount(WORD *wIntCount);

- **Input Parameter :** *wIntCount : address of wIntCount

- **Return Value :**
  NoError : OK
  VxdNoOpen : the NAPDIO.VxD no open
  GetIntCountError : **wIntCount** read error

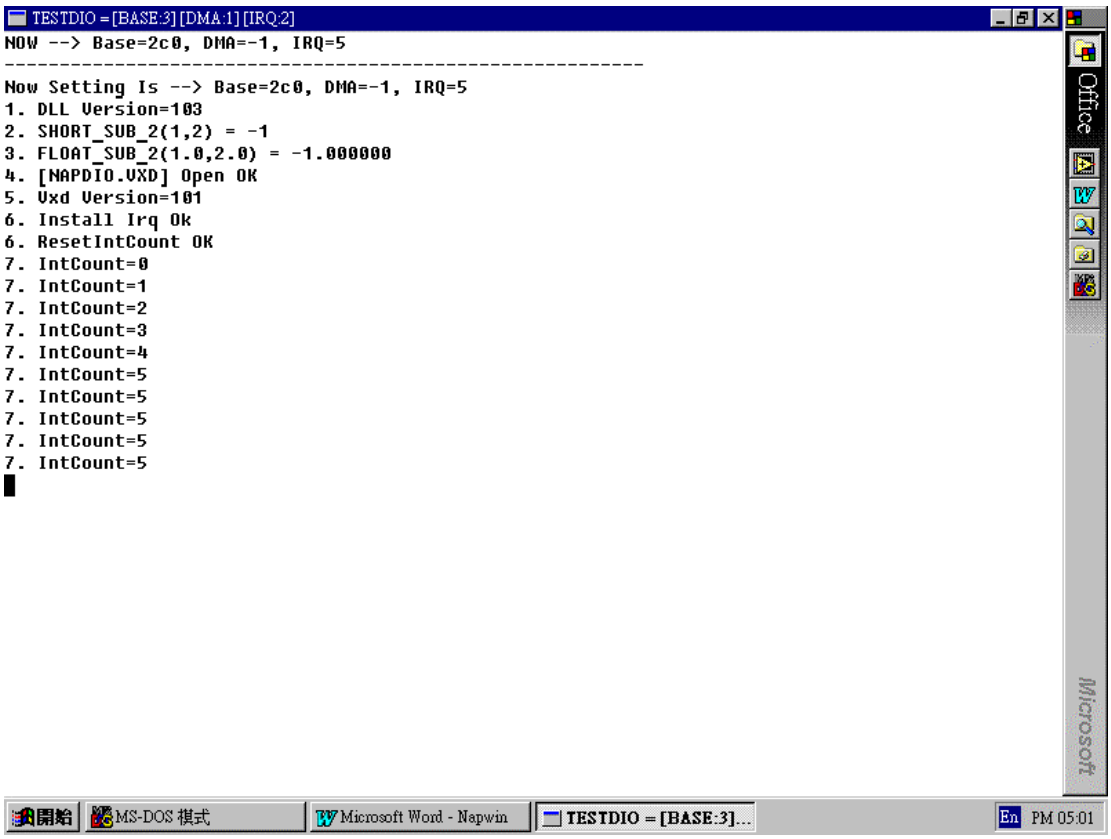
- **Demo Program :** Refer to Sec. 3.13

A. DIO Demo Program

```
void TEST_CMD(HWND hwnd, int x, int dx, int y, int dy)
{
char cBuf[80],cShow[80];
HDC hdc;
WORD wRetVal,wChannel,wConfig,wType,wBuf[10],wCount,i,j;
short nRetVal;
float fRetVal,fVal,fBuf[10];

iLine=0;
hdc=GetDC(hwnd);
HideCaret(hwnd);
SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));

sprintf(cShow,"------------------------------------------------------------");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;
sprintf(cShow,"Now Setting Is --> Base=%x, DMA=%d,
IRQ=%d",wBase,nDMA,nIRQ);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

wRetVal=DIO_GetDllVersion();
sprintf(cShow,"1. DLL Version=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

nRetVal=DIO_ShortSub2(1,2);
sprintf(cShow,"2. SHORT_SUB_2(1,2) = %d",nRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

fRetVal=DIO_FloatSub2((float)1.0, (float)2.0);
sprintf(cShow,"3. FLOAT_SUB_2(1.0,2.0) = %f",fRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;
```

```
wRetVal=DIO_OpenVxd();    // NOTE : this function must call once before
                          //       the other Vxd-service functions are
                          //       called. Refer to manual for details.
if (wRetVal==0) sprintf(cShow,"4. [NAPDIO.VXD] Open OK");
else sprintf(cShow,"4. [NAPDIO.VXD] Open Error");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

wRetVal=DIO_GetVxdVersion(&i);
if (wRetVal==0) sprintf(cShow,"5. Vxd Version=%x",i);
else sprintf(cShow,"5. Get Vxd Version Error");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

wRetVal=DIO_InstallIrq(wBase, (WORD)(nIRQ));
if (wRetVal==0) sprintf(cShow,"6. Install Irq Ok");
else sprintf(cShow,"6. Install Irq Error");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

wRetVal=DIO_ResetIntCount();
if (wRetVal==0) sprintf(cShow,"6. ResetIntCount OK");
else sprintf(cShow,"6. ResetIntCount Error");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

for (j=0; j<10; j++)
{
wRetVal=DIO_GetIntCount(&i);
if (wRetVal==0) sprintf(cShow,"7. IntCount=%x",i);
else sprintf(cShow,"7. GetIntCount Error");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;
Sleep(1000);
}

DIO_CloseVxd();   // NOTE : this function must call once before
                  //       program exit  (if Vxd-service functions
                  //       are called) Refer to manual for details.

ShowCaret(hwnd);
```

```
ReleaseDC (hwnd,hdc);
```


TESTDIO = [BASE:3] [DMA:1] [IRQ:2]
NOW --> Base=2c0, DMA=-1, IRQ=5
Now Setting Is --> Base=2c0, DMA=-1, IRQ=5
1. DLL Version=103
2. SHORT_SUB_2(1,2) = -1
3. FLOAT_SUB_2(1.0,2.0) = -1.000000
4. [NAPDIO.VXD] Open OK
5. Vxd Version=101
6. Install Irq Ok
6. ResetIntCount OK
7. IntCount=0
7. IntCount=1
7. IntCount=2
7. IntCount=3
7. IntCount=4
7. IntCount=5
7. IntCount=5
7. IntCount=5
7. IntCount=5
7. IntCount=5
Office
Microsoft
開始
MS-DOS 模式
Microsoft Word - Napwin
TESTDIO = [BASE:3]...
En
PM 05:01

I. RS-232 Driver

A. UART.H

```
#define EXPORTS extern "C" __declspec (dllexport)
// return code for UART_SEND_STR
#define    UART_START_SEND          1
#define    UART_PORT_ERR                2
#define    UART_HANDLE_ERR          3
#define    UART_STR_LENGTH_ERR      4

// return code for UART_CLEAR_INPUT_BUFFER
#define    UART_NO_ERR              0
// #define UART_PORT_ERR                2
// #define UART_HANDLE_ERR              3

// return code for UART_READ_SEND_STATUS
#define    UART_THREAD_INIT         5
#define    UART_THREAD_START            6
#define    UART_THREAD_OVER             7
// others                                  invalidate

// return code for UART_RECEIVE_1_CHAR
#define    UART_RECEIVE_ERR         8
// #define UART_NO_ERR                  0
// #define UART_PORT_ERR                2

EXPORTS short CALLBACK SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD  CALLBACK Get_DLL_Version(void);
```

```
// for general purpose UART applications
EXPORTS WORD  CALLBACK OPEN_COM(char port, DWORD baudrate, char
checksum);
EXPORTS BOOL  CALLBACK CLOSE_COM(char port);

// reserved
EXPORTS WORD  CALLBACK SEND_CMD(char port, char cmd[], WORD
wTimeout);
EXPORTS WORD  CALLBACK READ_COM_STATUS(char port, char Buf[], WORD
*status);

// for general purpose UART applications
EXPORTS WORD  CALLBACK UART_SEND_STR(char port, char cmd[]);
EXPORTS WORD  CALLBACK UART_CLEAR_INPUT_BUFFER(char port);
EXPORTS WORD  CALLBACK UART_READ_SEND_STATUS(char port);
EXPORTS WORD  CALLBACK UART_READ_RECEIVE_STATUS(char port);
EXPORTS WORD  CALLBACK UART_RECEIVE_1_CHAR(char port, char
*cReadChar);
```

A. SHORT_SUB_2

- **Description :**

Compute C=A-B in **short** format, **short=16 bits sign integer.** This function is provided for testing purpose. To test this DLLs can be called by your programming language, call this subroutine for testing. If this subroutine return the correct value, the other DLLs will work OK also.

- **Syntax :**

short SHORT_SUB_2(short nA, short nB);

- **Input Parameter :**

nA : short integer

nB : short integer

- **Return Value :**

return=nA-nB symbol 224 \f "Wingdings" \s 12→❝ short integer

- **Demo Program :** void

A. FLOAT_SUB_2

- **Description :**

Compute C=A-B in **float** format, **float=32 bits floating pointer number.** This function is provided for testing purpose. To test this DLLs can be called by your programming language, call this subroutine for testing. If this subroutine return the correct value, the other DLLs will work OK also.

- **Syntax :**

float FLOAT_SUB_2(float fA, float fB);

- **Input Parameter :**

fA : floating point value

fB : floating point value

- **Return Value :**

return=fA-fB symbol 224 \f "Wingdings" \s 12→“ floating point value

- **Demo Program :** void

A. Get_DLL_Version

- **Description :**

Read the software version of the NAP6000 DLLs.

- **Syntax :**

WORD Get_DLL_Version(void) ;

- **Input Parameter :**

void

- **Return Value :**

return=0x102 symbol 224 \f "Wingdings" \s 12→“ Version 1.2 **(WORD=16 bits**

unsigned integer)

- **Demo Program :** void

A. OPEN_COM

- **Description :**

This function will initialize the COM port. This function must be called before sending any command string.

- **Syntax :**

WORD OPEN_COM(char cPort, DWORD dwBaudRate, char cChecksum)

- **Input Parameter :**

cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
dwBaudRate : 1200/2400/4800/9600/19200/38400, others=invalidate
cChecksum : reserved

- **Return Value :**

1 = open COM OK
2 = port number error (validate only foe 1/2/3/4)
3 = baud rate value error (validate only for 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200)
4 = open COM port error
5 = set COM port mask error
6 = set buffer size error
7 = set time out error
8 = set DCB error

- **Demo Program :**

Refer to Sec. 4.14

A. CLOSE_COM

- **Description :**

This function will free all the resources used by OPEN_COM. This function must be called after program exit. The OPEN_COM will return error message if the program exit without calling CLOSE_COM function.

- **Syntax :**

BOOL CLOSE_COM(char cPort)

- **Input Parameter :**

cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

- **Return Value :**

1 = close COM OK

0 = port number error (validate only foe 1/2/3/4)

- **Demo Program :**

Refer to Sec. 4.14

A. SEND_CMD(reserved)

B. READ_COM_STATUS(reserved)

A. UART_SEND_STR

- **Description :**

This function will create a thread to send out string. **The input string is terminated with 0** . After create the thread, this function will return to the caller at once. **So the caller and the thread will be executed in multitasking environment.** The thread will auto terminated when the input string is sent over. **The caller can call "UART_READ_SEND_STATUS" to read the status of this thread**.

- **Syntax :**

WORD UART_SEND_STR(char cPort, char szCmd[])

- **Input Parameter :**

cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

szCmd : the starting address of the command string

- **Return Value :**

UART_START_SEND = create a thread to send input string and return at once

UART_PORT_ERR = port number error (validate for 1/2/3/4 only)

UART_HANDLE_ERR = COM port not initialize or OPEN ERROR

UART_STR_LENGTH_ERR = input string too long

- **Demo Program :**

  Refer to Sec. 4.14

A. UART_CLEAR_INPUT_BUFFER

- **Description :**

  This function will clear the UART input buffer.
- **Syntax :**

  WORD UART_CLEAR_INPUT_BUFFER(char cPort)
- **Input Parameter :**

  cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
- **Return Value :**

  UART_NO_ERR = function OK

  UART_PORT_ERR = port number error (validate for 1/2/3/4 only)

  UART_HANDLE_ERR = COM port not initialize or OPEN ERROR
- **Demo Program :** void

A. UART_READ_SEND_STATUS

- **Description :**

  This function will read the status of the send_thread.
- **Syntax :**

  WORD UART_READ_SEND_STATUS(char cPort)
- **Input Parameter :**

  cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate
- **Return Value :**

  UART_THREAD_INIT = create send_thread

  UART_THREAD_START = send_thread start to execute

  UART_THREAD_OVER = send_thread send input string OVER

- **Demo Program :** Refer to Sec. 4.14

A. UART_READ_RECEIVE_STATUS

- **Description :**

  This function will read the status of the UART receive buffer.

- **Syntax :**

  WORD UART_READ_RECEIVE_STATUS(char cPort)

- **Input Parameter :**

  cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

- **Return Value :**

  0 = no char

  others = number of chars are received in receive buffer

- **Demo Program :** Refer to Sec. 4.14

A. UART_RECEIVE_1_CHAR

- **Description :**

  This function will read 1 char from the UART receive buffer.

- **Syntax :**

  WORD UART_RECEIVE_1_CHAR(char cPort)

- **Input Parameter :**

  cPort : 1=COM1, 2=COM2, 3=COM3, 4=COM4, others = invalidate

- **Return Value :**

  UART_NO_ERR = receive 1 char OK

  UART_PORT_ERR = port number error (validate for 1/2/3/4 only)

  UART_RECEIVE_ERR = UART receive error

- **Demo Program :** Refer to Sec. 4.14

A. Demo Program

```
void Wait_COM(HWND hwnd, int yy )
{
char cBuf[80],cShow[80],ch;
HDC hdc;
WORD ret,i,t;

hdc=GetDC(hwnd);
SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));

ret=OPEN_COM(COM_PORT,9600L,0);
if (ret!=1)
  {
  strcpy(cShow,"Open Com Error");
  goto ret_label;
  }

for (;;)
   {
   ret=UART_READ_RECEIVE_STATUS(COM_PORT);
   if (ret!=0)
     {
     UART_RECEIVE_1_CHAR(COM_PORT, &ch);
     }
   else break;
   }

t=0;
for (;;)
   {
   ret=UART_READ_RECEIVE_STATUS(COM_PORT); /* finished or timeout */
   if (ret!=0)
     {
     UART_RECEIVE_1_CHAR(COM_PORT, &ch); /* finished or timeout */
```

```
      if (ch==0x0d) break;
      t=0;
      }
    else Sleep(1);
    t++;
    if (t>100)
      {
      strcpy(cShow,"Time Out");
      goto ret_label;
      }
    }

t=0; i=0;
for (;;)
    {
    ret=UART_READ_RECEIVE_STATUS(COM_PORT); /* finished or timeout */
    if (ret!=0)
      {
      UART_RECEIVE_1_CHAR(COM_PORT, &ch); /* finished or timeout */
      if (ch==0x0d) break;
      t=0;
      cBuf[i]=ch; i++;
      }
    else Sleep(1);
    t++;
    if (t>100)
      {
      strcpy(cShow,"Time Out");
      goto ret_label;
      }
    }
    cBuf[i]=0;

    strcpy(cShow,"Receive string : ");
    strcat(cShow,cBuf);
ret_label:
```

```
    TextOut(hdc,1,yy,cShow,strlen(cShow));
    ReleaseDC (hwnd,hdc);
    CLOSE_COM(COM_PORT);
}
```

I. Windows NT Applications

The system driver registration steps of Windows NT are given as following:

1. Enter DOS prompt
2. A:
3. cd wnt
4. cd sys
5. copy NAPWNT.SYS c:\WINNT\SYSTEM32\DRIVERS*.*
6. REGINI NAPWNT.INI
7. shutdown the NT, power off
8. power on, the NAPWNT.SYS will be in NT system

A. C Call DLLs(refer to Sec. 2.1)

B. MFC Call DLLs(refer to Sec. 2.2)

C. BC++ Call DLLs(refer to Sec. 2.3)

D. VB Call DLLs(refer to Sec. 2.4)

E. Delphi Call DLLs(refer to Sec. 2.5)

F. Demo Program(refer to Sec. 2.6)

II. DIO

This driver is design for **DIO Cards & WDT-01 Card**. The user must refer to the original "???? User Manual" for details.

A. DIO.H

```
#define EXPORTS

#define NoError      0
#define OpenError  1
#define DaChannelError 1

EXPORTS WORD  CALLBACK NAPWNT_INIT(WORD wPort);
EXPORTS void  CALLBACK NAPWNT_CLOSE(void);
EXPORTS void  CALLBACK DIO_OutputByte(WORD wBase, WORD wHexValue);
EXPORTS WORD  CALLBACK DIO_InputByte(WORD wBase);
```

A. DIO_OutputByte

- **Description :**
  This subroutine will send the 8 bits data to the desired I/O port.

- **Syntax :**
  void DIO_OutputByte(WORD wPortAddr, WORD wOutputVal);

- **Input Parameter :**
  wPortAddr : I/O port address, for example, 0x220
  wOutputVal : 8 bits data send to I/O port

- **Return Value :** void

- **Demo Program :** Refer to chapter 7

A. DIO_InputByte

- **Description :**
  This subroutine will input the 8 bits data from the desired I/O port.

- **Syntax :**
  WORD DIO_InputByte(WORD wPortAddr);

- **Input Parameter :**
  wPortAddr : I/O port address, for example, 0x220

- **Return Value :**
  16 bits data with the leading 8 bits are all 0

- **Demo Program :** Refer to chapter 7

A. NAPWNT_INIT

- **Description :**

Initialize the NAPWNT DLLs. This function must be called before the other DLLs.

- **Syntax :**

WORD NAPWNT(WORD wBase) ;

- **Input Parameter :**

wBase : I/O port base address

- **Return Value :**

0x8000 : invalidate handle

0x4000 : open handle error

NoError : OK

- **Demo Program :** Refer to chapter 7

A. NAPWNT_CLOSE

- **Description :**

Close the NAPWNT DLLs. This function must be called before program stop

- **Syntax :**

WORD NAPWNT_CLOSE(void) ;

- **Input Parameter :**

void

- **Return Value :**

0x1000 : status error

NoError : OK

- **Demo Program :** Refer to chapter 7

I. Demo for P16R16DIO

```
void TEST_CMD(HWND hwnd, int x, int dx, int y, int dy)
{
char cBuf[80],cShow[80];
HDC hdc;
WORD wRetVal,wChannel,wConfig,wType,wBuf[10],wCount,i,j,wData,wVal;
short nRetVal;
float fRetVal,fVal,fBuf[10];

if (wOpen==0)          /* need to init once */
      {
      NAPWNT_INIT(wBase);
      wOpen=1;
      }

iLine=0;
hdc=GetDC(hwnd);
HideCaret(hwnd);
SelectObject(hdc,GetStockObject(SYSTEM_FIXED_FONT));

sprintf(cShow,"------------------------------------------------------------");
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;
sprintf(cShow,"Now Setting Is --> Base=%x, DMA=%d,
IRQ=%d",wBase,nDMA,nIRQ);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

DIO_OutputByte(0x7ff4,0x55);
wRetVal=DIO_InputByte(0x7ff4);
sprintf(cShow,"0. 0x55 --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

DIO_OutputByte(0x7ff4,0xAA);
wRetVal=DIO_InputByte(0x7ff4);
```

```
sprintf(cShow,"0. 0xAA --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;
```

```
DIO_OutputByte(0x7f7c,0x00);
wRetVal=DIO_InputByte(0x7f7c);
sprintf(cShow,"0. 0x00 --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

DIO_OutputByte(0x7f7c,0xff);
wRetVal=DIO_InputByte(0x7f7c);
sprintf(cShow,"0. 0xff --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;
goto ret_label;

/* this demo is validate for P16R16 */
for (i=0; i<3; i++)         /* relays ON-OFF 5 times */
{
DIO_OutputByte(0x300,0x00);  /* R0-R7  OFF */
DIO_OutputByte(0x301,0x00);  /* R8-R15 OFF */
Sleep(500);
DIO_OutputByte(0x300,0xff);  /* R0-R7  ON  */
DIO_OutputByte(0x301,0xff);  /* R8-R15 ON  */
Sleep(500);
}

DIO_OutputByte(0x300,0x55);  /* R0-R7  OFF */
Sleep(500);
wRetVal=DIO_InputByte(0x300);
sprintf(cShow,"1. R0-R7=0x55 --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

DIO_OutputByte(0x300,0xAA);  /* R0-R7  OFF */
Sleep(500);
wRetVal=DIO_InputByte(0x300);
sprintf(cShow,"2. R0-R7=0xAA --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

DIO_OutputByte(0x301,0x55);  /* R0-R7  OFF */
Sleep(500);
```

```
wRetVal=DIO_InputByte(0x301);
sprintf(cShow,"3. R8-R15=0x55 --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

DIO_OutputByte(0x301,0xAA);  /* R0-R7  OFF */
Sleep(500);
wRetVal=DIO_InputByte(0x301);
sprintf(cShow,"4. R8-R15=0xAA --> read back=%x",wRetVal);
TextOut(hdc,x*dx,(y+iLine)*dy,cShow,strlen(cShow)); iLine++;

ret_label:
ShowCaret(hwnd);
ReleaseDC (hwnd,hdc);
}
```

I. RS-232 Driver

Refer to Chapter 4 for details. The UART.DLL is a WIN32 driver. It can be run in the Windows 95 & Windows NT.