Implementing a Windows CE .NET stream Interface

Objectives

After completing this lab, you will be able to develop a stream interface device driver.

Prerequisites

Before working on this lab, you must have:

- Familiarity with the Microsoft® Windows® CE Architecture.
- Familiarity with Microsoft Platform Builder
- Knowledge of debugging a device driver.
- Knowledge about the difference between a workgroup and a domain.

1. Developing a Stream Interface Device Driver

In this exercise, you will write a simple stream interface device driver that allows access to a simple string buffer. The stream interface device driver allows you to:

- Write a string (up to 256 characters) to the buffer.
- Read strings from the buffer.

To create a new DLL project for the device driver

1. Open the Platform Builder.
2. Open the workspace from Lab1 (MyPlatform).
3. Verify that the active configuration is a debug configuration
4. On the **File** menu, click **New Project or File**.
5. In the **New Project or File** dialog box, on the **Projects** tab, click **WCE Dynamic-Link library** as the project type.
6. In the **Project name** edit box, name your project **STRINGS**.

By default, the project is created inside the public directory under the project directory.

7. Click **OK**.
8. The next dialog box asks you what kind of DLL you want to create. Select **an empty project**.
9. Click **Finish**.

To add Startup files to project

1. Under the **LAB** directory, you will find two files named **STRINGS.CPP** and **STRINGS.DEF**. (see sources below). Use these files as the skeleton of the device driver. Copy the two files to the project directory you just created.

2. In Platform Builder, ensure that you are in **Workspace** window simultaneously by clicking the **View** menu and then click **Workspace.** You can also switch to Workspace window by pressing the **ALT** and **0** keys.

3. In the Workspace window, click the **FileView** tab.

4. Expand the **STRINGS** files item in the treeview. Right-click the **Source File** item and click **Add Files to Folder** in the popup menu.
5. In the **Insert Files into Project** dialog box, browse to the project directory, set **Files of type** to **All Files (*.*)**, and then, select both **STRINGS.CPP** and **STRINGS.DEF**, and then click **OK**.

```
;
; STRINGS.DEF
;
; Minimum Windows CE Serial Driver. Written by Paul Yao
;
LIBRARY STRINGS
EXPORTS
    STR_Close
    STR_Deinit
    STR_Init
        STR_IOControl
        STR_Open
        STR_PowerDown
        STR_PowerUp
        STR_Read
        STR_Seek
        STR_Write
```

To implement String reading and writing

1. Open the Strings.cpp file.

Implementing a Stream Interface Driver

2. Define a constant for the buffer size.
#define BUFSIZE 256

3. Define a global buffer for reading to and writing from
`WCHAR achBuffer[BUFSIZE];`

```
#define _WIN32_WINNT 0x0400
#include <windows.h>
#include <tchar.h>

#define  BUFSIZE 256

HANDLE g_hInstance;

WCHAR  achBuffer[BUFSIZE];
```

4. From within the driver's initialization function, set the buffer to zero using the **memset** command. Return from initialization by setting the return to a non-zero value.

```
DWORD STR_Init(DWORD dwContext)
{
    DWORD dwRet = 1;
    RETAILMSG(1,(TEXT("STRINGS: STR_Init\n")));
    memset(&achBuffer, 0, BUFSIZE * sizeof(WCHAR));
    return dwRet;
}
```

5. In the **Open** function of the driver, set the return to a non-zero value.

```
DWORD STR_Open(DWORD hDeviceContext, DWORD AccessCode, DWORD ShareMode)
{
    DWORD dwRet = 1;
    RETAILMSG(1,(TEXT("STRINGS: STR_Open\n")));
    return dwRet;
}
```

6. Add the following buffer reading code to the **Read** function of the driver.

```
DWORD cbBuffer = wcslen(achBuffer) + 1;
dwRet = min(cbBuffer, Count);
wcsncpy((LPWSTR)pBuffer, achBuffer, dwRet);
```

```
DWORD STR_Read(DWORD hOpenContext, LPVOID pBuffer, DWORD Count)
{
    DWORD dwRet = 0;
    RETAILMSG(1,(TEXT("STRINGS: STR_Read\n")));

        DWORD cbBuffer = wcslen(achBuffer) + 1;
        dwRet = min(cbBuffer, Count);
        wcsncpy((LPWSTR)pBuffer, achBuffer, dwRet);

    return dwRet;
}
```

7. Add the following code to the **Write** function of the driver.

```
dwRet = min(BUFSIZE, NumberOfBytes);
wcsncpy(achBuffer, (LPWSTR)pSourceBytes, dwRet);
```

```
DWORD STR_Write(DWORD hOpenContext, LPCVOID pSourceBytes, DWORD NumberOfBytes)
{
    DWORD dwRet = 0;
    RETAILMSG(1,(TEXT("STRINGS: STR_Write\n")));

        dwRet = min(BUFSIZE, NumberOfBytes);
        wcsncpy(achBuffer, (LPWSTR)pSourceBytes, dwRet);

    return dwRet;
}
```

8. Build the DLL through the **Build** menu by selecting **Build STRINGS.dll**.

9. On your **Workspace** window, on the **ParameterView** tab, modify **PROJECT.REG** (or platform.reg) in **ParameterView** tab to add the necessary entries for loading the stream driver during system boot.

10. Enclose the new entries in the file, with a conditional IF statement using **STRINGS_DRIVER**.

11. Ensure that at least the following values are set under **HKEY_LOCAL_MACHINE\Drivers\BuiltIn\STRINGS** and set the Prefix equal to **STR**:

```
IF STRINGS_DRIVER

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\STRINGS]
   "Dll"="STRINGS.DLL"
   "Prefix"="STR"
   "Index"=dword:1
   "Order"=dword:0
   "FriendlyName! = "KoanSoftware.com sample driver"

ENDIF
```

12. On the **Platform** menu, click **Settings**.

13. On the **Environment** tab, click **New**.

14. In the **Variable Name** edit box, type **STRINGS_DRIVER.**

15. In the **Variable Value** edit box, type **1**.

16. Click **OK** to return to the Platform Settings dialog box.

17. Click **OK** to return to the Platform Builder IDE.

18. Build the platform and verify that there were not build errors [**Sysgen**]

19. Reset the CEPC (close emulator).

20. On the **Target** menu, select **Attach Device**.

21. Select the **Output** window and click on the **Debug** tab. When the image starts running, check for messages from the **STRINGS** driver.
Another way to check if the **STRINGS.DLL** module has been loaded by the system is to check the **system modules** list. To display the module list, in the Platform Builder, on the **Target** menu, click **CE Modules and Symbols**, or **[Tools / Remote System Information]** or **[Tools / Remote File Viewer]**

2. Developing a Simple Application that calls STRINGS

In this exercise, you will create a simple Windows CE application that uses Win32 API file functions to request services from the STRINGS driver you wrote in the previous exercise.

To create a new WCE application project

1. On the **File** menu, click **New Project or File**.

2. In the **New** dialog box, click the **Projects** tab. Then select **WCE Application** as the project type.

3. In the **Project name** edit box, name your project as **STRINGAPP**.

4. Click **OK**.

5. The next dialog box asks you what kind of application you want to create. Select **A simple Windows CE application**.

6. Click **Finish**.

7. Under the **LAB** directory, you will a files named **STRINGAPP.CPP**. (see sources below). Use these files as the skeleton of the device driver. Copy the file to the project directory you just created.

8. Build the project (DO NOT sysgen).

9. Start emulator [**Target / Attach device**]

To implement access to the STRINGS device driver

1. Call the Win32 **CreateFile** function to open the **STR1:** device file.

2. If the call succeeds, that is if you receive a valid handle, call **WriteFile** to write a string.

3. Read the string by calling **ReadFile**. Display the string in a message box.

4. Build **StringApp.exe.**

5. Run it on the target using the **Target**, **Run Programs** menu.
By doing this, you do not have to build the StringApp.exe into the actual CE image.

Source code

```
/*++
STRINGS.CPP

THIS CODE AND INFORMATION IS PROVIDED \"AS IS\" WITHOUT WARRANTY OF
ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
PARTICULAR PURPOSE.
Copyright © 2002 Microsoft Corporation
--*/

#define _WIN32_WINNT 0x0400
#include <windows.h>
#include <tchar.h>

#define  BUFSIZE 256

HANDLE g_hInstance;

WCHAR  achBuffer[BUFSIZE];

//------------------------------------------------------------------------------
//------------------------------------------------------------------------------
BOOL WINAPI
DllEntryPoint(HANDLE hinstDLL,
              DWORD dwReason,
              LPVOID /* lpvReserved */)
{
    switch(dwReason)
    {
    case DLL_PROCESS_ATTACH:
        MessageBox(NULL, L"Driver Loading Now!", L"STRINGS", MB_OK);
        g_hInstance = hinstDLL;
        RETAILMSG(1,(TEXT("STRINGS: DLL_PROCESS_ATTACH\n")));
        return TRUE;
    case DLL_THREAD_ATTACH:
        RETAILMSG(1,(TEXT("STRINGS: DLL_THREAD_ATTACH\n")));
        break;
    case DLL_THREAD_DETACH:
        RETAILMSG(1,(TEXT("STRINGS: DLL_THREAD_DETACH\n")));
        break;
    case DLL_PROCESS_DETACH:
        RETAILMSG(1,(TEXT("STRINGS: DLL_PROCESS_DETACH\n")));
        break;
#ifdef UNDER_CE
    case DLL_PROCESS_EXITING:
        RETAILMSG(1,(TEXT("STRINGS: DLL_PROCESS_EXITING\n")));
        break;
    case DLL_SYSTEM_STARTED:
        RETAILMSG(1,(TEXT("STRINGS: DLL_SYSTEM_STARTED\n")));
        break;
#endif
    }

    return TRUE;
}
```

```
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
BOOL STR_Close(DWORD hOpenContext)
{
    BOOL bRet = TRUE;
    RETAILMSG(1,(TEXT("STRINGS: STR_Close\n")));
    return bRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
BOOL STR_Deinit(DWORD hDeviceContext)
{
    BOOL bRet = TRUE;
    RETAILMSG(1,(TEXT("STRINGS: STR_Deinit\n")));
    return bRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
DWORD STR_Init(DWORD dwContext)
{
    DWORD dwRet = 1;
    RETAILMSG(1,(TEXT("STRINGS: STR_Init\n")));

    memset(&achBuffer, 0, BUFSIZE * sizeof(WCHAR));

    return dwRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
BOOL STR_IOControl(DWORD hOpenContext,
                   DWORD dwCode,
                   PBYTE pBufIn,
                   DWORD dwLenIn,
                   PBYTE pBufOut,
                   DWORD dwLenOut,
                   PDWORD pdwActualOut)
{
    BOOL bRet = TRUE;
    RETAILMSG(1,(TEXT("STRINGS: STR_IOControl\n")));
    return bRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
DWORD STR_Open(DWORD hDeviceContext, DWORD AccessCode, DWORD ShareMode)
{
    DWORD dwRet = 1;
    RETAILMSG(1,(TEXT("STRINGS: STR_Open\n")));
    return dwRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
void STR_PowerDown(DWORD hDeviceContext)
{
    RETAILMSG(1,(TEXT("STRINGS: STR_PowerDown\n")));
}
```

```
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
void STR_PowerUp(DWORD hDeviceContext)
{
    RETAILMSG(1,(TEXT("STRINGS: STR_PowerUp\n")));
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
DWORD STR_Read(DWORD hOpenContext, LPVOID pBuffer, DWORD Count)
{
    DWORD dwRet = 0;
    RETAILMSG(1,(TEXT("STRINGS: STR_Read\n")));

        DWORD cbBuffer = wcslen(achBuffer) + 1;
        dwRet = min(cbBuffer, Count);
        wcsncpy((LPWSTR)pBuffer, achBuffer, dwRet);

    return dwRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
DWORD STR_Seek(DWORD hOpenContext, long Amount, DWORD Type)
{
    DWORD dwRet = 0;
    RETAILMSG(1,(TEXT("STRINGS: STR_Seek\n")));
    return dwRet;
}

//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
DWORD STR_Write(DWORD hOpenContext, LPCVOID pSourceBytes, DWORD NumberOfBytes)
{
    DWORD dwRet = 0;
    RETAILMSG(1,(TEXT("STRINGS: STR_Write\n")));

        dwRet = min(BUFSIZE, NumberOfBytes);
        wcsncpy(achBuffer, (LPWSTR)pSourceBytes, dwRet);

    return dwRet;
}
```

```
/*++
STRINGAPP.CPP

THIS CODE AND INFORMATION IS PROVIDED \"AS IS\" WITHOUT WARRANTY OF
ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
PARTICULAR PURPOSE.
Copyright © 2002 Microsoft Corporation
--*/


#include <windows.h>

#define BUFFER_SIZE  256  // The buffer size is the same as the driver's buffer size


int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPTSTR     lpCmdLine,
                   int       nCmdShow)
{

       // Open the STRINGS driver with READ and WRITE access
       // ------------------------------------------------------------------
    HANDLE hStr = CreateFile(TEXT("STR1:"), GENERIC_READ | GENERIC_WRITE, 0, NULL,
OPEN_EXISTING, 0, 0);
    if (INVALID_HANDLE_VALUE == hStr)
    {
        MessageBox(NULL, _T("Cannot open STR1:"), _T("StringApp"), MB_OK);
        return 0;
    }


    // Write a string to the driver.
       // ------------------------------------------------------------------
    DWORD  dwWritten = 0;
    WCHAR* pString   = TEXT("This is a test of the String Driver. This is only a test");
    WriteFile(hStr, pString, (_tcslen(pString)+1), &dwWritten, NULL);



    // Read string from driver.
       // ------------------------------------------------------------------
    WCHAR wch[BUFFER_SIZE];
    DWORD dwBytesRead = BUFFER_SIZE;
       memset(&wch, '\0', BUFFER_SIZE * sizeof(WCHAR));

       ReadFile(hStr, wch, sizeof(wch), &dwBytesRead, NULL);

    MessageBox(NULL, wch, TEXT("StringApp"), MB_OK);



    // Disconnect from driver.
       // ------------------------------------------------------------------
    CloseHandle(hStr);

       return 0;
}
```