7. Programming with the Windows API

7.1 Overview

In this chapter we want to show you how to write programs with the POW! environment which directly access the application programmers interface (API) of Windows. To understand the chapter you should be familiar with the general mechanisms of Windows programming.

Direct use of the Windows interface requires a clear understanding of the Windows programming model and also the concepts of event driven program design. There is a wide variety of publications available on this subject and so a general introduction is not provided in this chapter. The book written by Charles Petzold and listed in the literature section is recommended as an introduction to Windows programming.

7.2 Language extensions

On the whole, three Oberon-2 language extensions are required to allow access to the programming interface of Windows.

- The character "_" is permitted in symbols
- Structures and procedures may be marked by the keyword "[WINDOWS]". So structures become compatible with Windows (or C) as far as the memory alignment and calling conventions are concerned. In this case an appropriate linker symbol suitable for Windows (or C) is created for these procedures.
- In a definition module the interface of Windows-compatible Dynamic Link Libraries (DLLs) can be defined. For procedures only declarations, without code, are declared. The whole Windows API is accessed using this mechanism, as well as all other DLLs.

These three language extensions are provided by the Robinson Oberon-2 compiler which is supplied with the POW! environment for Windows.
7.3 Interfaces to “foreign” DLLs

7.3.1 Definition module

To use a DLL written in a language other than Oberon-2 it is necessary to define a compatible module interface for it. Once this is done the DLL can be used just like a normal Oberon-2 module.

The interface of a “foreign” DLL must be defined in an Oberon-2 definition module. Structures must be declared with the suffix `[WINDOWS]` to avoid differences in the memory representation due to the alignment of Oberon-2. Procedures must declared as `[WINDOWS]` to enable the creation of a linker symbol without the module name at the beginning.

The following example shows a drastically shortened version of the interface definition for the Windows API:

```oberon2
DEFINITION Windows;

CONST
  WM_CHILDACTIVATE = 22H;
  WM_CLOSE = 10H;
  WM_COMMAND = 111H;

TYPE (* Define "C" types in terms of Oberon-2 types *)
  WORD = INTEGER;
  DWORD = LONGLONG;
  BOOL = INTEGER;
  ADIR = LONGLONG;
  FARPROC = LONGLONG;
  HANDLE = UINT;
  UINT = WORD;
  HDC = UINT;
  HWND = UINT;

BITMAP = RECORD [WINDOWS]
  bmType: INTEGER;
  bmWidth: INTEGER;
  bmHeight: INTEGER;
  bmWidthBytes: INTEGER;
  bmPlanes: BYTE;
  bmBitsPixel: BYTE;
  bmBits: ADIR;
END;

PROCEDURE [WINDOWS] GetTickCount (): DWORD;
PROCEDURE [WINDOWS] GetTimerResolution (): DWORD;
PROCEDURE [WINDOWS] ShowWindow (a: HWND; b: INTEGER): BOOL;

END Windows.
```

7.3.2 Data types

In C far more basic data types are available than in Oberon-2. So when it comes to defining an Oberon-2 interface a direct equivalent cannot always be found. Unsigned integer types are particularly difficult as they do not exist in Oberon. As long as such variables only serve for comparisons with