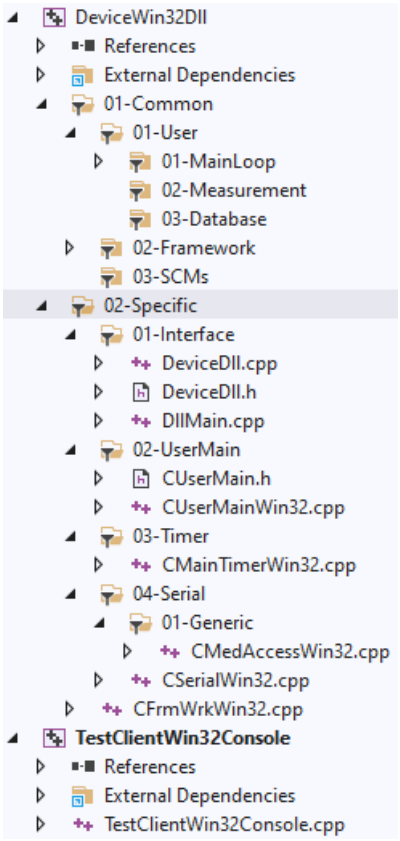
Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon: +49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Embedded Devices Simulation

The Interface

It is arguably a fact that most firmware developers use a Windows PC for their work. It is therefore obvious that a PC simulation for an embedded device should run under Windows.

When designing a reasonably neutral framework, the following picture quickly emerged.



The simulation consists of two components. The functionality of the device is fully encapsulated in a Windows DLL (DeviceWin32Dll). A client (TestClientWin32Console.cpp) is therefore required to load and operate the DLL.

The DLL in turn consists of two areas. The 01-Common area contains the *reusable source code of the device*. The area named '02-Specific' provides the source code that is required for adapting the DLL to the system environment.

However, it is important to see that not only a so-called HAL (Hardware Abstraction Layer) has to be implemented here, but also a part that maps certain functions to the System API (Application Layer Interface), which in our case is Windows.

For the reason mentioned, some of the modules are also implemented in two parts. One as platform independent (e.g. CMedAccess.cpp) and another as platform specific (e.g. CMedAccessWin32.cpp). However, there is always only one version of the header files (e.g. CMedAccess.h).

Figure 1: Structure of the Visual Studio Project

Some developers may find it a simplification if both the device and the test client are implemented in one programming language. The present draft was initially coded exclusively in C++. But it would also be easy to realize the test client in C#, Python or Java.

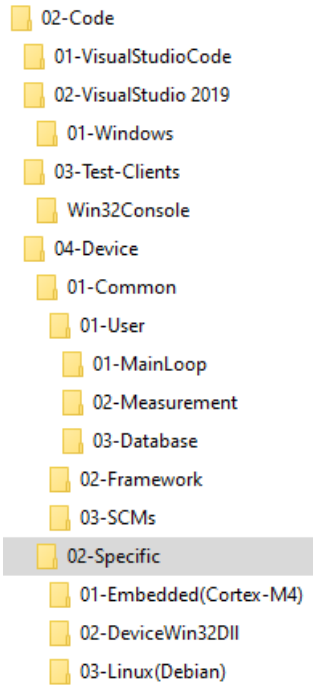
The interface to the DLL is kept very simple and consists of only five exported functions.

```
SIMDLL_DEVICE_API uint8_t  WINAPI  DEVSIM_Start(uint16_t _com_port);
SIMDLL_DEVICE_API void     WINAPI  DEVSIM_Terminate();
SIMDLL_DEVICE_API uint8_t  WINAPI  DEVSIM_GetStatus(void);
SIMDLL_DEVICE_API uint8_t  WINAPI  DEVSIM_GetComPort(void);
SIMDLL_DEVICE_API uint32_t WINAPI  DEVSIM_GetSystemTime(void);
```

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon: +49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

The Directory Structure

Of course, there is a little more to see in the directory structure than in the structure of just one project, as shown on the previous sheet.



Initially, it is planned to implement projects for Visual Studio 2019. However, the structure already provides a place for Visual Studio Code too. Even if I don't think that anyone really uses Visual Studio Code on a machine like a Linux computer, in the end it will probably be a matter of taste which version of Visual Studio is used.

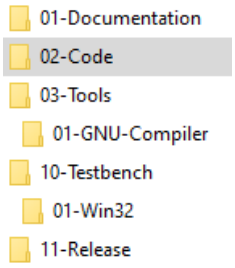
The first two directories contain the solutions for the two intended Visual Studio versions and the associated settings.

Test clients are programs that are used to operate or test the devices or device simulations. The projects for this are in the directory '03-Test-Clients'.

Finally, the source code for the device is stored in the fourth folder ('04-Device'). Just like in the concrete project, there are the areas '01-Common' and '02-Specific'. While the structure of '01-Common' is determined by the structure of the device, the subdirectory '02-Specific' is initially divided according to the intended platforms.

Figure 2: The Directory Structure

In addition to the directory '02-Code' shown above, there are four other main directories.



Only the '01-Documentation' path contains documentation related to the device.

The directory '03-Tools' is intended for the subject of software tools. Such a tool could be the compiler for the main processor on the final electronics. Subdirectories for additional tools such as additional compilers and/or debuggers should also be listed here.

Figure 3: Top Level Directories

The debug builds are output in the directory area '10-Testbench'. All executables (apps), DLLs and test clients for testing and debugging end up here.

The simulations running in this directory behave at least 95% like the final device, including the required interfaces. Therefore, *this environment makes it possible to develop functional acceptance tests even when no hardware is yet available*.

The files intended for delivery are finally copied into the directory area '11-Release'.