EPICS SCA CLIENTS ON THE .NET X64 PLATFORM*

C. Timossi[1] and H. Nishimura[2], LBNL, Berkeley, CA 94720, U.S.A.

Abstract

We have developed a .NET assembly, which we call SCA.NET, which we have been using for building EPICS [1] based control room applications at the Advanced Light Source (ALS)[2]. In this paper we report on our experiences building a 64-bit version of SCA.NET and the underlying channel access libraries for Windows XP x64 (using a dual core AMD Athlon CPU). We also report on our progress in building new accelerator control applications for this environment.

SIMPLE CHANNEL ACCESS AT ALS

Simple Channel Access (SCA)[3] is a library that provides a simplified API for developing Channel Access (CA) clients. SCA was developed at LBNL and has been in heavy use for both accelerator and beamline controls. Since SCA's most common use has been on Windows platforms, we originally packaged it as an ActiveX Control called SCA.COM[4]. This control is easily called by any Windows client supporting Active X (e.g. Labview, Visual Studio).

Although ActiveX controls can be accessed from .NET assemblies we believed a more seamless integration was important for two reasons. First, .NET will be the standard development framework for Windows in the future--on Vista, it will be the only development framework. Second, it looked like a relatively simple task to re-package the ActiveX control as a .NET assembly. In fact, as often happens during a re-write, we found many optimizations that resulted in better performance than the previous component, in the context of the 32-bit version of SCA.NET[5].

MIGRATION TO 64-BIT

Need to Support 64-bit

PCs with 64-bit processors and operating systems, such as Windows XP x64, are finally becoming widely available. Scientific applications, which are accelerator tracking programs for us [6], are already taking advantage of the larger address space and faster execution speed.

On the other hand, typical machine control applications have little need for these advantages as 32-bit has been mostly sufficient.

It is convenient, however, when building model based 64bit control applications, to have access to 64 bit versions of controls libraries to avoid mixing 32 bit and 64 bit libraries.

*Work supported by the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

1. CATimossi@lbl.gov. 2. H_Nishimura@lbl.gov

Building CA for Windows XP x64

Building CA on x64 bit platforms has been done previously. Even so, some help from the author [7] of the CA software was needed to add a macro to CA base code for the AMD architecture on Windows. Further, although the EPICS build system is both powerful and flexible, we decided to build the libraries in the Visual Studio environment to take advantage of its rich debugging tools.

Finally, a patch previously posted by AMD, which synchronizes the core time stamp counters, needed to be installed [8].

The 2 DLLs built in this fashion (Com.dll and Ca.dll) are categorized by .NET as *unmanaged* because they were built for the win32 environment, not for .NET.

SCA.NET

Unlike the above DLLs, ALS.dll, which implements SCA.NET, is built as a .NET assembly. When .NET assemblies call routines in unmanaged libraries, they do so through a special interface called *Platform Invoke (P/Invoke)*. C# has syntax for P/Invoke that uses the *DllImport* keyword, for example, as shown below:

```
[DllImport("ca.dll")]
 unsafe public static extern
    char * ca_message(uint ca_status);
```

.NET considers any code that manipulates pointers as *unsafe*. The C# compiler will generate an error unless code using pointers are labelled with the *unsafe* keyword.

The CA routines used by SCA.NET are similarly wrapped.

It's worth noting that although the CA libraries are separately compiled for both 64-bit (x64) and 32-bit (x86) versions, ALS.dll is a single binary of the .NET assembly built with the Visual Studio build option of "Any CPU". The OS is responsible for loading either the x64 or x86 versions of the DLLs that ALS.dll needs.

32-bit Programs on 64-bit Windows

Although both 32-bit and 64-bit programs can run on 64-bit Windows, 32-bit programs must run in a compatibility layer called *WOW64*. This layer wraps the application in its own 32-bit environment from which it can only call directly into 32-bit DLLs. The resulting overhead from this layer is architecture dependent--AMD processors can execute 32-bit code directly whereas Intel processors have to emulate 32-bit instructions. A 32-bit process may also use inter-process communications (IPC) when calling into a 64-bit library. Interface options are: pipes, messages, signals, ActiveX/COM out-of-process servers, and networking APIs.

Assigning the CPU as a Build Option

When building SCA.NET, it's most convenient to build and deploy separately for x64 and x86. So first the sources for CA and ALS.dll are compiled as native x64 libraries then an installer project is built. This process is repeated for the 32-bit version. Basically, the installer places the output binaries in either "Program Files\LBNL" for x64 binaries or in "Program Files (x86)\LBNL" for 32 bit binaries.

EXAMPLE

When we program SCA/NET client programs that are portable on x86 and x64 platforms, we must ensure that all the libraries are also portable if they are by 3rd parties. We currently use open-source libraries: SourceGrid [9] for string grid and ZedGraph[10] for chart. Both are managed code in C# and portable.

Fig.1 is an example program that reads and displays all the ALS storage ring magnet EPICS channels (287 magnets and 1669 channels) at 1 Hz by using 13 SourceGrid controls on WinForm.

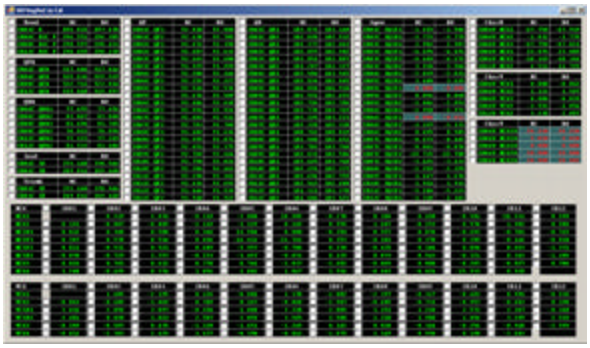

Fig.1. SCA.NET client example

This program runs both on x86 and x64 Windows without rebuilding.

We have also created EPICS database client programs that use ADO.NET 2.0 to access static EPICS database record information such as process variable names from a MySQL[11] database. ADO.NET also allows saving of device information and configured/edited information to XML files for runtime use on both x86 and x64 Windows.

ACKNOWLEDGEMENTS

The authors appreciate useful advices from T. Scarvie, and the system management help from C. Ikami and T. Kellogg.

REFERENCES

[1] L. R. Dalesio, et al., ICALEPCS '93, Berlin, Germany, 1993. http://www.aps.anl.gov/epics/

[2] LBL PUB-5172 Rev. LBL, 1986.

A. Jackson, IEEE PAC93, 93CH3279-7(1993)1432

[3] http://www-controls.als.lbl.gov/epics_collaboration/sca/

[4] C. Timossi and H. Nishimura, IEEE PAC'97, 0-7803-4376-X/98, p805, 1998

http://www-controls.als.lbl.gov/epics_collaboration/sca/win32

[5] H. Nishimura and C. Timossi, PCaPAC 2005, Hayama, Japan, 2005.

[6] H. Nishimura and T. Scarvie, EPAC 2006, Edinburgh, Scotland, 2006.

[7] Jeff Hill, http://www.aps.anl.gov/epics/contacts.php

[8] http://www.amd.com/us-en/Processors/TechnicalResources/0,,30_182_871_13118,00.html.

[9] http://zedgraph.org

[10] D. Icardi, http://www.devage.com/

[11] http://www.mysql.com/