# AN EMBEDDED DISTRIBUTED SYSTEM BASED ON TINE AND WINDOWS CE

A. Pazos[#], U. Ristau and S. Fiedler, European Molecular Biology Laboratory EMBL, Hamburg Unit, Germany
P. Duval, DESY, Hamburg, Germany

*Abstract*

We present an embedded distributed system based on the integration of the control system (TINE) inside an embedded-PC running Windows CE RTOS. As Windows CE is different to Desktop Windows and requires a cross-compilation of the source modules, porting TINE to Windows CE turned out to be straightforward, but non-trivial. Having a dedicated Windows CE TINE library allows to create device servers inside the embedded operating system, close to the hardware application layer. The embedded-PC is the master of the hardware line, where different hardware devices are connected through a real-time Ethernet field bus. On the one hand, there is a low level control of this hardware performed by a set of programmable logic controllers (PLC) running in fast cycling and on the other hand, there is a higher level control performed by the TINE server devices. The server is responsible for providing an interface to the external world, exporting the functionality of the system through the Ethernet control network. It is also possible that the server acts as a TINE client for other external servers, constituting a network of embedded nodes. We present a practical development that demonstrates the proposed system.

## INTRODUCTION

Beamline control software needs flexible and robust system architecture. Although the main parts of the system are fixed, it is very common to extend them and to have temporal devices running together with the existing ones. Moreover, the operation of each component involves synchronization between the different hardware components and the possibility of remote control. Fast feedback systems are also desirable. This makes it necessary to have a distributed system. In this point we identify two kinds of distribution. The first one is hardware related, where the individual hardware components (motor controller, analogue/digital signals, sensors, etc) are connected together through a common field bus system. Architectures with more than one field bus or mixed with the computer bus are also common [1]. The second one is the distribution of the control software thanks to the use of a control system (CS). EPICS, TINE, TANGO are examples of CS used in the accelerator control community [2]. Normally they are based on a client/server architecture. The server is responsible for the hardware access and for the distribution of the system parameters using an Ethernet network. The central services provided by the control system, such as archiving, alarming, naming and logging, etc. are also crucial in this type of structure. In many systems different layers of complexity in the server side are also considered.

Recently, one sees the emergence of real-time Ethernet (RTE) buses [3]. These are an alternative to the afore-mentioned field buses. Both distribution points (hardware and software) are in this case based on an Ethernet network. This makes for less complexity in the integration of the system components. At this point we present a complete system based on embedded-PC control, which will treat both distribution points together. We combine the embedded hardware with the embedded software. In this design the integration of the control system running inside the embedded-PC and a close access to the hardware is essential. This means that all the logic (low-level and high-level) can run inside the embedded-PC. Similar systems based on a VME bus have been used in many places with the inconvenience of the difficult extension of the system and the high cost of the hardware [4]. Our solution is based on the known Windows CE OS running in an embedded-PC together with the TINE control system. The rest of the hardware is connected by means of a real-time Ethernet field bus.

## SYSTEM OVERVIEW

In this section we present a description of the designed system that fulfils many of the requirements of beamline software automation. The main piece of our embedded distributed system is an embedded-PC running Windows CE OS. This is a compact DIN-rail Industrial PC (IPC). The IPC is the master of the real-time Ethernet field bus where different hardware devices can be connected. The system provides a low level programming interface, where a set of PLCs runs in a fast cycling mode. At this point the low-level functionality accessing the hardware is solved. There is the necessity of running higher level software that has direct access to the control of the PLC. The integration of the control system inside the OS has to be performed.

All the components of the system are well known and proven technologies, close to the open source world and not dependent on specific hardware developments. This makes it a flexible and extendable system.

The following figure (Fig. 1) shows a schematic representation of the design of the system.

[#]andres.pazos@embl-hamburg.de

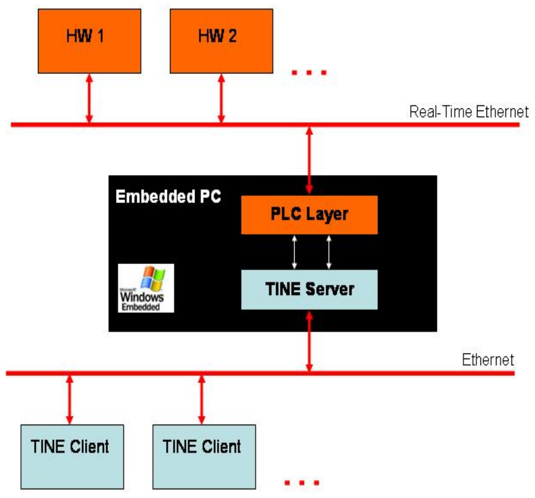


Figure 1: Distributed embedded design.

### Windows CE

Windows CE is a 32-bit architecture operating system (OS) released by Microsoft for covering a full range of embedded devices. Its application goes from mobile devices to industrial automation systems [5]. It is real-time, lightweight, multithreaded and component-based OS. This makes it possible to build a very light kernel by only selecting the components one needs to support, for example an optional graphical user interface. It has a priority based pre-emptive thread scheduler, supports 256 levels of priority and synchronization objects such as semaphores, mutexes and critical sections are also available.

The first thing one learns when using Windows CE is that it is not like the Windows desktop OS. Specific tools for programming are necessary and also new compatible libraries. The last release of the OS is Windows CE 6.0 (2008). This has been an important improvement in terms of available resources of the OS, but always trying to maintain it as a small footprint. The most radical improvements have been the increase from 32 processes to 32000, and the virtual memory from 32 MB to 2 GB. Windows CE is classified as a hard real-time operating system (RTOS) [6]. This special feature and the full control of the OS make it very attractive for automation applications.

A specific programming infrastructure is necessary in order to develop software for the WINDOWS CE OS. The standard way of proceeding in order to develop a Windows CE application is to implement the software in a desktop PC. The first step is to choose the development environment. We have selected the Microsoft embedded Visual Studio 4.0 which is freely available from Microsoft www.microsoft.com/downloads. The standard supported programming languages are C/C++, VisualBasic and C#. There is also support for Java from third-party development environments. We have focused our server development in C/C++ like the kernel of the TINE libraries.

It is also necessary to install a software development kit (SDK) of the specific Windows CE before starting a project. This is normally provided by the device manufacturer and not by Microsoft. There are also standard SDK that support the operative system in general provided by Microsoft. Before testing the software into the real device, it is very convenient to use the Windows CE emulator. Like with the SDK, there are specific device emulators and general ones. When you want to develop a hardware application or a network application the emulator can be useful only in first term. It is also possible to debug the application running it directly in the device. The prerequisite is to establish a connection between the desktop PC and the Windows CE device.

### Control System

The higher level distribution of the system is possible thanks to the use of the TINE (Three-fold Integrated Network Environment) CS [7]. This provides the required network services as well as central services of archiving, alarming, naming and logging. It is a multi-platform CS supporting Windows, UNIX, Linux, MAC OS, VxWorks, and others. But until now it was not support for the Windows CE OS. In order to integrate TINE inside the Windows CE a cross-compilation of the Windows source code was necessary. A first beta version has been released for the architecture x86 (February 2008). The service pipe and remote pipe options are not available for the Windows CE version. Two new files have been added to wrap the unsupported windows functions under CE (wincelib.c and wincelib.h). The final code is fully compatible with the other TINE distributions and the integration into WindowsCE is transparent to the user. A Windows CE version of the TINE server example *sineServer* was also compiled and tested under Windows CE 6.0. The common device interface (CDI) [8] and the hardware plugs have likewise been cross-compiled and tested with real hardware under CE. This will be detailed in the next section. Most porting issues were related to ANSI string and time functions. This is because Windows CE uses Unicode coding, so in many cases it is necessary to convert ANSI functions or use the TCHAR library. Also several porting difficulties were related with the set-up of a correct Windows CE programming environment. An infrastructure and knowledge for this family of embedded system is now available.

## PRACTICAL APPLICATION

A testing prototype that follows the presented ideas has been implemented. This is composed by a 1 GHz Intel embedded-PC provided by Beckhoff (www.beckhoff.de). The operating system is a light-weight compilation of the Windows CE 6.0 OS with a size of 20 Mbytes.

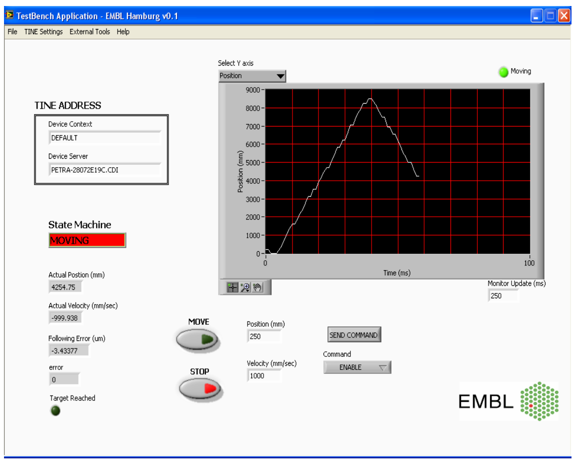
### Hardware-side

The embedded-PC, as mentioned beforehand, is the master of the real-time Ethernet where a set up of hardware is connected. In our case, we use the standard

EtherCAT (www.ethercat.org). The testing prototype is constituted by two stepper motor controllers with three different kinds of encoders (linear encoder, LDVT and potentiometers).

### *Software-side*

On the software side, a low-level PLC program was developed, following a simple state machine. Every transition in the state machine is steered from an external TINE device server that talks directly to the PLC program. An interchange of data is possible between both layers. We have implemented within our device server both a polling system, monitoring functions with a cycle of 250 ms, and also an event system, where callback functions wait for a value change. At this stage, the server that directly accesses the PLC program has been semi-automatically generated by CDI. This CDI library provides, by filling a data sheet, an automatically generated TINE server that maps the PLC variables. The procedure in cross-compiling the CDI layer is very similar to that of the TINE CS. In the embedded-PC, we can run many different TINE device servers. In our case, we access the control network to get some relevant data of other servers and correlate it with the data of the connected hardware. Finally, an external TINE client was developed. TINE provides a rich API for programming clients (C++, Java, Visual Basic, Labview). At this point, a Labview client which runs in an external PC was developed. Here we have implemented a thin client, because the logic was included inside the TINE server layer. It is also compatible in the present design to create rich clients. A screenshot of the system is presented in the next figure (Fig. 2).



TestBench Application - EMBL Hamburg v0.1

TINE ADDRESS
Device Context: DEFAULT
Device Server: PETRA-28072E19C.CDI

State Machine: MOVING

Actual Postion (mm): 4254.75
Actual Velocity (mm/sec): -999.938
Following Error (um): -3.43377
error: 0

Select Y axis: Position
Monitor Update (ms): 250
Position (mm): 250
Velocity (mm/sec): 1000
Command: ENABLE

Figure 2: Labview TINE Client of the practical application.

## CONCLUSIONS

With this approach we present a final embedded distributed system. This is a low-cost solution compared to other embedded systems such as VME. We have released a beta version of the TINE Windows CE compilation already available in http://tine.desy.de. This is the first open source control system running embedded in the Windows CE OS family. Real time possibilities are also possible thanks to the capabilities of the OS. Future efforts have to be undertaken in this line, in order to synchronize the hardware layer together with the server devices. We have provided a new programming framework of embedded programming inside Windows CE. Although Windows Desktop programming expertise is very helpful for entering the Windows CE embedded community, new expertise is needed.

As we have explained, we provide a complete solution making the connection of the server layer to the hardware layer independent of the network. More security issues are possible, implementing the logic of the system inside the embedded-PC, which is powerful enough to perform these operations.

Besides Microsoft, there are very active communities supporting Windows CE, as well as forums, etc. With regard to the control system, the power and the flexibility of TINE demonstrates that it is possible to adapt it to new incoming systems, programming languages or architectures. The possibility of running the control system in mobile devices has been not yet investigated, but with the availability of TINE inside Windows CE now it is also possible to run it in Windows Mobile OS. This makes possible to program thin and rich clients that run inside a mobile device like a PDA. This is a line of future developments which includes also the combination with wireless networks.

## REFERENCES

[1] R. Bacher, "The New Control System for the Future Low-Emittance Light Source PETRA3 at DESY: from Conceptual Design Work to Realization", ICALEPS'07, Knoxville, Tennessee USA, 2007 , TPPB27, p.217

[2] P. Duval and Z. Kakucs, "The Babylonization of Control Systems Part II- the Rise of Fallen Tower", ICALEPS'03, Gyeongju, Korea, 2003, p. 513

[3] R. Zurawski and L. Lavagno "Embedded Systems: Towards Networking of Embedded Systems", from Embedded Systems Handbook, ISBN 0-8493-2824

[4] J. Yan and et al, "Ethernet Based Embedded IOC for FEL Control System", ICALEPS'07, Knoxville, Tennessee, 2007, FOAB03, p.720

[5] "Windows CE 5.0 in Automation Technology Applications", Beckhoff PControl Magazine, May 2006

[6] "Windows CE 5.0 on an x86 Platform", RTOS Evaluation Program, Dedicated Systems Experts, EVA-2.9-TST-CE-x86-01. Oct 2004

[7] P. Bartkiewicz and P. Duval. "TINE as an accelerator control system at DESY" Meas Sci Technol, 18:2379–2386, 2007, p. 2379-2386

[8] P. Duval, H. Wu, "Using the Common Device Interface in TINE", PCaPAC'06, Virginia, USA, 2006