USING WINDOWS XP EMBEDDED BASED SYSTEMS IN A CONTROL SYSTEM

Tim Gray, Bob Mannix, ISIS, Rutherford Appleton Laboratory, United Kingdom

Abstract

Linux is popular in the Controls community, so discussion of a system based around Microsoft Windows XP Embedded is probably unusual. Having to replace an obsolete front-end IO system, Windows XP Embedded was chosen as the platform mainly due to the familiarity of Windows. The author will describe configuring and using Windows XP Embedded based CompactPCI systems to deliver an operating system and software platform that is used to communicate between the Controls System and hardware deployed around the accelerator using 'in-house' designed IO cards; including configuring the system to boot over the network, using HTTP (HyperText Transfer Protocol) and XML (Extensible Markup Language) to exchange data with the controls system and providing a simple C/C++ API to communicate with the Controls System database..

WHY DID WE DO THIS

The Control System for ISIS runs on a cluster of OpenVMS servers using Vista Control Systems[1] software. The hardware we might want to control or monitor could be a power supply, magnet or beam chopper for example. Our existing setup placed a chassis based on the STEbus standard between the Control Systems and the hardware. This STEbus standard and the particular design of it we are using had become obsolete and a new solution was needed.

THE NEW PLATFORM

We examined various options for the replacement of the STEbus based systems. Our starting point though was the hardware platform. We rapidly decided that we wanted our new systems to be based on the CompactPCI standard. It is widely supported throughout the controls and instrument industry and looks set to maintain that support in the future.

Windows CE Embedded, Windows XP Embedded, Embedded Linux, and QNX were evaluated but the similarity of Windows XP Embedded to the standard Windows XP won the day, requiring less skills acquisition within the group.

REQUIRED SOFTWARE

Windows Embedded Studio

The software for building and maintaining a Windows XP Embedded image is all included with Windows Embedded Studio, which has to be purchased. It's major components are:

1. Target Designer
2. Component Database Manager
3. Target Designer
4. Component Designer
5. First Boot Agent (FBA) and FBreseal.exe
6. SDI Loader & sdimgr.exe
7. Remote boot Manager

Target Analyzer

Use this to create a template for your XP Embedded image. It examines the devices your hardware platform contains.

Component Database Manager

All Windows XP Embedded features, programs and drivers etc. are components and stored in an SQL database. A component database needs to be configured somewhere before you build an image.

Target Designer

This is the main design application. The results from Target Analyzer build your initial image description in here. This is then used to add further components and features to your image.

Component Designer

Custom components may be designed to install custom applications or drivers or configure the image in another required way. These components can then be added to the component database.

First Boot Agent (FBA) and FBreseal.exe

FBA runs the first time an image is booted on a target device. It completes the configuration of the image. Once FBA has run, the target device is complete, or FBreseal.exe if deploying the image to multiple target devices.

SDI Loader & sdimgr.exe

We have our Windows XP Embedded image loading over the network. These programs are needed to build the file containing the image that loaded.

Remote Boot Manager

This is used to specify which image a network client loads and along with the Remote Boot Service fulfils this.

BUILDING A WINDOWS XP EMBEDDED IMAGE

By running these applications in the order listed a Windows XP Embedded Image can be built. Certain steps such as 3 – 5 are usually iterated to perfect your image.

This was the process used to configure our Windows XP Embedded image for our CompactPCI hardware As stated, we chose to load our image to our CompactPCI chassis over the network, but other devices such as hard disk, CDROM, CompactFlash and USB stick are supported. The choice to boot our image over the network was largely down to failures of similar hard disk based systems in certain areas of ISIS and was one of the main reasons for us using an embedded operating system.

THE APPLICATION PROGRAM

The task of our application is via communications with the Control System running on VMS to interface with CompactPCI IO cards in the chassis and control or monitor the status of the equipment running ISIS. The application was designed as a component and added to our Windows XP Embedded image.

The application was written in C++ and designed to run as a Windows Service. An HTTP server is run within the application, accepting communications from the Control System using the HTTP GET & POST methods. The information comprising the request, and the subsequent response, is packaged in XML. The use of an HTTP server and XML does give the advantage of being able to examine a system in the field using a browser.

The HTTP server is supplied as a class in a program library, along with various other classes, most notably a Database class. The Database object is a representation of the Database in the Vista Control System, this being a collection of channels which represent items of interest for control or monitor on the relevant hardware.

```
POST postdatabase HTTP/1.1
Content-Length: 531

<?xml version="1.0" encoding="UTF-8"?>

<database>
    <channel name="STATUS">
        <params type="integer"/>
    </channel>
    <channel name="THING:VOLTAGE">
        <params type="integer">
            <hparams>0 0 1 5 7 8 9 4</hparams>
            <cparams>3.1 1.5</cparams>
        </params>
    </channel>
    <channel name="ARRAY:READ_DATA">
        <params type="integer" size="100">
            <hparams>0 0 7 9 8 3 5 9 1</hparams>
            <sparams>FT0</sparams>
         </params>
    </channel>
</database>
```

Figure 1: Example XML used to load a Database.

By starting the HTTP server the application program absolves itself of all network communication, it all being handled by this server. The data that the HTTP server works with is shared with the application in the form of a Database object, which is a representation of the database on the Vista Control System. See Figure 1 for an example of the HTTP/XML data used to load a Database object. The Database class is written to be thread safe, as it has to be being shared in this manner.

The application program interacts with this Database object using various available methods, such as IterateChannels(), UpdateChannel(), GetValue() etc. Via CompactPCI IO cards installed in the chassis data is read from and to the equipment at regular intervals according to information in this Database object. See Figure 2 to illustrate this interaction.

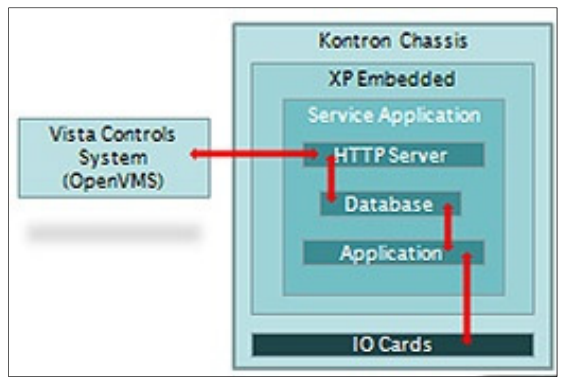

Figure 2: Application Data Flow.

CONCLUSION

We have currently deployed 14 Windows XP Embedded Systems around ISIS. They have proved an excellent replacement for our old STEbus based systems. The initial load time of a system if repowered is slower than the old STEbus systems, but the infrequency of doing it mitigates this. The advantages of being able to use an internet browser to examine the current Database on a running system are very useful. The familiarity of Windows XP is a benefit to all members of the group when managing these systems. The image does however have to be regularly updated with Windows updates and running relevant security software.

REFERENCES

[1] www.vista-controls.com