12-2010

Implementation of a Microsoft Windows embedded standard system.

Kristopher L. Kumler
University of Louisville

IMPLEMENTATION OF A MICROSOFT WINDOWS EMBEDDED STANDARD SYSTEM

By

Kristopher L. Kumler
B.S., University of Louisville, 2002

A Thesis
Submitted to the Faculty of the
University of Louisville
J.B. Speed School of Engineering
in Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Computer Engineering and Computer Science

December 2010

IMPLEMENTATION OF A MICROSOFT
WINDOWS EMBEDDED STANDARD SYSTEM

Submitted by: _____
Kristopher L. Kumler

A Thesis Approved On

December 8, 2010
(Date)

by the Following Reading and Examination Committee:

_____
Ahmed H. Desoky, Thesis Director

_____
Adel S. Elmaghraby

_____
John F. Naber

ACKNOWLEDGMENTS

I would like to express my appreciation to my advisor, Dr. Ahmed Desoky, for his guidance and patience. I also thank my committee, Dr. Adel Elmaghraby and Dr. John Naber.

Special thanks to Mr. David Goffinet and Image Vault, LLC. for the use of equipment and the use of Image Vault's software product as an example host application. Thanks to Eric Kramer, for his help with advice, proofreading, and editing.

I want to express my appreciation to my wife, Mandy Kumler, for pushing me to finish. Her tremendous love and dedication throughout the past twelve years is ineffable.

ABSTRACT

Many dedicated-use computer systems sold as complete products require a turn-key design delivered to the customer. This requires a system which is stable, secure, and serviceable. Adaptability of the system to existing software applications is a key consideration for many vendors.

This thesis attempts to establish and gather best practices for designing, configuring, and building a Microsoft Windows Embedded Standard 2009 system. An existing real-world system will be used as a case study and example implementation. The end result will be a relatively compact, secure, and efficient Microsoft Windows Operating System image to support the target software application.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Modern dedicated computer systems are in use throughout the consumer marketplace. Devices such as ATMs, retail kiosks, gaming or security systems, medical devices, or other application-dedicated systems require a stable and secure platform for their Operating System. The repercussions of failures in either system stability, with crashes or denial of service, or security, with user or external breaches, can result in the loss of life, money, or business-related licenses.

The popularity of the Microsoft Windows XP Operating System (OS) as a target for applications has led to numerous software solutions in the marketplace. Image Vault has publicly marketed Digital Video Recorders (DVRs) for security surveillance since 1998. The flagship Image Vault DVR solution is the PRO-Command product, targeted toward numerous operations: convenience stores, groceries, banks, restaurants, retail stores, and several other markets. The PRO-Command product interfaces with analog cameras, network-based cameras, Point-of-Sale (POS) devices, safes, and alarm systems. It is desirable to utilize existing software applications with minimal changes in a more secure and stable operating system. In 2004, the PRO-Command codebase was ported to operate on a Windows XP-based platform.

The Windows XP OS, while being extremely popular, can still suffer from many security vulnerabilities and stability issues. For Windows XP, device

drivers total 85% of failure reports [1]. By strictly controlling hardware devices, device drivers, and software applications, a higher degree of system stability can be obtained. In addition, this stability can be improved by carefully testing all of these components.

The primary goals in most companies are to generate a profit and deliver a quality product. In the case of an existing software application, porting it to another more economical or secure OS may be commonly suggested. Porting an existing application, or developing a new application while an existing solution exists, for an alternative OS can be prohibitively expensive for many organizations. Additionally, not all desired implementation or programming options may be present on alternative OS options. For these reasons, the cost of porting a software application seldom outweighs the benefits.

Microsoft markets several products for embedded systems, with a wide range of applicable target devices. The Windows Embedded family includes: Windows Embedded CE, for small footprint (300KB at minimum), typically portable devices; Windows Embedded Standard, for reduced footprint systems (40 MB OS at minimum) with full 32-bit Windows functionality, compatible with Windows XP Professional; Windows Embedded for Point of Service, an OS optimized for Point of Service Devices (cash registers, etc.); Windows Embedded Enterprise, which is Windows XP Professional or Windows Vista with a restrictive license; and Windows Embedded NavReady, for portable navigation devices. Windows Embedded for Point of Service and Windows Embedded

NavReady are specializations of Windows Embedded Standard and Windows Embedded CE, respectively. [2] Microsoft Windows Embedded Standard 7, based on the Windows 7 product, is targeted toward similar applications as Windows Embedded Standard 2009. However, Windows Embedded Standard 7 does not scale to the smaller PC devices that WES 2009 supports. [19]

The solution presented in this thesis is to demonstrate a reliable and secure Windows Embedded Standard (e.g. Windows XP-compatible) Operating System configuration and implemented for the Image Vault PRO-Command DVR. The configuration will be designed with Windows Embedded Standard 2009, released November 2008, and incorporates Windows XP Professional Service Pack 3 (SP3) files and features. This solution includes the initial hardware configuration and analysis; requirements for the hosted software application, which will be observed as a black-box system for this project; and best practices for configuration and future maintenance.

Chapter two will include an overview of relevant literature concerning this project. The third chapter will describe the system and discuss the overall requirements along with all equipment. The fourth chapter will cover the project implementation details. The implementation details will serve as a tutorial and guide for implementations by others with a systems engineering background. The fifth chapter will show the final implementation results, and discuss and expand on the best practices utilized in the implementation. The final chapter will show possible paths for future work.

II. LITERATURE REVIEW

A. System Reliability

For a security system of any type, let alone video surveillance, system reliability is a topmost concern. With software systems, reliability is defined as "the probability of failure-free software operation for a specified period of time in a specified environment." [3] A complete system consists of both software and hardware, and each much be addressed independently, as well as in conjunction with each other. As well, the aspect of cost must be addressed, as with all engineering endeavors.

As previously discussed, for the Windows XP Operating System, hardware device drivers total 85% of OS failure reports and as such must be a large consideration in overall system reliability. [1] The first priority in limiting exposure to unstable device drivers is to only make required device drivers available to the Operating System. This limits the number of devices that may be connected, but in the case of this closed security system, that will also be a design goal. With a limited set of hardware devices and associated drivers, it may be possible to have a close relationship with the driver vendor, allowing for more in-depth research and feedback with issues. A final and more hands-on approach to ensuring reliability of device drivers is to perform independent testing of the driver. Microsoft provides the Driver Verifier tool to detect illegal function calls or actions that may corrupt or de-stabilize the system. Use of the Driver Verifier

is highly involved, but can reveal many potential issues before they are realized in the final product. [4]

The hardware platform used for the system is also highly important. Hardware designed for high-availability systems also tends to have with it a very high cost. However, the use of individual parts intended for industrial applications, that carry a longer mean-time-to-failure (MTTF), can be quite advantageous in increasing hardware reliability and balancing the cost of the system. [5] The inclusion of an Uninterruptible Power Supply device can extend the operational life of the system and maintain uptime. [6] Prudent requirements include adequate testing of the hardware components before integration into a shipping product. Testing of individual components within an integrated system can be made easier by simplifying the software used in the scenario.

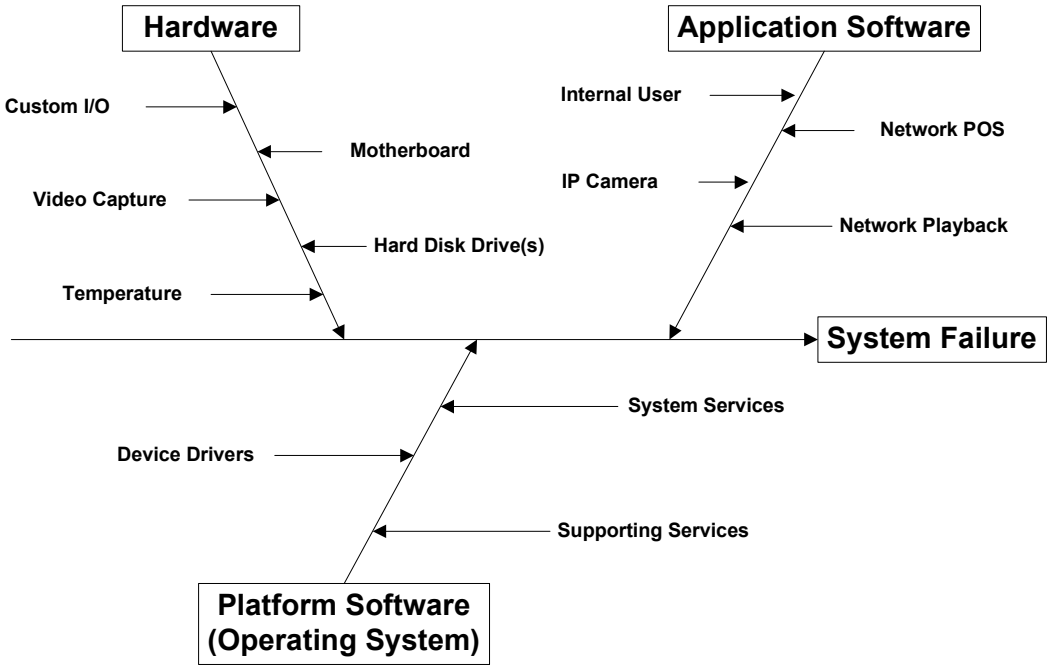

FIGURE 1. System Reliability Failure Fault Tree

A simplistic fault tree for system reliability failure is represented in FIGURE 1. The root causes of failure for many commodity computer systems will be quite similar. For this design application, the additional portions are related to video capture, supporting services, and the application software itself. The aforementioned figure includes all of the items discussed concerning system reliability.

B. System Security

Closely related to, and a component of, the topic of system reliability, is that of security vulnerabilities. External threats to the security and stability of the system come primarily from physical access, internal or "trusted" users of the system, and network exposure. Limiting physical access in the DVR scenario is primarily the responsibility of the end-user during installation. Utilizing the front access panel lock or placing the entire system within a locking container box is one step that may be taken. Also, the DVR units in a retail scenario are usually in a security room or the business manager's office. Reducing problems introduced by malicious internal users is a pervasive problem, but this will rely primarily in the hosted software application, and as such will not be introduced here.

The vulnerabilities possible by the system network exposure may be addressed via several standard methods. The first method is to not run or execute system services or applications that operate or listen on the network in the first place. Good network firewall practices can block access to open ports for services that cannot be disabled due to various circumstances or requirements. [7] The Microsoft Windows XP Service Pack 2 Firewall primarily controls ingress network traffic, and also has several pertinent features: excepted traffic by program or destination port, excepted traffic by source scope, startup (boot) security, and programmatic control. [8]

C. Microsoft Windows Embedded Products

The Image Vault PRO-Command system was designed for a Microsoft Windows-based system. The current target platform is Windows XP Professional-compatible. The implementation is ordinarily restricted to pure language standards-compliant functions, except where necessity or performance requires specialization. Because of the high cost of porting the application to any other Operating System, such as Linux, this product must be deployed on a Microsoft Windows XP-based system. In the future the application may be adapted for later Windows releases.

Microsoft's Windows Embedded family of products is targeted toward several different markets and architectures. Determination of the correct Operating System product for a project is of critical importance. The Microsoft product family contains several "core" Windows Embedded products. Windows Embedded for Point of Service and Windows Embedded NavReady, while distinct products are essentially specialized versions of other family products (Windows Embedded Standard and Windows Embedded CE, respectively). The core Windows Embedded products are listed in TABLE I, and are Windows Embedded CE, Windows Embedded Standard, and Windows Embedded Enterprise. Windows Embedded Enterprise is itself a family of three different product offerings: Windows XP Professional for Embedded Systems, Windows Vista Business for Embedded Systems, and Windows Vista Ultimate for Embedded Systems. The Windows Embedded Enterprise family products consist

of the full version of the Operating Systems, with restricted licensing for embedded situations. [2]

TABLE I.
WINDOWS EMBEDDED FAMILY COMPARISON

Product	**Smallest Footprint**	**Component Granularity**	**Processor Architectures**	**Example Applications**
Windows Embedded CE	300 KB	700 Components	ARM, MIPS, SHx, x86	Digital Picture Frames, Portable Media Players, Portable Navigation Devices, Voice Over Internet Protocol (VoIP) Phones, Handheld Terminals
Windows Embedded Standard	40 MB	12,000 Components	x86	Media Servers, Digital Video Recorders, Automatic Teller Machines, Point of Service Kiosks
Windows Embedded Enterprise	**XP Professional**: 128 MB RAM, 1.5 GB Hard Drive **Vista**: 1 GB RAM, 40 GB Hard Drive	N/A	**XP**: x86 **Vista**: x86, x64	Existing applications with no customization possibilities.
Windows Embedded Standard 7	~500 MB	~150 OS feature sets	x86, x64	Feature-rich kiosks, video applications

D. Windows Embedded Standard

Windows Embedded Standard (WES) is the current product, in November 2008 replacing Windows XP Embedded (XPe), which is a more well-known trademark. Most existing documentation references the trademark Windows XP Embedded. Additionally, all documentation and knowledge concerning XPe will apply to WES. The release of WES contains several API features and updated components that XPe did not include, such as Microsoft Silverlight, .NET Framework 3.5, Windows Server 2008 features, Windows Media Player 11, Internet Explorer 7, Internet Explorer 8, and Microsoft Baseline Security Analyzer. [9]

2.4.1 Boot and Storage Options

Microsoft identifies several features in the Windows Embedded Family as *Embedded Enabling Features* (EEF's). These are features which are unique to the embedded products and design scenarios. The EEF's related to system boot and storage are listed in TABLE II, although not all options may be able to be used at the same time, while others may need to be used in combination. [10] For flexibility of boot options throughout a product lifetime, the OS image

footprint should be minimized if at all possible, to enable changes to the boot operation.

TABLE II.
BOOT AND STORAGE OPTIONS

Feature Name	Feature	Example
Remote Boot	Boot image from network server	Boot device from PXE server
Enhanced Write Filter	Prevent write access at volume-level	Enables boot from read-only media
File Based Write Filter	Prevent write access to files or directories	Protection of critical system files
Flash Technology Support	Support for booting from flash technology	Flash devices such as PCMCIA-ATA, Compact Flash, MultiMediaCard, or Memory Stick
El Torito Support	Bootable CD-ROM format specification	Boot from CD-ROM
USB Boot	Booting from USB flash device	Boot from USB flash device. e.g. “thumb drive”

2.4.2 Deployment and Management Technologies

It is necessary, when building the system, to be able to deploy the Operating System image to the target software. Additionally, servicing and management tools are required to maintain the system. Microsoft has provided several tools and technologies with the WES product to assist in the serviceability aspect.

The deployment features include the First Boot Agent (FBA), System Deployment Image (SDI) Manager, and the Windows Pre-Installation Environment (WinPE). The FBA allows the designer to perform tasks that must be executed on the run-time image and cannot be authored offline using

development tools. [10] The SDI Manager allows for the deployment of WES images to virtual disks and later delivery to target Hard Disk Drives in the field. The WinPE feature is a hardware independent Windows environment that gives a bootable platform to access the target hardware, allowing tools such as the SDI Manager to operate.

Service and management features include the Device Update Agent (DUA), and the Active Directory Client. Additionally, a highly useful Windows XP Embedded utility is the Image Difference Engine. The DUA tool allows for running local or remote scripts to modify device settings or update system binaries and applications. The Active Directory Client allows for the deployed device to participate in a Microsoft Active Directory domain and be maintained by domain administrators. [10] The Image Difference Engine is an advanced tool to compare two deployed images in order to identify specific changes to facilitate binary updates.

2.4.3 Platform Development Tools

The Windows Embedded Standard product includes several development tools required to facilitate the design process. The Target Analyzer tool operates on the target hardware, under a Windows XP Professional or Windows Pre-Installation Environment, to catalog the hardware devices in the system. The Target Designer is the primary development tool, used to select all components of

the configuration along with their dependencies, estimate the storage footprint, and assemble the binary files to be deployed for the image. The command-line application, XPECMD, is analogous to Target Designer and allows for scripting configurations. The Component Designer allows for the creation of components to be used in the system configuration and can be leveraged to build a library of specialized components to allow for speedy delivery of unique configurations. The Component Database Manager runs primarily in the background of the development system, tracking all components and relationships, along with their physical location in the repository of files. The Command Line Tool can be used to automate the entire process and investigation of component relationships.

2.4.4 Windows Embedded Component System

Windows Embedded Standard 2009 enables the creation of customized Operating System run-time images by the breakdown of Windows XP Professional (WES 2009) into a set of discrete components. A set of components in an image configuration with specific additional settings, fully describe the features of the Operating System image. An individual component may contain information on files, registry data, and particular resources, such as commands to be executed during the build process, actions, Device IDs, help documents, or a dependency on another component. The component may describe one small portion of a larger application, or the entire application. A component may

inherit all properties from another component, utilizing it as a prototype. Additionally, a macro component is one that groups other components together and has no files or other data included.

Windows Embedded Standard controls the order in which components are built during the run-time image build process by utilizing a dependency on a build order. The build order dependency allows a component to modify data placed by a previously built component. The build proceeds with the build order, determined by dependencies, and then by an ordinal phase number, from 0 to 65,535, going from initial build through the device boot process. The phase number is optional for most components, but required for advanced components where exact control is needed on its execution in system startup.

III. INSTRUMENTATION AND EQUIPMENT

A. Hardware Platform

The target hardware consists of the base devices that usually compromise a modern x86 computer system along with some specialized devices. Of the base devices, the motherboard and processor are the core defining components, and the rest, ordinarily, may safely be considered generic and are supported by Microsoft-provided function drivers. [12] The specialized devices are few: one or more video capture cards, and custom Image Vault I/O board (FK-145). The list of major hardware devices is presented in TABLE III.

TABLE III.
HARDWARE DEVICES

Device Type	**Instance**
Motherboard	Advantech AIMB-763
Processor	Intel Pentium 4 651, 3.4 GHz
I/O Board	Image Vault FK-145
Video Capture Card	UDP Technology NCP3000V2: 16 Channel, 120 FPS
Hard Disk Drive	Seagate, SATA, 250 GB
Memory	512 MB DDR2
Optical	Optiarc DVD+RW AD-7200A

The specific processor is of particular importance at the time of image configuration. Since WES runtime executed on target hardware does not provide for dynamically selecting the Hardware Abstraction Layer (HAL) automatically by any setup program, the correct architecture must be selected at configuration. The HAL choices include: Advanced Configuration and Power Interface (ACPI) PC, MPS Uniprocessor PC, ACPI Uniprocessor PC, Standard PC (non-ACPI),

ACPI Multiprocessor PC, and MPS Multiprocessor PC. [14] For a system with a single processor, the Uniprocessor components will have a performance benefit. Most modern configurations will support ACPI, and with the proliferation of multi-core processors, the multiprocessor components will be desired. Therefore, the most common HAL component in use will be the ACPI Multiprocessor PC.

The system motherboard for the target hardware, the Advantech AIMB-763, utilizes a chipset provided by Intel Corporation, the Intel 945G. Also on the motherboard is the Intel I/O Controller Hub 7 (ICH7). [15] These devices are responsible for much of the system operation and drivers required. Favorably, they are commonly encountered devices and, as such, much information is known about them and drivers are readily available.

The video capture card for the target system is one of several models provided by the same vendor (UDP Technology) for use in Image Vault systems. The drivers are provided by the vendor as standard Windows driver packages. Public access to documentation is restricted.

The additional I/O board is a custom board and a generic driver interface and package is provided by an existing Image Vault project. [17] The I/O board is extremely basic, providing support for RS-485 communication, used for serial Pan-Tilt-Zoom (PTZ) cameras; general purpose input/output (GPIO) channels, for communicating with devices such as alarm panels or door sensors; and a

watchdog alarm, which will sound a piezo buzzer if the software application is non-responsive. [13]

B. Hosted Application

The primary purpose of the complete system is to host the Image Vault PRO-Command Recorder application (the "server" process). The software suite for the Image Vault PRO-Command Recorder consists of two primary applications and their dependent files: IVRecord, the PRO-Command Recorder server process; and AutoUpdate, a device servicing and update application. The IVRecord application consists of the main executable, IVRecord.exe, and numerous file dependencies included with the application. The OS directly executes the AutoUpdate application on startup, which in turn handles any system updates and then executes the IVRecord application. A certain amount of documentation and specification is provided for the external dependencies of these applications.

C. Windows Embedded Tools

Several useful tools are provided in the Windows Embedded Studio distribution, which will almost always be used in the process of an image configuration. The WES tools are the Target Analyzer Probe (TAP), the Windows Pre-Installation Environment (WinPE), Component Designer, and Target Designer. Some additional third-party tools are commonly employed in order to

more easily produce higher quality configurations, such as Dependency Walker or InCtrl5.

The Target Analyzer Probe (TAP) utility is responsible for gathering information on known devices in a computer system. TAP executes on the target hardware system with an existing Operating System and generates a device information file (PMQ) which contains a comprehensive list of identified system devices. [14]

Microsoft also provides the Windows Preinstallation Environment (WinPE), a basic distribution of Windows XP which may be booted from CD-ROM. WinPE allows a developer to execute TAP on target hardware without an existing, permanently installed Operating System.

The determination of any dependent file modules for a desired application is necessary so that those dependencies can be added to the final image. This is foremost provided by the vendor or engineering specification documents. Otherwise, the applications must be analyzed to determine what modules are necessary. Dependency Walker is a free utility that can scan and analyze a Windows executable, statically or via observation at run-time, to generate a list of dependent modules, as well as which functions of those modules are used. [16]

Dependency analysis on Windows systems can also extend beyond the installation or program execution. Post-reboot actions commonly take place and prove more difficult to discover. The InCtrl5 program records system state – including file locations, registry data, and INI or text file contents – at two

separate temporal points. The difference of the two system states details the installation and configuration of a program and its actions. [18]

IV. DEVELOPMENT OF AN EMBEDDED OPERATING SYSTEM PLATFORM

The implementation begins with the hardware analysis for each device, and the creation of custom Windows Embedded components when necessary. Likewise, the analysis of software components continues in the same fashion, with analysis of software execution for dependencies.

A. Motherboard

The motherboard, and its on-board components, represents the bulk of the workload for hardware analysis and component creation. Only the bare minimum components are included in the system at this time, to create more purely representative data for the motherboard – this is very important for component re-use. The execution of the Target Analyzer Probe (TAP) within the Windows PE environment on the motherboard device generates a complete listing of devices present. The TAP execution is very straightforward, as can be seen in FIGURE 2. Some of the devices listed may be software-enumerated devices and are rarely desired in the final image. However, using Windows PE as the Operating System during the target probe does leave out several of the possible software-enumerated devices. See APPENDIX I for the TAP output (PMQ file) and modifications.

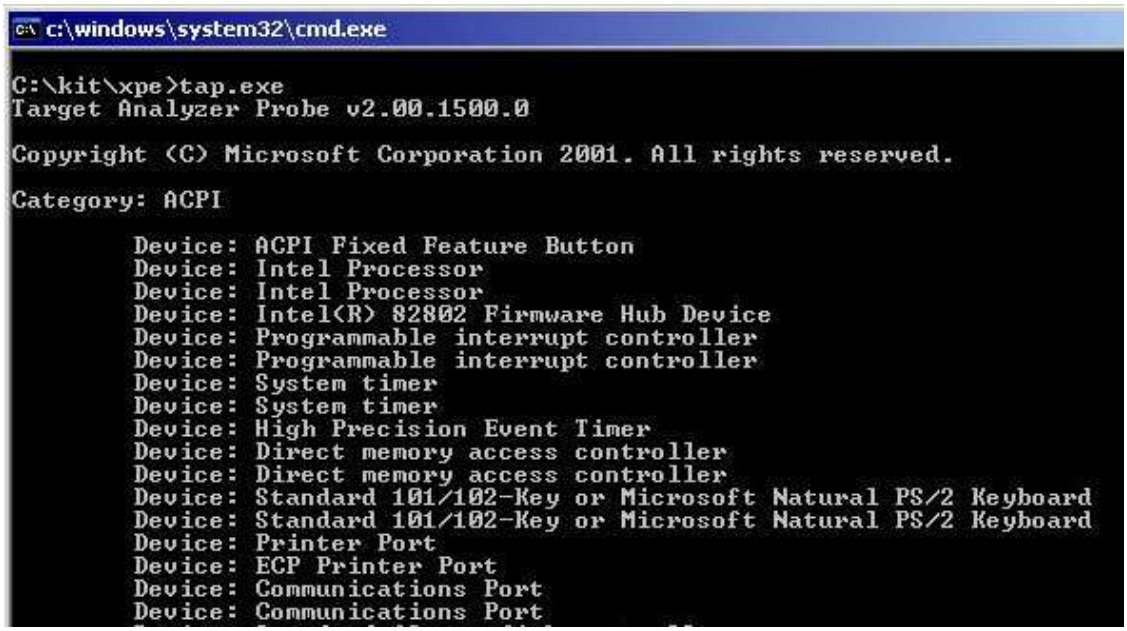

FIGURE 2. Target Analyzer Probe Execution

To proceed with the creation of a component for the motherboard, the Component Designer tool will be used to import the modified PMQ file from the analysis phase. It is advisable to log the import process for later error investigation and verification. Component Designer creates a basic component with dependencies on all hardware devices from the previously created PMQ file. Any devices without a matching compatible component will be listed as errors or warnings at the conclusion of the import process. The motherboard component is completed by assigning correct description data (e.g. device name, manufacturer, author name, etc.), and applying the *Selector Prototype Component* as a prototype, as seen in FIGURE 3. The Selector Prototype Component allows the individual components of the motherboard to be enabled

or disabled during the Target Designer implementation, allowing for a more flexible building block.

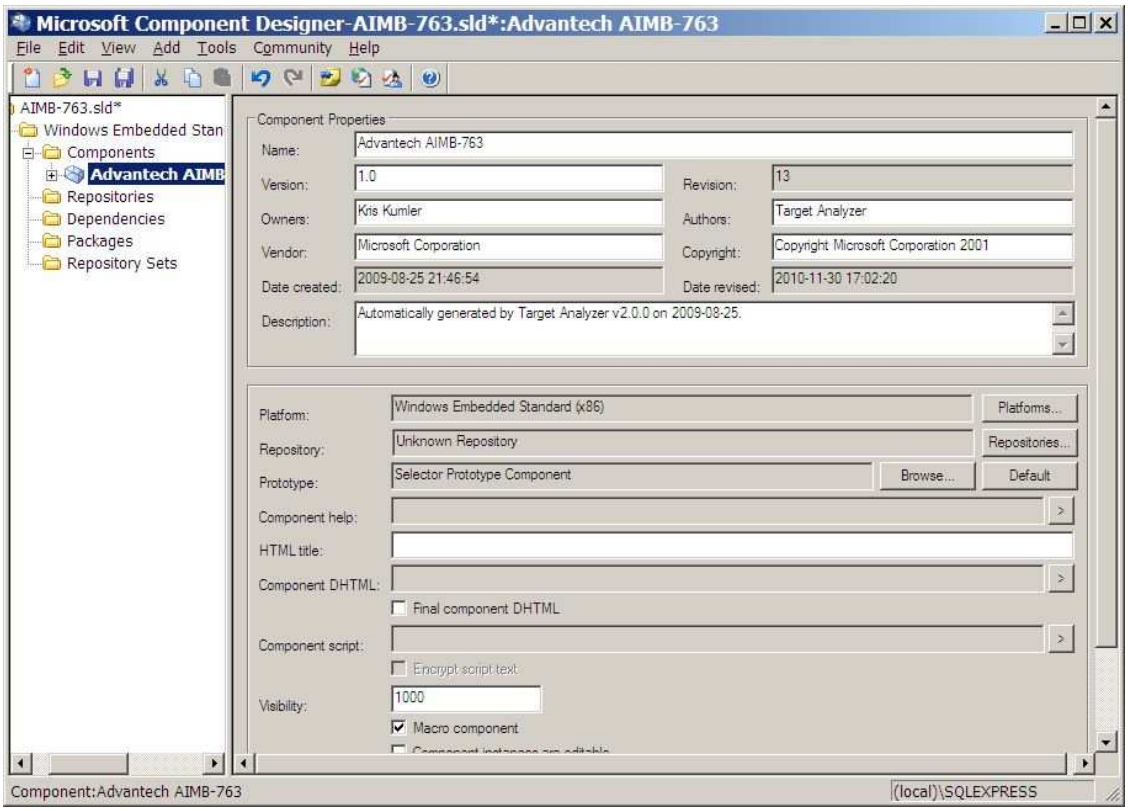

FIGURE 3. Selector Prototype Used for Macro Component

The final step of creating the component, as with all components, is to make it available for configurations by its import into the Component Database via the Microsoft Component Database Manager, available via the Tools menu in Component Designer, as seen in FIGURE 4. Importing the component adds it to the component database and, when applicable, copies specified files into the server repository.

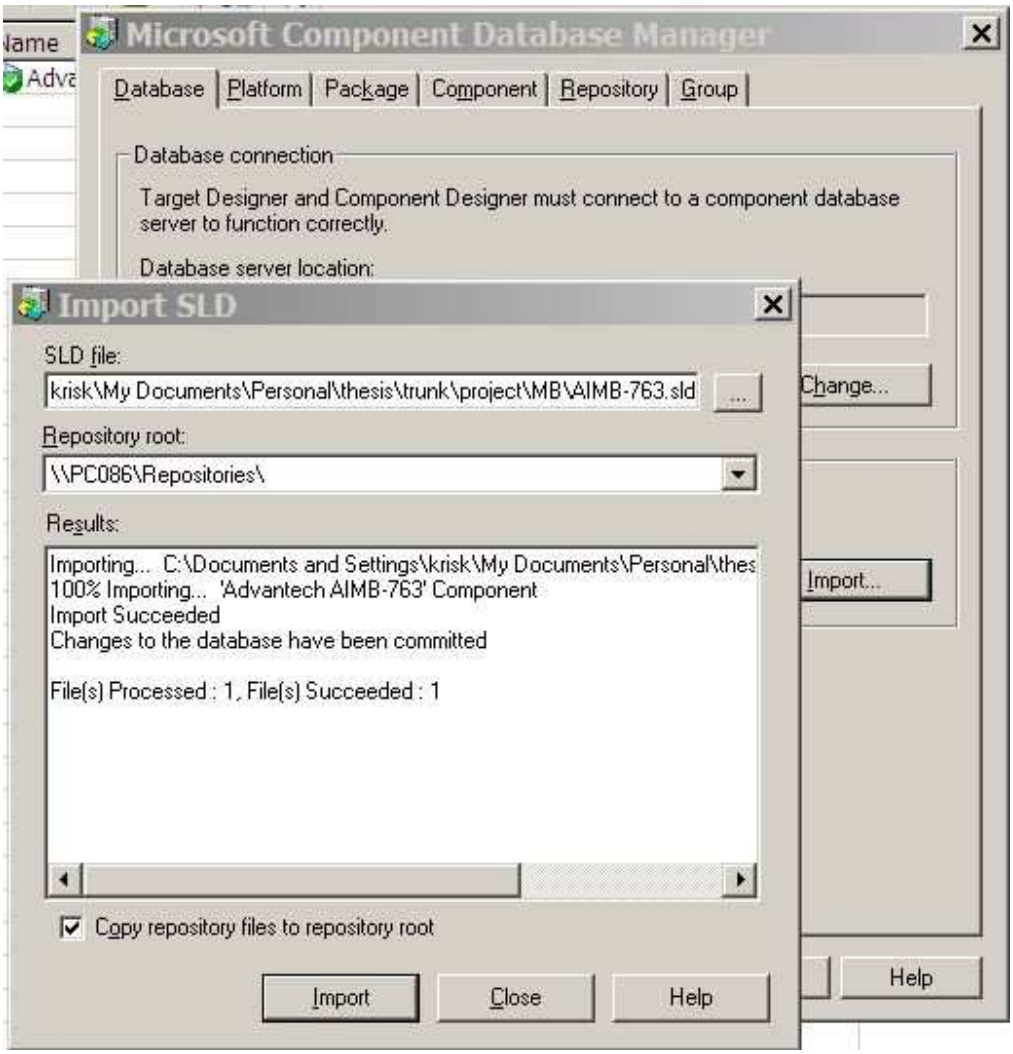

FIGURE 4. Component Database Manager Import

B. Device Drivers

The drivers for the video capture card consist of two sets of drivers, one for the audio portion and one for the video portion, including driver files (*.SYS files) and their associated information files (*.INF files). Retaining the distribution driver files in a unique directory is a good practice for maintainability. The

import of the INF file into Component Designer is straightforward. Some drivers include definitions for multiple PNP Device ID entries when this is not necessary, such as when all files, services, and registry keys are otherwise identical. To correct this, the PNP Device ID entries may be copied from the *Resources* entry in the component design. This has been done, as can be seen in FIGURE 5, for the video capture card component, which includes "A," "B," "C," and "D" entries for multiple devices presented to the PCI subsystem from a single card.

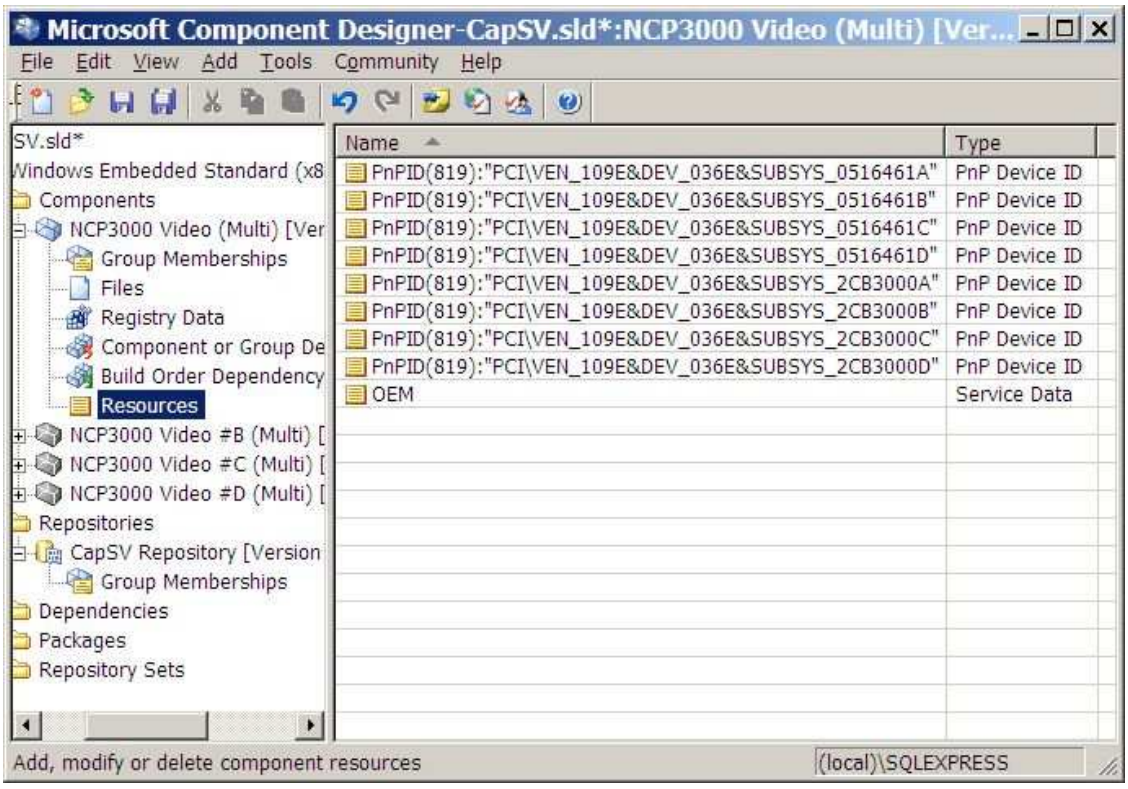

FIGURE 5. Video Capture Device Resources

One important aspect is creating a Repository entry for the driver files. This repository is associated with the physical directory on the development system containing the driver files, and it is re-used for any device driver definitions

requiring the same set of files. For the purposes of the video capture card audio drivers (CapAM), the file directory and repository is located in a relative directory of CapAM/files. The repository is defined under the *Repositories* tree in the component configuration file (sld) and referenced from each component in the repository field of the component properties (FIGURE 6). The component file is saved into a separate directory at CapAM/sld and imported into the Component Database. The same process is repeated for the video capture card video drivers (CapSV).

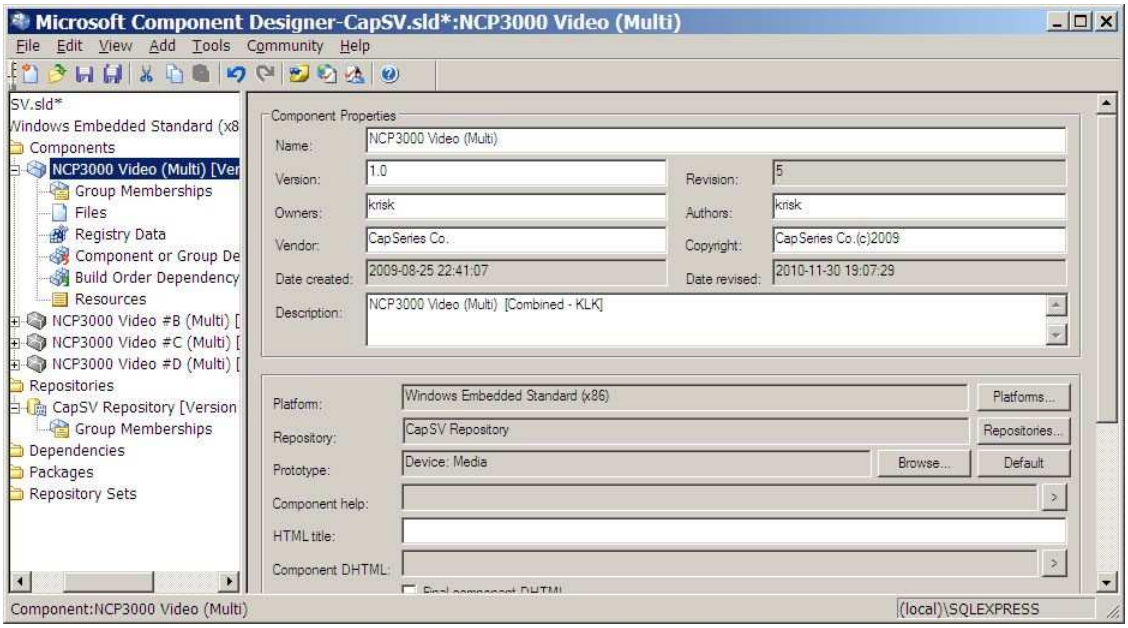

FIGURE 6. Component Repository Selection

The custom Image Vault I/O Card utilizes a generic Windows kernel driver, provided by Jungo Systems' WinDriver product. The driver INF import was quite seamless, especially considering the age of the driver implementation.

The driver was recognized as a non-Plug and Play device but no errors manifested during component creation.

C. Host Application

The dependency analysis for the host application (IVRecord) is broken into two distinct portions: software specifications, provided by the software developers; and run-time execution observation. A good software specification will provide most, if not all, direct dependencies on shared libraries or other system resources. However, there may be dependencies overlooked by the developers or hidden by the development tools themselves, but which can be revealed during run-time execution.

The dependency specification for the Recorder application may be found in APPENDIX II. The specification includes both files that may be considered dependencies but included in the deployment package as well as known Operating System library dependencies. Any additional dependencies required for associated utilities should be included in the specification.

The run-time execution for the Recorder application will be observed through the Dependency Walker application. It is necessary to fully exercise the application to expose all modules that are loaded later in the program execution. The resultant listing, as partially seen in FIGURE 7, of loaded modules for the application is rather large and must be manually analyzed afterward (see

APPENDIX III). Some amount of experimentation or research may be needed to separate actual dependencies from modules that the Operating System, or software components, may load simply if they are present in the file system at execution time. Any further dependencies will easily be revealed during testing of the complete Windows Embedded image. The inclusion, as dependencies, of standard Operating System components that will be included in nearly every configuration can be a desirable best practice. With the great variability and future upgrade paths of Windows Embedded products, this will ensure no issues at upgrade points, or with other developers.

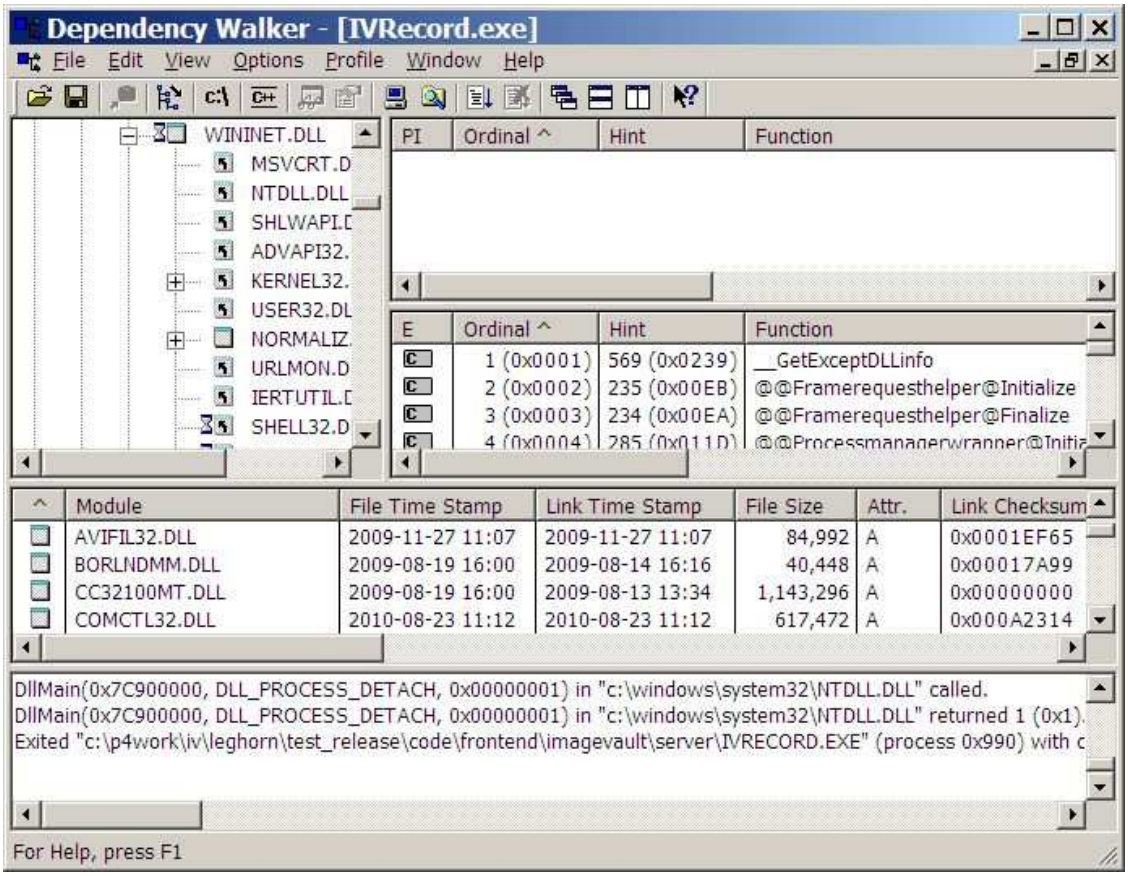

FIGURE 7. Dependency Walker Analysis

The implementation of the IVRecord component begins with the file repository and the addition of the package files to the 'Files' section of the component. The Operating System component dependencies are added to the 'Component or Group Dependency' section of the component, as seen in FIGURE 8. Multiple raw file dependencies may be included in the same system component.

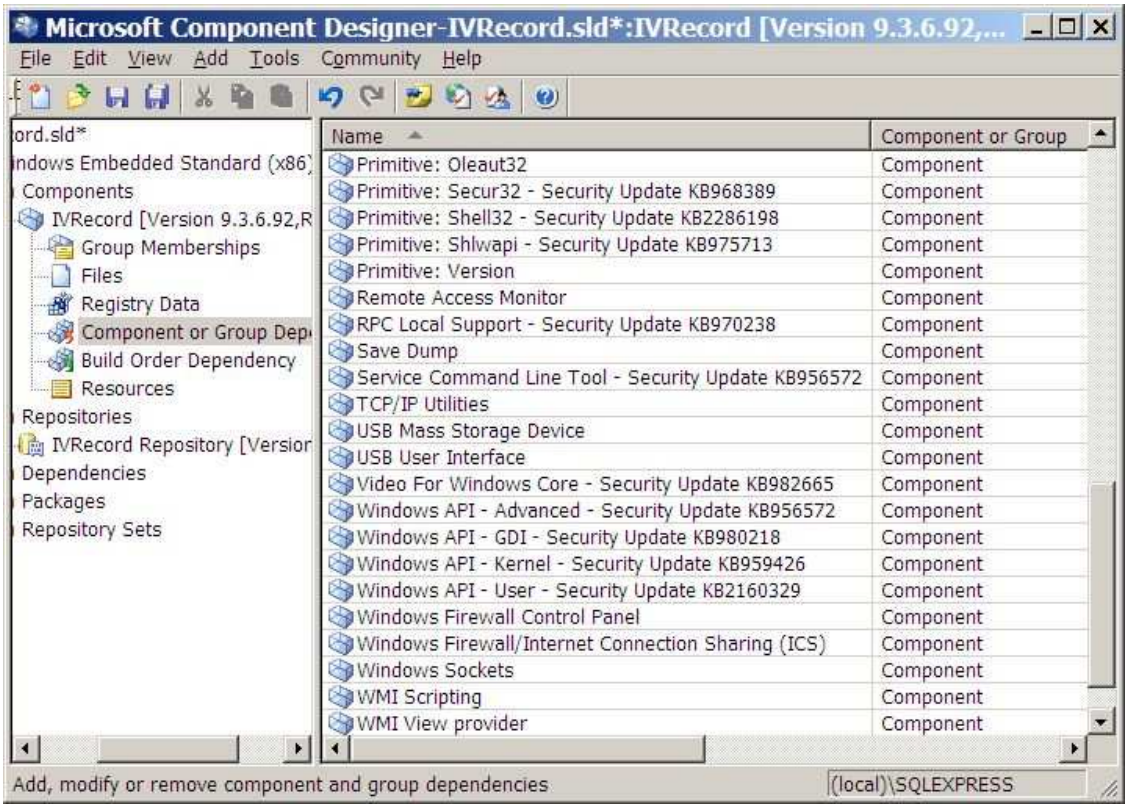

FIGURE 8. Recorder Software Component Dependencies

Further exploration of dependencies will reveal system level components that should be used instead of dependencies on primitive libraries. For example, instead of directly depending on the WinSock 2 DLL, WS2_32.dll, having a

dependency on the system component “Windows Sockets” would usually be preferable. The full steps undertaken to create the IVRecord component may be found in APPENDIX IV.

D. AutoUpdate Application

The maintenance of the deployed device is a serious issue to be considered. There is an impact on security, customer service, and software upgrades with the maintenance system. Image Vault’s aforementioned AutoUpdate program is able to add and modify files on the device, execute programs, and make registry modifications. All actions are specified in proprietary XML document files for which AutoUpdate will search and process upon execution. The creation of the AutoUpdate component specification is beyond the scope of this treatment. However, the final image will be required to execute the AutoUpdate application on startup and treat it as the first shell of the Windows system. The steps required to create the AutoUpdate component may be found in APPENDIX V.

E. Image Configuration

The configuration of the target image will begin with the basic components already built from the analyzed hardware devices and software applications, overall configuration settings, and dependency checks. The original, minimal set of components included in the saved configuration should be maintained in that state. Maintaining this minimal set separate from the later configuration which includes all dependencies will allow for longer-term usage of the image configuration while retaining a smaller set of components. The full detail of the steps may be found in APPENDIX VI. For this study, the Product Identification Key (PID) field, as seen in FIGURE 9, which activates the run-time license of the image, is left blank and will therefore create a limited-time evaluation version of the run-time image.

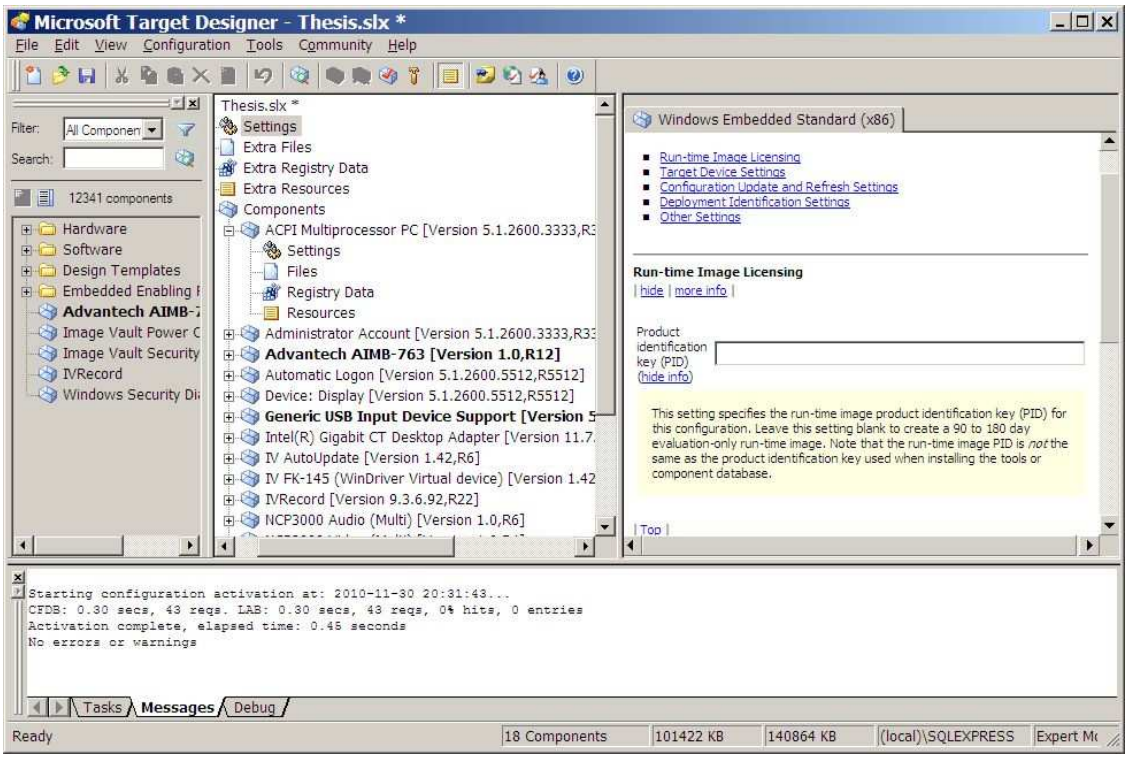

FIGURE 9. Configuration Settings – Run-time Image Licensing

Several base components are added to the base configuration for further settings customization. These are components that would have been added via dependencies otherwise, but only with default settings. The "Administrator Account" and "Automatic Logon" components are used to create an account and configure the system to logon without user interaction. Additionally, the customization of the "Windows Logon (Standard)" component prevents many of the graphical logon dialogs from being presented. The "Windows subsystem" component is customized, via the ErrorMode registry value, to suppress system messages (e.g. low virtual memory errors") and instead record the errors in the system event log.

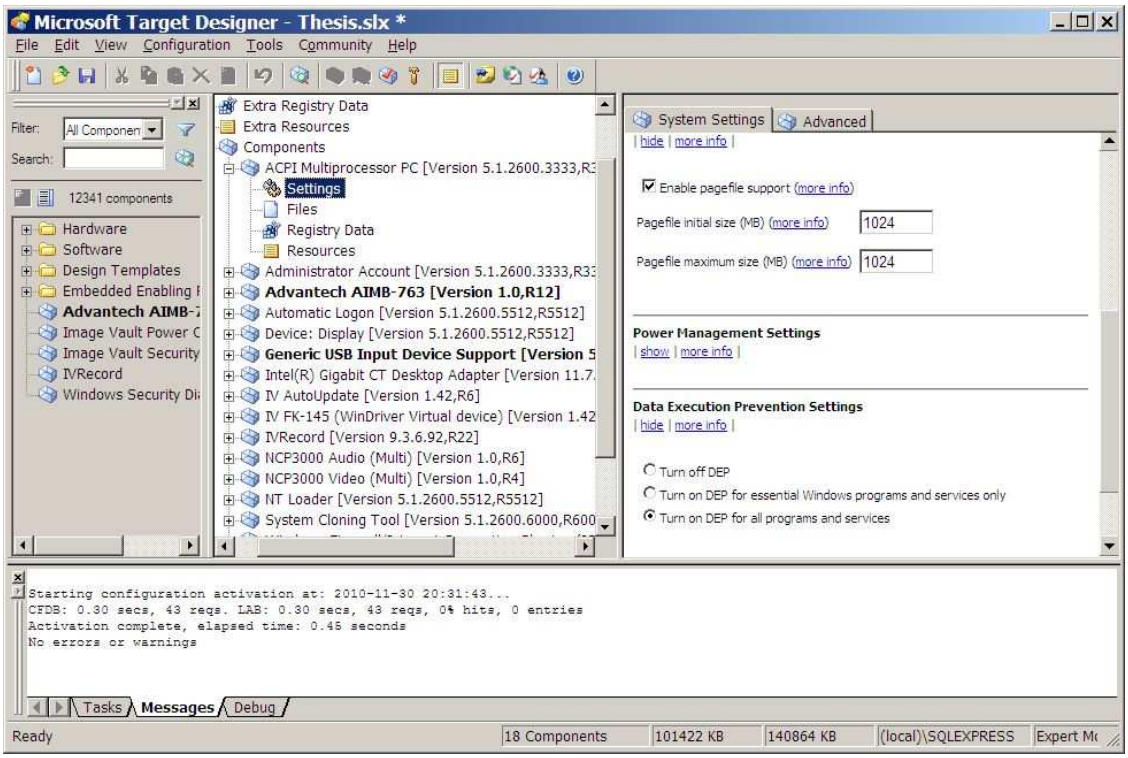

FIGURE 10. ACPI Multiprocessor PC Configuration

The “ACPI Multiprocessor PC” component is customized to enable a system paging file and, regarding security, forcing Data Execution Protection (DEP) for all processes (see FIGURE 10). Further security-related, the “Windows Firewall/Internet Connection Sharing (ICS)” component as seen in FIGURE 11 allows the firewall exception for the host application and prevents the notification that may allow unauthorized access for other programs.

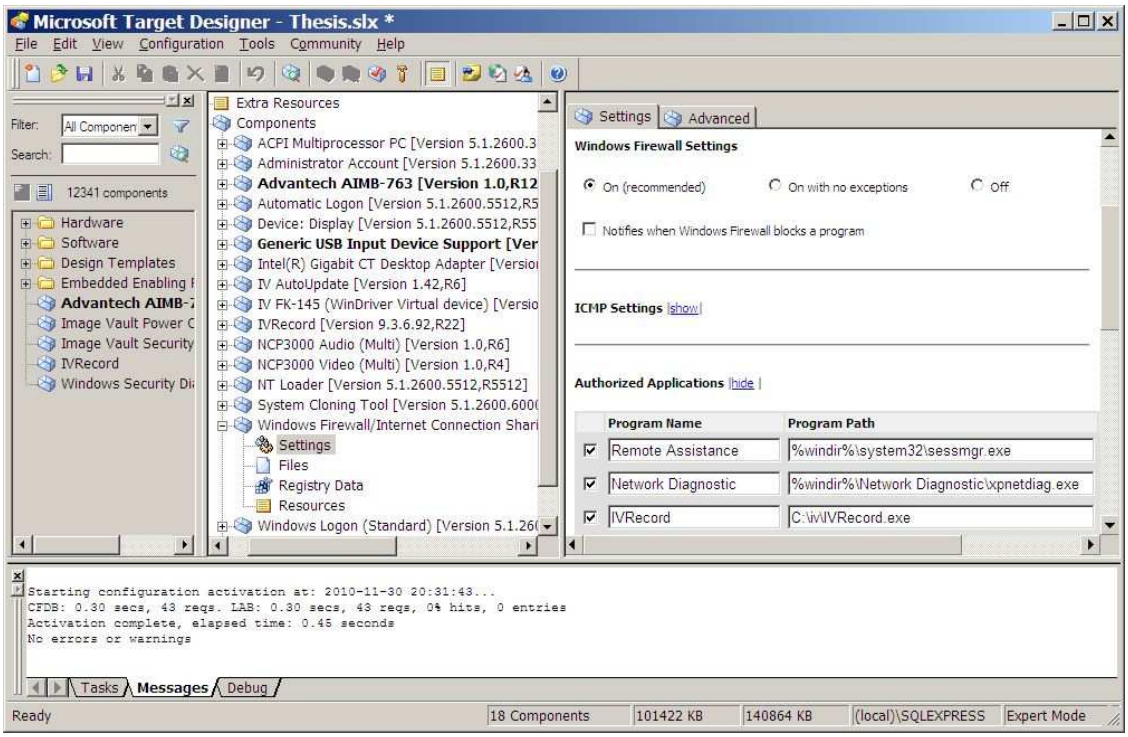

FIGURE 11. Windows Firewall Configuration

The System Cloning Tool is a very important feature of the Windows Embedded system and care should be taken to configure properly.. The tool is used during the manufacturing process to ensure that each target device possesses a unique computer security identifier (SID) and name, which is required on Windows-based networks. This device customization is performed both during the reseal phase, as the master device is configured, and primarily during the cloning phase, which takes place on each target device as it boots the first time. Disabling the reseal option “Remove AutoLogon Settings” keeps intact the Automatic Logon settings described earlier.

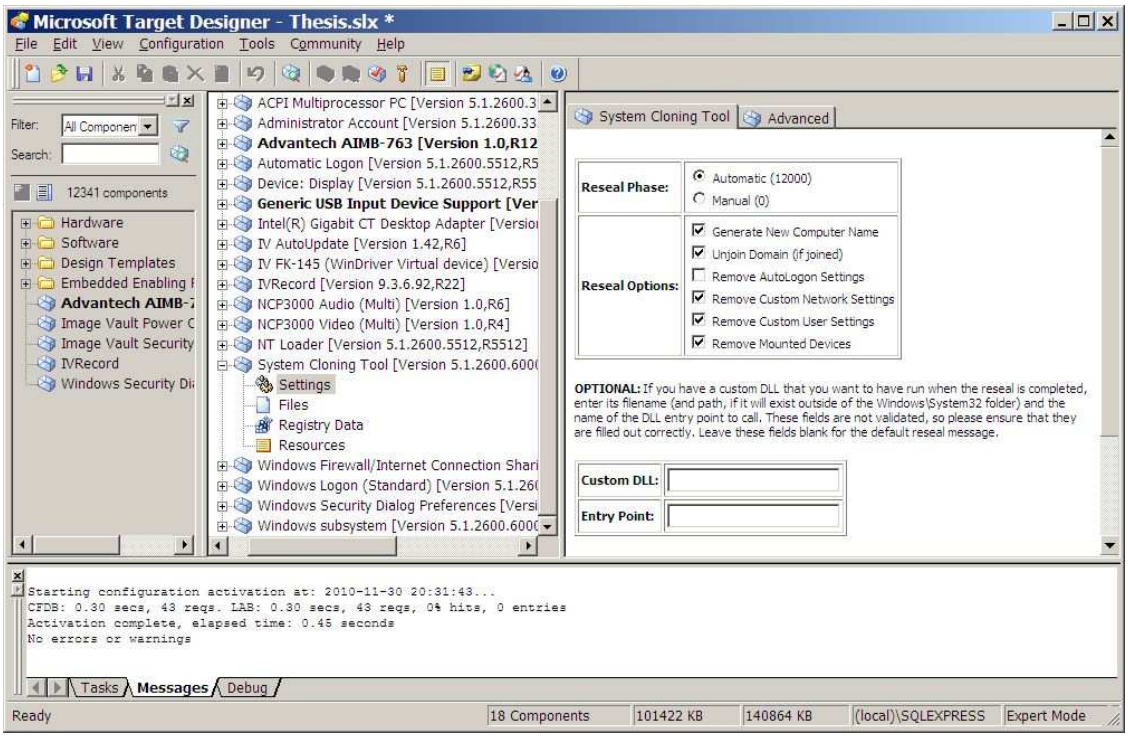

FIGURE 12. System Cloning Tool Configuration

The next major steps in the image configuration process are the dependency check and executing the image build command. For the dependency check, Target Designer will examine each component already in the base configuration and add all dependent components, as can be seen in FIGURE 13. After the dependency check completes without error, the image should be in a component-complete state. Building the target image can be immediately executed. The target image will be generated and placed in the desired directory. It is desirable to save this completed, built image configuration as a separate file (e.g. appending "-built" to the filename) for system configuration management.

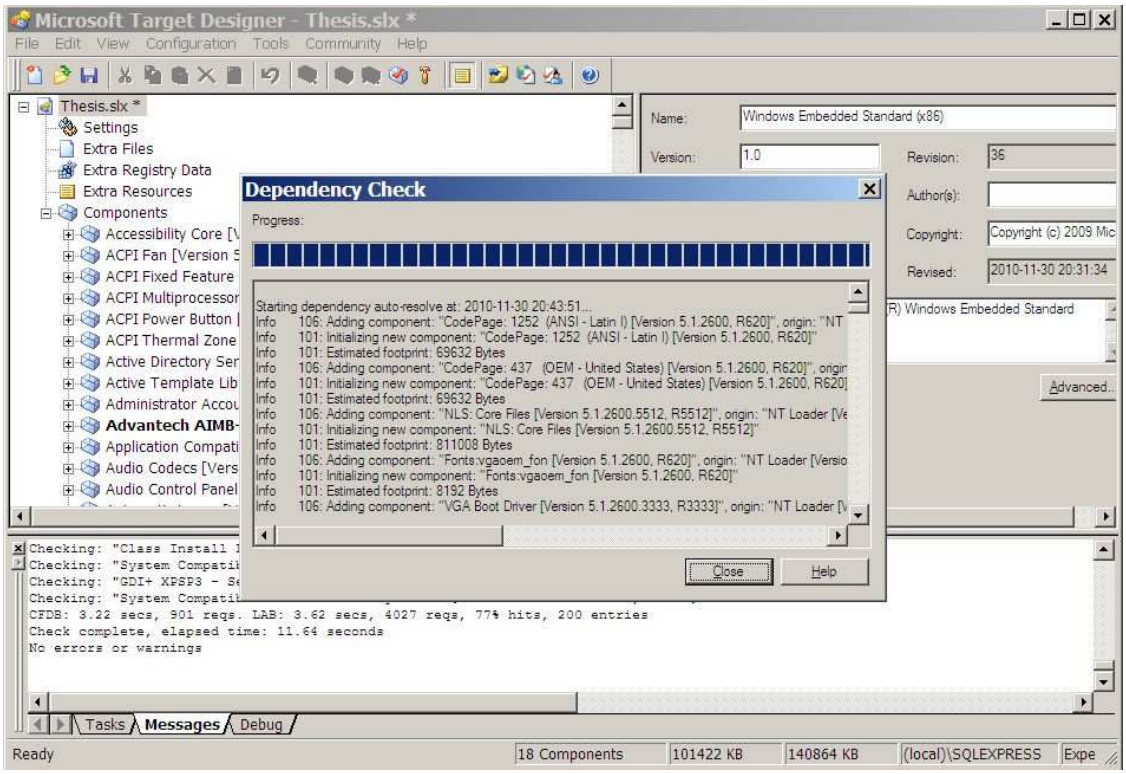

FIGURE 13. Configuration Dependency Check

To assist image maintenance, it is important to identify image configurations after deployment. This is possible through the use of the Deployment Identification Settings in the image configuration settings tree. The device model information is a GUID format string which may be used for this purpose. It may only be necessary to utilize a few characters of this string for identification, reserving the others for a future use. In a nearby field, a Runtime OEM revision may be set, assisting in the deployment of patches and service packs to devices. [14]

F. Target Image Preparation

The target image is prepared by allowing the First Boot Agent (FBA) to execute and reseal on the target device hardware. For this device, the target image is copied to a clean disk drive and placed into the target device. After powering on, the FBA executes and will take a non-trivial amount of time to complete. The FBA goes through the process of adding all hardware devices found to the Windows Device Manager, installing PNP devices (see FIGURE 14), activating drivers, running specified actions, etc.

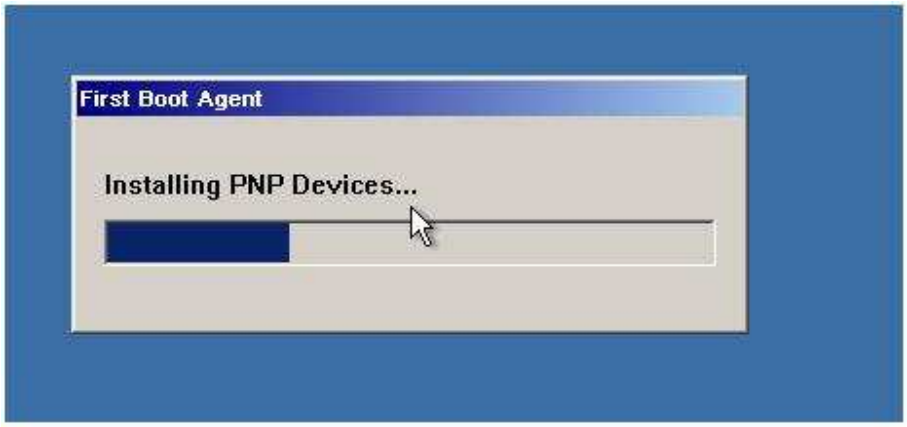

FIGURE 14. First Boot Agent Executing

Once the First Boot Agent has resealed the unit, a dialog will be presented notifying the developer (see FIGURE 15). After powering down the target device, the disk drive is removed, and all files copied and stored as the "master image." This master image is ready to be deployed onto numerous target devices in production.

FIGURE 15. First Boot Agent Complete

The first boot on a production system begins with a momentary delay as a short First Boot Agent process initializes the unique hardware devices (differing addresses, serial numbers, etc.). This delay does not occur again on the same instance of hardware after deployment. The system will boot immediately and proceed to run the specified AutoUpdate and Recorder applications, as seen in FIGURE 16, with little indication of the underlying Operating System.

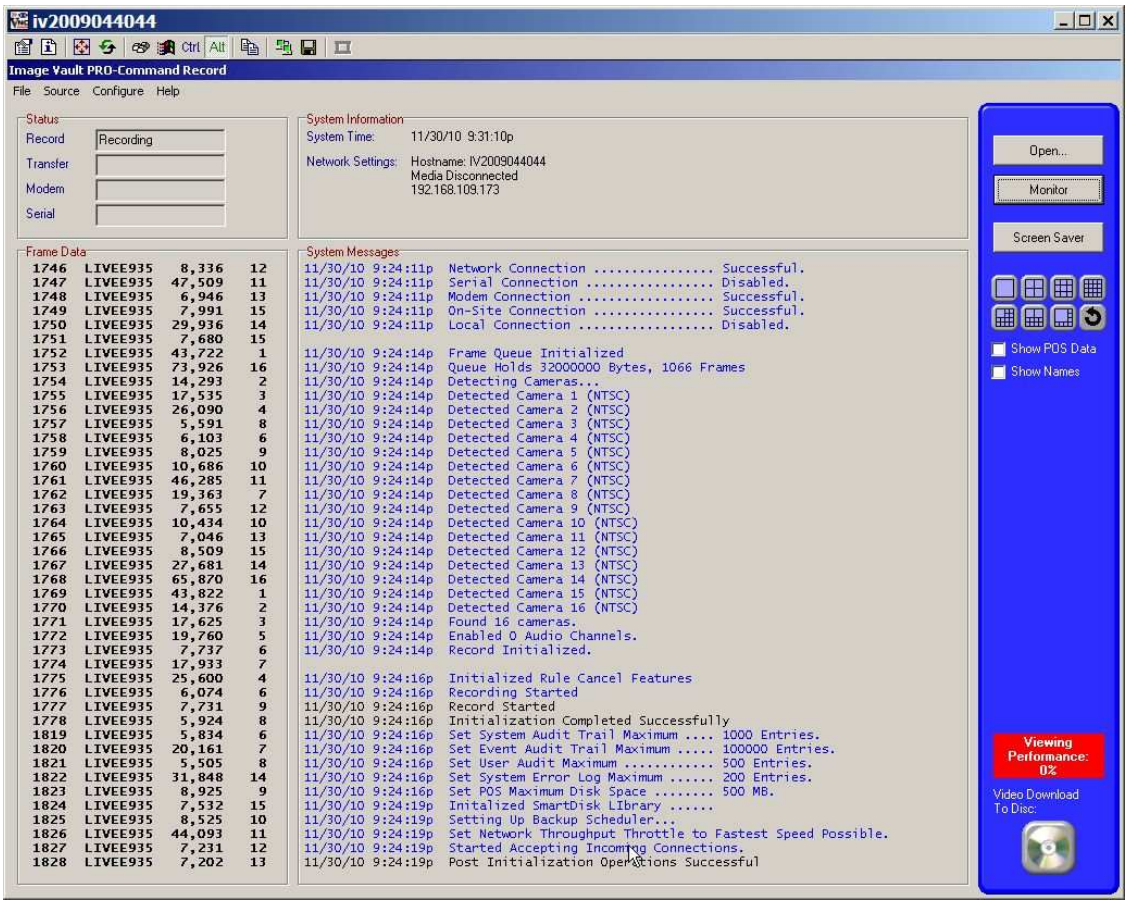

FIGURE 16. Final System Running Image Vault's Recorder Application

V. DISCUSSION OF RESULTS

The final image generated from the configuration developed will be applied to ongoing development and production of shipping product at Image Vault. The requirements set forth in this thesis have been met through the resolutions as listed in TABLE IV.

TABLE IV.
REQUIREMENT RESOLUTIONS

Requirement	Reference	Resolution
Hosted software application	I.	Analyzed, all dependencies met
Future maintenance	I., 2.4.2	Available between proprietary AutoUpdate, OS support functions
Configuration best practices	I.	Componentization and image configuration practices documented
Hardware testing	II. A.	Reduced OS software variables for better isolation
System reliability	II. A.	Fewer running drivers, system services, supporting services, and applications
System security	II. B.	Removal of network and software execution vectors
OS footprint	2.4.1	Complete image size 252 MB + optional 1024 MB page file.

The OS footprint represents both a reduction in the Operating System file size, as well as an increase in system functionality from the previous configuration. This image configuration creates a fixed page file on the deployed system during operation. For this reason, the deployment is limited to hard disk

drives (a given requirement). To enable use on a flash drive, the page file would be disabled and file-based protection filters utilized.

Integrated hardware component testing will be eased with reduced variables from software overhead. As well, system reliability is far greater; with no direct Operating System faults recorded to date (all recorded faults are due to hardware or application software failure). The application software is more reliable with fewer services and applications interfering.

Numerous best practices are expressed through component design and image configuration design. Utilizing a minimal, non-dependency checked image configuration allows for a quick turnaround for changing hardware requirements while controlling image footprint growth. As well, it allows for easier maintenance of image configuration revisions, along with easier utilization within version control systems. The built-in macro components are a design tool that should be frequently used to ensure maximum reusability of created components.

VI. CONCLUSIONS

The usage of Windows Embedded Standard product for the Operating System on Image Vault Recorders has led to increased system reliability and security. The streamlined installation of the Recorder software in the image will allow for easier creation of systems in the unit manufacturing process. The serviceability of deployed systems allows for the ongoing usage of systems without unit returns for Operating System or software upgrades. All of the requirements from stakeholders in the production development process have been met in this configuration.

The best practices set forth in this thesis allow for enhanced developer productivity, image configuration maintenance, and configuration control. The advantages become evident through the production lifetime and are characterized by a consistent footprint and feature set.

VII. RECOMMENDATIONS

Microsoft released the next version of desktop-class embedded Operating System, Windows Embedded Standard 7, during the course of this research. Both Windows Embedded Standard 2009 and Windows Embedded Standard 7 will be offered for sale; the increased options to the developer are beneficial. Many developers will choose the Operating System product based upon existing application design as their first criterion.

Exploration of Windows Embedded Standard 7 (WES7) is worthwhile, but the product is significantly different from Windows Embedded Standard 2009. Foremost, the larger granularity available in WES7 provides lesser control over system components – WES7 includes approximately 150 OS feature sets to WES 2009's approximately 1000 components. Driver granularity is also decreased, with WES7 including approximately 500 driver packages to WES 2009's approximately 9000 driver components. The minimum image footprint changes accordingly, with the smallest possible image in WES7 to be ~500 MB versus WES2009's ~40 MB. The remote and optical boot options available in WES 2009 (remote, PXE, and CD/DVD) are not available in WES7. [19]

Windows Embedded Standard 7 does introduce numerous features previously unavailable in Microsoft's desktop embedded offerings, beginning with 64-bit (x64) processor support. Serviceability options are also expanded, allowing for the use of automatic servicing via Microsoft Windows Update in

addition to the OEM and manual device servicing functions. Security options are greatly increased through full certified IPv6 support, AppLocker (access control for allowed applications), and BitLocker (full disk encryption). The Dialog Filter feature of WES7 allows for tight control of dialog messages with customized responses, as well as expandable custom actions through software. [19]

For the developer productivity, Windows Embedded Standard 7 offers one very notable feature, for the automatic updating of the development tools via the Windows Embedded Developer Update tool. "Windows Embedded Developer Update automatically checks the development environment to ensure developers have the most current product revisions and updates, categorizes the available downloads and readies the product updates for download by subscribers." [20] This tool will remove the disruption of twice-monthly updates to be considered and manually applied by the developer.

For both Windows Embedded Standard versions, a further exploration of the disk write filters available is worthwhile to explore security and reliability improvements. The File-Based Write Filter (FBWF) allows for maintaining a stateless disk during system execution and with intervening restarts.

APPENDIX I. TARGET ANALYSIS RESULTS

Advantech AIMB-763 Target Analysis Probe Devices

- ACPI Fan
- ACPI Fixed Feature Button
- ACPI Multiprocessor PC
- ACPI Power Button
- ACPI Thermal Zone
- CD-ROM Drive
- Communications Port
- Direct memory access controller
- Disk drive
- Intel Processor
- Intel® 82801 PCI Bridge – 244E
- Intel® 82801G (ICH7 Family) PCI Express Root Port – 27D0
- Intel® 82801G (ICH7 Family) PCI Express Root Port – 27D2
- Intel® 82801G (ICH7 Family) SMBus Controller – 27DA
- Intel® 82801G (ICH7 Family) USB Universal Host Controller – 27C8
- Intel® 82801G (ICH7 Family) USB Universal Host Controller – 27C9
- Intel® 82801G (ICH7 Family) USB Universal Host Controller – 27CA
- Intel® 82801G (ICH7 Family) USB Universal Host Controller – 27CB
- Intel® 82801G (ICH7 Family) USB2 Enhanced Host Controller – 27CC
- Intel® 82801GB/GR (ICH7 Family) LPC Interface Controller – 27B8
- Intel® 82801GB/GR/GH (ICH7 Family) Serial ATA Storage Controller – 27C0
- Intel® 82802 Firmware Hub Device
- Intel® 82945G Express Chipset Family
- Intel® 945G/GZ/GC/P/PL Processor to I/O Controller – 2770
- Intel® PRO/1000 PL Network Connection
- ISAPNP Read Data Port
- Logical Disk Manager
- Microsoft ACPI-Compliant System
- Microsoft UAA Bus Driver for High Definition Audio
- Motherboard resources
- Numeric data processor
- PCI bus
- Plug and Play Software Device Enumerator
- Primary IDE Channel
- Printer Port
- Printer Port Logical Interface
- Programmable interrupt controller
- PS/2 Compatible Mouse

- Secondary IDE Channel
- RAS Async Adapter
- Standard 101/102-Key or Microsoft Natural PS/2 Keyboard
- System board
- System CMOS/real time clock
- System speaker
- System timer
- USB Mass Storage Device
- USB Root Hub
- Volume Manager

Advantech AIMB-763 Removed Devices

- RAS Async Adapter

APPENDIX II. RECORDER APPLICATION DEPENDENCY SPECIFICATION

This listing represents internal engineering specifications as to the dependent files needed for the Recorder software application in the deployed image.

Software Package Contents:

```
IVRecord.exe
borlndmm.dll [vendor]
cc3250mt.dll [vendor]
DUNZIP32.DLL [vendor]
Trace_Mgr.dll
dlls\Audio_Mgr.dll
dlls\Audit_Mgr.dll
dlls\CapAM.dll [vendor]
dlls\CapSeries.dll [vendor]
dlls\Capture_Mgr.dll
dlls\Catalog_Mgr.dll
dlls\Config_Mgr.dll
dlls\Connect_Mgr.dll
dlls\Daughter_Card_Mgr.dll
dlls\ECPSV.dll [vendor]
dlls\ErrorLog_Mgr.dll
dlls\IVPOS_Mgr.dll
dlls\lame_enc.dll [vendor]
dlls\POS_Mgr.dll
dlls\PreEvent_Mgr.dll
dlls\Process_Mgr.dll
dlls\Rem_Disk_Mgr.dll
dlls\SmartDisk.dll
dlls\tmp4core.dll [vendor]
dlls\UserAudit_Mgr.dll
dlls\UUCore.dll [vendor]
```

Non-package file dependencies:

```
iphlpapi.dll [Microsoft]
setupapi.dll [Microsoft]
user32.dll [Microsoft]
cmd.exe [Microsoft]
```

Non-package Operating System component dependencies:

```
Windows Management Instrumentation (core functionality)
```

APPENDIX III. RECORDER APPLICATION ANALYZED DEPENDENCIES

The execution of Dependency Walker [16] is:

1. Execute depends.exe
2. File->Open..., select IVRecord.exe (see FIGURE 17)
3. Profile->Start Profiling...
4. Execute OK button
5. Exercise application functionality
6. Exit application
7. Analyze tool output (see FIGURE 7)

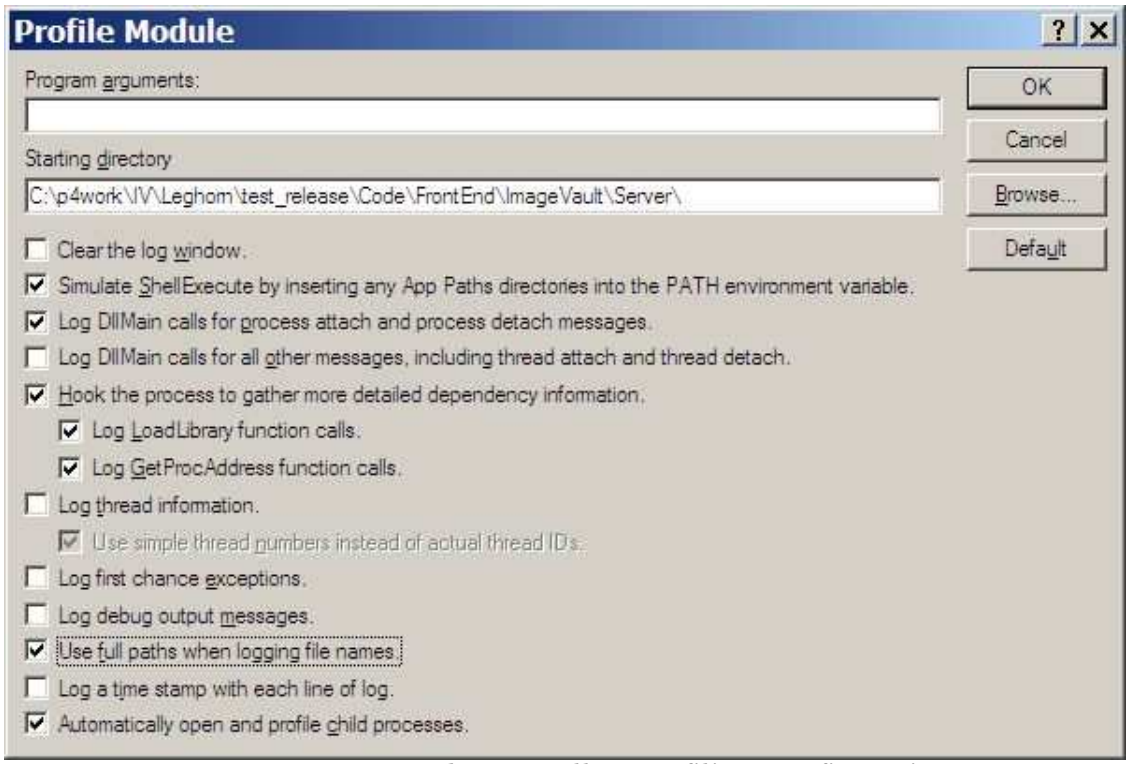

FIGURE 17. Dependency Walker Profiling Configuration

The analysis of the Dependency Walker output is primarily a matter of identifying into which category each module falls. The first column of the module list view seen in FIGURE 7 identifies the classification. A detailed text listing for each classification follows.

Category: Loaded / warning
KERNEL32.DLL
CAPSERIES.DLL
MIDIMAP.DLL
MPR.DLL

Category: Normal
ADVAPI32.DLL
AVIFIL32.DLL
BORLNDMM.DLL
CC3250MT.DLL
COMCTL32.DLL
COMDLG32.DLL
DCIMAN32.DLL
DDRAW.DLL
DUNZIP32.DLL
GDI32.DLL
GLU32.DLL
IPHLPAPI.DLL
IVRECORD.EXE
MSACM32.DLL
MSVCRT.DLL
MSVFW32.DLL
NTDLL.DLL
OLE32.DLL
OLEAUT32.DLL
OPENGL32.DLL
RPCRT4.DLL
SECUR32.DLL
SHELL32.DLL
SHLWAPI.DLL
USER32.DLL
VERSION.DLL
WINMM.DLL
WINSPOOL.DRV
WS2_32.DLL
WS2HELP.DLL
WSOCK32.DLL

Category: Marked as delay-load dependency
ADVPACK.DLL
APPHELP.DLL
AUTHZ.DLL
BROWSEUI.DLL
CABINET.DLL

```
CDFVIEW.DLL
CERTCLI.DLL
CFGMGR32.DLL
CLUSAPI.DLL
CREDUI.DLL
CRYPTDLL.DLL
CRYPTUI.DLL
CSCDLL.DLL
DBGHELP.DLL
DEVMGR.DLL
DHCPCSVC.DLL
DOT3API.DLL
DOT3DLG.DLL
DUSER.DLL
EAPOLQEC.DLL
EAPPCFG.DLL
EAPPPRXY.DLL
EFSADU.DLL
ESENT.DLL
GDIPLUS.DLL
HLINK.DLL
IMGUTIL.DLL
IMM32.DLL
INETCOMM.DLL
LINKINFO.DLL
LSASRV.DLL
LZ32.DLL
MFC42U.DLL
MLANG.DLL
MOBSYNC.DLL
MPRUI.DLL
MSGINA.DLL
MSHTML.DLL
MSI.DLL
MSIMG32.DLL
MSLS31.DLL
MSOERT2.DLL
MSRATING.DLL
MSSIGN32.DLL
MSVCP60.DLL
NETCFGX.DLL
NETMAN.DLL
NETPLWIZ.DLL
NETRAP.DLL
NETSHELL.DLL
```

NETUI0.DLL
NETUI1.DLL
NETUI2.DLL
NTDSAPI.DLL
NTLANMAN.DLL
ODBC32.DLL
OLEACC.DLL
OLEDLG.DLL
OLEPRO32.DLL
ONEX.DLL
PAUTOENR.DLL
POWRPROF.DLL
PRINTUI.DLL
PSAPI.DLL
QUERY.DLL
QUTIL.DLL
RASDLG.DLL
REGAPI.DLL
SAMSRV.DLL
SCECLI.DLL
SHDOCVW.DLL
SHSVCS.DLL
URLMON.DLL
USP10.DLL
UTILDLL.DLL
UXTHEME.DLL
W32TOPL.DLL
WINHTTP.DLL
WININET.DLL
WINSCARD.DLL
WINSTA.DLL
WMI.DLL
WTSAPI32.DLL
WZCDLG.DLL
WZCSAPI.DLL
WZCSVC.DLL

Category: Dynamically Loaded
ACTIVEDS.DLL
ADSLDPC.DLL
ATL.DLL
AUDIO_MGR.DLL
AUDIT_MGR.DLL
CAPAM.DLL
CAPTURE_MGR.DLL

```
CATALOG_MGR.DLL
COMCTL32.DLL
CONFIG_MGR.DLL
CONNECT_MGR.DLL
CRYPT32.DLL
DAUGHTER_CARD_MGR.DLL
DIGEST.DLL
DNSAPI.DLL
DSOUND.DLL
ECPSV.DLL
ERRORLOG_MGR.DLL
HNETCFG.DLL
IMAGEHLP.DLL
IVPOS_MGR.DLL
KSUSER.DLL
LAME_ENC.DLL
MPRAPI.DLL
MSACM32.DRV
MSAPSSPC.DLL
MSASN1.DLL
MSNSSPC.DLL
MSV1_0.DLL
MSVCRT40.DLL
MSWSOCK.DLL
NETAPI32.DLL
POS_MGR.DLL
PREEVENT_MGR.DLL
PROCESS_MGR.DLL
RASADHLP.DLL
RASAPI32.DLL
RASMAN.DLL
REM_DISK_MGR.DLL
RICHED20.DLL
RICHED32.DLL
RTUTILS.DLL
SAMLIB.DLL
SCHANNEL.DLL
SETUPAPI.DLL
TAPI32.DLL
TMP4CORE.DLL
TRACE_MGR.DLL
USERAUDIT_MGR.DLL
USERENV.DLL
WDMAUD.DRV
WINRNR.DLL
```

```
WINTRUST.DLL
WLDAP32.DLL
WSHTCPIP.DLL
ES1371MP.SYS
UUCORE.DLL
```

APPENDIX IV. IVRECORD COMPONENT CREATION

The steps for creating the IVRecord component are as follows:

1. Create new file in Microsoft Component Designer
2. Add new Repository
    a. Name “IVRecord Repository”
    b. Set Source Path to “.\..\FILES”
3. Add new component
    a. Name “IVRecord” (see FIGURE 18)
    b. Set Version “9.3.6”
    c. Set Repository to “IVRecord Repository”
    d. Add Files from “Software Package Contents” in APPENDIX II.
    e. Add Component or Group Dependencies (see FIGURE 8):
        i. Audio Control Panel
        ii. CDFS
        iii. Client for Microsoft Networks
        iv. CMD – Windows Command Processor
        v. Common Control Libraries
        vi. Common File Dialogs
        vii. Control Panel Command Line Support
        viii. Cryptographic Network Services
        ix. Device Manager

x. DirectSound

xi. Disk Management Command Line Utility

xii. Disk Management MMC Snap-In

xiii. Display Control Panel

xiv. Dr. Watson Debugger

xv. English Language Support

xvi. Error Reporting

xvii. FAT

xviii. FAT Format

xix. FAT/NTFS Common Format/Tools Files

xx. Internet Authentication Service (IAS) and Remote Access Common Files

xxi. Kernel Audio Support

xxii. Kernel Streaming User Mode Support

xxiii. Microsoft Visual C++ Run Time

xxiv. Microsoft WINMM WDM Audio Compatibility Driver

xxv. Misc. Command Line Tools

xxvi. Misc. File System Utilities

xxvii. Multi-Protocol Router Service Messages Library

xxviii. Network Command Shell

xxix. Network Command Shell Interface Context

xxx. Network Routing

xxxi. NTFS

xxxii. NTFS Format

xxxiii. NTFS Management Utility

xxxiv. OpenGL Support

xxxv. Primitive: Ntdll

xxxvi. Primitive: Ole32

xxxvii. Primitive: Oleaut32

xxxviii. Primitive: Secur32

xxxix. Primitive: Shell32

xl. Primitive: Shlwapi

xli. Primitive: Version

xlii. Realtek High Definition Audio

xliii. Remote Access Monitor

xliv. RPC Local Support

xlv. Save Dump

xlvi. Service Command Line Tool

xlvii. TCP/IP Utilities

xlviii. USB Mass Storage Device

xlix. USB User Interface

l. Video For Windows Core

li. Windows API – Advanced

lii. Windows API – GDI

liii. Windows API – Kernel

liv. Windows API – User

lv. Windows Firewall Control Panel

lvi. Windows Firewall/Internet Connection Sharing (ICS)

lvii. Windows Sockets

lviii. WMI Scripting

lix. WMI View provider

lx. WMI Win32 Provider

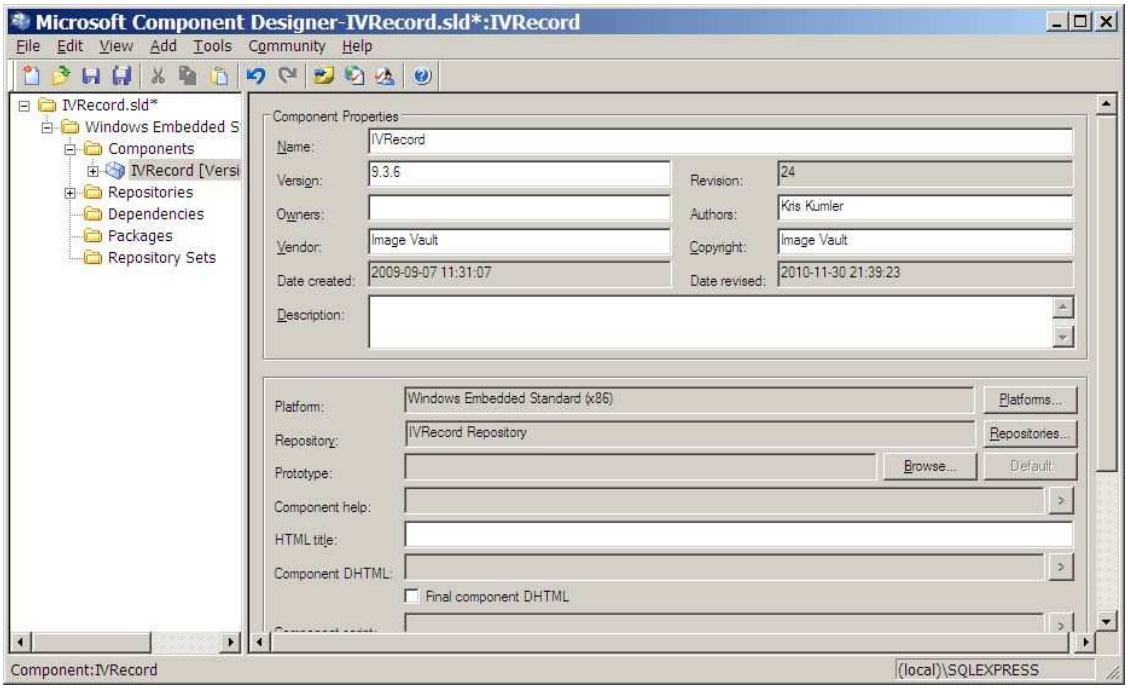

FIGURE 18. IVRecord Component Properties

APPENDIX V. AUTOUPDATE COMPONENT CREATION

1. Create new file in Microsoft Component Designer
2. Add new repository
    a. Name “AutoUpdate Repository”
    b. Set Source Path to “.\..\FILES”
3. Add new component (see FIGURE 19)
    a. Name “IV AutoUpdate”
    b. Set Repository to “AutoUpdate Repository”
    c. Set Prototype to “Shell prototype component”
    d. Edit Advanced Properties, adding extended property “cmiShellPath” as a String with value of “C:\iv\autoupdt.bat”
    e. Add Group Membership to “Shell” group.
    f. Add Group Membership to the “Software\System\User Interface\Shells” category.
    g. Add AutoUpdate files (AutoUpdate.exe, autoupdt.bat, autoupdt.ini), with destination path of “C:\iv\”

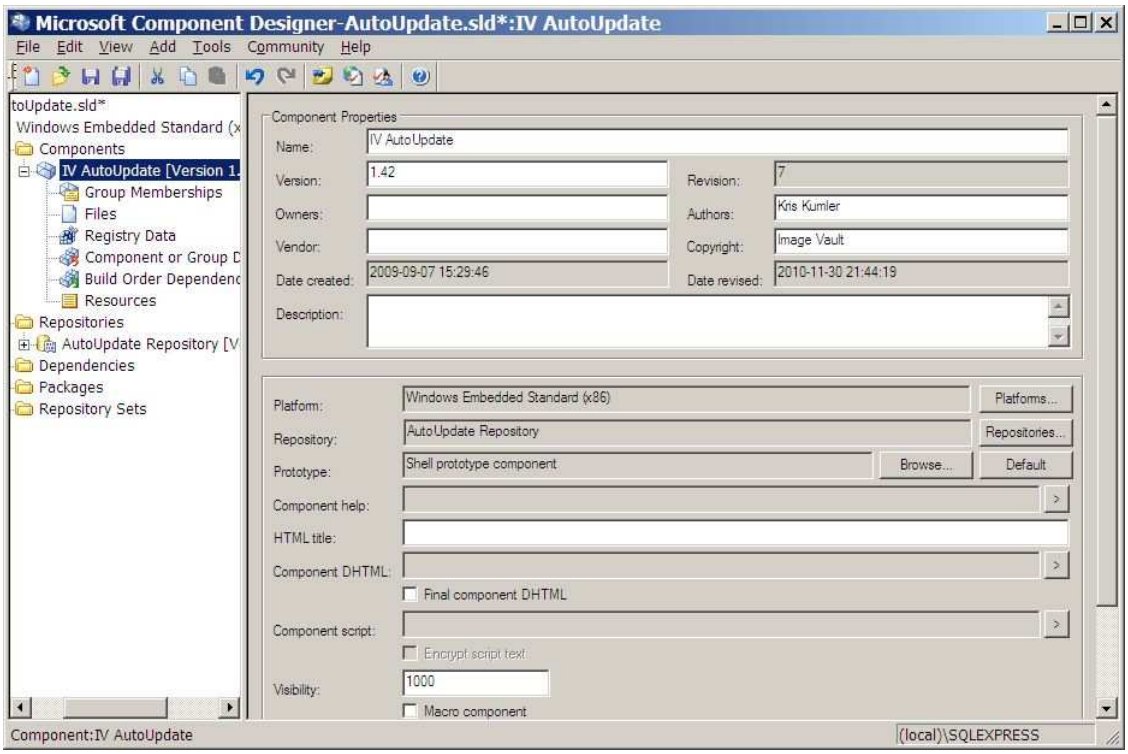

FIGURE 19. AutoUpdate Component Properties

APPENDIX VI. WINDOWS SECURITY DIALOG PREFERENCES

The process for creating the Windows Security Dialog Preferences component follows the same pattern as previous components created. See FIGURE 20 for the registry data view of the Component Designer.

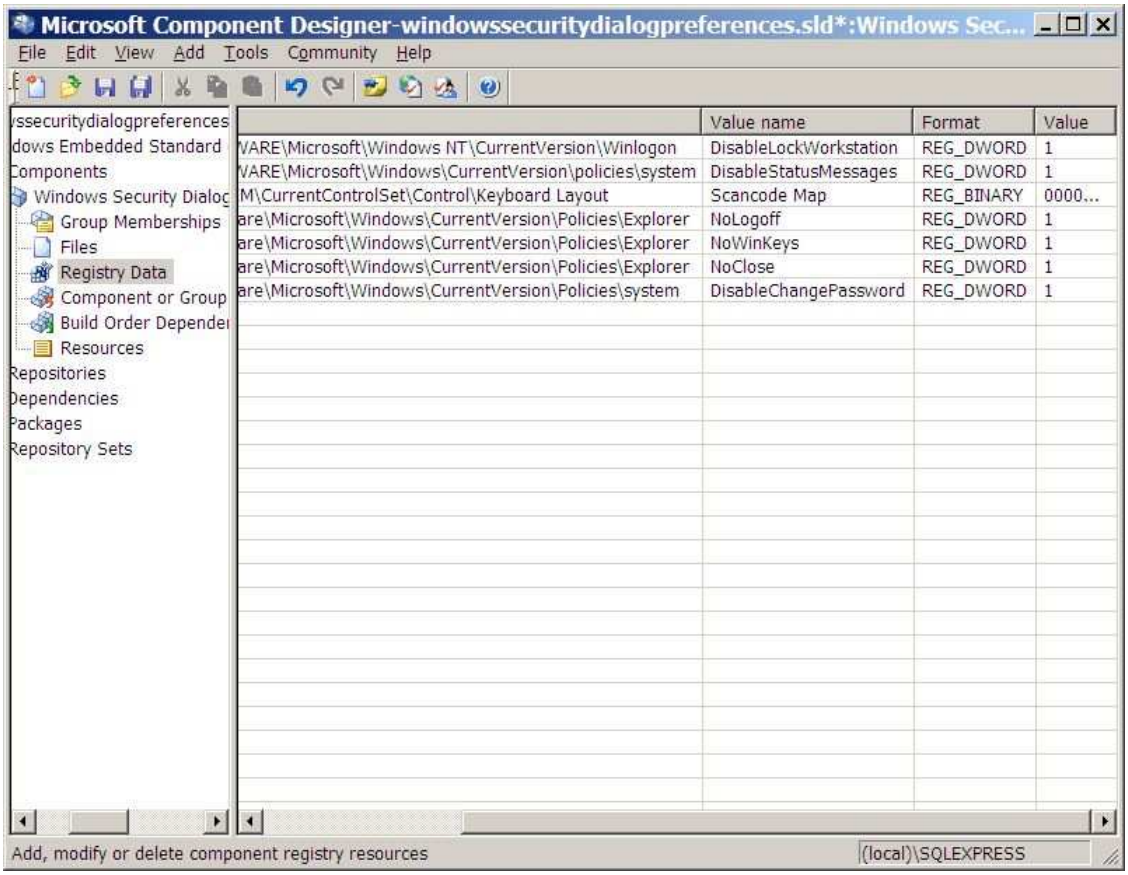

FIGURE 20. Component Designer Registry Data

1. Create new file in Microsoft Component Designer
2. Add new component
    a. Name “Windows Security Dialog Preferences”
3. Add Registry Data, to disable to the “Lock Workstation” button:

a. Root: HKEY_LOCAL_MACHINE
b. Key Name: SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
c. Value Name: DisableLockWorkstation
d. Type: REG_DWORD
e. Value: 0x1

4. Add Registry Data, to disable status messages on logon and logoff:
    a. Root: HKEY_LOCAL_MACHINE
    b. Key Name: SOFTWARE\Microsoft\Windows\CurrentVersion\policies\system
    c. Value Name: DisableStatusMessages
    d. Type: REG_DWORD
    e. Value: 0x1
5. Add Registry Data, to disable the “logoff” button:
    a. Root: HKEY_USERS
    b. Key Name: .DEFAULT\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
    c. Value Name: NoLogoff
    d. Type: REG_DWORD
    e. Value: 0x1
6. Add Registry Data, to disable the Windows hotkeys:

a. Root: HKEY_USERS
b. Key Name:
   .DEFAULT\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
c. Value Name: NoWinKeys
d. Type: REG_DWORD
e. Value: 0x1

7. Add Registry Data, to disable the “shutdown…” button:
   a. Root: HKEY_USERS
   b. Key Name:
      .DEFAULT\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
   c. Value Name: NoClose
   d. Type: REG_DWORD
   e. Value: 0x1
8. Add Registry Data, to disable the “Change password” button:
   a. Root: HKEY_USERS
   b. Key Name:
      .DEFAULT\Software\Microsoft\Windows\CurrentVersion\Policies\system
   c. Value Name: DisableChangePassword
   d. Type: REG_DWORD

e. Value: 0x1

APPENDIX VII. IMAGE CONFIGURATION CREATION

The following is a list of detailed steps for the image configuration process discussed in CHAPTER IV. DEVELOPMENT OF AN EMBEDDED OPERATING SYSTEM PLATFORM.

1. Create new configuration within Microsoft Target Designer
2. Under the Configuration Settings page, set *Target Device Settings -> Boot partition size (MB)* to "5000"
3. Add component "Advantech AIMB-763 Motherboard"
4. Add component "IVRecord"
5. Add component "IV AutoUpdate"
6. Add component "IV FK-145 (WinDriver Virtual device)"
7. Add components "NCP3000 Audio (Multi)"
8. Add components "NCP3000 Video (Multi)"
9. Add component "Windows Security Dialog Preferences"
10. Add component "Generic USB Input Device Support"
11. Add component "System Cloning Tool" (see FIGURE 12)
    a. Edit Reseal Option: uncheck "Remove AutoLogon Settings"
12. Add component "Administrator Account"
    a. Edit setting Password: PitEpEksOvbogVopbedk [randomly generated]
13. Add component "Automatic Logon"

a. Edit settings
    i. User name: Administrator
    ii. Password: PitEpEksOvbogVopbedk

14. Add component “NT Loader”
15. Add component “Device: Display”
    a. Edit settings
        i. Screen resolution: 1024 by 768 pixels
        ii. Screen refresh rate: 75 Hertz
16. Add component “Windows Logon (Standard)”
    a. Edit settings
        i. uncheck “Show Friendly Winlogon”
        ii. uncheck “Show `Welcome to Windows’ screen before Winlogon”
17. Add component “Windows Firewall/Internet Connection Sharing (ICS)” (see FIGURE 11)
    a. Edit settings
        i. Check “ICMP Settings->Allow incoming echo request”
        ii. Uncheck “Windows Firewall Settings->Notifies when Windows Firewall blocks a program”
    b. Add Authorized Application
        i. Program Name: IVRecord
        ii. Program Path: C:\iv\IVRecord.exe

iii. Scope: Any source

18. Add component “ACPI Multiprocessor PC” (see FIGURE 10)
    a. Edit settings
        i. “System Identification->Registered Owner”: IV
        ii. “System Identification->Registered Organization”: IV
        iii. Check “System Pagefile->Enable pagefile support”
        iv. “System Pagefile->Pagefile initial size (MB)”: 1024
        v. “System Pagefile->Pagefile maximum size (MB)”: 1024
        vi. Check “Data Execution Prevention Settings->Turn on DEP for all programs and services”
19. Add component “Windows subsystem”
    a. Edit registry data, for suppressing system messages
        i. Edit properties for “HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Windows\ErrorMode”
        ii. Value: 0x1
20. Add Extra Registry Data (see FIGURE 21), for suppressing the New Hardware Wizard:
    a. Root: HKEY_LOCAL_MACHINE
    b. Key Name: System\CurrentControlSet\Services\PlugPlay\Parameters
    c. Value Name: SuppressUI

d. Type: REG_DWORD
e. Value: 0x1

21. Add Extra Registry Data (see FIGURE 21), for suppressing system status messages:
    a. Root: HKEY_LOCAL_MACHINE
    b. Key Name: SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
    c. Value Name: DisableStatusMessages
    d. Type: REG_DWORD
    e. Value: 0x1
22. Add Extra Registry Data (see FIGURE 21), for changing desktop background color to red:
    a. Root: HKEY_USERS
    b. Key Name: .DEFAULT\Control Panel\Colors
    c. Value Name: Background
    d. Type: REG_SZ
    e. Value: 128 0 0
23. Run Tools -> Dependency Check

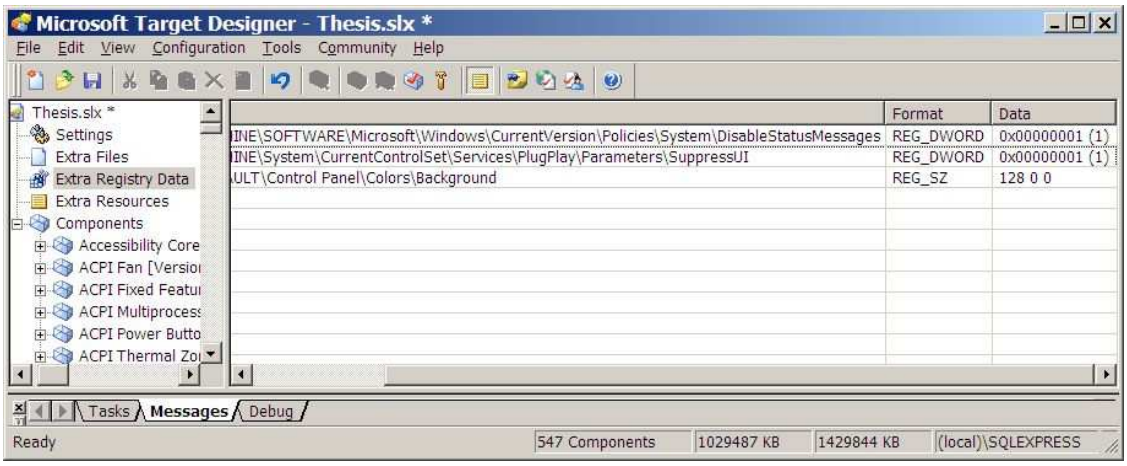

FIGURE 21. Image Configuration Extra Registry Data

LIST OF REFERENCES

[1] Swift, M., Bershard, B., Levy, H. 2004. Improving the Reliability of Commodity Operating Systems. *ACM Transactions on Computer Systems,* Vol. 23, No. 1, February 2005.

[2] Microsoft Corporation. "Which Windows Embedded Product is Right for Me?" *Internet source*, Available at http://www.microsoft.com/windowsembedded/en-us/products/whichproduct/default.mspx; access 28 October 2008.

[3] ANSI/IEEE, *Standard Glossary of Software Engineering Terminology*, STD-729-1991, ANSI/IEEE, 1991

[4] Microsoft Corporation. 2008. "Driver Verifier," Windows Driver Kit: Driver Development Tools Documentation. Redmond, Washington.

[5] Advantech Corporation. "Industrial Motherboards and Embedded Systems Solutions," *Advantech Corporation Press Room*, Available at http://www.advantech.com/pressroom/corporate_news.aspx?doc_id={5FD2A138-6F1A-4128-8775-A4EA53A2B33, published 4 June 2008

[6] Siddens, S. 1 February 2007. "UPS on the front line." *Plant Engineering*, Oak Brook, Illinois.

[7] Ingham, K., Forrest, S. 2002. A History and Survey of Network Firewalls. Computer Science Department, University of New Mexico.

[8] Davies, J., 2005 "Manually Configuring Windows Firewall in Windows XP Service Pack 2," *TechNet Library, The Cable Guy*. Microsoft Corporation, Redmond, Washington.

[9] Microsoft Corporation. 2008. *Windows Embedded Standard Datasheet*. Redmond, Washington.

[10] Microsoft Corporation. 2008. *Windows Embedded Standard: Product Overview*. Redmond, Washington.

[11] Chamberlain, M. 2008. *Microsoft Windows XP Embedded Developer Resource Kit*. Microsoft Corporation, Redmond, Washington.

[12] Russinovich, M., Solomon, D. 2005. *Microsoft Windows Internals*. Microsoft Press. Redmond, Washington.

[13] *Image Vault PRO-Command Product Manual*. 2008. Image Vault, LLC. New Albany, Indiana.

[14] Microsoft Corporation. *2007. Windows XP Embedded Documentation*. Redmond, Washington.

[15] *AIMB-763 User Manual, 2nd Edition*. 2007. Advantech Corporation. Taipei, Taiwan.

[16] Miller, S. 2006. *Dependency Walker 2.2 Help*. Microsoft Corporation. Redmond, Washington.

[17] *Image Vault Internal Project. I/O Daughter Card*. 1998-2009. Image Vault, LLC. New Albany, Indiana.

[18] InCtrl5. 2000. Neil J. Rubenking, Ziff-Davis Media, Inc. San Francisco, California.

[19] Microsoft Corporation. April 2010. "Feature Comparison of Windows Embedded Standard 7 vs. Windows Embedded Standard 2009" Microsoft Corporation. Redmond, Washington.

[20] Microsoft Corporation. "Windows Embedded Developer Update (WEDU)," *Internet source*. https://www.microsoft.com/windowsembedded/en-us/products/westandard/developer-update.mspx; access 17 November 2010

VITA

Name: Kristopher L. Kumler

Address: 9615 Long Rifle Ln
Louisville, KY 40228

Education: B.S., Computer Engineering and Computer Science
University of Louisville
1998-2002

Awards: Raymond I. Fields Award
ACM Distinguished Student Award

Professional Societies: Association for Computing Machinery

Association for Computing Machinery,
University of Louisville Chapter;
Linux Users Special Interest Group

Upsilon Pi Epsilon Computer Engineering Honor Society

Order of the Engineer