ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WECPL05

MIGRATING TO TINY CORE LINUX IN A CONTROL SYSTEM

R. A. Washington, ISIS, Rutherford Appleton Laboratory, Didcot, U. K.

Abstract

The ISIS Accelerator Controls (IAC) group currently uses Microsoft Windows Embedded Standard 2009 (WES) as its chosen Operating System (OS) for control of front-line hardware. Upgrading to the latest version of Microsoft Windows Embedded is not possible without also upgrading hardware or changing the way that the software is delivered to the hardware platform. The memory requirements are simply too large for this to be considered a viable option. A new alternative needed to be sought; which led to Tiny Core being selected due to its frugal memory requirements and ability to run from a RAM disk. This paper describes the process of migrating from Windows Embedded Standard to Tiny Core Linux as the OS platform for IAC hardware.

A NEED TO UPGRADE

The ISIS Control System uses Vista Control Systems [1] Vsystem software as the primary user interface. The Control System [2] monitors some 29,000 values, the majority of which are acquired through custom hardware designed by the ISIS Accelerator Controls (IAC) group. The latest version of this hardware is the IAC CompactPCI Standard (CPS) system which run Microsoft Windows Embedded Standard 2009 (WES) as the embedded Operating System (OS) and platform for the CPS software handlers. The CPS systems have been very reliable but the extended support for WES ended on January 8th 2019 meaning that an upgrade to the OS was already well overdue.

A NEW PLATFORM

CPS systems are built around a CompactPCI backplane with slots for up to seven peripheral cards and a system slot containing a processor card. CPS processor cards that are currently used are the Kontron [3] CP305, CP306 and CP3004.

The natural choice for the new OS, given the previous experience of the IAC group, was to pursue the latest embedded version of Microsoft Windows, Windows 10 IoT Enterprise. However it quickly became clear that this OS would not meet the particular deployment requirements. The CPS systems are required to remotely acquire an OS image over the network and continue to run the OS as a RAM disk. This is because conventional hard disk drives, specifically the disk controllers, fail due to radiation within the area of the inner synchrotron. To this end, the OS image is downloaded via Pre-boot Execution Environment (PXE) into the on-board RAM of the processor card, where it continues to run as a RAM disk.

All this is possible in Windows 10 but the size of the Windows 10 OS image footprint (typically around 8 GB) is far greater than that of the WES image (350 MB) and therefore requires more available RAM on the processor card. For reference, the older CPS processor cards have just 1 GB of available RAM. A decision was made to look at alternative OS solutions that have a smaller footprint and therefore smaller RAM requirement.

A comparison of several Linux distributions resulted in Tiny Core (TC) [4] being selected as the new platform. TC is a minimal Linux distribution and has the added advantage of being designed to run completely from a RAM disk.

THE CPS SERVICE

The CPS Service is a software manager that handles hardware reads and writes that have been initiated by Vsystem. It provides the link between the hardware in a CPS system and the control screen operated by the end user. The flowchart for the CPS Service is illustrated in Fig. 1.

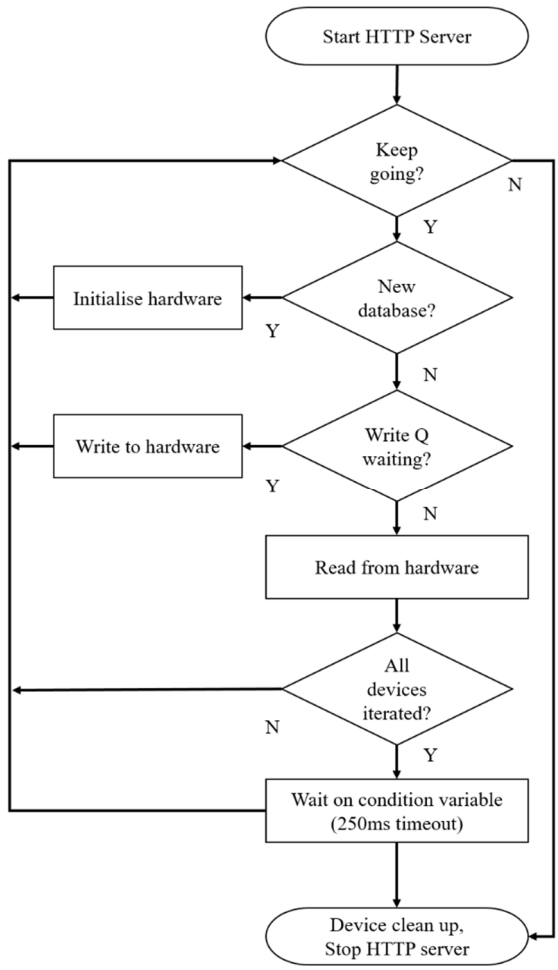

Figure 1: CPS Service flowchart.

The CPS Service code is organised into three projects: CPSService, HTTPServer and PCIDeviceClasses.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WECPL05

CPSService

This is the main thread. The HyperText Transfer Protocol (HTTP) Server is started in a separate thread while the main thread enters the service loop. This will run indefinitely until a stop signal is received. Each CPS system is defined by a local database containing 'channels' that represent items to control or monitor. The service loop iterates through the database until all the database channels have been processed. This process then repeats.

HTTPServer

The HTTPServer contains the code for communication over the ISIS Controls network with Vsystem. The CPS systems communicate by passing Extensible Markup Language (XML) files over HTTP. The web server also provides the valuable ability to view the database contents of a CPS system in a web browser for diagnostics and monitoring purposes.

PCIDeviceClasses

The PCIDeviceClasses project contains all the application specific code for the various CPS peripheral control cards.

BUILDING A TINY CORE IMAGE

The CPS Service was originally written for a Windows OS environment [5]. As a result it relied heavily on various Microsoft libraries and Microsoft XML Core Services (MSXML).

Before the transition to TC could take place, the CPS Service project needed to be refactored to remove all Windows-related dependencies and apply modern (C++17) techniques to simplify the code where possible.

The outcome of this process is that the HTTP communications server and XML parsing is now provided by the open source gSOAP XML web services toolkit [6].

The Build System

In order to create a stand-alone image for the CPS Systems that will run entirely in RAM, the initial ramdisk (initrd) image requires 'remastering'.

While any Linux distribution could be used as the basis for a build system, TC was chosen in order for the build and target system to have as much in common as possible.

A TC virtual machine (VM) was created using the TC CorePlus.iso image file as the installation media. The VM also includes the TC EZRemaster tool (available as an installation option) which allows a base TC image to be easily created and remastered.

Remastering

The process of building the TC image is relatively simple. A base image is created using the native EZRemaster tool, then all the extra files, tools, extensions and drivers that are necessary to build and run the CPS Service are added.

Extensions

Extensions are software applications that provide the required additional functionality to the TC image. The following extensions are included in the initrd image:

- bash
- glibc_base-dev
- kmaps
- libpci
- openssh
- tzdata

bash The Bourne Again Shell is not strictly needed but it is the most commonly used shell in Linux so is included for convenience.

glibc_base-dev Provides the shared libraries, libpthread and libdl, that are used by the CPS Service.

kmaps Allows the correct UK keyboard map to be applied.

libpci Provides the shared library libpci, which is required by the CPS Service code.

openssh Provides the Secure Shell (SSH) daemon which allows for remote logging into the CPS System.

tzdata Provides time zone data so the system clock will update in accordance with seasonal changes.

Boot Codes

Additional configuration of the system is done by including 'boot codes' in the image. These are a way to configure the system by providing information during the boot process. The TC image for CPS includes the following boot codes:

- base
- norestore
- lang=en_GB.UTF8
- kmap=qwerty/uk
- tz=Europe/London
- user=CPSAdmin
- 8250.nr_uarts=32

base This loads the base system only, without any additional extensions outside of the initrd.

norestore As each CPS System will boot a fresh image each time, this code prevents anything being restored. In combination with the 'base' boot code, it also serves to prevent TC from looking for any disks to mount.

lang=en_GB.UTF8 Set the system language to British English.

kmap=qwerty/uk This works in conjunction with the kmaps extension to set the keyboard map to the United Kingdom keyboard layout.

tz=Europe/London Sets the system time zone to London.

user=CPSAdmin Configure the default user as CPSAdmin.

8250.nr_uarts=32 The kernel is limited to 4 serial ports by default. However the CPS System is required to support up to 32. This code changes the default limit.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WECPL05

Other Customisations

After the extensions and boot codes have been added, the EZRemaster tool can be used to finish the remastering process, resulting in a directory containing the remastered image.

There are now a number of manual customisations to be done before we have the final TC image. These are:

- Add the hardware drivers and kernel modules to the extracted directory.
- Add the CPS Service binary file.
- Add a script to start/stop the CPS Service.
- Generate the system locale using the getlocale extension and then copy the output to the extracted image.
- Set up the Network Time Protocol (NTP) configuration file.
- Edit the /etc/inittab file in the extracted image to ensure the CPS Service software runs on start up.
- Generate a server key for SSH.
- Set a user password.

After this, the image customisation is now complete and an initrd can be built from the extracted image directory by following instructions in the TC documentation [4]. This gives the initrd output, CPScore.gz.

DEPLOYING THE IMAGE

While the long term plan is to move to a Linux Remote Boot Server; the hosting of CPS System OS images currently uses a Windows Remote Boot Service Trivial File Transfer Protocol (TFTP) server.

The boot program that is used for booting Linux via PXE is pxelinux.0, which is part of the SYSLINUX [7] extension (available with most Linux distributions). This file along with ldlinux.c32 needs to be placed in the root directory of the TFTP server. A separate configuration file, rbsprov.ini, gives details of the boot program to use for each CPS System.

By default, pxelinux.0 looks for its configuration in the pxelinux.cfg directory; where it looks for a file that is named with the following (in order of preference):

1. Client's Universally Unique Identifier (UUID).
2. Client's Media Access Control (MAC) address.
3. Client Internet Protocol (IP) address[1].
4. A file named 'default'.

During testing, this iterative process was found to be prohibitively long. However, it can be bypassed using the pxelinux-options program, which is provided with SYSLINUX, to force the configuration file to use 'default' in the first instance.

All that is left to do is copy the kernel (vmlinuz) and the initrd (CPScore.gz) to the TFTP server and include these file locations in the configuration file named 'default'.

[1] This is recursively checked, dropping one character each time, until the end of the IP address is reached.

CONCLUSION

The IAC CPS hardware has now been upgraded from the unsupported Microsoft WES 2009 to use Tiny Core Linux as the embedded OS. This has provided the following additional benefits:

- TC is covered by the GNU General Public License, meaning that recurrent purchases of licenses are no longer necessary.
- The OS is flexible and open source, with an active open source community.
- The size of the image at 15 MB is much smaller than the WES image size of 350 MB. This has resulted in a much quicker boot time for the CPS systems. Typically <45 seconds, compared to 3-4 minutes previously.
- The amount of RAM required is now ~40 MB which is small enough to run on the oldest of the CPS processor cards; removing the need for costly hardware upgrades.

ACKNOWLEDGEMENTS

Ben Warrington for work completed in refactoring the CPS Service project and development of the Tiny Core image.

Tim Gray for his previous work on development of the CPS Service project.

REFERENCES

[1] Vista Control Systems, Inc., https://www.vista-control.com

[2] R. P. Mannix, "Vista Controls' Vsystem at the ISIS Pulsed Neutron Source", in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, paper WOAA04, pp. 284-284.

[3] Kontron, https://www.kontron.com

[4] The Core Project, http://tinycorelinux.net

[5] T. G. Gray and R. P. Mannix, "Using Windows XP Embedded Based Systems in a Control System", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper THA003, pp. 636-637.

[6] Genivia, https://www.genivia.com/products.html

[7] The Syslinum Project, https://www.syslinux.org