GPU Passthrough

BY CORVIN KÖHNE AND DANIEL KERKHOFF

Beckhoff has been working on industrial automation technology since it was founded in 1980. In 1986, the concept of PC-based machine control was born. Back then, Beckhoff was moving away from microcontroller-based PLCs and started using standard PC technology as a base for the PLC, while casing it in industrial housing. This boosted performance massively, as PCs were already far more powerful than microcontrollers.

With the rise of Windows, the idea to make a general-purpose operating system part of the machine control system itself was born. The Windows environment was already familiar to machine builders and operators. The Windows environment could also be enriched with additional user applications, which proved beneficial since it added value to the overall control system. Windows itself, however, did not provide the required execution environment for control logic with hard, real-time constraints.

TwinCAT (which stands for The Windows Control Automation Technology) software had been developed and was first released in 1996. TwinCAT was designed as an extension of Windows NT to assure deterministic execution of control tasks on Windows-based control systems from Beckhoff.

Over the years, the hardware components of PCs, such as CPUs, memory, bus systems and I/O devices have become more powerful, as have PC-based control systems. In 2002, Beckhoff invented the embedded PC family, a DIN-rail mountable, industrial PC with a direct connection to the Beckhoff I/O system. As these devices used low power CPUs (AMD SC2200) with compact flash cards, an embedded operating system with a small footprint was needed. At the time, Windows CE was the dominant player on the market and was the obvious choice for this embedded PC family. Today, Beckhoff offers a broad product family of industrial PCs, ranging from small single core ARM CPUs to the latest AMD Ryzen CPUs and Intel Xeon based server class devices. Consequently, control system engineers can choose the right performance class for their applications.

TwinCAT runtime is the common denominator for all Beckhoff IPCs, and naturally, TwinCAT itself has evolved over time to enable control systems to harness the power of modern PC hardware. Windows has become the standard operating system for PC-based control systems at Beckhoff and also for its customers.

In recent years, however, Windows CE has become obsolete and customer demand for a non-Windows based control system has increased, so Beckhoff started to search for an alternative operating system which could become the new TwinCAT host. Ultimately, FreeBSD was chosen as the non-Windows operating system to host TwinCAT. Beckhoff combined FreeBSD and TwinCAT to create a new PC-based control operating system, named TwinCAT/BSD.

TwinCAT/BSD

Currently, FreeBSD 13.2 serves as the foundation for TwinCAT/BSD. Consequently, TwinCAT/BSD benefits from the FreeBSD base system, including FreeBSD user space tools as well as kernel services. In addition, Beckhoff has also imported the majority of TwinCAT features from Windows-based control systems to TwinCAT/BSD. TwinCAT/BSD is made up of both FreeBSD base system and the TwinCAT features. Figure 1 shows the structure of the system.

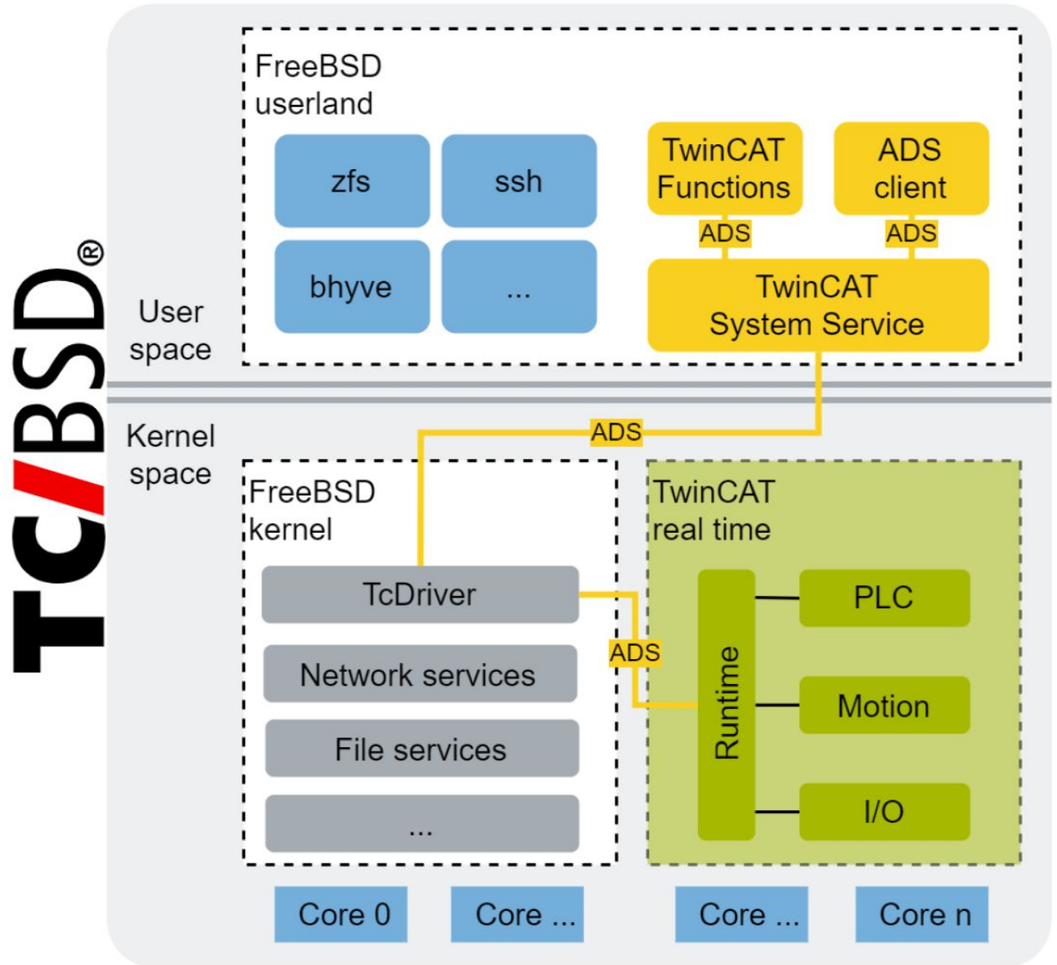

Figure 1: TwinCAT integration on FreeBSD

The core components of a TwinCAT-based control system are the TwinCAT runtime and the TwinCAT System Service including the ADS message router. The TwinCAT runtime is responsible for the execution of real-time control tasks. Real-time control tasks typically involve programmable logic controller (PLC) tasks, motion tasks such as axis positioning or CNC, and the integration of industrial I/O devices connected via fieldbus systems.

To be able to run control tasks with hard real-time requirements, TwinCAT configures different system settings. For example, TwinCAT uses isolated CPU cores for control tasks. Isolated cores are removed from all CPU groups of the FreeBSD scheduler. This allows TwinCAT to fully utilize isolated cores to execute control tasks. The TwinCAT runtime determines the execution of the configured, real-time tasks. Each real-time task has its own cycle time which specifies the frequency at which the task has to be executed.

The TwinCAT runtime provides an interface for the Automation Device Specification protocol (or ADS for short) so that the TwinCAT runtime can send and receive ADS messages. ADS is an application protocol developed by Beckhoff. The whole information exchange in a TwinCAT system is driven by ADS. The TcDriver is a standard FreeBSD kernel module which provides an ADS channel between the TwinCAT runtime and the user space. This channel is used by the TwinCAT System Service to control the TwinCAT runtime. Likewise, the TwinCAT System Service implements an ADS message router to route ADS messages between different applications like the TwinCAT runtime and TwinCAT Functions. Thanks to the ADS message router, the TwinCAT system can easily be extended with other services that implement the ADS protocol. Similarly, ADS client applications only need access to the message router to access ADS services such as the TwinCAT runtime to read and write the PLC task's data, for example. The ADS specification and libraries are publicly available so that ADS-based applications can be implemented easily.

In general, the TwinCAT architecture described for TwinCAT/BSD is similar to the architecture for Windows-based operating systems. However, Beckhoff customers particularly favor TwinCAT/BSD as it is a more compact and robust operating system. Due to its small footprint, it is often used on embedded industrial PCs. These embedded industrial PCs can be found in control applications ranging from building automation to CNC machines to automated guided vehicles. Although TwinCAT/BSD is already used in a variety of control applications, sophisticated control applications always require more flexibility to extend the capabilities of the control systems.

The Need for Virtual Machines

For PC-based control systems, the host operating system serves two purposes. As with any operating system, it is responsible for providing a clean, abstract interface to the subordinate hardware. Equally, the operating system environment can be used to run applications and services which extend the functionality of the control system. The user can thus install or develop their own applications for the control system while interacting with the system as they do with any other PC.

Furthermore, the operating systems serve as hosts for the TwinCAT runtime. FreeBSD, as a host operating system, is responsible for managing the majority of the hardware in an appropriate manner. This allows TwinCAT to focus on scheduling and executing real-time control tasks. However, without a running host operating system, the TwinCAT runtime would not be able to run at all. In terms of the overall control system, the system operator needs to pay a great deal of attention when interacting with the host operating system, since a restart or crash in the host operating system would also result in the real-time control being stopped. This is true for both Windows and TwinCAT/BSD PC-based control systems.

FreeBSD, as a host operating system, is responsible for managing the majority of the hardware in an appropriate manner.

bhyve offers the option of using a virtual machine as user environment, which can be rebooted without affecting the TwinCAT real-time system. In this setup, the TwinCAT/BSD host operating system would still serve as the underlying PC-based control system, but the operator would no longer directly use the host operating system for human-machine interaction; a guest operating system would be used inside a virtual machine environment instead.

With virtual machines, the system designer can take advantage of a wider range of operating systems to deploy additional services on the control system. A virtualized Windows environment could enable the Windows desktop environment and already developed Windows user interfaces to be used for machine interactions on TwinCAT/BSD systems. Environments such as these are especially useful for customers who are used to Windows-based control systems. A virtualized Linux environment would, in turn, enable containerized Linux applications to be used on the IPC via Docker, Podman or Kubernetes. Again, customers who are already using Linux container deployments to run services or algorithms for analyzing process data could deploy those applications directly on the control system itself.

Virtual machines hosted by bhyve also benefit from OpenZFS, which is the default file system on TwinCAT/BSD. For customers, high availability of both the TwinCAT-based control system itself and user applications is crucial. As a result, any updates on a system which may lead to downtime or inoperability are inherently high-risk. Being able to create ZFS snapshots of datasets which show the functional state of the base system and of virtual machine disks enables a functional system state to be restored, thus cutting the risk of long downtimes if software updates malfunction.

Furthermore, running operating systems in virtual machines can also be harnessed to improve system security via increased availability. The capacity to access process data from a control system is becoming more relevant as a result of digitization, because this data forms the basis for optimized production processes. Acquisition, processing, and interpretation of process data should be as automated, in-process and timely as possible. However, the access to the control system which is required by applications and/or operators also increases the risk that the control system's open interfaces may be exploited or operated incorrectly. If the system is accessed in this way, this can in turn influence the correct operation of the control system and, in the worst case, lead to downtime and thus to uncontrolled processes. A virtual machine can be used as an additional gateway here to restrict both user and network access to the control system. To provide the aforementioned benefits through virtual machine environments, we have started incorporating bhyve into TwinCAT/BSD.

Virtual Machine Configuration

bhyve was integrated into TwinCAT/BSD in order to configure a system setup like the one shown in figure 2.

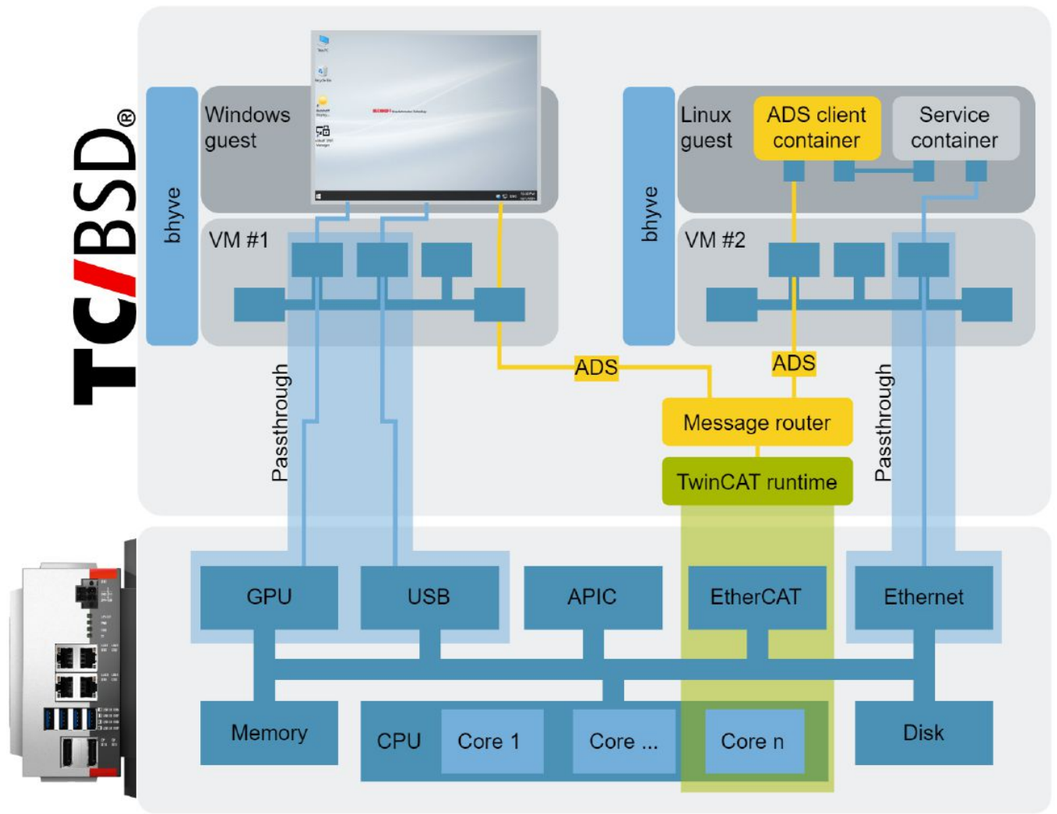

Figure 2: Sample VM configurations on TwinCAT/BSD

In this setup, the execution of virtual machines should be optional as not every TwinCAT/BSD user requires separate execution environments for additional guest systems. If needed, however, the customer would be able to configure a virtual machine with bhyve that could support either the use case of a Windows or Linux guest operating system or both.

Linux guest operating systems are usually needed for headless server applications either installed directly on the Linux guest OS or deployed and managed as Linux containers. In terms of system setup, this means that network interface configuration tasks such as routing and package filtering are handled on host and guest operating sites. In most cases, this type of VM configuration consists of two network interfaces. One of these is used in a host-only network to access the TwinCAT ADS message router on the host operating system.

Process data from the TwinCAT control system can be gathered with access to the ADS message router. These data can then be further analyzed by other service containers inside the guest operating system. The information gained is then usually served or transmitted via a second network interface which is explicitly assigned to the virtual machine.

On industrial PCs that contain modern Intel or AMD CPUs, hardware support is provided for IO virtualization. This is termed VT-d for Intel and AMD-Vi for AMD. Both forms of support are similar and allow a hypervisor to give a guest direct access to a hardware device. The device memory can be mapped onto the guest memory space. Additionally, interrupts can be remapped to be directly delivered to the guest. This allows a guest to access a device without any hypervisor intervention to increase the performance.

bhyve already supports VT-d and AMD-Vi. These are used for PCI passthrough. As a result, a PCI device can be exclusively assigned to a single guest. This device is no longer shared with the host and the guest has direct access to it.

With PCI passthrough, one Ethernet controller can be isolated from the TwinCAT/BSD host and explicitly assigned to the virtual machine. As a result, all network traffic is directly passed to the virtual machine environment, meaning that no additional filter or routing rules need be applied on the TwinCAT/BSD host. Likewise, network traffic can be processed much faster as it is directly available inside the guest.

All network traffic is directly passed to the virtual machine environment, meaning that no additional filter or routing rules need be applied on the TwinCAT/BSD host.

As mentioned above, virtual machines with Windows guest operating systems should serve as an isolated environment for human-machine interaction. Consequently, the physical I/O interfaces of the industrial PC must be passed to the virtual machine. On Beckhoff IPCs, these interfaces are the USB controller for user input via keyboard, mouse or touch panel and, of course, the graphics controller for video output on any connected display.

As usual, PCI passthrough is used to explicitly assign the USB controller to the Windows guest. PCI passthrough works for almost all PCI devices. However, graphics cards are one class of PCI devices in which PCI passthrough isn't supported by bhyve. As this is a common issue in hypervisors, it's often referred to as GPU passthrough.

GPU Passthrough

Most graphics cards have some special requirements which aren't handled by bhyve yet. These special requirements are dependent on the graphics card vendor and the graphics driver. Different operating system graphics drivers require different features within the graphics card. Some graphics drivers need a VBIOS while others need access to special memory regions. All of these elements require extra handling, which has to be implemented for bhyve. Beckhoff has developed patches for bhyve to enable GPU passthrough on AMD and integrated Intel graphics cards. We're also working on upstreaming all of these patches.

GPU passthrough of AMD graphics cards requires a PCI ROM emulation. In the first step, the VBIOS of the graphics card has to be extracted from the host system. It is then used by

the guest BIOS to initialize the graphics card. Additionally, some guest graphics drivers require the VBIOS.

The VBIOS can be extracted using a variety of different methods. There's no common method for extracting the VBIOS on FreeBSD yet. Therefore, a different operating system such as Linux must be booted to extract the VBIOS. This ensures that the right VBIOS version is used for GPU passthrough. If a different version is used, there could be incompatibilities which may damage your device in the worst-case scenario.

Once the VBIOS has been extracted, it has to be passed to the guest. Therefore, we added a PCI ROM emulation to bhyve so that the guest can read the PCI ROM to get the VBIOS. The patches required to support the PCI ROM emulation have already been upstreamed. The PCI ROM emulation will be included in FreeBSD versions 13.2, 14.0 and subsequent versions.

Sadly, supporting the PCI ROM emulation isn't enough to support a VBIOS. It has to be executed and shadowed into main memory. This should be done by the guest BIOS. However, bhyve's current edk2 port is not capable of this. The patches that would make this possible have been placed in the upstream but have not yet been accepted. Therefore, a modified guest BIOS must be used to fully support GPU passthrough for AMD devices.

The VBIOS can be extracted using a variety of different methods. There's no common method for extracting the VBIOS on FreeBSD yet.

Adding a VBIOS to graphics cards also provides another benefit. A VBIOS includes an UEFI graphics driver. When the guest executes the VBIOS, the VBIOS installs an UEFI graphics card driver. This will be picked up by the guest BIOS to produce graphic output. It makes the graphics card available in an early boot phase. For that reason, graphic output can be produced in the UEFI and bootloader phase when a VBIOS is passed to the guest. Without a VBIOS, the first graphic output is produced after the OS driver is loaded.

In contrast to AMD graphics cards, GPU passthrough for integrated Intel graphics cards has not yet been upstreamed. Intel graphics cards have two special memory regions which need special attention. Intel calls these memory regions OpRegion and Graphics Stolen Memory. They have to be reserved by the guest BIOS and reported in a PCI register. Therefore, bhyve needs to detect and report those regions to the guest. We have patched bhyve to provide an E820 table to the guest. It includes all memory regions and reports where the OpRegion and Graphics Stolen Memory reside in the memory. Our guest BIOS is modified to parse this E820 table and reserve all memory regions as reported by the table. bhyve's PCI emulation reports the addresses of these memory regions by PCI register.

Our current implementation works for Intel processors from the 3rd generation, called Ivy Bridge, up to 9th generation Intel processors, called Coffee Lake Refresh. Newer Intel processors require slight modifications for emulation. Support will be added in the future. We have not yet tested GPU passthrough for AMD graphics cards on a variety of different graphics cards. According to some feedback from bhyve users who have tested the patches, it is supported on a wide range of different AMD graphics cards.

As previously stated, GPU passthrough is the same as PCI passthrough. Due to this, it does not differ from PCI passthrough in terms of usage. bhyve's -s option can be used for GPU passthrough as shown in the following code snippet.

```
bhyve \
        -s 0,hostbridge \
        -s 2,passthru,0/2/0 \
        -s 31,lpc \
        -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \ my-vm
```

If you wish to pass a VBIOS to the guest, we've added a ROM option to bhyve's -s option.

```
-s 2,passthru,0/2/0,rom=/home/user/vbios.rom
```

Conclusion

FreeBSD, in combination with Beckhoff TwinCAT automation software, forms the foundation of the TwinCAT/BSD operating system. TwinCAT/BSD is Beckhoff's Unix alternative to Windows-based control systems and is already used in a variety of control systems.

Now, by integrating bhyve into TwinCAT/BSD, PC-based control systems benefit from virtual machine environments. With the added GPU passthrough feature, virtual machine graphic output can now be displayed on connected screens. This is particularly useful for applications which provide a human-machine interface. Since these applications often originate from former Windows-based control systems, they can now be used on TwinCAT/BSD.

Virtual machines, in combination with the PCI passthrough feature, also provide an advanced system setup in which user interfaces and network interfaces can be isolated from the host operating system. Consequently, virtual machine environments can provide an additional layer to improve the security of the control system.

CORVIN KÖHNE is a software developer at Beckhoff Automation GmbH & Co. KG. He's a maintainer and developer of Beckhoff's FreeBSD fork, which is called TwinCAT/BSD. He focuses on x86-based systems and hypervisor technology. Due to his contributions to the bhyve project, he became a FreeBSD committer in 2022.

DANIEL KERKHOFF started his career as an application engineer for industrial automation solutions at Beckhoff Automation GmbH & Co. KG. After graduating with a master's degree in information technologies, he joined the TwinCAT product management team. In his role, he is responsible for queries from customer who use bhyve and TwinCAT/BSD as part of Beckhoff's PC-based control systems.