



KUKA Roboter GmbH

SOFTWARE

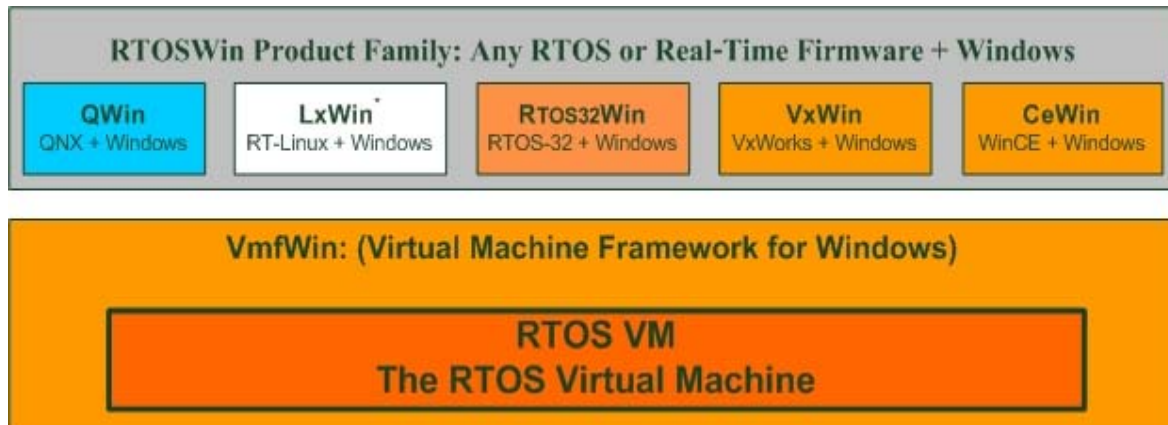
RTOSWin Product Family

Whitepaper

Edition: 2008-09-24

1	THE KUKA RTOSWIN PRODUCT FAMILY	3
2	KUKA RTOS VIRTUAL MACHINE OPERATION MODES	4
2.1	SINGLE-CORE SYSTEMS: SHARED MODE OPERATION.....	4
2.2	DUAL-CORE AND MULTI-CORE SYSTEMS: SHARED MODE OPERATION	5
2.3	DUAL-CORE AND MULTI-CORE SYSTEMS: EXCLUSIVE MODE OPERATION	6
2.4	DUAL-CORE AND MULTI-CORE SYSTEMS: SMP EXCLUSIVE MODE OPERATION	7
2.5	DUAL-CORE AND MULTI-CORE SYSTEMS: SMP SHARED MODE OPERATION	8
3	REAL-TIME DEVICE MANAGEMENT	9
4	VIRTUAL MACHINE FRAMEWORK	10
4.1	VMF ARCHITECTURE	10
4.2	BASIC VMF SERVICES (HARDWARE ABSTRACTION LAYER)	11
4.2.1	<i>Overview</i>	11
4.2.2	<i>HAL Services</i>	11
4.3	COMMUNICATION AND OTHER SERVICES	12
4.4	PORTABILITY	13
4.5	MEMORY LAYOUT	14
4.6	CONCLUSION	15
5	THE RTOS LIBRARY	15
6	EXAMPLE IMPLEMENTATION: VXWIN	16
7	EXAMPLE IMPLEMENTATION: RTOS32WIN	17

1 The KUKA RTOSWin Product Family



The KUKA RTOSWin family is a family of Windows virtualization solutions for multiple real-time operating systems and Microsoft Windows.

The key component of all these solutions is the RTOS Virtual Machine (RTOS-VM). The real-time operating systems are executed on top of the RTOS-VM.

KUKA Roboter's strategy is

- to provide a virtualization platform for Multi-OS and Multi-Core systems
- to offer well-proven technology for embedded applications
- to build an alliance group with experienced RTOS partners

RTOSWin product family members

- CeWin: Windows XP/Vista + Windows CE
- VxWin: Windows XP/Vista + VxWorks
- RTOS32Win: Windows XP/Vista + On Time RTOS-32
- QWin: Windows XP/Vista + QNX
- LxWin: Windows XP/Vista + RT/Linux (currently not available)

KUKA Roboter (alliance group) provides the complete solution:

KUKA and RTOSWin providers are partners of the provided operating systems

- **VxWin**
 - provided by KUKA's distributors/partners
 - KUKA has deep knowledge in Windows XP and VxWorks
 - KUKA has close partnership with both, Microsoft and Wind River
- **CeWin**
 - provided by KUKA's distributors/partners
 - KUKA has deep knowledge in Windows XP and Windows CE
 - KUKA has close partnership with Microsoft
- **RTOS32Win**
 - provided by acontis technologies GmbH
 - RTVmf component provided by On Time Informatik GmbH
 - acontis is On Time Informatik and KUKA alliance partner
- **QWin**
 - provided by IBV and IBV's distributors/partners
 - IBV is QNX Premier Distributor and has deep QNX knowledge
 - IBV is KUKA alliance partner

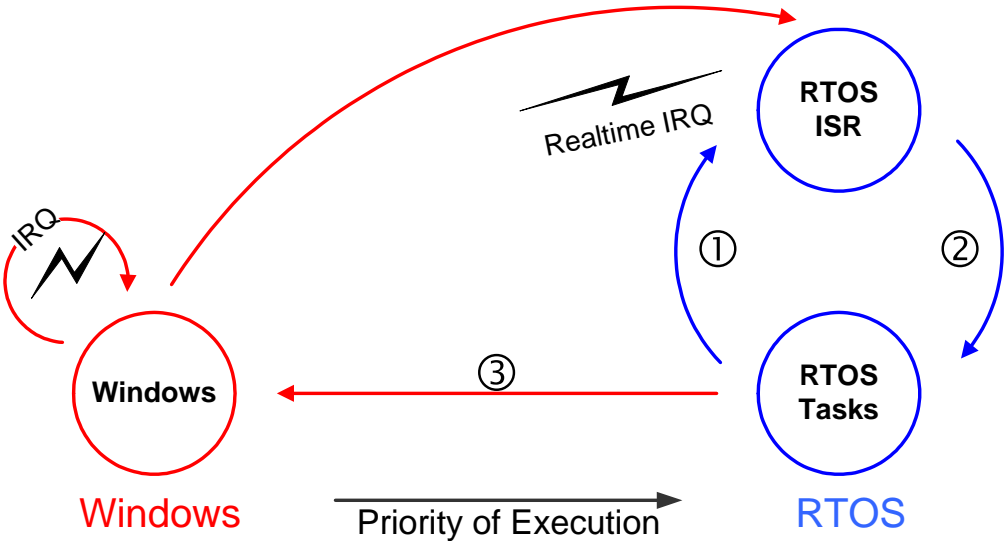
2 KUKA RTOS Virtual Machine Operation Modes

The KUKA RTOS-VM provides a light-weight real-time virtualization platform for Windows. On top of this platform real-time firmware or off-the-shelf real-time operating systems can be executed. Different operation modes are available.

2.1 Single-Core Systems: Shared Mode Operation

Windows and the RTOS both run on one single core CPU system. Windows will only get CPU time when the RTOS becomes idle.

The following figure shows shared mode operation on a single-core system:

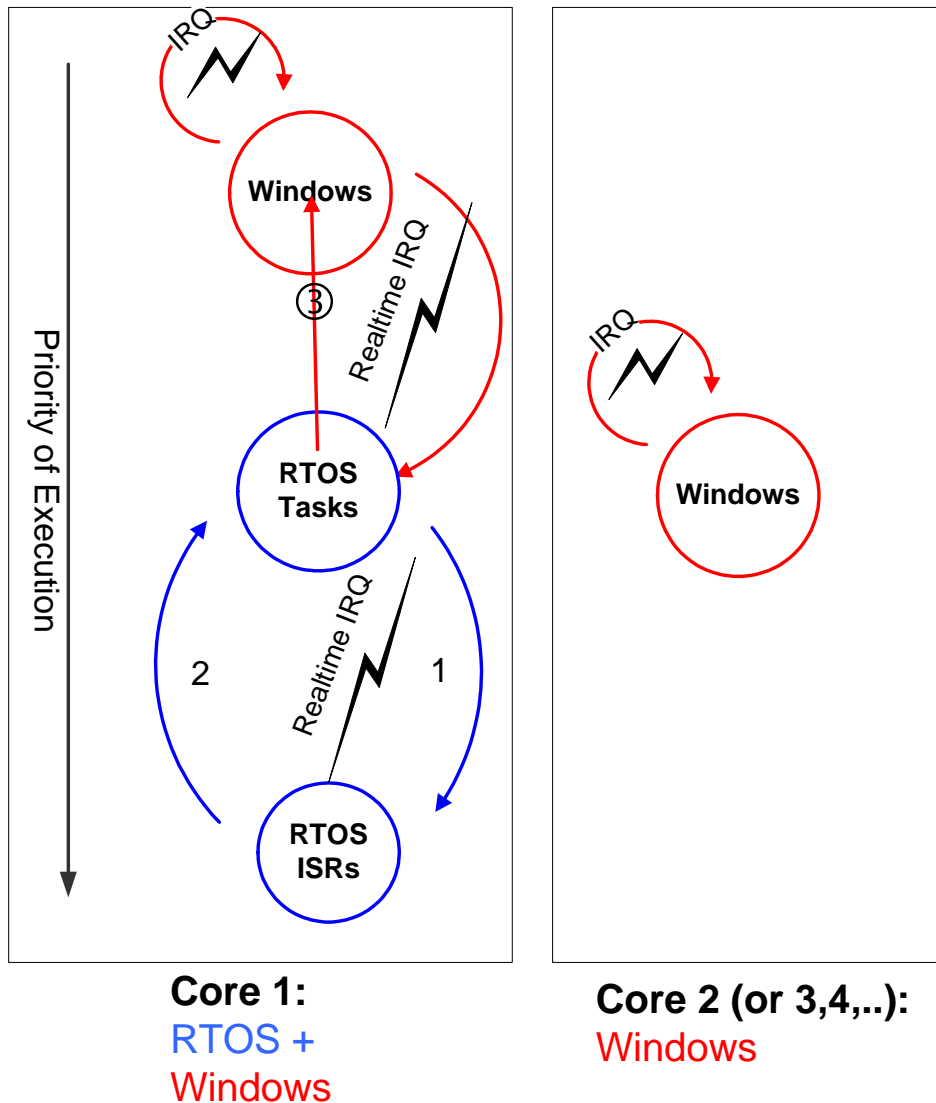


2.2 Dual-Core and Multi-Core Systems: Shared Mode Operation

When Windows utilizes all CPU cores in the system, the RTOS may run on an arbitrary CPU core. Thus, the core where the RTOS is running will be shared by Windows and the RTOS. Windows will only get CPU time on this core when the RTOS becomes idle.

Note: if the RTOS doesn't become idle all Windows activities on that core will cease which will also block all other Windows cores to operate correctly.

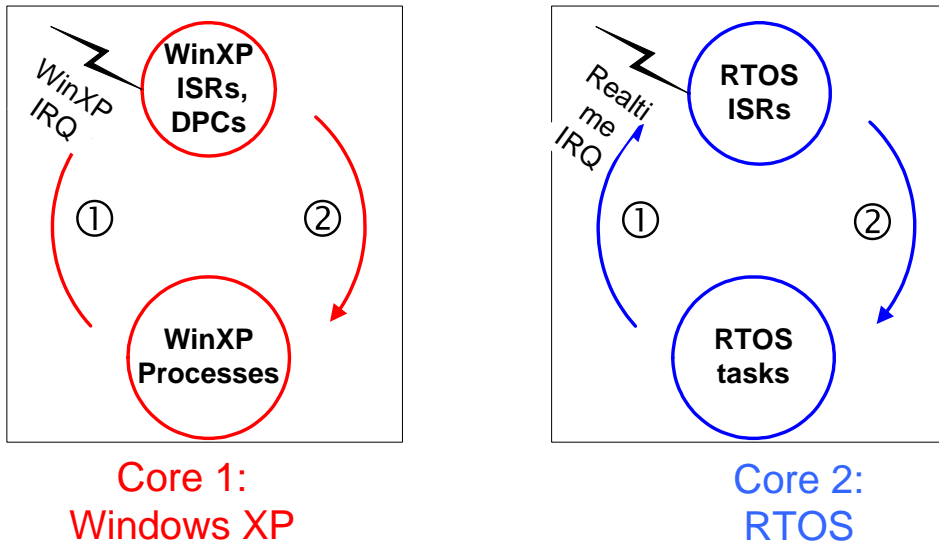
The following figure shows shared mode operation on a dual-core or multi-core system, where the RTOS runs on the first CPU core.



2.3 Dual-Core and Multi-Core Systems: Exclusive Mode Operation

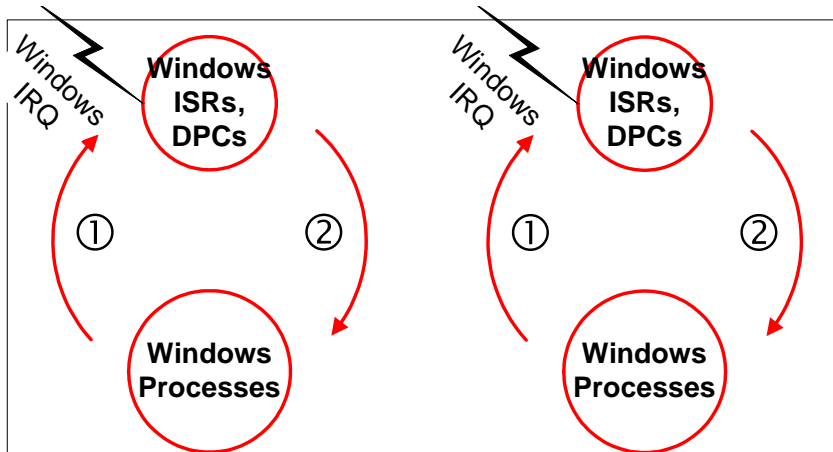
When the RTOS operates in exclusive mode (only available on multiprocessor/multicore systems) the last CPU core in the system is dedicated to the RTOS. All remaining CPUs are used by Windows XP.

The following diagram illustrates the flow of control on a dual-core system:

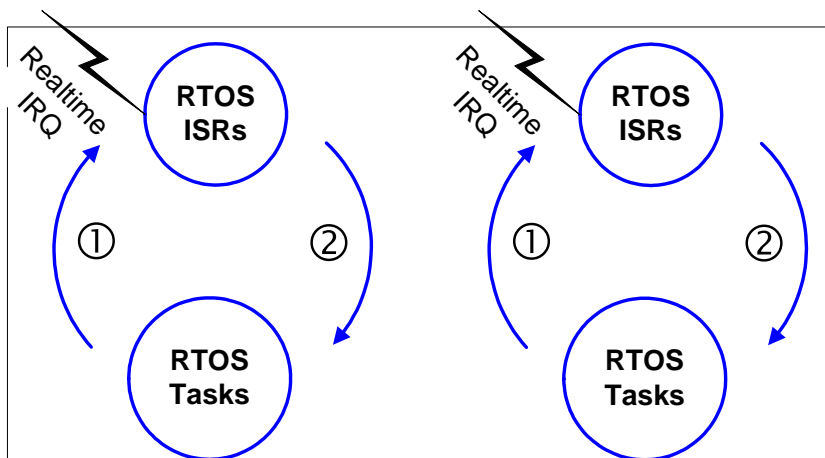


2.4 Dual-Core and Multi-Core Systems: SMP Exclusive Mode operation

On a system with n CPU cores Windows will use the first w CPU cores and the RTOS will use the remaining r CPU cores (where $r > 1$). The RTOS thus will use more than one core and run in SMP mode (Symmetric Multiprocessing mode). Both operating systems run completely independent from each other.



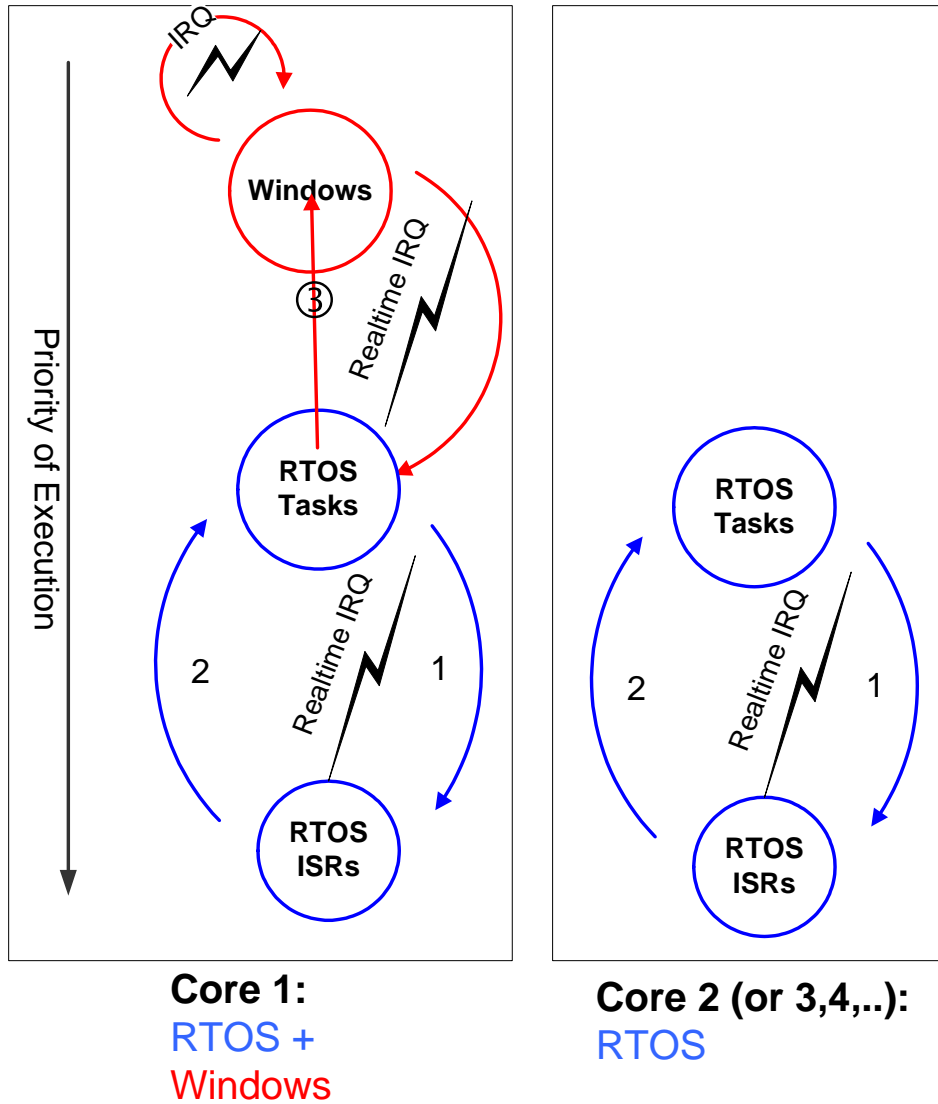
Cores 1 + 2: Windows in SMP Mode



Cores 3 + 4: RTOS in SMP Mode

2.5 Dual-Core and Multi-Core Systems: SMP Shared Mode operation

On a multi core system Windows utilizes only the first CPU core, the RTOS will use all other CPU cores in SMP mode. Thus, the first core will be shared by Windows and the RTOS. Windows will only get CPU time when the RTOS becomes idle on this core.

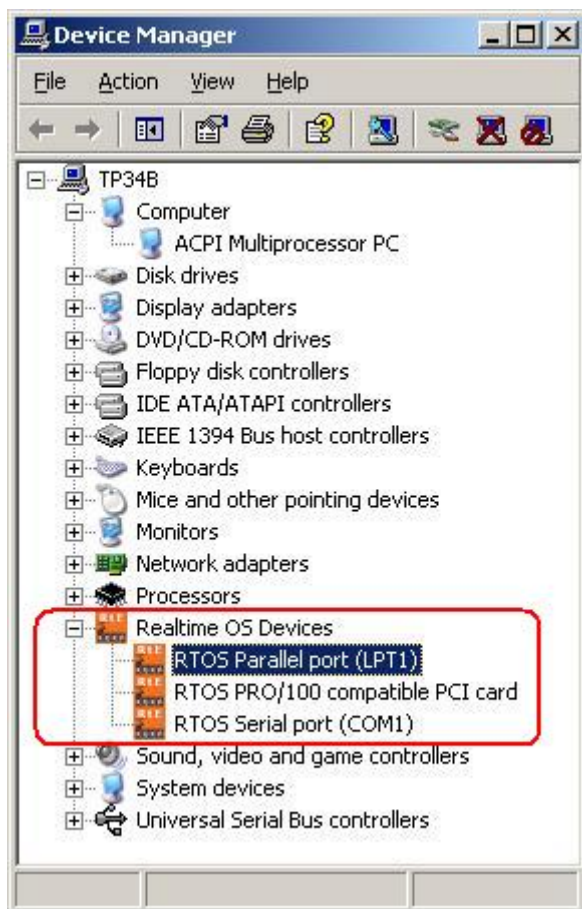


3 Real-time Device Management

To achieve real-time behavior the RTOS will have to directly access its hardware devices. In fact, hardware devices are never emulated, neither in Windows nor in the RTOS. Every specific device, e.g. a PCI network adapter card will, then either be used by Windows or by the RTOS exclusively. All hardware devices which shall be used by the RTOS will be managed by the Windows RtosPnp driver shipped with the KUKA RTOS-VM.

Using the KUKA Real-Time Device Manager tool the user can select which device shall be used by the RTOS and which by Windows.

Within the Windows Device Manager all RTOS devices will then appear in the "Realtime OS Devices" tree:



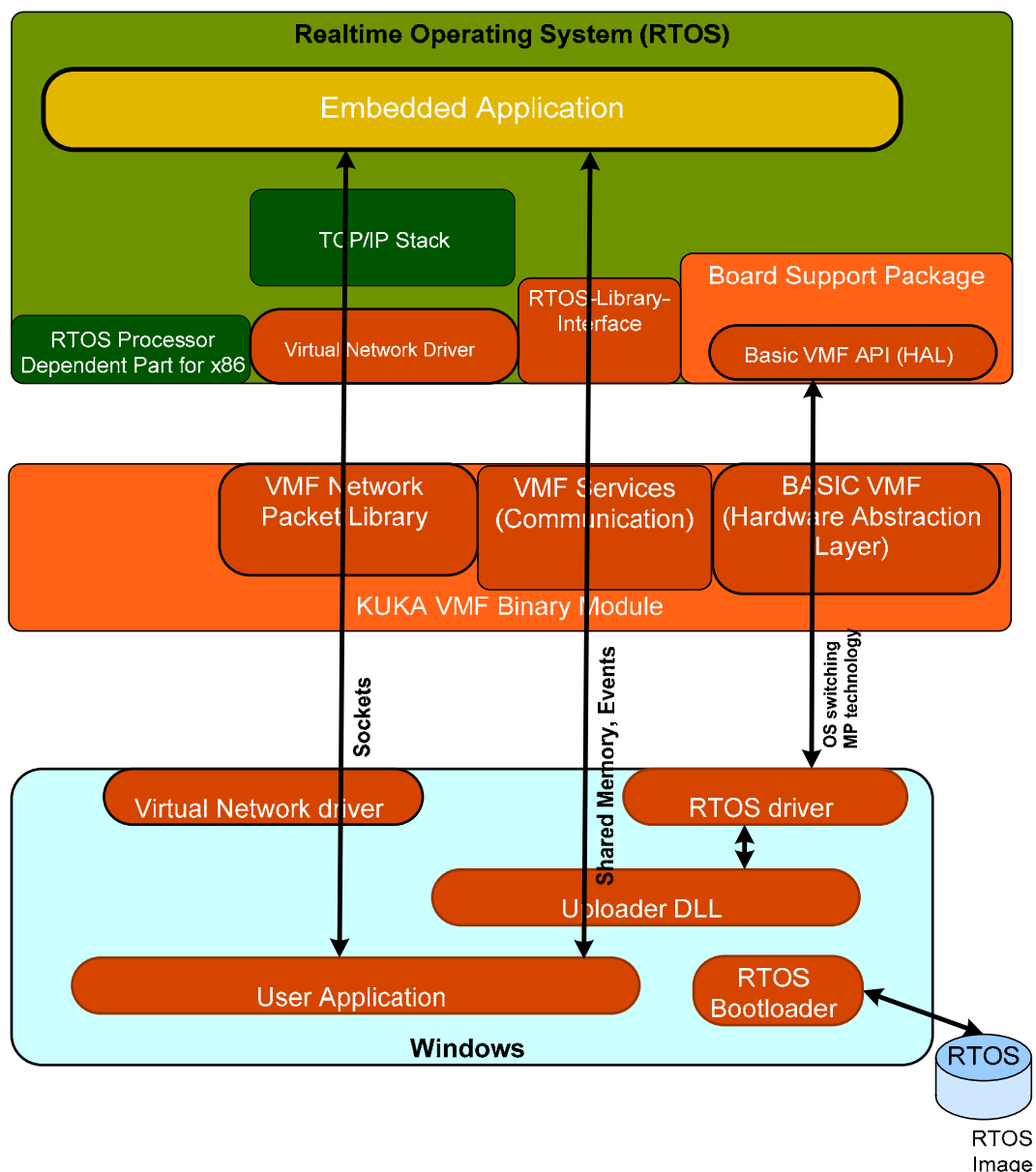
Within the RTOS there are two methods for detecting whether a device can be accessed or not. For PCI devices usually the PCI vendor and device ID can be used. For other devices (as well as for PCI devices) the device name (e.g. RTOS PRO/100 PCI card) can be used.

4 Virtual Machine Framework

Using the KUKA RTOS-VM there is no need to understand the complex hardware of modern PC systems. The basic hardware components of the PC (architecture specific processor registers, timer, interrupt controller, memory handling/partitioning) can be accessed in the real-time software by simply calling the appropriate functions that the RTOS-VM **hardware abstraction layer** (HAL) provides. The application interface between the real-time software and the RTOS-VM is called the Virtual Machine Framework (VMF).

4.1 VMF Architecture

The following figure shows the general architecture of the VMF when a RTOS is embedded within Windows. Besides the basic VMF API (the HAL) which usually is required to build a RTOS BSP (Board Support Package) the VMF contains services for communication between Windows and the RTOS (e.g. shared memory, events, network packet library). On top of the network packet library a virtual network driver can be built which will then provide a virtual network connection between Windows and the RTOS.



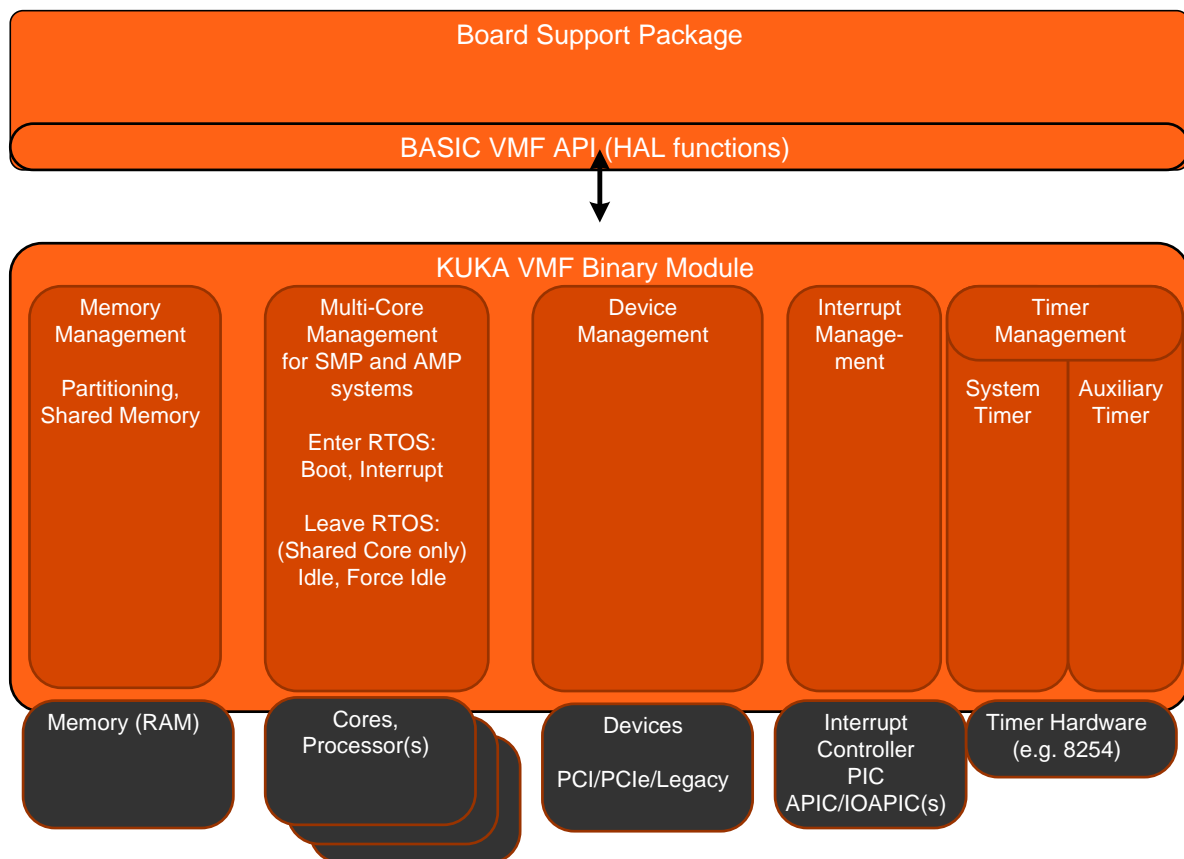
4.2 Basic VMF Services (Hardware Abstraction Layer)

When calling VMF hardware functions the hardware will be directly accessed and not emulated. These functions are called the VMF Hardware Abstraction Layer (HAL) functions.

4.2.1 Overview

The basic VMF services provide a simple programming interface to access the otherwise complex PC hardware.

The following figure shows in detail the basic VMF services which usually are used within a RTOS Board Support Package.



When porting system software (e.g. a RTOS Board Support Package) to run with the KUKA RTOS-VM there is no need to directly access PC hardware like timers or interrupt controllers.

The VMF as well provides a generic method for booting the system software (e.g. a RTOS) and for setting up the RTOS memory context (virtual memory).

When running on multi-core systems the VMF also provides methods for executing a RTOS which supports Symmetric Multiprocessing (SMP).

4.2.2 HAL Services

The VMF Hardware Abstraction Layer offers the following services:

- Multiprocessor Management Services (e.g. start and stop CPU cores)
- Device Management Services (e.g. to determine which physical device is available)
- Interrupt Management Services (interrupt controller abstraction, interrupt id management)
- Timer Abstraction (two virtual independent timers)
- Real Time Clock (Software RTC)
- Time Stamp Timer (a system wide time stamp timer for event logging)

4.3 Communication and other Services

Besides the HAL functions the RTOS-VM provides additional services, especially for communication between the RTOS and Windows:

- Network Packet Library: basic Ethernet data transfer service (interrupt and/or polling mode)
- Virtual I/O Service (to provide a virtual serial channel between the RTOS and Windows)
- Basic Shared Memory Area: Direct access to shared memory area using pointers
- Enhanced Shared Memory areas
 - Write protect option
 - Optionally initialized by a file
 - Optionally saved when the RTOS is stopped (persistent data)
 - Can be used to access a RAM Disk image
- Shared Events: Notification services using named events
- Data Access Synchronization: Interlocked Data Access
- Message Box (RTOS may show a message box on the Windows Screen)
- BSOD notification (RTOS will be informed if a Windows Blue Screen occurs)

Besides communication services the following additional services are available:

- OS Switching Service (when running on a single-core system)
- RTOS configuration services (Registry Services, e.g. for dynamically setting the IP address of the virtual network)
- Date and Time Synchronization (assures the RTOS and Windows run with the same time, either the RTOS or Windows may serve as clock master).
- Performance Measurement Services (CPU utilization measurement)
- Optional Windows Hibernate Support (If the RTOS supports power management it may be included into the Windows Hibernate Mechanism)

4.4 Portability

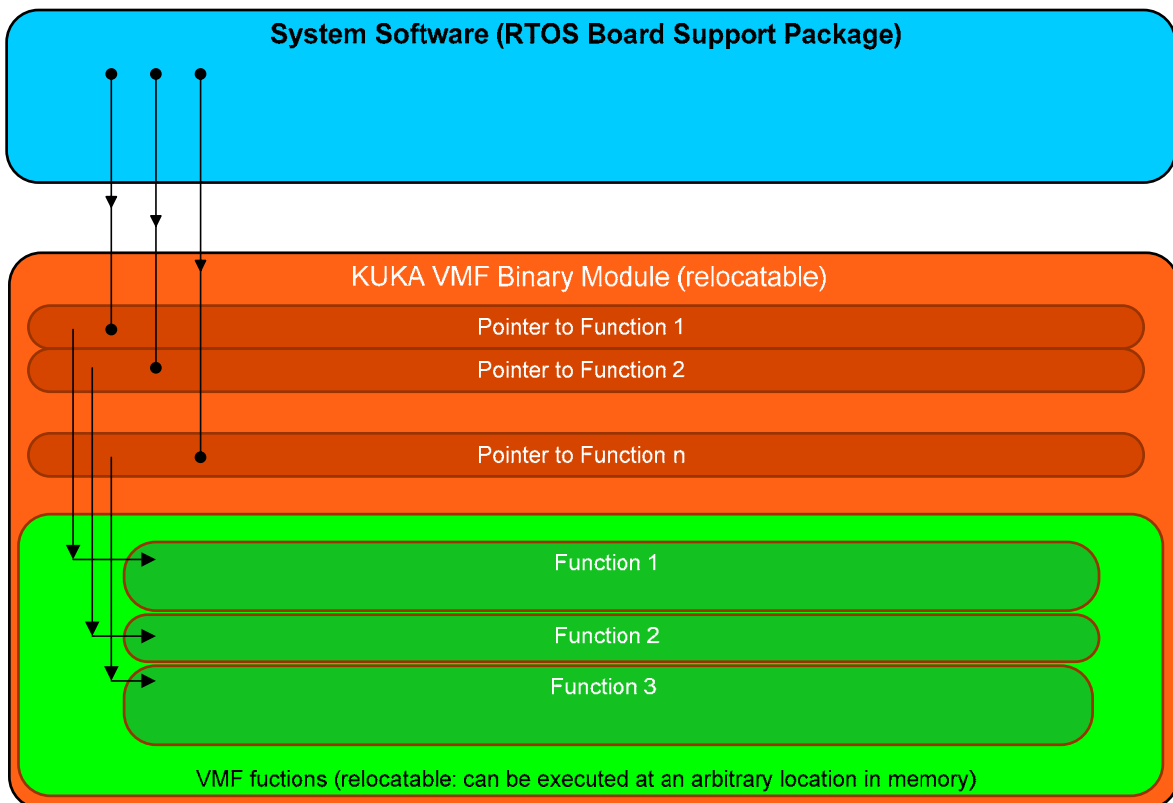
When using standard frameworks or libraries the customer usually gets source-code which in a first step would have to be ported to his specific environment (operating system, compiler, linker). In cases where the supplier does not want to ship the source-code the customer would have to wait until a version for the framework/library is available for his environment.

To avoid these restrictions, the KUKA VMF is not shipped as a library or source code but as a relocatable binary module. This binary module will be loaded by the KUKA RTOS-VM at an arbitrary location in the memory (the VMF code can be executed at any location in memory!).

Every call to a VMF function will then be redirected via well-known locations inside a jump table, this jump table is stored at a well-defined location inside the binary module.

Thus there is no need to port one single line of C language or assembly language code (and no need to add the VMF as an additional library to the customer's environment).

The only requirement is to include one small header file. Within this header file the VMF functions are simply defined as macros which call the appropriate functions using the function pointer in the jump table.



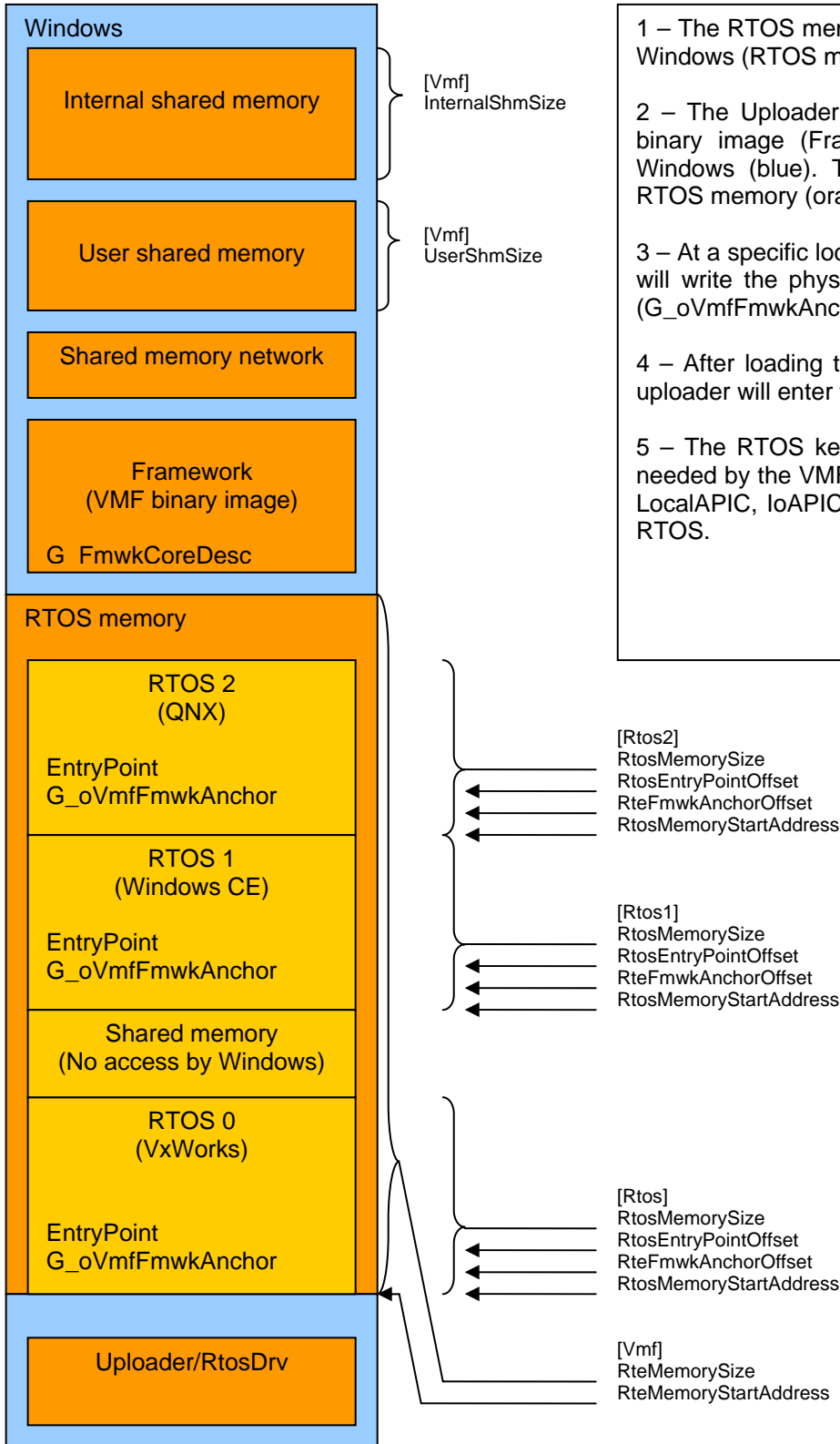
Summarized, using the VMF binary module leads to the following advantages:

- No porting necessary, just include a C header file.
- No change necessary in the system software when new VMF versions are released, just exchange the binary module by the new one.
- The same binary VMF module will be used together with different RTOSes; this ensures a higher quality than if the VMF code would have been ported individually for any RTOS.

4.5 Memory Layout

VMF = Virtual Machine Framework

RTOS Framework = RTOS interface (VMF interface functions)



1 – The RTOS memory area (orange) will not be used by Windows (RTOS memory configuration)

2 – The Uploader (RTOS Bootloader) copies the VMF binary image (Framework) into an area allocated by Windows (blue). The RTOS image is copied into the RTOS memory (orange).

3 – At a specific location in the RTOS image the uploader will write the physical base address of the VMF image (G_oVmfFmwkAnchor).

4 – After loading the RTOS image into the memory the uploader will enter the RTOS boot entrypoint.

5 – The RTOS kernel will then boot. All memory areas needed by the VMF (Internal / User Shm, virtual network, LocalAPIC, IoAPICs etc.) will have to be mapped by the RTOS.

4.6 Conclusion

The VMF defines a virtual machine platform to run one or multiple secondary operating systems (RTOS) on top of a primary operating system (Windows).

The VMF is a binary module which is loaded at a predefined physical address. The interface function entry points are located at fixed offsets within this binary module.

All functions of the VMF are fully relocatable, thus the VMF may be located at any physical address and mapped into the RTOS memory context at an arbitrary location without the need to be recompiled or relinked.

Summarized, using the VMF one gets the following advantages:

- Support for all x86 processors (Intel, AMD): no hardware virtualization processor support (e.g. Intel VT) required.
- Fully virtualized hardware access (via Hardware Abstraction Layer functions). No need to understand the complex PC hardware.
- Either run the RTOS and Windows together on one single-core or use dedicated cores exclusively for each operating system.
- The **same** RTOS image can be run either on a shared or a non-shared CPU core.
- Sophisticated Multi-Core Support
 - Run the RTOS on one single or on multiple cores (SMP)
 - A RTOS can run in SMP mode even on dual-core CPUs

5 The RTOS Library

The RTOS library provides higher level communication services for synchronizing Windows with the RTOS or to exchange data between the operating systems.

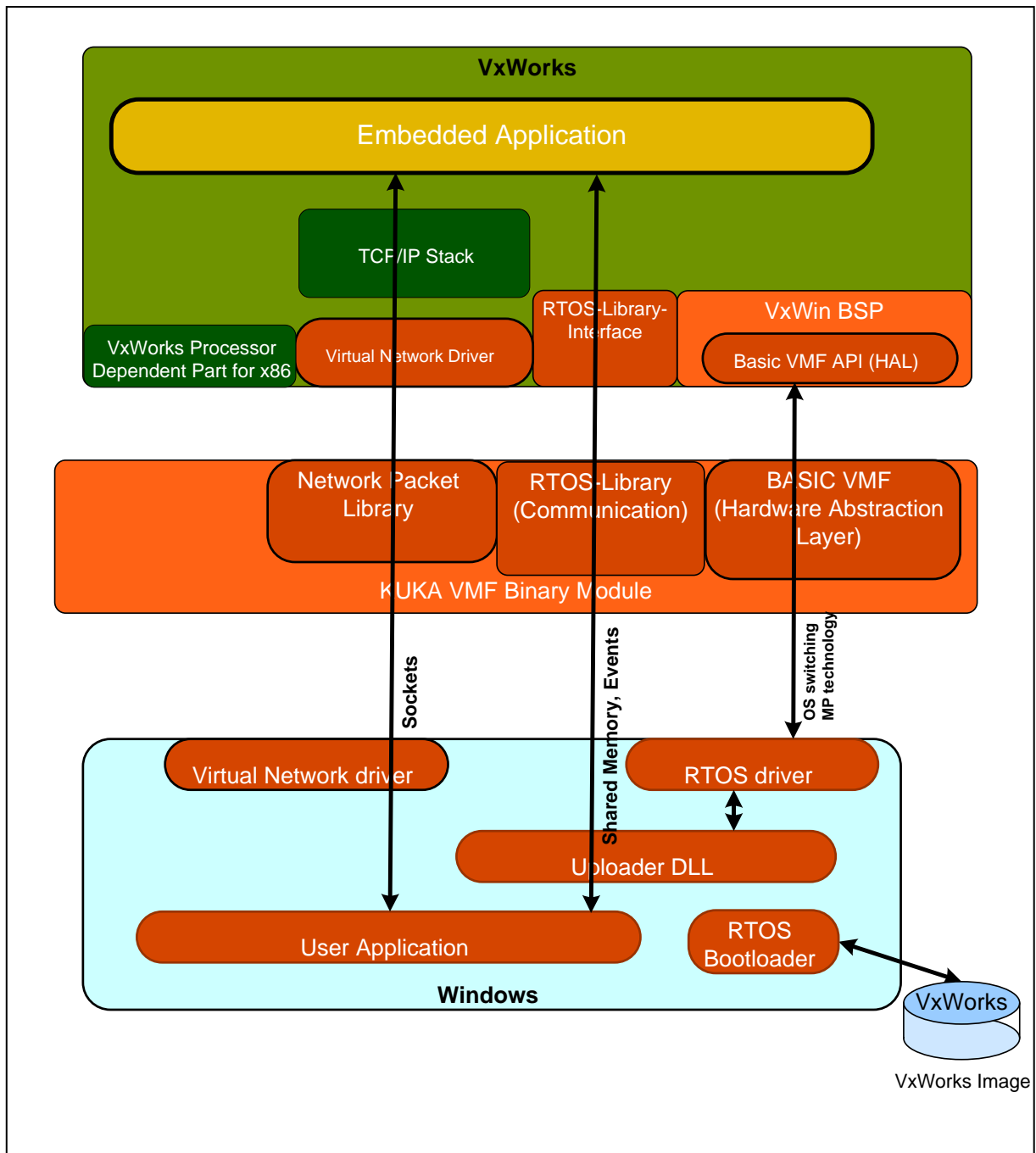
The RTOS library is based on VMF functions which provide the basic communication functionality.

6 Example Implementation: VxWin

VxWin is split into two main components:

- a) The KUKA RTOS Virtual Machine runtime (VMF runtime)
- b) The KUKA VxWin Board Support Package for VxWin

The VxWin BSP provided by KUKA enables VxWin to run on top of the RTOS Virtual Machine. It is using Virtual Machine Framework (VMF) functions to access the RTOS-VM.



7 Example Implementation: RTOS32Win

RTOS32Win is split into two main components:

- The KUKA RTOS Virtual Machine runtime (VMF runtime)
- The On Time Informatik RTVmf-32 component. This component is provided by On Time Informatik.

The RTVmf-32 component provided by On Time Informatik enables the On Time RTOS-32 operating system to run on top of the RTOS Virtual Machine.

It is using Virtual Machine Framework (VMF) functions to access the RTOS-VM.

