Kithara | The Pulse of Real-Time

›kithara.com

Kithara RealTime Suite

- Concept
- Implementation
- Modules

Kithara RealTime Suite

- Real-time extension for Windows
- Comprehensive hardware support
- Beginner-friendly and future-proof
- Modular function library
- Cost-efficient real-time system

Combining Windows and PC hardware: A powerful team.

- Modern and efficient, appealing GUI
- Compatible with excellent driver support
- Widespread, popular with developers and users
- Also widely used in industry

The problem:

- Windows is not real-time capable
- Real-time extension required

The solution:

- Windows and real-time system run simultaneously without negatively impacting each other

Kithara RealTime Suite – The RTOS for Windows

- "Hard" real time on PC hardware
- Hardware: dual- and multicore, 32 and 64 bit
- For Windows 10 (x86 and x64), Windows 11 (x64) and Windows Server 2016, 2019, 2022 (x64)
- Supports any programming language and compiler that is able to generate native code such as C/C++, Delphi

- 32-bit programs on 32-bit Windows
- 64-bit programs on 64-bit Windows
- 32-bit programs on 64-bit Windows (with KiK64)

Comprehensive hardware support

- Current AMD and Intel processors, min. dual core
- Optional hyper-threading with multicore, currently up to 48 logical CPUs usable, beyond that on request
- ACPI (Advanced Control and Power Interface)
- APIC (Advanced Programmable Interrupt Controller)
- Scalable according to requirements
- Generally platform-neutral and hardware-neutral, such as for Ethernet, EtherCAT, CAN, UART, GigE Vision
- Therefore largely free choice of hardware manufacturer (“second source”)

Beginner-friendly and future-proof

- Designed for easy usability
- Familiar programming language and environment
- No separate real-time system required, Windows is retained as platform
- Development and testing under Windows
- Any GUI and libraries usable
- All modules developed with coherent concept (“from the same mould”)

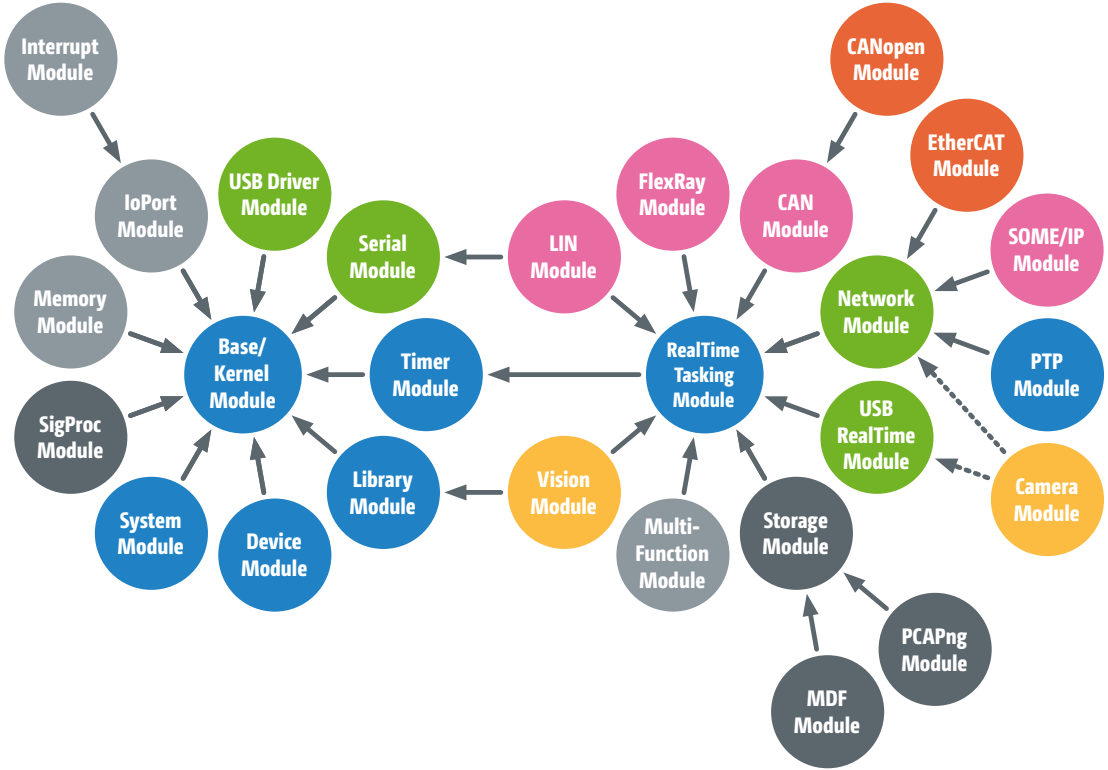
Modular function library

- Functionality (real time and hardware) in SYS driver
- Accessed from application context (DLL driver)
- Header files and import libraries for various compilers and programming languages are included

Consists of more than 20 modules:

- Compiled modularly
- Streamlined solution, geared towards demand
- Extendable at any time
- Continuous further development

Relations between modules



Cost-efficient real-time system

Familiar programming language and environment:

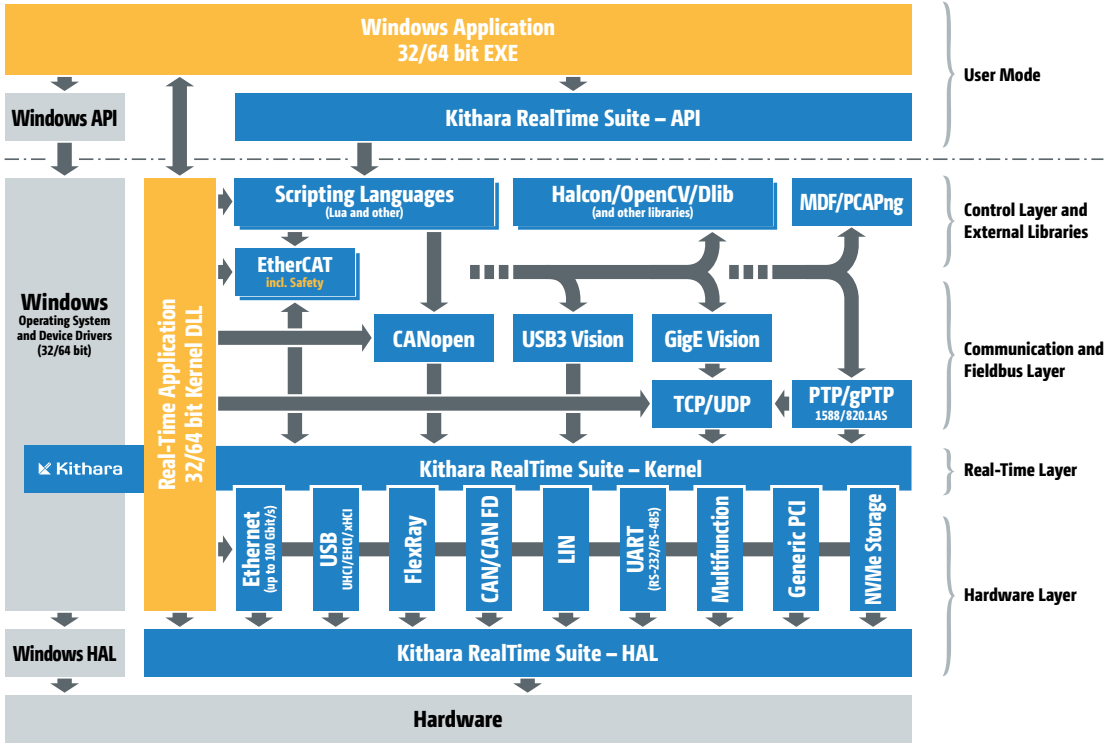
- Next to no additional training required
- Familiar tools, usage, libraries
- Easy migration (“step-by-step to real time”)

Real time under Windows means:

- Easier implementation of GUI and real time
- Development platform = test platform

Customer-specific application is made of two parts:

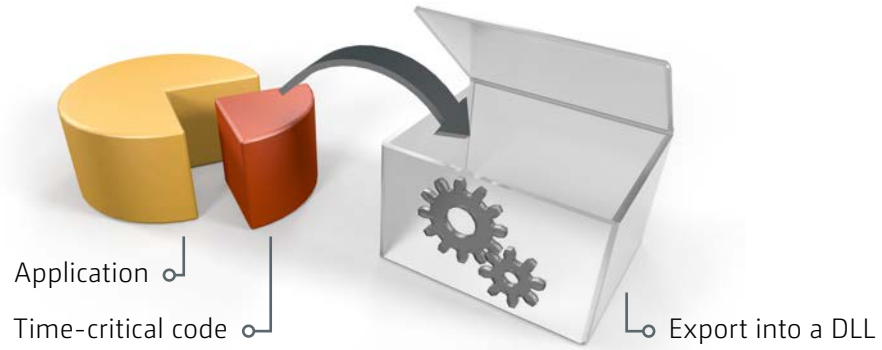
- Windows application (EXE)
- RealTime application (DLL)
- Direct communication between the two possible



Real-time capabilities, but how?

Real-time is only possible at kernel level. But how can application code get there?

With a clean flexible solution: As a DLL.



A detailled description can be found at:

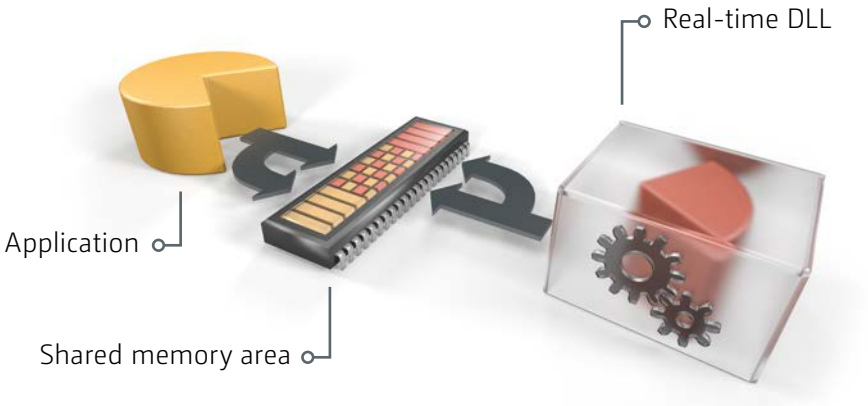
› **Future-Proof Architecture with Kithara RealTime Suite**

How can both parts communicate with each other?

- **Windows application** (EXE) in application context
- **Real-time application** (DLL) in real-time context

Communication via:

- Events
- Shared memory
- Data and message pipes
- Sockets



Usable programming languages

- **Windows application** in application context (EXE):
  Any programming language (e. g. C/C++, Delphi, C#, F#, Java, VB.NET and others) as 32-bit or 64-bit application

- **Real-time application** in real-time context (DLL):
  Compiler has to be able to generate native machine code (x86/x64), such as in case of C/C++ and Delphi as well as match the respective Windows OS (32/64 bit)

Application context vs. real-time context

Most mechanisms are also usable from the Windows context:

- With all programming languages (for example, .NET)
- Easy testing with integrated debugger
- No real time, yet convenient testing capability

Real-time mechanisms from the real-time context:

- Native machine code required (x86 or x64)
- Limited number of programming languages (e. g. C/C++, Delphi)
- “Hard” real time

Distinction is made during function call via flag.

How does this work?

Ingredients:

- **SYS file** from Kithara RealTime Suite (kernel driver):
  contains essential functionality
- **DLL file** from Kithara RealTime Suite (application driver):
  contains access to kernel driver
- **EXE file** from the user (Windows application):
  GUI, user interaction, communication
- **DLL file** from the user (real-time application):
  real-time processing, hardware access, I/O
- Header files, import libraries, INF files:
  regulate connections among themselves

How to start?

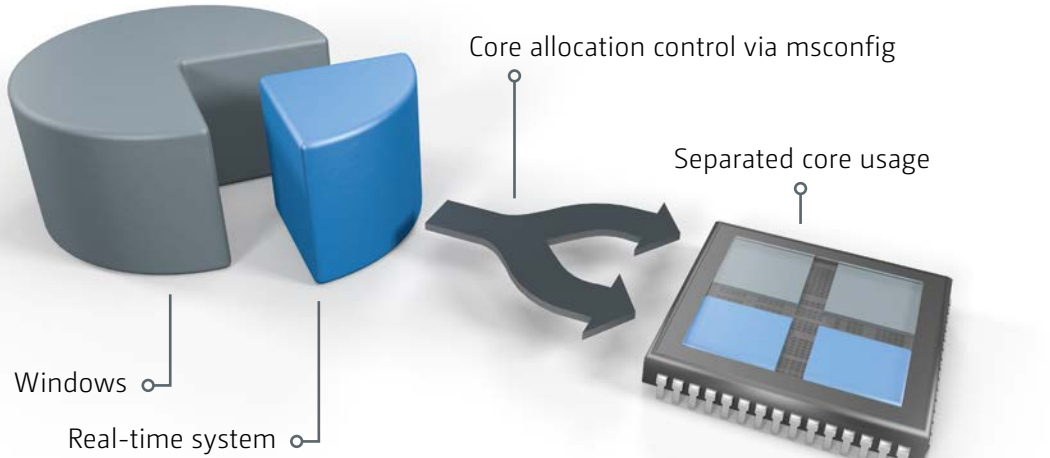
- Kernel driver from Kithara RealTime Suite is loaded during Windows booting process, while staying completely passive in the background; hardware is only addressed when used
- Opening the driver activates the real-time kernel
- Real-time kernel boots the real-time system on CPU cores, determines PC resources, calibrates timers, afterwards:
- Load real-time application in real-time context
- Generate resources (memory, pipes, events)
- Generate callback functions and real-time tasks
- Activate timers, automation, communication etc.

Requires available dedicated CPU cores

Dedicated real-time mode

Windows and real time run on separate CPU cores

- No negative impact on each other
- Can be freely allocated
- One-time setup during installation (with msconfig.exe)



Kithara RealTime Suite – modular system consists of:

- **Real-time system:**
  Timer, multitasking, dedicated real time
- **Communication:**
  Ethernet, TCP/UDP, UART, USB
- **Automation:**
  EtherCAT, CANopen
- **Machine vision:**
  GigE Vision, USB3 Vision, Halcon, OpenCV
- **Automotive:**
  FlexRay, CAN, CAN FD, LIN, BroadR-Reach
- **Data storage:**
  SSD, RAID, MDF, PCAPng, XML
- **Hardware access:**
  I/O, Memory, Interrupt, Multifunction I/O

Real-Time System

- Precise time measurement, short time delay and simple Windows-based timers
  › **Timer Module**
- High-frequency real-time timer functions and Preemptive real-time multitasking on exclusively-used CPU cores:
  › **RealTime Tasking Module**
- Synchronization with other computers or GPS:
  › **PTP Module**

What is real time?

- Immediate **action** to function calls, such as sending of data, setting of I/Os etc. Must not be interrupted by other code (requires high system priority)
- Immediate **reaction** to external events, such as timers, receiving data etc. This needs to immediately interrupt other code (requires high system priority)

Real time under Windows requires the highest system priority before any other Windows activity.

Full control due to dedicated mode.

Dedicated mode: Exclusively-used CPU cores

- Nowadays, CPUs with multiple cores are very common
- Windows is able to run on a fewer number of CPU cores
- On the other available CPU cores, a pure Kithara real-time system is booted

Advantages:

- “Hard” real time due to exclusive CPU allocation (real time is not negatively impacted by Windows or other drivers)
- Hyperthreading usable
- Currently up to 47 logical CPUs usable
- All CPU resources usable (e.g. IO-APIC, PCIe lanes etc.)
- Multi-socket systems: **Kernel NUMA Extension** recommended

Staggered real time

Real time with callback functions:

- Multiple callback functions can be announced, without priorization between them, however. Every callback is completely executed until the end and not interruptible.

Real-time multitasking system (additionally):

- Real-time tasks are prioritized
- Preemptive, with priority adjustment and priority inheritance
- Adjustable load distribution to multiple CPU cores

Recommendation: Real-time tasks

How do external events (timers, data, etc.) trigger a real-time reaction?

- Events are set:
  activation of Windows threads or real-time tasks
- Callbacks are invoked:
  in application context or real-time context
- Real-time tasks are triggered on exclusively-used CPU cores

Recommendation: Real-time tasks

Timer Module

High-precision measurement of system time

- Various formats such as milli-, micro- or nanoseconds
- Bound to system start, absolute, world time (UTC) etc.
- Custom-defined time formats
- Short-time delay in 0.1 µs increments
- All internal PC clocks are addressable such as PC timer, PM timer, HPET, TSC
- All clocks are long-term synchronized
- Internal 96-bit conversion: No overflow
- Additionally: simple Windows-based timers (no realtime!)

More information about the › **Timer Module**

RealTime Tasking Module

- High-resolution real-time timers, one-time or cyclical
- Programmable resolution in 0.1 µs increments
- Real-time context accuracy in microsecond range
- Timer frequencies up to and above 20 kHz possible
- Preemptive priority-based real-time multitasking
- Tasks with up to 255 priority levels, dynamic
- Priority inheritance for avoiding priority inversion
- Multiple tasks per priority level (round robin)
- Semaphores, mutexes, events, suspend/resume
- Tasks freely allocatable to exclusively used CPU cores

More information about the › **RealTime Tasking Module**

PTP Module

- Precise PC synchronization according to IEEE 1588v2, based on raw Ethernet/IP/UDP
- Synchronization locally or globally via GPS
- Best master clock algorithm
- Achievable accuracy with hardware timestamping: < 1 µs

- **NMEA Extension:** Synchronization with NMEA 0183
- **gPTP Extension:** Synchronization with “generalized Precision Time Protocol” (standardized PTP profile according to IEEE 802.1AS)

More information about the › **PTP Module**

Communication

- Ethernet communication, TCP and UDP sockets:
  - **› Network Module**
- CAN communication:
  - **› CAN Module**
- Serial communication via COM ports:
  - **› Serial/UART Module**
- USB device drivers:
  - **› USB RealTime Module**
  - **› USB Driver Module**

Network Module

- Ethernet communication (currently up to 100 Gigabit/s in real time)
- Any Ethernet protocol usable, including IP
- Connectionless (UDP) or connection-oriented (TCP)
- Query-driven or event-driven communication in real time
- Automatic address inquiry due to ARP support, IP and MAC multicast, broadcast, jumbo frames
- manufacturer-neutral API (e. g. for Intel, RealTek)

- **Thunderbolt Extension:** Support for external hardware
- **Network PTM Extension:** Passthrough and mirroring (communication with Windows, data traffic monitoring)

More information about the › **Network Module**

Serial/UART Module

Serial communication via UART/COM:

- Either via UART-16550-compatible hardware supported by specialized drivers (UART API = real time)
- Or via any COM ports with installed Windows drivers (COMM API = no real time)

More information about the › **Serial/UART Module**

USB RealTime Module

- Control USB devices up to USB 3.2 in real time via direct and exclusive access to the XHCI controller
- Low, Full, High and SuperSpeed
- Control, bulk, interrupt and isochronous transfer

More information about the › **USB RealTime Module**

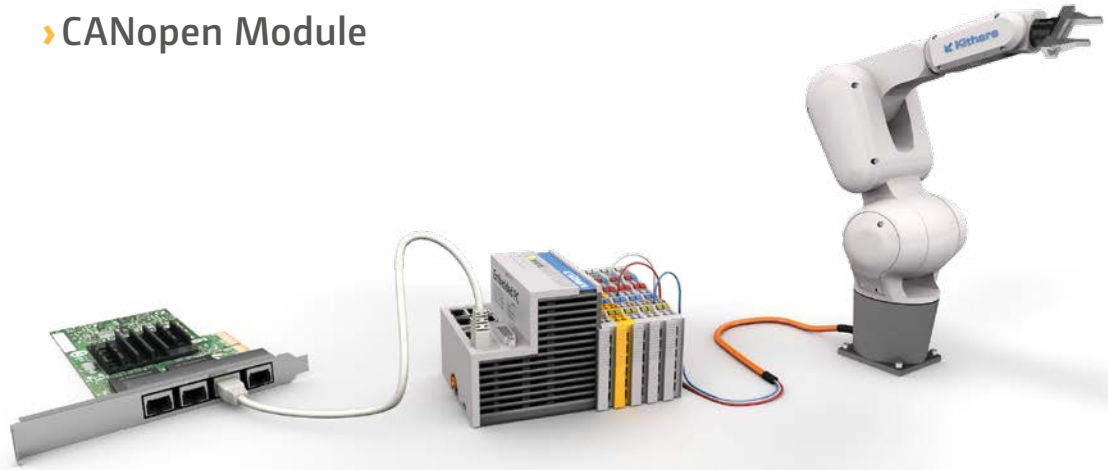
USB Driver Module

- Control USB devices up to USB 3.2 via Windows driver stack
- Low, Full, High Speed
- Control, bulk, interrupt and isochronous transfer

More information about the › **USB Driver Module**

Automation

- Real-time EtherCAT master
  - › **EtherCAT Module**
- Real-time CANopen master
  - › **CANopen Module**



EtherCAT Module

- EtherCAT master in real time, up to 50 µs cycle time
- Automatic EtherCAT topology detection
- Process data and service data communication (PDO+SDO, mailbox communication, CoE

The basic system is extendable with various functions, among them:

- Distributed Clocks (DC), Safety-over-EtherCAT (FSoE), Ethernet-over-EtherCAT (EoE), File-access-over-EtherCAT (FoE), ServoDrive-Profile-over-EtherCAT (SoE)
- Hotplug capability for dynamic topologies
- Cable redundancy

More information about the › **EtherCAT Module**

EtherCAT Module

Also available:

- **EtherCAT PC Slave Device Extension:** The PC itself becomes an EtherCAT slave. PC plug-in card, programmable as EtherCAT slave, e. g. ESD ECS-PCIe/FPGA.

- **EtherCAT Automation Protocol:** EtherCAT-like communication between PCs. Real-time data traffic with up to 100 Gbit/s via regular network cards.

CANopen Module

- CANopen master in real time
- Automatic CANopen topology detection
- CANopen slave state management
- Process data and service data communication (PDO+SDO, mailbox communication
- Also usable with EtherCAT terminal EL6651 by Beckhoff

More information about › **CANopen Module**

Machine Vision

- Real-time image capture with GigE Vision and USB3 Vision:
  › **Camera Module**
- Real-time image processing with Halcon and OpenCV:
  › **Vision Module**

Camera Module

- Image capture with USB3 Vision cameras (via XHCI controller)
- Image capture with GigE Vision cameras (up to 10 Gbit/s Ethernet)
- Configuration according to GenICam 2.0
- Event-based or request-based, error management, hotplug capability, multiple cameras simultaneously
- GigE Vision 2.0: time synchronization, link aggregation

- **Camera Acceleration Extension:** Support for GigE Vision frame grabber acceleration card PLC2 PGC-1000 (up to 4×10 Gbit/s) for almost complete CPU-offload

More information about the › **Camera Module**

Vision Module

Consists of Library Module and the following extensions:

Halcon Extension

- Image processing in real time with Halcon (proprietary: MVTec)
- Programming in C++ or with HDEV scripts
- Supports versions 10 to 13, 18.11 and 20.11
- Requires separate license by MVTec

OpenCV Extension

- Real-time image processing with OpenCV in C++ (open source)
- Supports versions 3.0, 3.4, 4.1 and 4.2

Automotive

- FlexRay: Windows PC as full-featured FlexRay node
  › **FlexRay Module**
- CAN/CAN FD: real-time communication via CAN
  › **CAN Module**
- LIN: Master or slave node via UART interface
  › **LIN Module**
- BroadR-Reach: Automotive Ethernet
  › **BroadR-Reach**

FlexRay Module

- Real-time FlexRay communication
- Independently combinable FlexRay channels
- Windows PC as full-featured FlexRay node
- Use as leading or following start node
- Baud rate flexibly adjustable (2,5/5/10 Mbit/s)

More information about the › **FlexRay Module**

CAN Module

- CAN communication in real time with specialized drivers
- CAN 2.0A and 2.0B
- Data rate increase with **CAN FD Extension**

- Vendor-neutral API:
  Supports interfaces from various manufacturers:
  (Peak, Star Cooperation, Ixxat, EMS, ESD, Kvaser)
- All received CAN telegrams are assigned accurate time stamps
- Real-time reaction to received CAN telegrams
- Filter handler installable

More information about the **› CAN Module**

LIN Module

- LIN data exchange in real time
- Easy implementation of master or slave nodes with a regular PC
- LIN transceiver required
- Bus collision detection
- Automatic checksum validation (LIN version 1.x and 2.x)
- Data rate of up to 20 Kbit/s
- Guaranteed latency times
- Reception of LIN messages can be done with callbacks or polling

More information about the › **LIN Module**

BroadR-Reach

- Physical Ethernet layer specifically for automotive networking
- High data rates for large amounts of data
- High scalability
- Cost-efficient in wiring and programming
- Usable as central backbone network
- BroadR-Reach as pure physical Ethernet layer is available as **Network BroadR-Reach Extension**

More information about › **BroadR-Reach**

Data Storage

- Fast file access to SSDs
  - › **Storage Module**
- Simple structured measurement data storage
  - › **PCAPng Module**
- Complex structured measurement data storage
  - › **MDF Module**

Storage Module

Very fast access to SSDs with NVMe interface:

- Supports SSDs from various manufacturers
- Storage with UDF file system
- Access can be switched between real time and Windows

Storage RAID Extension

- Further data rate increase to > 10 Gbyte/s
- Support for RAID 0
- Linking of 2, 4 or more SSDs

More information about the **› Storage Module**

PCAPng Module

Simple structured storage of measurement data from real-time context in PCAPng format:

- Viewable in Wireshark
- Storage in UDF file system
- Accessible from either real time or Windows

More information about the › **PCAPng Module**

MDF Module

Complex structured storage of measurement data from real-time context in MDF

- Support for MDF 4.1, downward compatible with earlier versions
- Writing of virtually arbitrarily large files (2^{64} bytes)
- Storage of raw messages from CAN, LIN, FlexRay and automotive Ethernet

More information about the › **MDF Module**

Hardware Access

- Access to I/O ports:
  - › **IoPort Module**
- Access to physical memory:
  - › **Memory Module**
- Interrupt handling:
  - › **Interrupt Module**
- Digital and analog I/O of multifunction cards:
  - › **MultiFunction Module**

Driver Programming

Driver development requires:

- I/O access
- Access to physical memory
- Handling of interrupt requests (IRQ)

With this, basically any PC hardware can be addressed.

Additional functions of the › **Base/Kernel Module**:

- Listing of devices by specific type
- Determination of device and driver information
- Switching drivers

IoPort Module

- Access to any I/O registers of a PC system
- I/O access from application context is also possible, either directly (after unlocking address; system-dependent) or indirectly (by switching to Kernel mode)
- Determination of PCI configuration data (current base adresses and IRQ numbers) based on plug&play mechanisms
- Determination of resource information

More information about the › **IoPort Module**

Memory Module

- Access to external physical memory (dual port RAM)
- Allocation of PC memory for external hardware (DMA)
- Copying of memory areas between application and real-time context

More information about the › **Memory Module**

Interrupt Module

Handling of hardware interrupts:

- Desktop: PCI and PCIe cards, ISA bus, PC104, PC104+ etc.
- Laptop: CardBus (PCMCIA), ExpressCard/34 or /54

Two operation modes:

- Either usage of operating system mechanisms for interrupt handling (no real time = regular driver)
- Or real-time interrupts (PCIe hardware required)

More information about the › **Interrupt Module**

MultiFunction Module

- Support for multifunction cards (PCIe bus) with vendor-neutral API
- Digital I/O, bitwise or wordwise
- Analog I/O as single value, channel sequence, limited series of sequences or continuous mode with switch buffer interrupt
- Real-time drivers for cards from various manufacturers, e. g. National Instruments (NI) X-Series

More information about the › **MultiFunction Module**

Base Functions

- General functions and resources:
  - › **Base/Kernel Module**
- Interception of system events:
  - › **System Module**
- Windows device interfaces:
  - › **Device Module**
- Standard function library in real time:
  - › **Library Module**

Base/Kernel Module

- Opening of real-time driver
- Error handling
- Fetching system information
- Debugging assistance
- Determining hardware devices, switching drivers
- Callback functions
- Event objects, mutex objects, semaphores
- Shared memory
- Pipes (data pipes, message pipes)
- Function execution
- Real-time memory management

More information about the › **Base/Kernel Module**

System Module

- Interception of system events such as protection faults, system crashes at kernel level or blue screen handler (failsafe handler)
- Hardware safety shutdown

More information about the › **System Module**

Device Module

- Provides a Windows programming interface for device communication: CreateFile, ReadFile, WriteFile, DeviceIoControl, CloseHandle
- Freely selectable device names, e. g. for virtual COM ports

More information about the › **Device Module**

Library Module

Real-time function library with

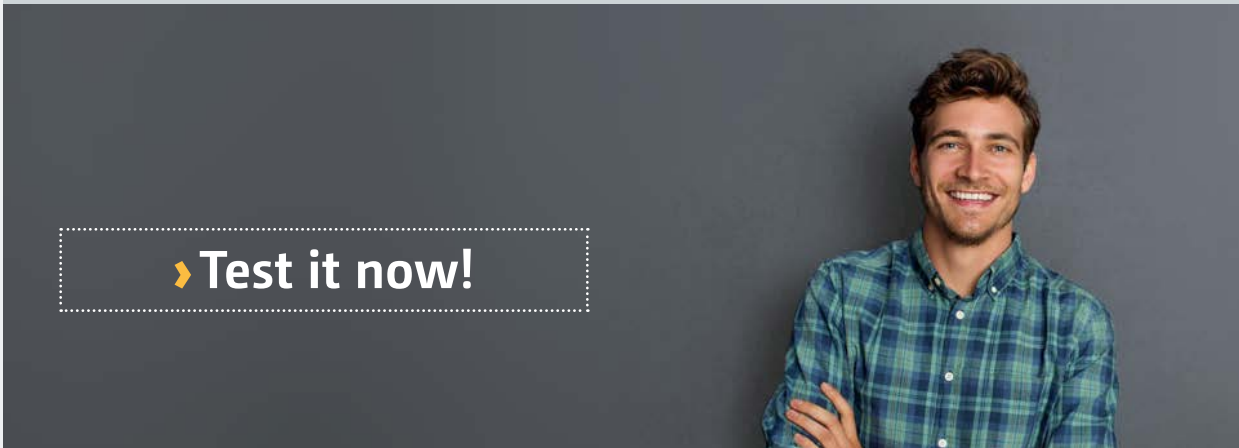
- Mathematical/trigonometrical functions
- String/memory functions
- Access to XML files

- **Halcon Extension:** Image processing library by MVTec
- **OpenCV Extension:** Open source image processing library
- **DLib Extension:** Machine learning

More information about the › **Library Module**

Kithara RealTime Suite: What to do next?

Request the free test version with complete functional range for a testing period of 90 full days!
Contact us to talk about your project, we will gladly advise you.



These are some of our customers

These are some of our partners

Contact us

Kithara Software GmbH
Alte Jakobstraße 78
10179 Berlin

Phone: +49 30 2789673-0
Fax: +49 30 2789673-20
› info@kithara.com
› kithara.com

© 1996–2023 Kithara Software GmbH