# PXROS-HR a Safty-Framework for embedded Systems

HighTec EDV-Systeme GmbH

Feldmannstraße 98
66119 Saarbrücken

1. Requirements of safety critical systems
    - Software Developing without Memory Protection
    - Base concepts of PXROS-HR
2. Benefits of TriCore Memory Protection Unit
    - Privilege Modes
    - Memory Protection Registers
3. MPU Usage in PXROS-HR
    - Characteristics of PXROS-HR
    - Migration
    - Function Integration
4. IEC 61508 Designer

# Requirements

- Prove correctness of software
- Operational availability

  ⟹ Functional Safety or Fail Safe
- Robustness and Testability
- Reduced complexity

# Safe system

## Safe system behaviour

If faults occur, the safety function will still be performed. The faults will be detected in time to prevent the loss of the safety function.

## Safe System

A safe system according to IEC 61508 is a system, that is always in a safe state

## Safe State

A state is safe, if the risk of an erroneous behaviour is tolerable.

## Risk

Error probability * extent of damage

# Problems without Memory Protection

- Enabling or disabling interrupts
- Full read/write access to the memory

  $\Longrightarrow$ Integrity is at risk
- Third party software cannot be integrated safely

  $\Longrightarrow$ Product liability
- Error propagation can cause loss of safety functionality

  $\Longrightarrow$ Fail Safe instead of Functional Safety
- Detection, evaluation and handling errors is difficult

  $\Longrightarrow$ Cost for testing
- Complexity through adverse effects

  $\Longrightarrow$ No modular testing

# Solution = Encapsulation

- Encapsulation of each component
- Each component can only access his own resources
- Encapsulation and its controlling have to be part of the OS
- The communincation mechanism have to be part of the encapsulation
- The encapsulation must be supported by hardware

# Base concepts of PXROS-HR for encapsulation

- PXROS is an object oriented system
- No interrupt disables and latencies
- All resources are controlled by the system
  - Time
  - Memory
  - Objects
- Each task has his own address space
- Communication only by message passing and events

# Concepts

**PXROS-HR**

implements a safe environment to execute any application

- Encapsulation of data and control of resources avoid error propagation
- Only access data which are necessary to perform a task
- Usage of assigned resources
- Communication via message passing including permissions

# Features of PXROS-HR

### Memory protection of the TriCore

PXROS-HR uses the TriCore protection mechanism to implement a hardware controlled separation of Task address spaces

### hardware requirements

- Distinction between privileged and non privileged modes
- The most privileged mode (supervisor mode) is reserved for the PXROS-HR kernel
- All tasks are executed in a less privileged mode (User-0 and User-1)
- Memory protection mechanism with a small granularity
- The TriCore protection mechanism allows the protection of memory areas of any size

# Features of PXROS-HR

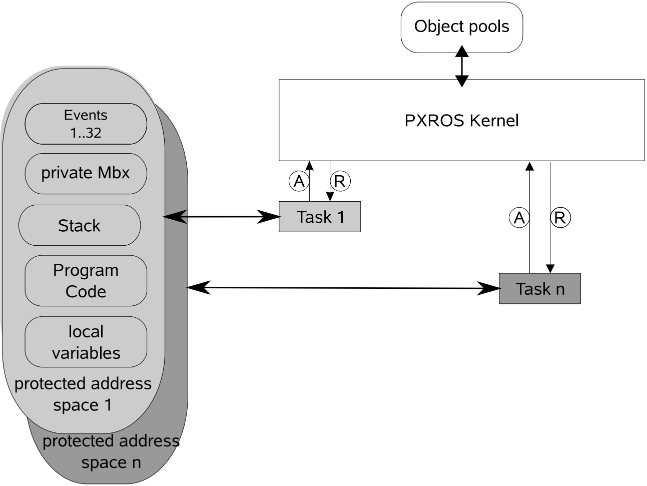
## Add protection areas to a task

add memory protection areas to the task context on creation.

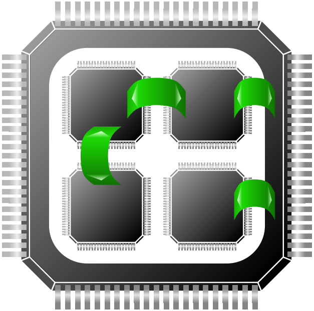
## Detect, evaluate and handle error

Add protection fault handling into the PXROS kernel If a protection trap occurs, the trap handler checks if the trap address is covered by a protection area. In this case the protection registers are changed temporarily and the trapped task can continue execution.

# PXROS Tasks


Object pools
PXROS Kernel
A
R
Task 1
A
R
Task n
Events
1..32
private Mbx
Stack
Program
Code
local
variables
protected address
space 1
protected address
space n

# Benefits of TriCore Memory Protection Unit

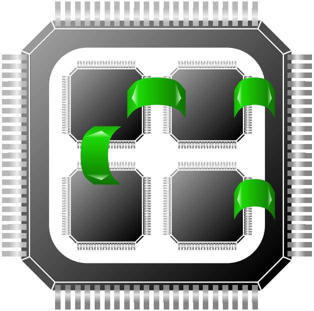


## Virtualisation

- Encapsulate components using MPU
- Components run on 'virtual' processors
  - No error propagation
  - Reduced complexity
  - Modular testing
  - Functional Safety
- Detect and handle software errors

⇒ Integrity of systems is kept even if one or more components fail

# Memory Protection Unit of TriCore



Different privilege modes of TriCore

- Supervisor mode
- User-1 mode
- User-0 mode

Tasks may run in User-1 or User-0 mode.

# Privilege Modes

### Supervisor mode

- Access to all peripherals
- Read/write access to core registers

### User-1 mode

- Access to regular peripherals
- Enabling and disabling of interrupts

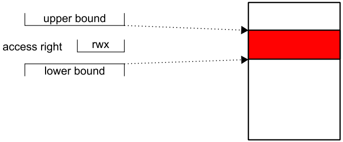
### User-0 mode

- No access to peripherals
- No enabling and disabling of interrupts

# Memory Protection Unit of TriCore

## Memory Protection

Encapsulate components using (2 code and 4 data) protection registers in supervisor or user mode.



Each protection register configures the scope of a memory view window

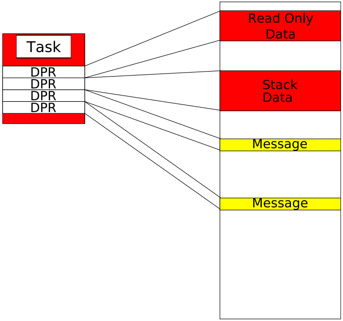
- upper bound
- lower bound
- control (read, write, execute)

⟹ TriCore MPU increases safety and reduces cost for testing

# Benefits of Memory Protection Unit

- Encapsulated access to memory and peripherals
- Simple and safe function integration
- Functional Safety
- Reduced complexity (no adverse effect)
- Illegal access will be detect via MPU
- Reduced costs for testing
- New dimension of software quality

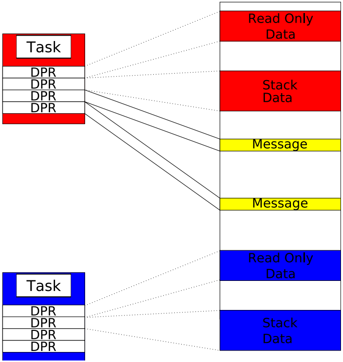
# Memory Protection in PXROS-HR



- Each component (PXROS task) has its own protection register context
- PXROS-HR manages the switching of memory protection of components
- An access violation leads to an error handling, e.g. the component is suspended

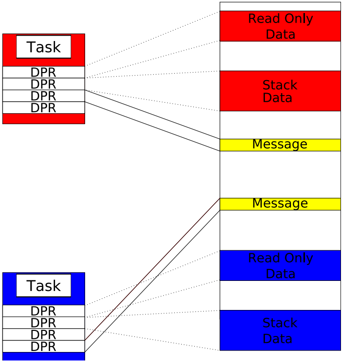
⇒ Stack overflow can be detected

# Memory Protection in PXROS-HR



- Each component (PXROS task) has its own protection register context
- PXROS-HR manages the switching of memory protection of components
- An access violation leads to an error handling, e.g. the component is suspended

# Memory Protection in PXROS-HR



- Each component (PXROS task) has its own protection register context
- PXROS-HR manages the switching of memory protection of components
- An access violation leads to an error handling, e.g. the component is suspended

# Characteristics of PXROS-HR I

- Virtualisation of peripherals

  ⟹ Portability
- Task can contain different software (AutoSAR, simple C-Code)
- Framework for testing software

  ⟹ No error propagation
- Abitrary granularity of memory protection

  ⟹ Efficient and safe

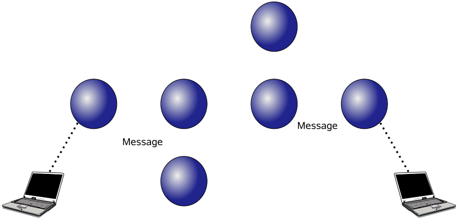
## Testability

- Dynamic loading of components
- Debug components in a running system
- Components can be debugged independently

# Migration

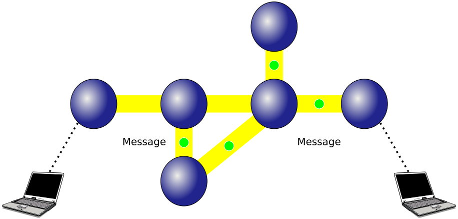
- Capsule without OS or with a different OS (e.g. OSEK)
- Put software into large capsule
- Modularize software step-by-step into capsules
- Allocate a capsule for supplied software

# Function Integration



- Reserve capsules for OEM components
- Configure mode for the capsule
- Add communication channel to the system
- Execute and test new component

# Function Integration



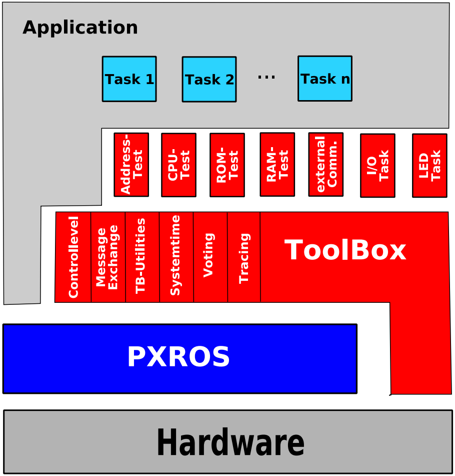
- Reserve capsules for OEM components
- Configure mode for the capsule
- Add communication channel to the system
- Execute and test new component

# Meta model of a ToolBox application

- Unique concept of components
- Tasks are organized in components
- Finite state machines to model the task's characteristics
- Tasks and other model elements (handler, classes, timeouts, etc.) are organized in a hierachic structure of components
- Unique, small communication interface: message objects and commands
- Standard components are used following a modular design principle

⇒ Toolbox is certified with SIL-4 with PXROS

# PXROS-HR ToolBox



- Layer architecture between application and PXROS
- Services above PXROS
    - unique communication (middleware)
    - system time, utilities, etc.
- Standard components
    - Safety (runtime tests)
    - Communication (protocols)
- Domain specific reference architecture for applications
    - Unique components/tasks

# SIL-4 certified ToolBox

Safety

Railway, certified (EN50126/28/29)

Proven in use

More than 10 projects and worldwide 10.000 system in operation

Efficiency

Focusing on safety and testabillity

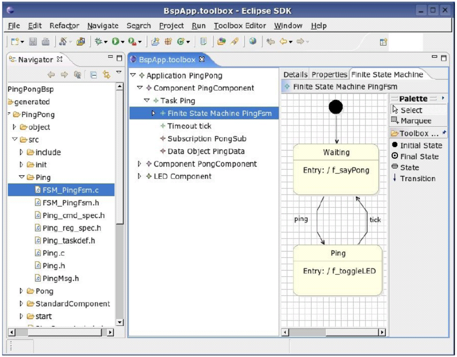
Operating experience

Extract re-usable functionality

⟹ Validated standard components

# Features of the ToolBoxDesigner

## Toolbox designer is graphical frontend of toolbox



- Based on the meta model of ToolBox
- GUI for simple, error reducing application model editing
- FSM (req. of IEC 61508)
- Code generators produce
  - C-Code.
  - Artefacts for the build process
- Integrated in eclipse