Design of an open-robotics low-level framework for the control of complex mechatronics devices

P. Laguillaumie, P. Vulliez,

RoBioSS, Pprime Institute / Auctus Inria

ITECA

Design of RTRMAC

RTRobMultiAxisControl: RTRMAC

- *Group of libraries, user extendable*
- *Multiaxis and multirobot control*

- *Based on hardware Real-Time constraints*
- *Respect of PLC standards*
- *Manufacturer independent*

=> "open robotics"

- *CNRS software patent*

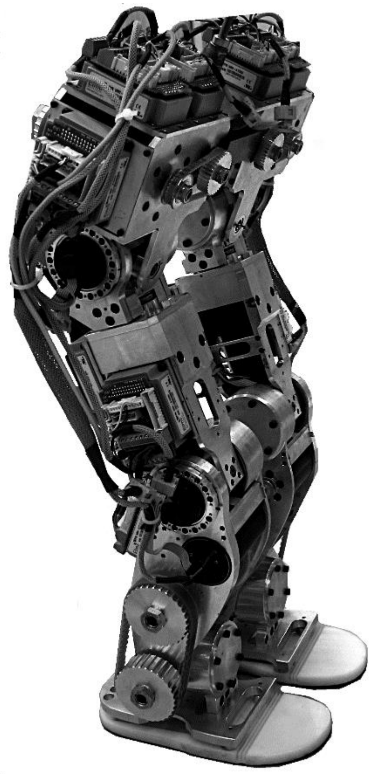
Presentation overview

- *Context*
- *Foundations*
- *"Manufacturer independent" feature*
- *Single axis / coordinated axis managers*
- *Control of a humanoid robot*
- *Conclusion*

Context

Context: situation in 2012

- *A different control technology for each experimental set-up*
  - *Humanoid / BIP: VxWorks + Inria ORCCAD*
  - *Humanoid / TIDOM: specific drive boards over CAN bus*
  - *Manipulation / LMS hand: CNC Galil Controllers*
  - *Large scale printing robot: NI Labview RT*
- *Partnership with B&R Automation / experience with industrial servodrives and PLC programming*



Context: strategy change

Systematic use of industrial-grade hardware
Creation of a single low-level control solution

- *Multi-axis control for reliable dynamic response to environmental disturbances*
- *Control of industrial robot architectures*
- *Force/velocity/position axis control*

With respect to:

- *Standards compliance*
- *Optimal use of the power drives*
- *Code reusability and maintainability*
- *Hard Real Time*
- *Feasible industrial transfer*

Evolution

Steps:

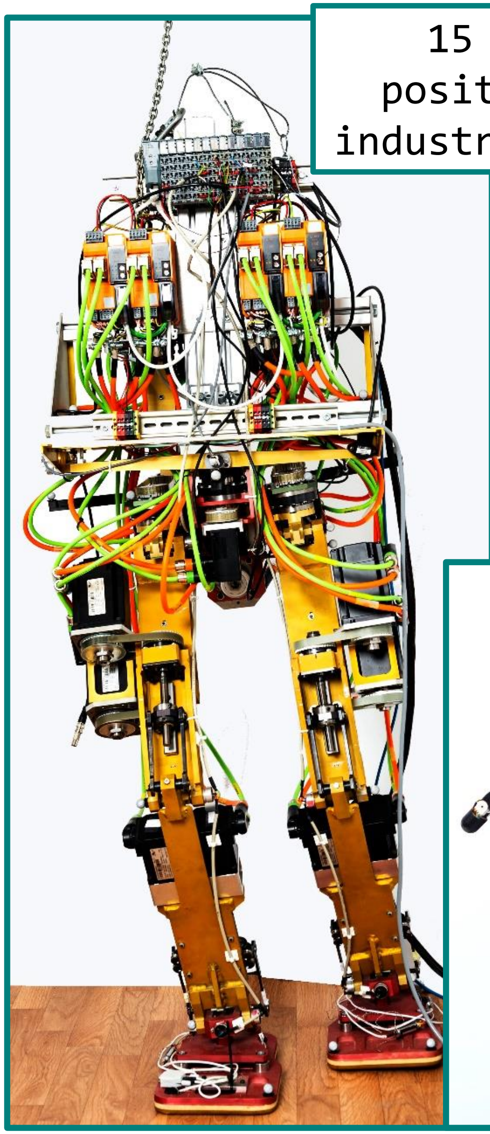
v1: 100% PLC world > v2: C >

v3: C++ / PLCopen coding recommandations

Currently used for all our mechatronics designs

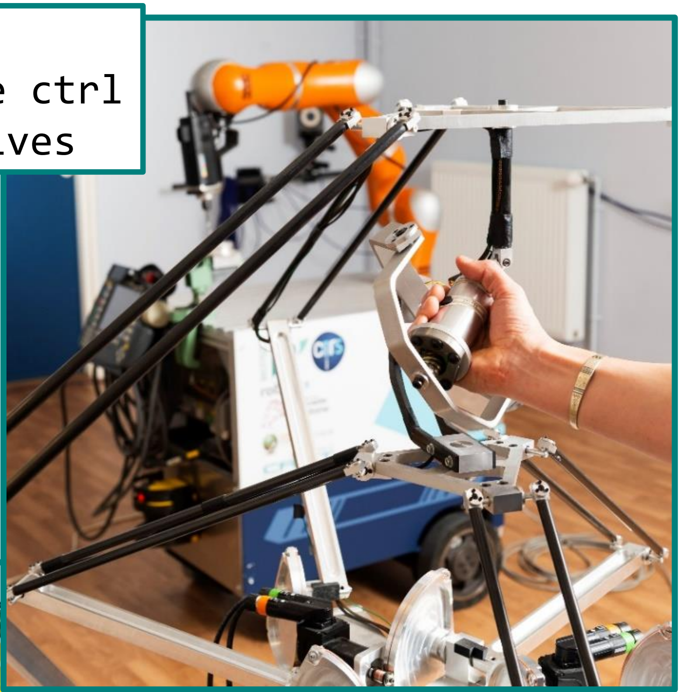
- *Robotic hands*
- *Cobotics with industrial arms*
- *Haptics and motion learning devices*
- *Test benches*
- *ORHRO humanoid*

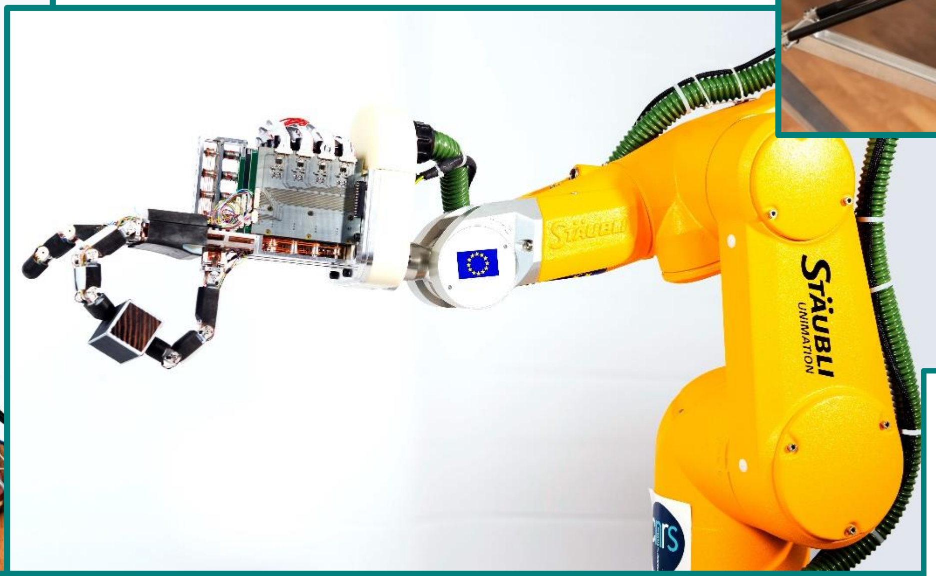
Some applications



15 axes /
position ctrl
industrial drives

6 axes
position / force ctrl
industrial drives





16+6 axes
position ctrl
robot cabinet
+ DC motors

Foundations

Fondations

- *PLC Real-Time hardware (IEC61131)*
- *1 Timer (RT Industrial Ethernet bus)*
- *Respect of main PLC standards (PLCopen international organization)*
  - *PCL compatiblity (sofware model & languages)*
  - *PLCopen Motion Control*

- *OOP (encapsulation, heritage, template)*
- *PLCopen Compliant Libraries requirements*
- *OPC-UA, OPC-UA/TSN*

PLC compatibility

- RT scheduler and tasks run periodically
  - A task class system controls the execution periods and the tasks' priorities
  - Variable sharing among tasks and I/O data mapping to variables
- IEC61131-3 languages can be used in the project
- Base program organization unit: Function Block (similar to a mono-method class)

FB examples

- *TON : Timer on delay (standard FB, synchronous)*

- *UdpOpen (supplier specific, asynchronous)*

TON	

VAR_INPUT
IN : BOOL
//Input signal
PT : TIME
//Delay time
	VAR_OUTPUT
	Q : BOOL
	//Rising edge of the input signal is delayed by PT
	ET : TIME
	//Elapsed time

UdpOpen	

VAR_INPUT
enable : BOOL
//Input signal
pIfAddr: UDINT
//IP address of the Ethernet interface
port : UINT
	VAR_OUTPUT
	status : UINT
	ident : UDINT

FB examples

- *TON : Timer on delay (standard FB, synchronous)*

- *UdpOpen (supplier specific, asynchronous)*

TON	

VAR_INPUT
IN : BOOL
//Input signal
PT : TIME
//Delay time
	VAR_OUTPUT
	Q : BOOL
	//Rising edge of the input signal is delayed by PT
	ET : TIME
	//Elapsed time

UdpOpen	

VAR_INPUT
enable : BOOL
//Input signal
pIfAddr: UDINT
//IP address of the Ethernet interface
port : UINT
	VAR_OUTPUT
	status : UINT
	ident : UDINT

Continuously read variables

Parameters (async. access)

Execution regulation of the function

PLCopen compliant lib.

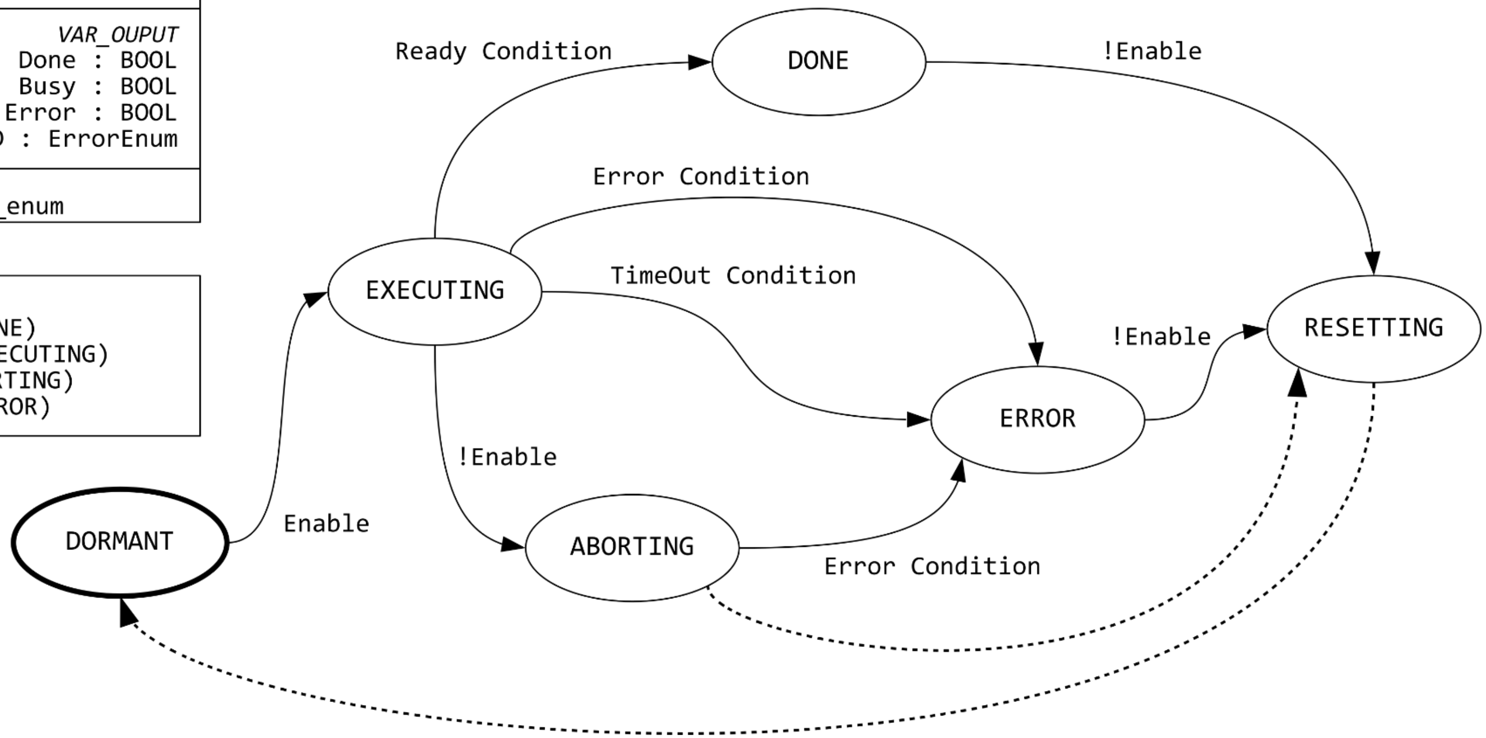
- *Definition of a limited number of execution behaviors required for control-command func.*
- *Standardized inputs / outputs*

- *Level-controlled / Edge triggered*
- *Time out limit*
- *Limit of execution time within a timer period*
- *Error handling and information*

Level-controlled FB

```
LConTo
VAR_INPUT
Enable : BOOL
TimeOut : REAL
VAR_OUPUT
Done : BOOL
Busy : BOOL
Error : BOOL
ErrorID : ErrorEnum
VAR
State : State_enum
```

```
Ouputs
Done  = (State==DONE)
Busy  = (State==EXECUTING)
      ||(State==ABORTING)
Error = (State==ERROR)
```



RTRMAC implementation

One Function Block = 2 classes

- *Run-time behavior*

(abstract classes implement the reference state machines)

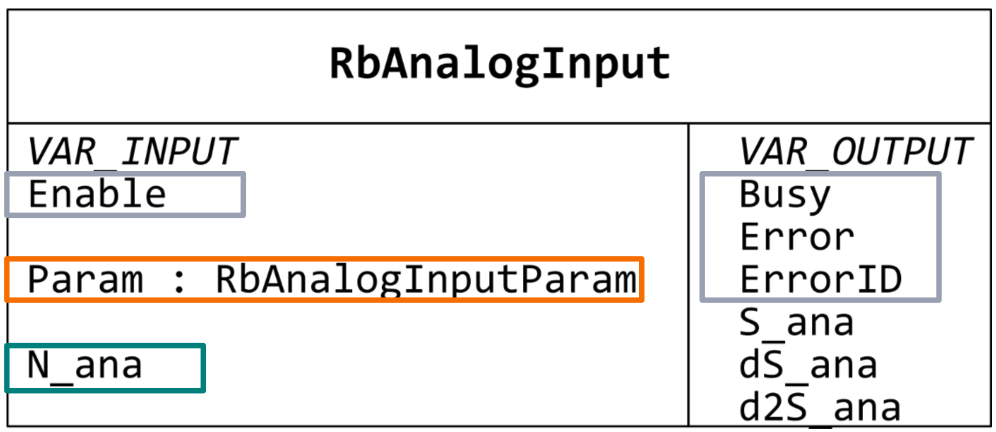
- *Parameter management*

(validation and update processes)

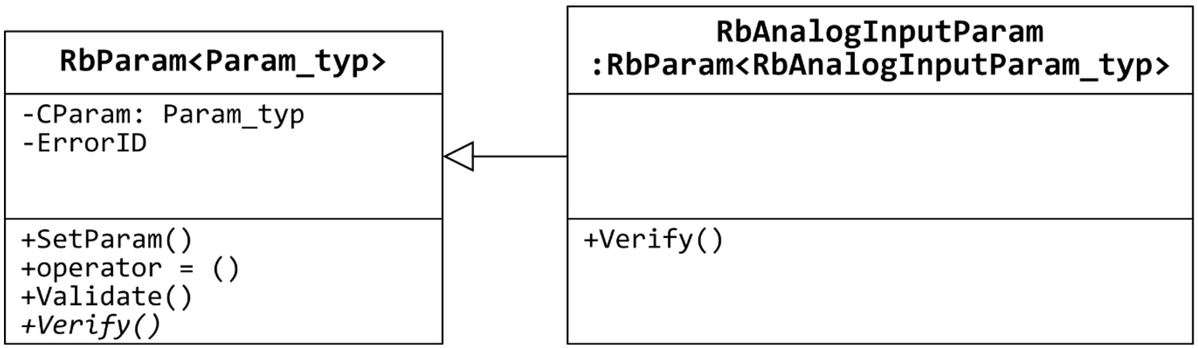
A simple example

Analog input conversion

integer from an input module -> float value with physical significance)



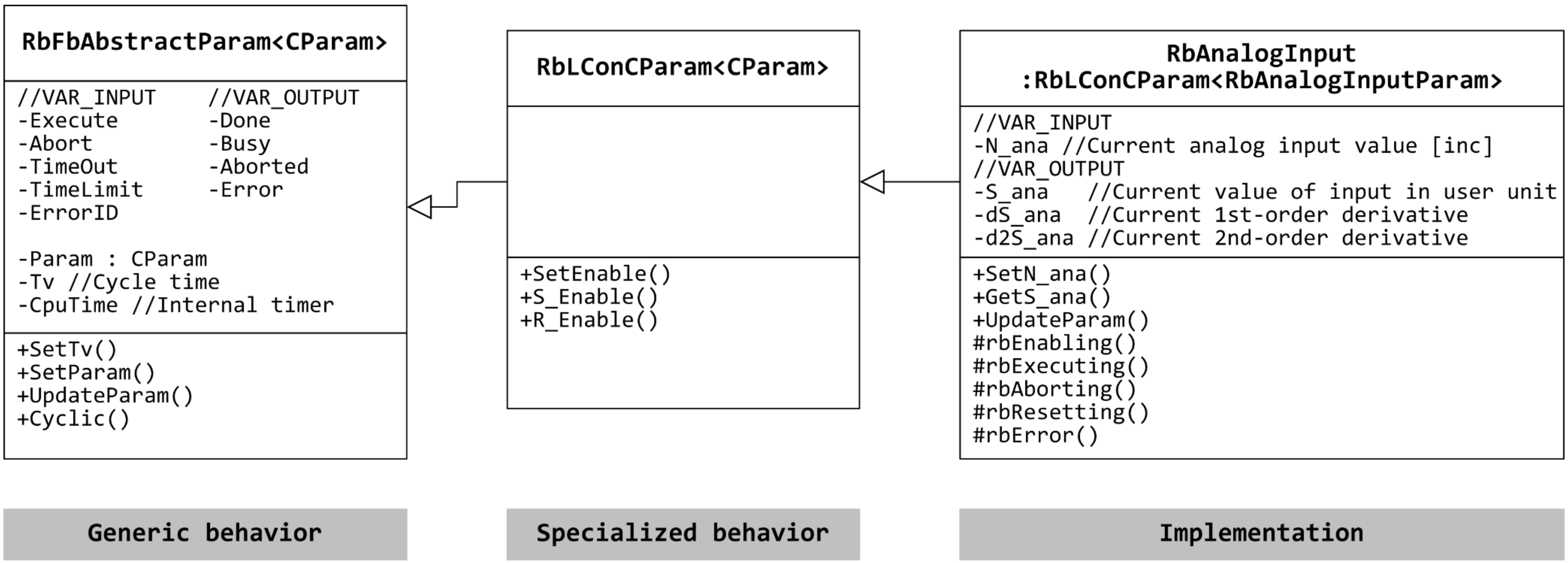
Parameter consistency




```
typedef struct
RbAnalogInputParam_typ
{
      rbLREAL Gain;//[Units/Inc]
      rbLREAL Offset;[Units]
      bool Differentiate;//if true,
computes signal derivatives
      rbLREAL TfVit;
      rbLREAL XiVit;
      rbLREAL TfAcc;
      rbLREAL XiAcc;
} RbAnalogInputParam_typ;

//Specific errors
const UINT rbERR_AI_GAIN = xxx ;
const UINT rbERR_AI_TF   = xxx ;
const UINT rbERR_AI_XI   = xxx ;
```

Execution behavior



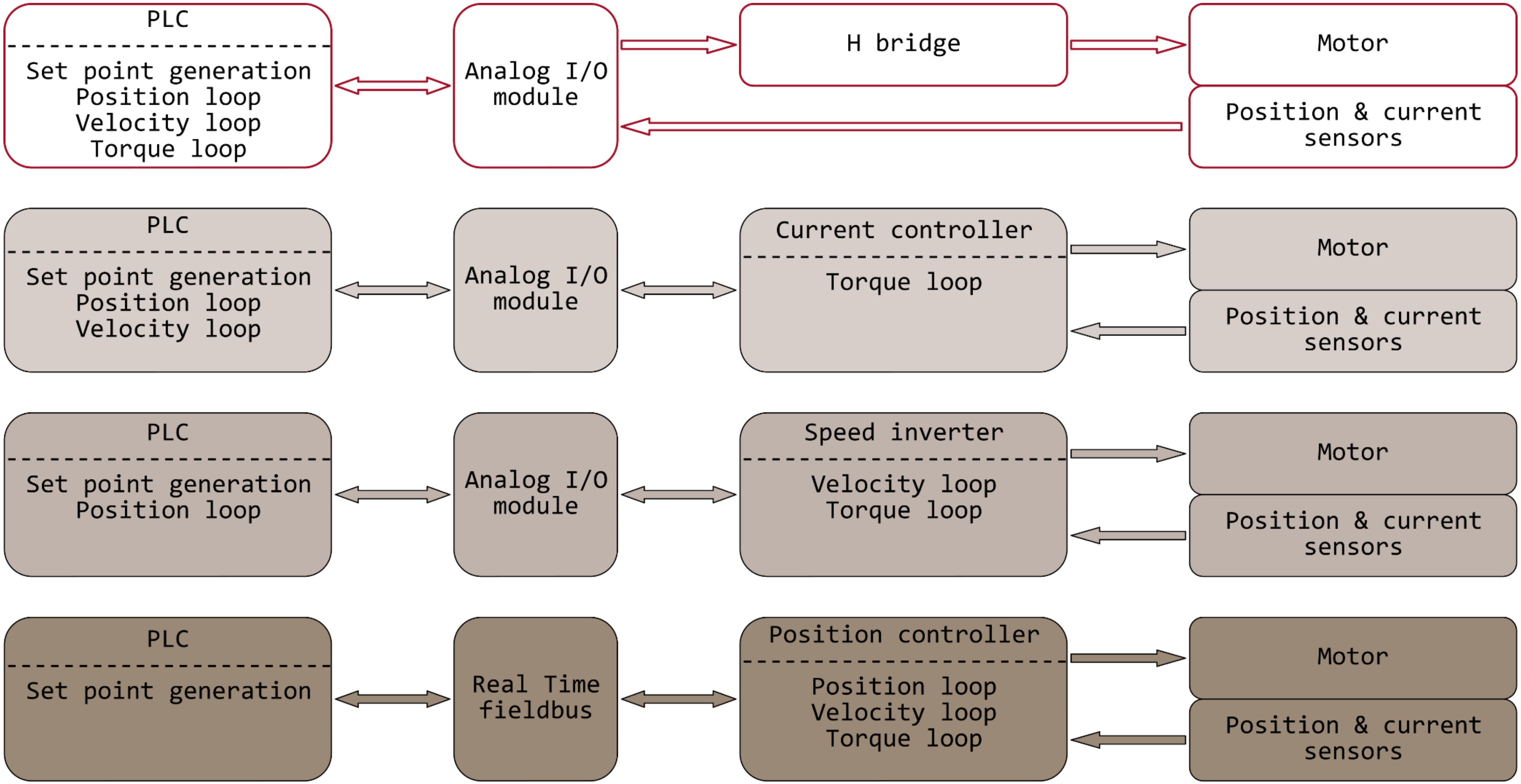
Independence from manufacturers

?

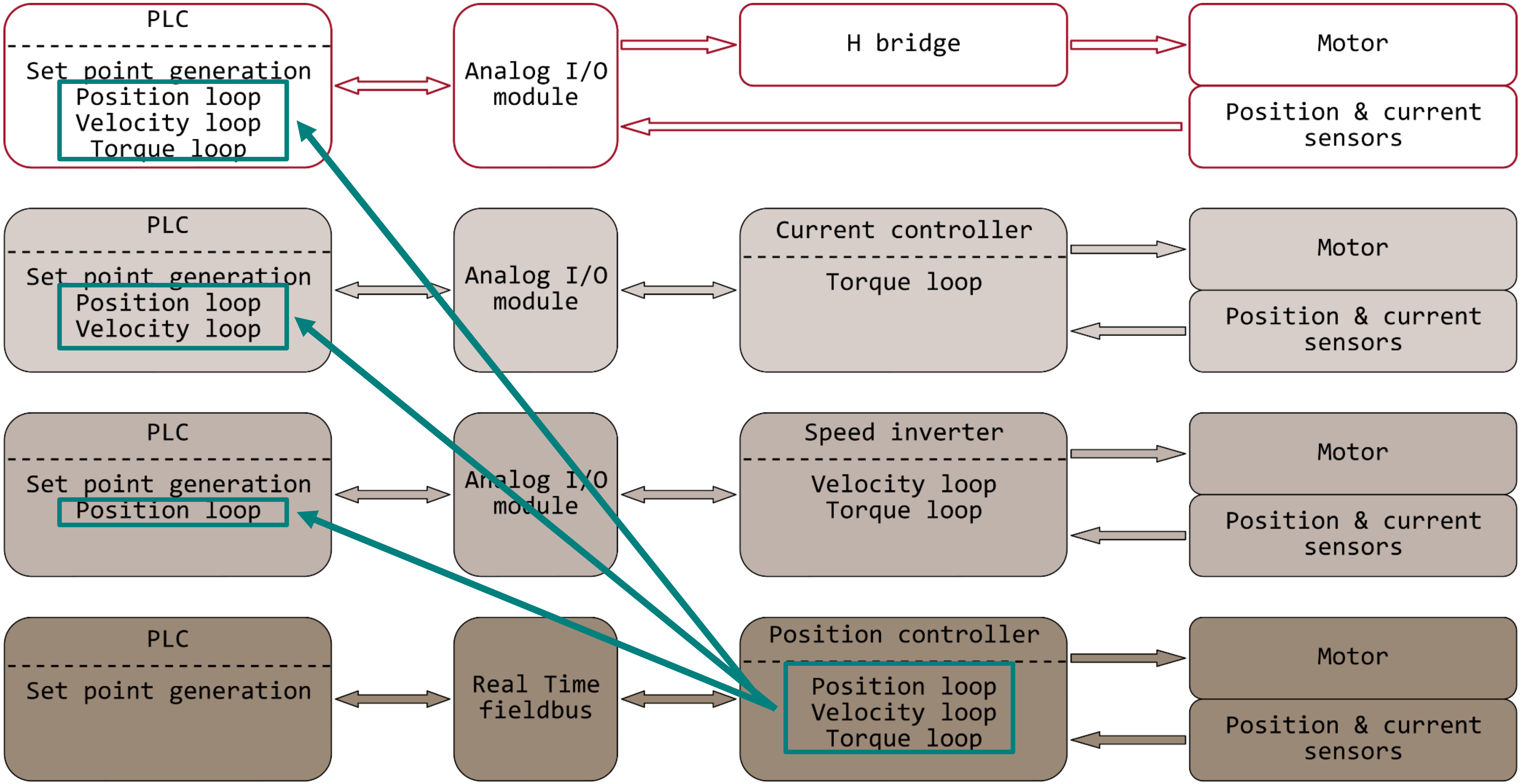
CPUs

- *Targets:*
  - *PLC / IPC B&R Automation*
  - *GPOS: Windows (Linux): simulation running on a periodic thread*
  - *Microcontroller*
- *Requirements:*
  - *C++11*
  - *Eigen*
  - *PLCopen Motion Control (for position controllers)*

Drive topologies



Drive topologies

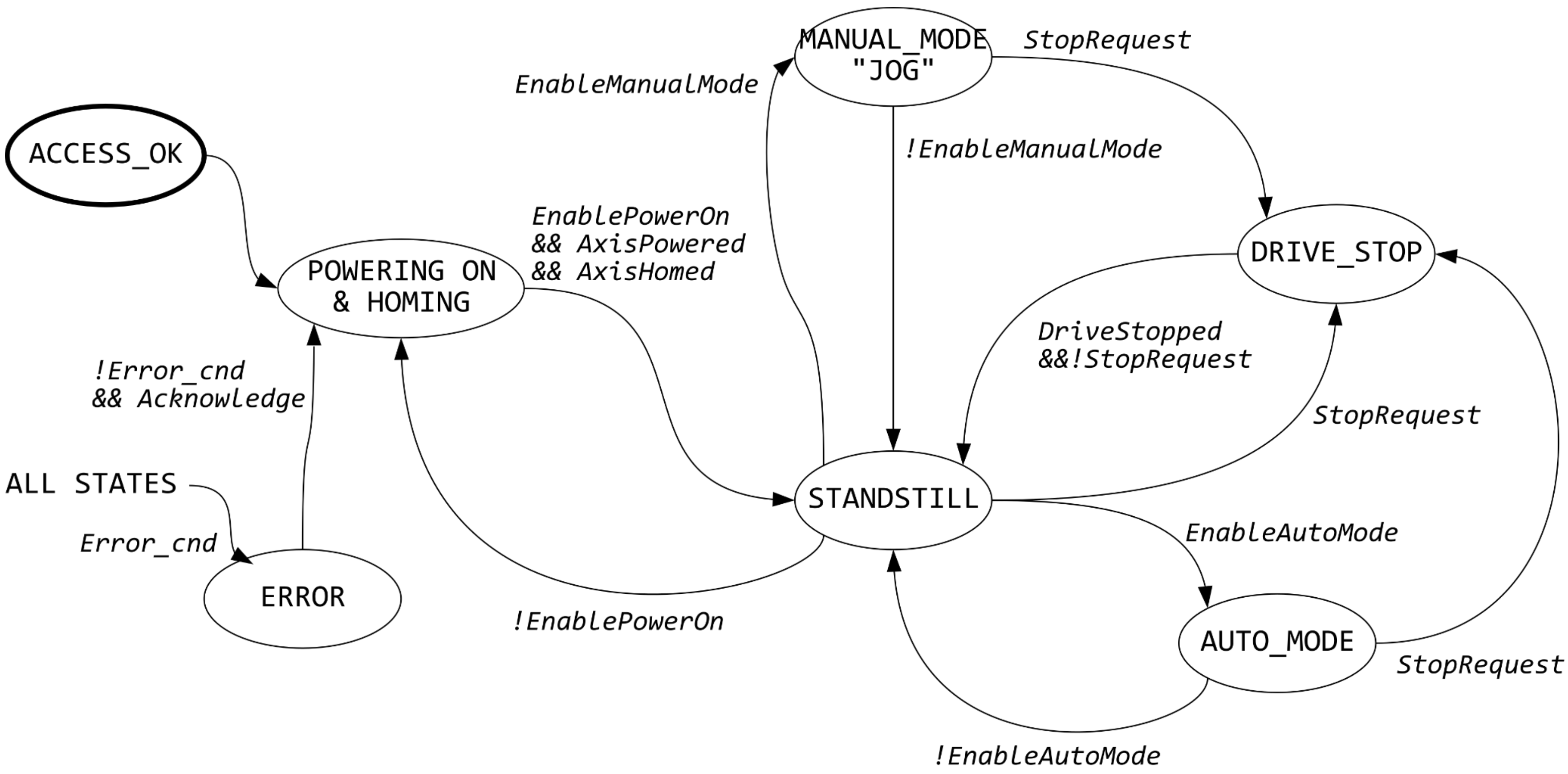


Single axis / multiaxis levels

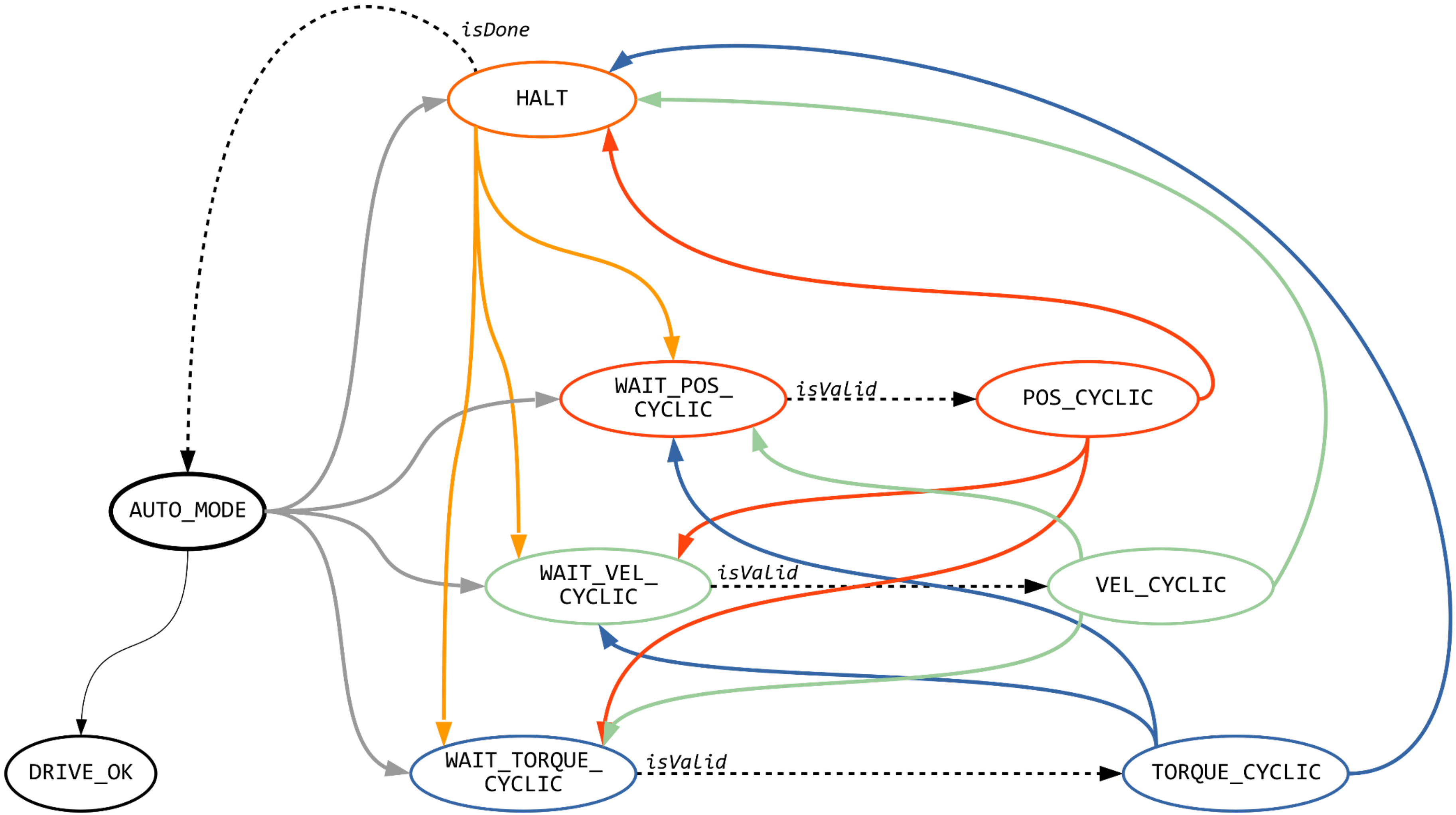
Single axis control

- *Servo control loops depending on the drive topology, completing the robust cascade position-velocity-torque control scheme*
- *PLCopen Motion Control compliance (function blocks, state machine)*
- *Generic robotic axis manager with specialized state machine*
- *Interface with PLC world with data flow structures*

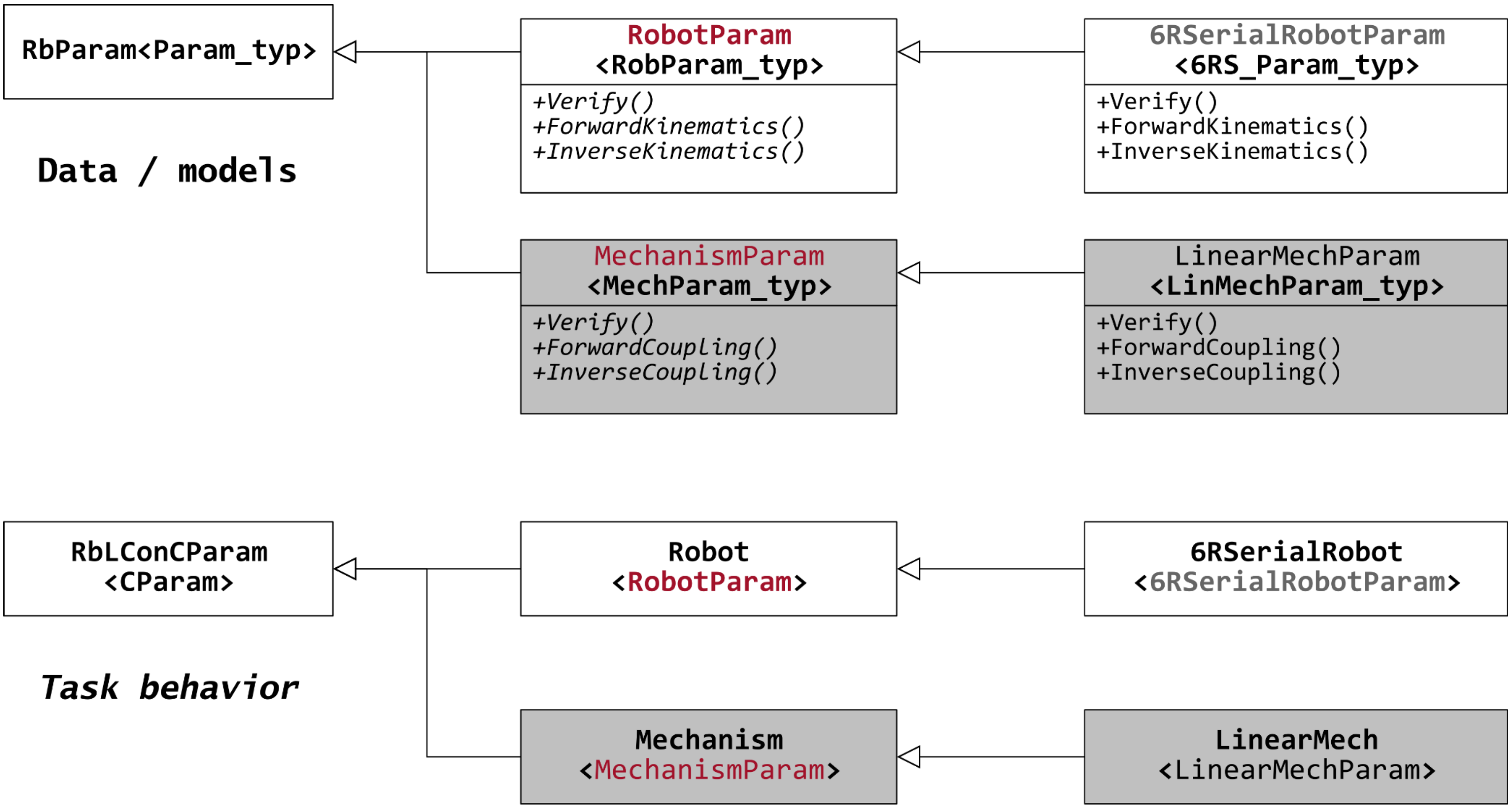
Axis manager, state machine



Axis manager, cyclic control



Robot level


RbParam<Param_typ>
RobotParam
<RobParam_typ>
+Verify()
+ForwardKinematics()
+InverseKinematics()
6RSerialRobotParam
<6RS_Param_typ>
+Verify()
+ForwardKinematics()
+InverseKinematics()
Data / models
MechanismParam
<MechParam_typ>
+Verify()
+ForwardCoupling()
+InverseCoupling()
LinearMechParam
<LinMechParam_typ>
+Verify()
+ForwardCoupling()
+InverseCoupling()
RbLConCParam
<CParam>
Robot
<RobotParam>
6RSerialRobot
<6RSerialRobotParam>
Task behavior
Mechanism
<MechanismParam>
LinearMech
<LinearMechParam>

RTRMAC implementation

- *Low level regulation functions*
- *Interfaces to main drive topologies*
- *PLCopen Motion Control Single Axis level*
- *Robotic axis manager*
- *Robot / coordinated axis manager*
- *I/O utilities*

Humanoid at Pprime

ORHRO's first leg

J. Gastebois PhD thesis (2017)

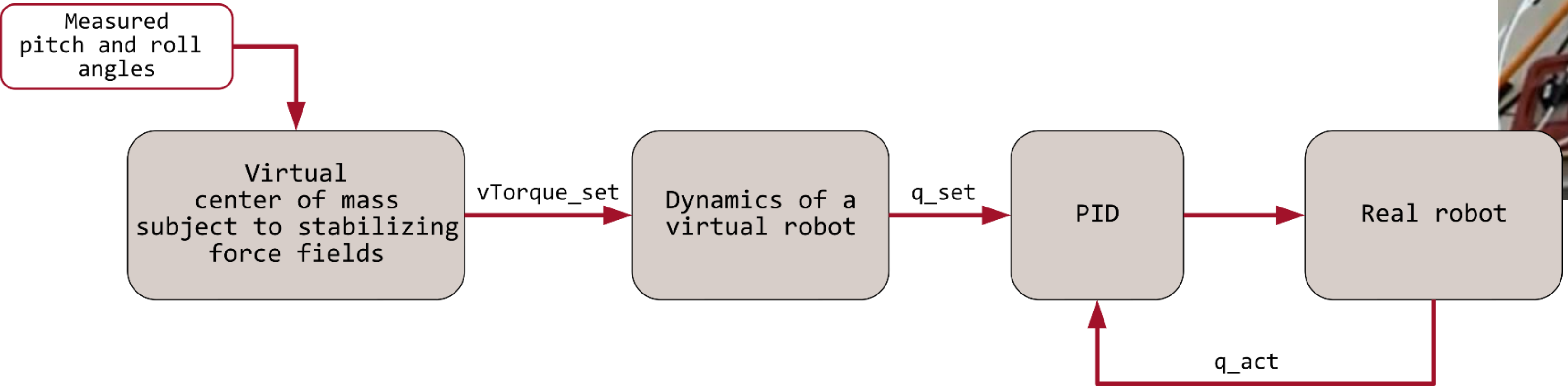
RTRMAC v2 (C)

Runs on a standard PLC

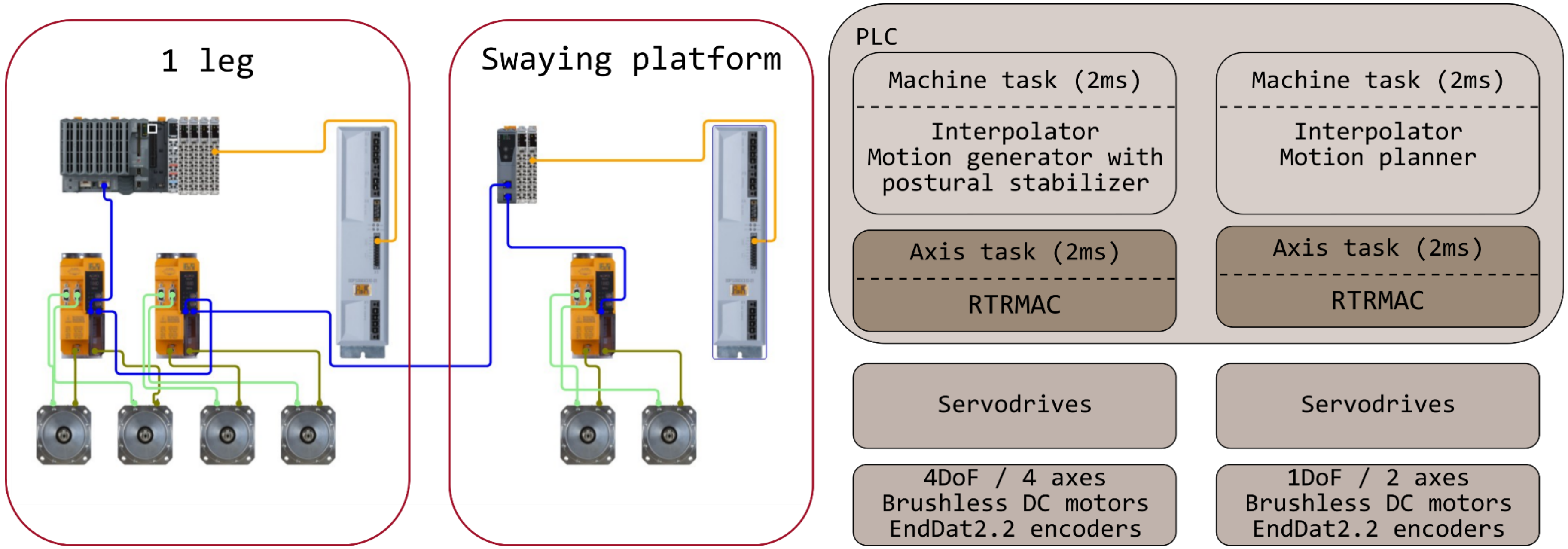
Application specific control software

Matrix library

Watch: youtube RoBioSS - Control of a robot leg using a postural stabilizer

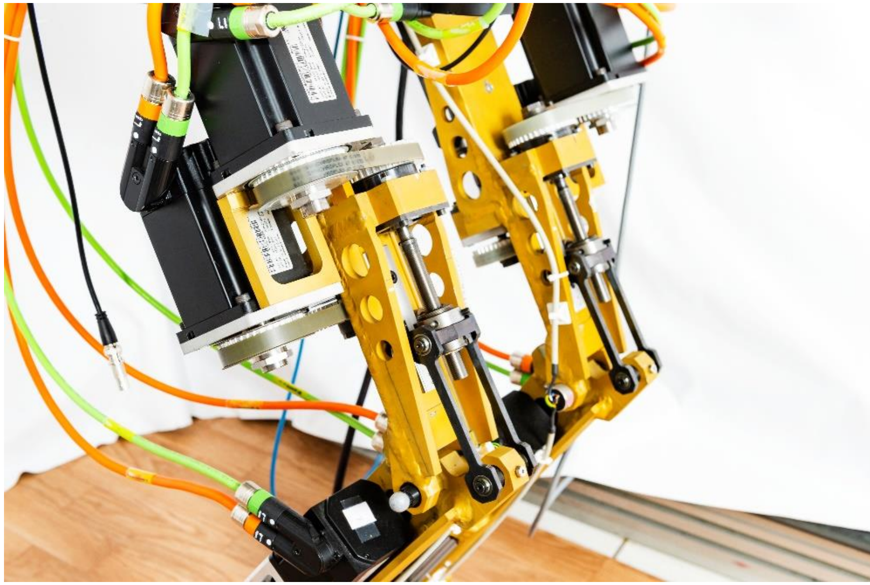


ORHRO's first leg

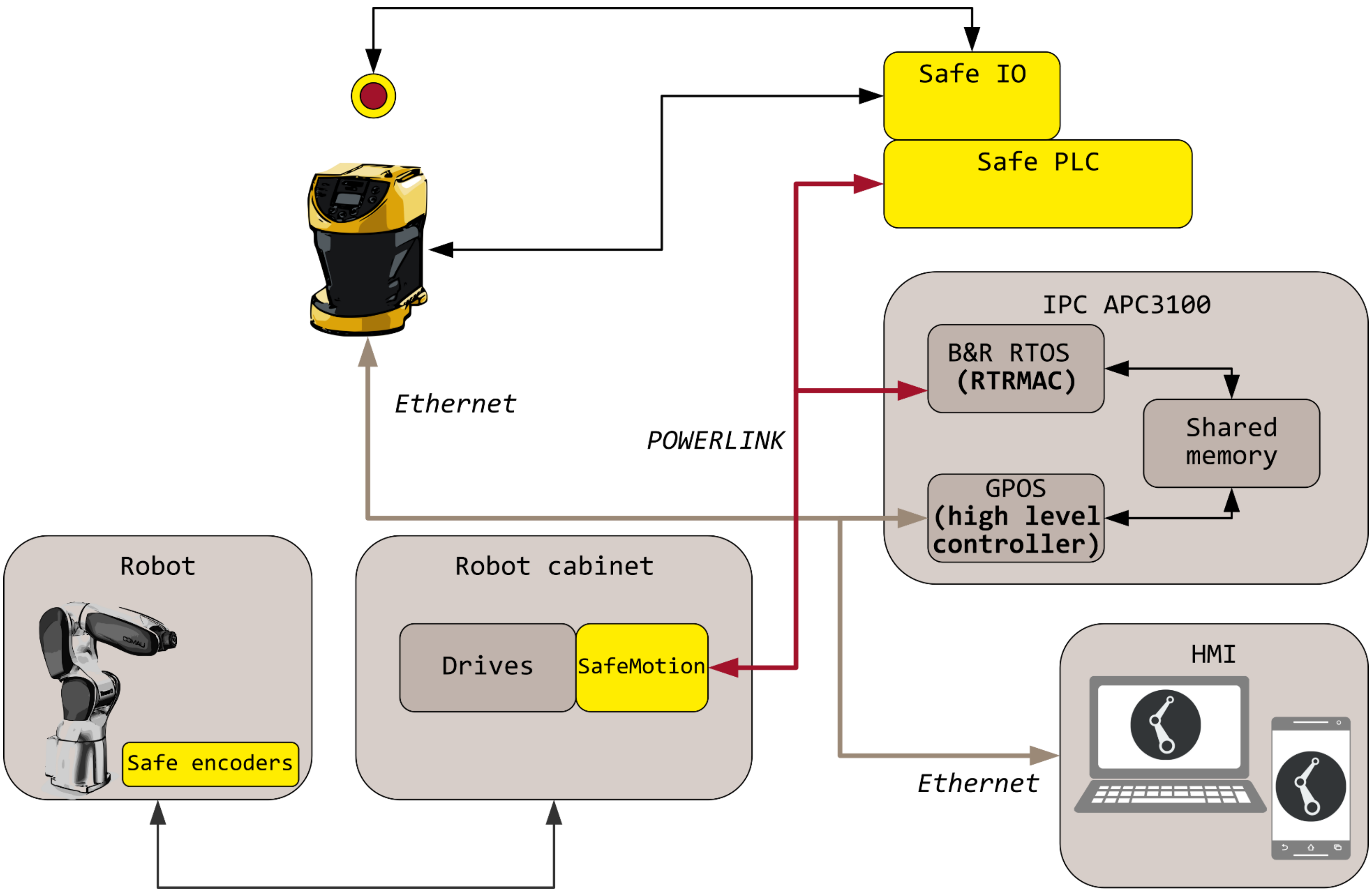


Short term evolution

- *Scale up to 2 legs + pelvis (open loop squatting, stepping, walking motions -> done)*
- *Mechanical and control design of ORHRO arms*
- *Implementation and validation of the « robot » level of RTRMAC:*
  complex architecture,
  parallel & non linear couplings,
  stabilizer control layer



Interfacing with ext. ctrl



Conclusion

Conclusion and perspectives

- *Motor interface layer and robotics axis level are fully functional*
- *Prototype commissioning is extremely fast, projects are easily scalable*
- *Robot level is currently under development*

- *Next integration challenges:*
  - *Control of ORHRO's next version (coordination of 23 axes)*
  - *Embedding into power drive boards*
  - *Machine simulation and control with a digital twin (ITECA SmartUpp)*

Thank you for your attention!

Design of an open-robotics low-level framework for the control of complex mechatronics devices

RoBioSS team, Pprime Institute, Poitiers

This work is sponsored by the ANR Labcom program (ANR-18-LCV2-0003)

MACH4 INSTITUT P' ITECA