PHILIPS

sense and simplicity

# Robot Arm
# Example Software

Rob Beekmans / Gérard Haagh
Philips Applied Technologies

9 December 2009

PHILIPS

# Contents

- Introduction
- System overview
- Controller component
- Servo
- I/O Layer
- Application
- CORBA interface
- User Interface component
- Build system
- Questions

# Introduction

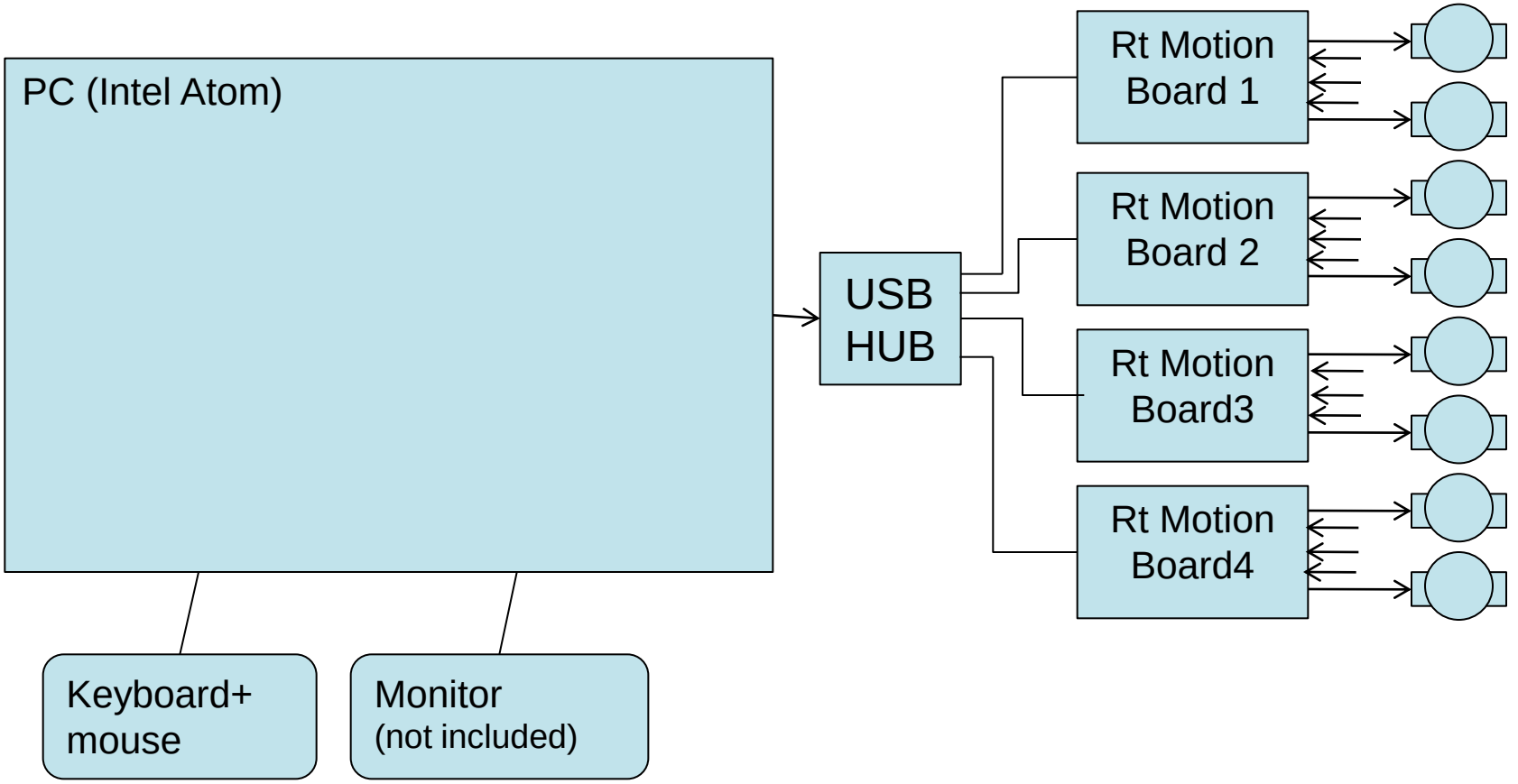
- Goals for the software in the robot arm project
  - Should be able to control the robot arm also during development
  - Uses the RtmotionUSB api
  - Example code that is simple to understand and extend
  - Using open source software

# Software result

- Linux system with Xenomai RealTime extension
- Realtime controller task @ 500Hz
  - Eight PID contoller loops
  - 2nd order setpointgenerator for each controller loop
  - Signal tracing
  - Interface with RtMotion-USB I/O hardware
- Graphical Userinterface
  - Control of each loop
  - Simple moves and jogs
  - Automatic execution of moves
  - Parameter reloading
- Written in C++

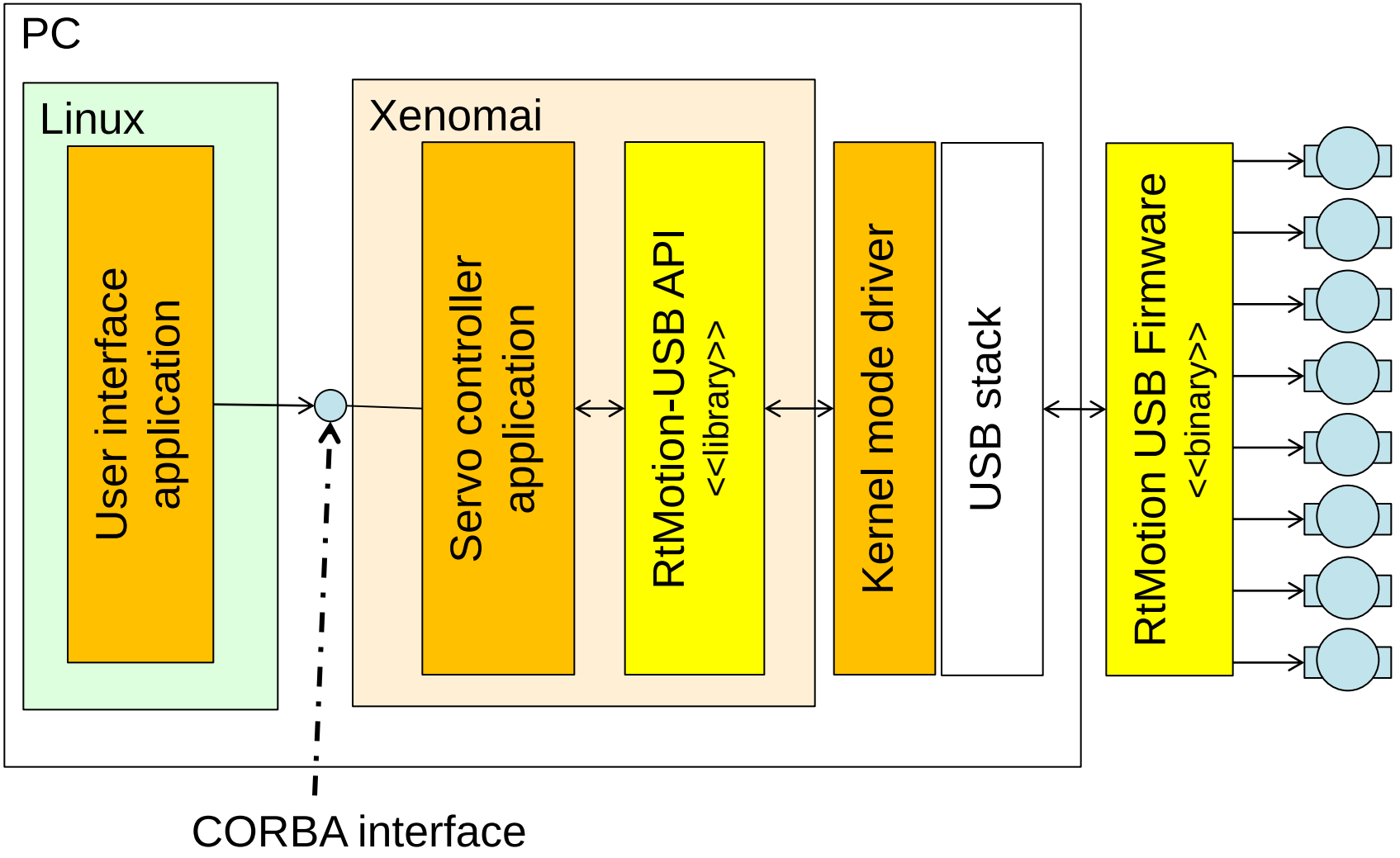
PHILIPS

# Physical overview

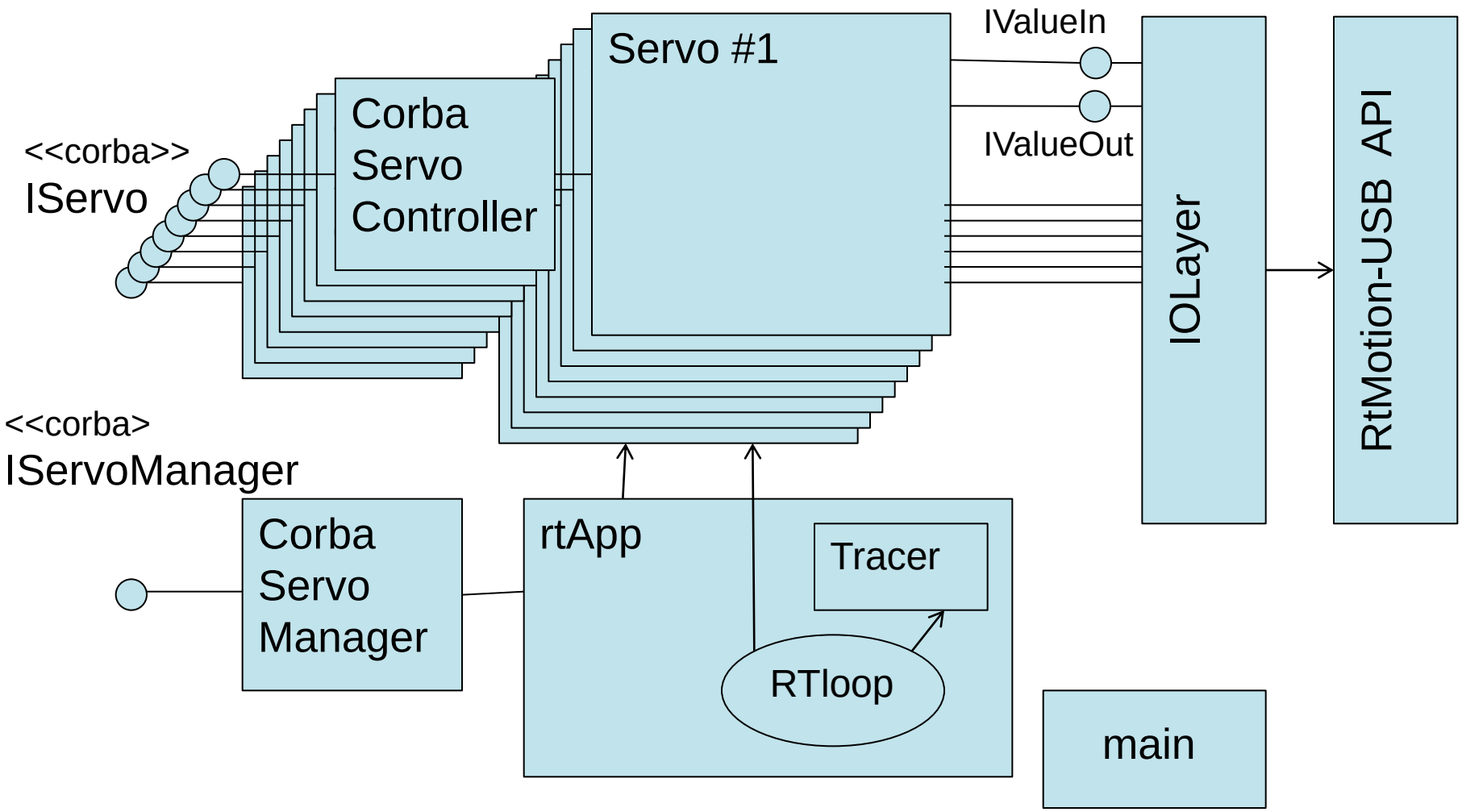


PHILIPS

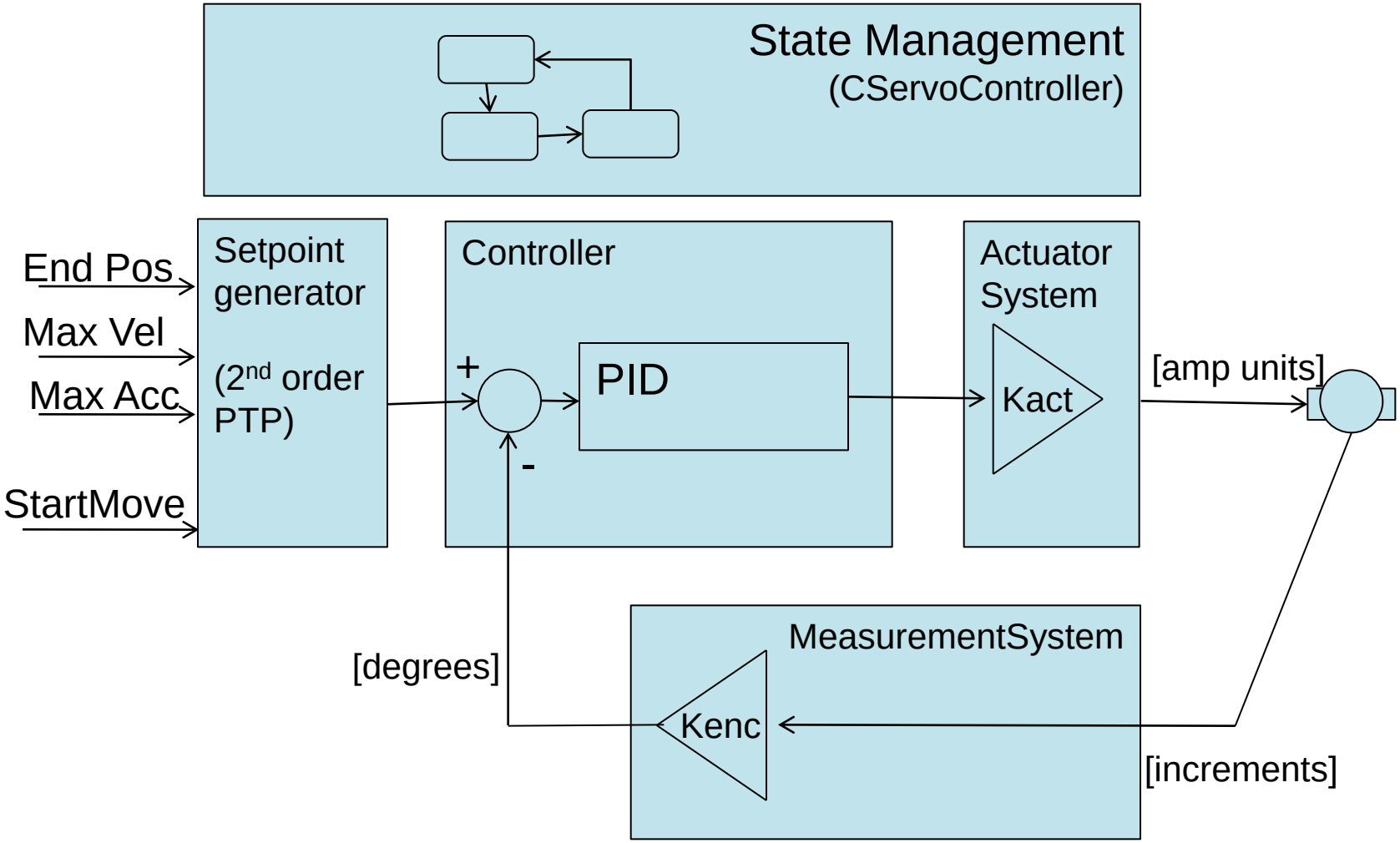
# SW-Components/Layers



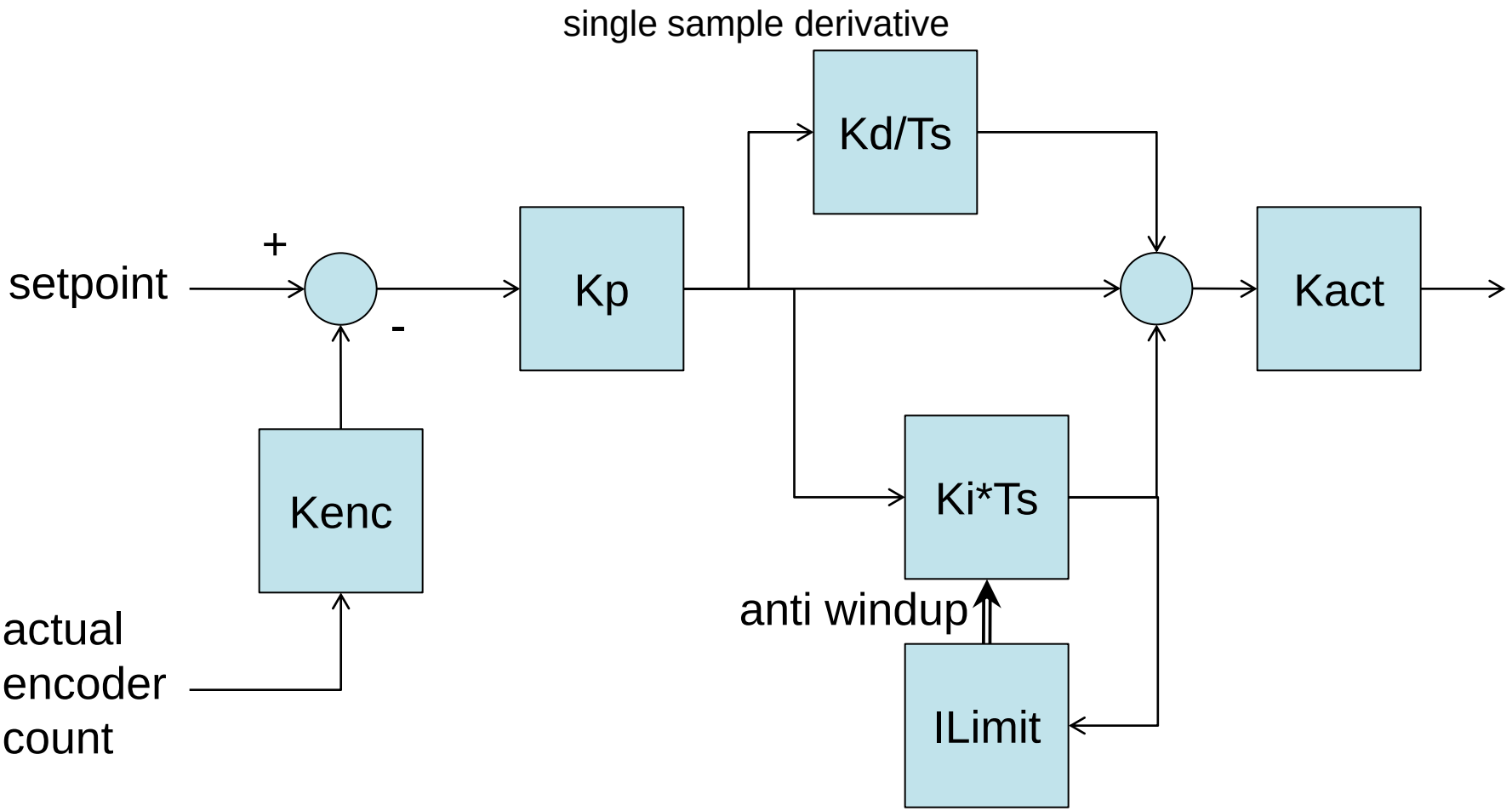
# Controller App. Components



# Servo


State Management
(CServoController)
End Pos
Max Vel
Max Acc
StartMove
Setpoint generator
(2nd order PTP)
Controller
+
-
PID
Actuator System
Kact
[amp units]
MeasurementSystem
Kenc
[degrees]
[increments]

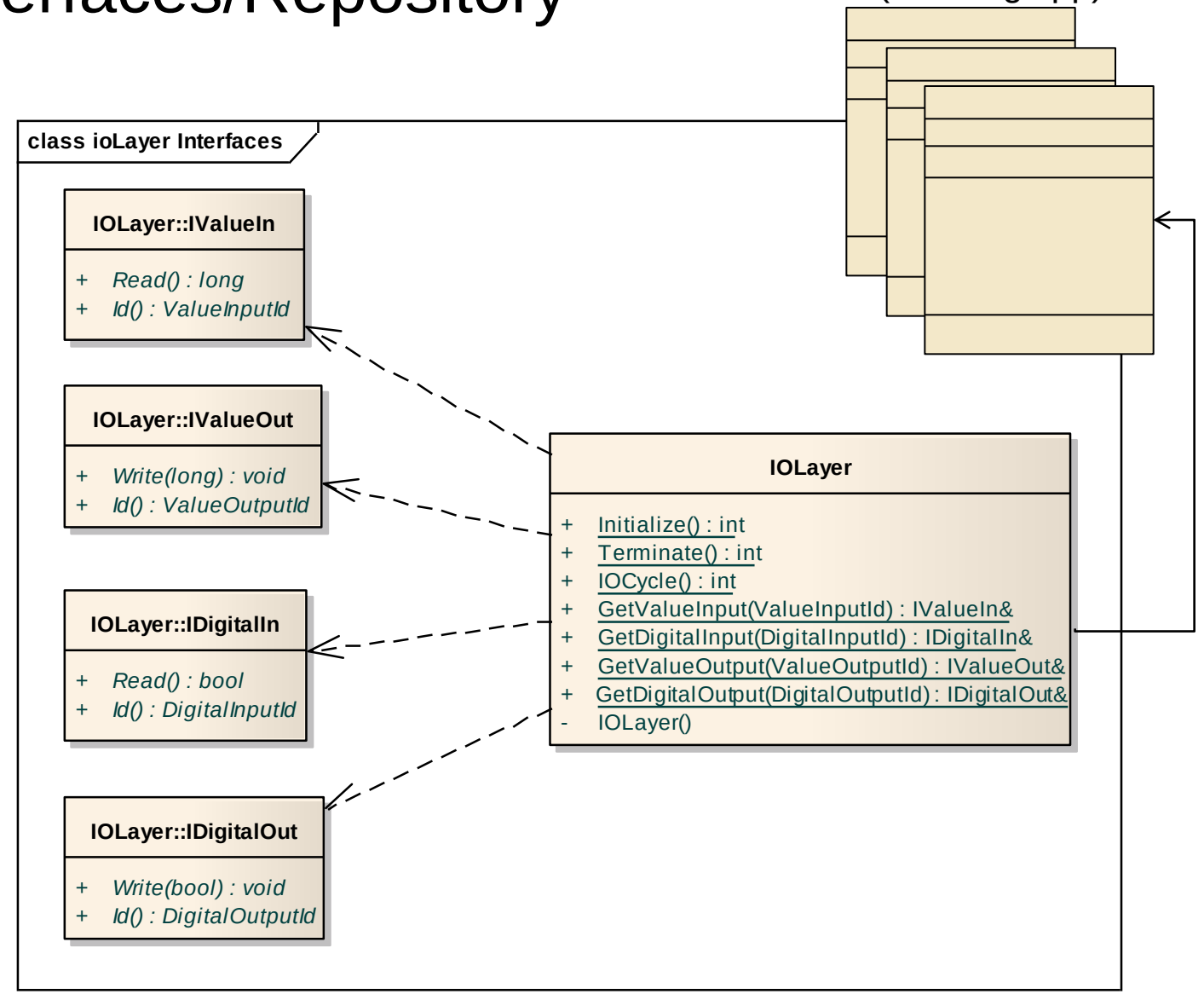
# Servo controller



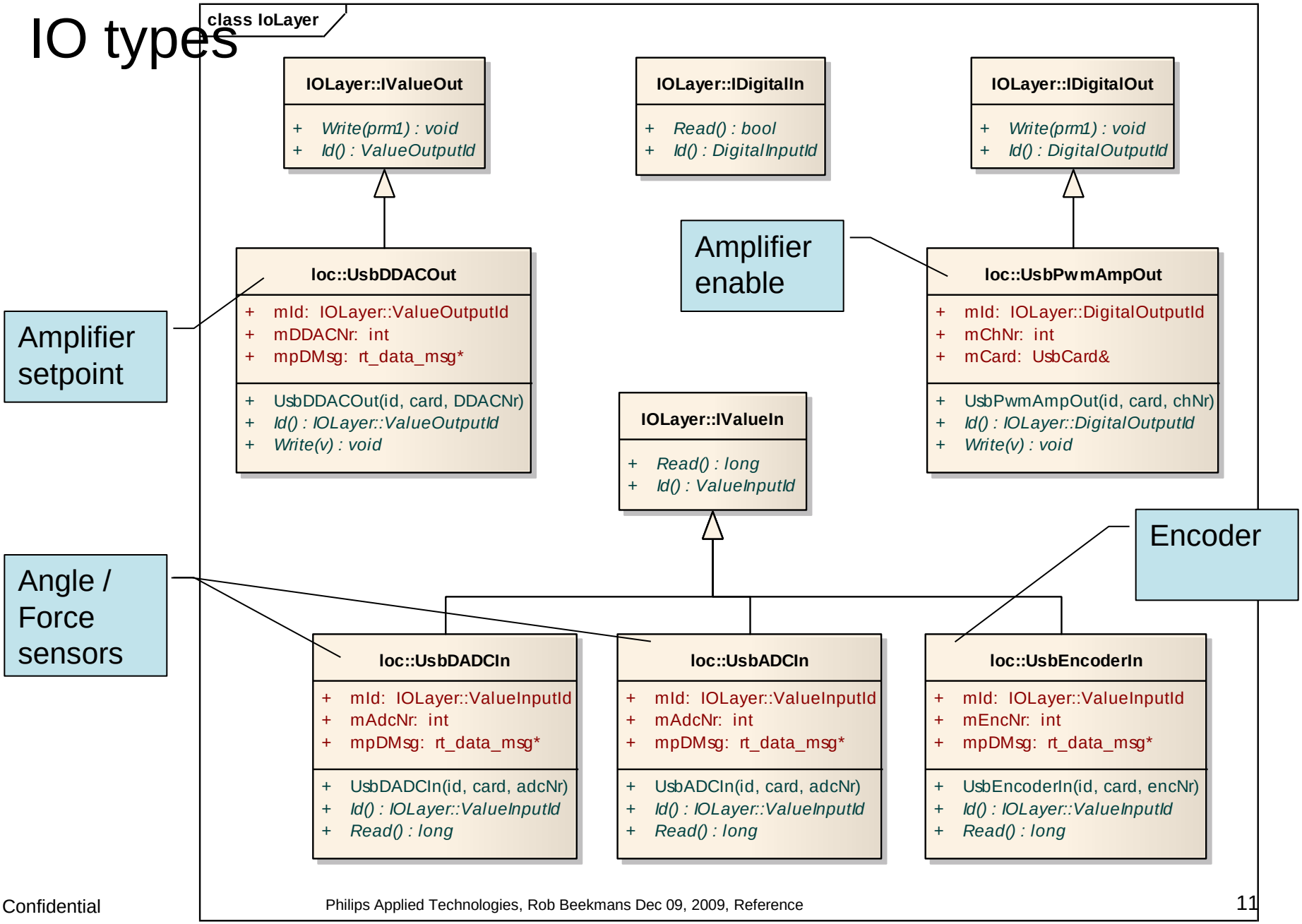
PHILIPS

# IO - Interfaces/Repository

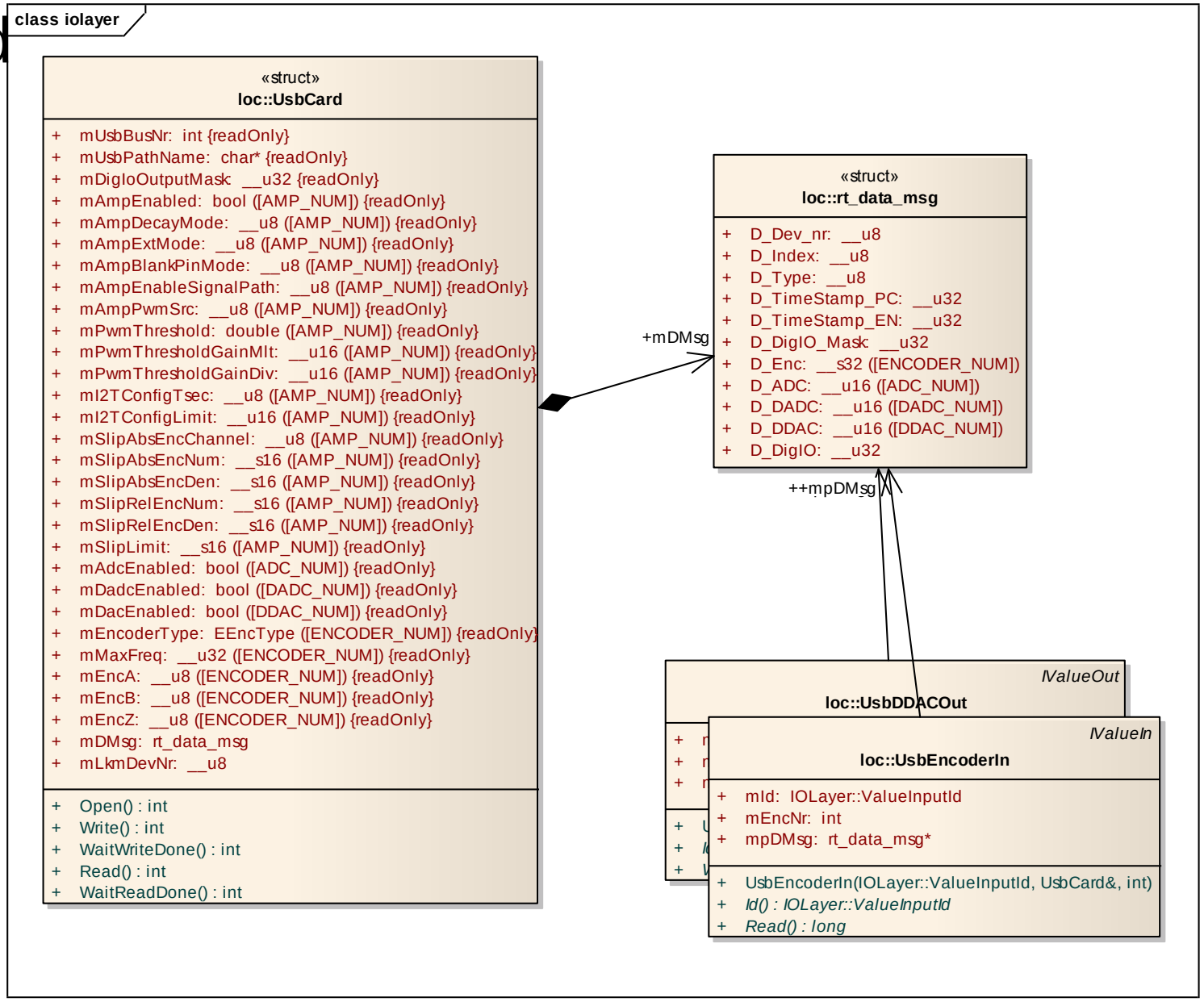
Tables with DI, AO, AI
(ioConfig.cpp)



# IO types


class IoLayer
IOLayer::IValueOut
+ Write(prm1) : void
+ Id() : ValueOutputId
IOLayer::IDigitalIn
+ Read() : bool
+ Id() : DigitalInputId
IOLayer::IDigitalOut
+ Write(prm1) : void
+ Id() : DigitalOutputId
Amplifier enable
Amplifier setpoint
Ioc::UsbDDACOut
+ mId: IOLayer::ValueOutputId
+ mDDACNr: int
+ mpDMsg: rt_data_msg*
+ UsbDDACOut(id, card, DDACNr)
+ Id() : IOLayer::ValueOutputId
+ Write(v) : void
Ioc::UsbPwmAmpOut
+ mId: IOLayer::DigitalOutputId
+ mChNr: int
+ mCard: UsbCard&
+ UsbPwmAmpOut(id, card, chNr)
+ Id() : IOLayer::DigitalOutputId
+ Write(v) : void
IOLayer::IValueIn
+ Read() : long
+ Id() : ValueInputId
Encoder
Angle / Force sensors
Ioc::UsbDADCIn
+ mId: IOLayer::ValueInputId
+ mAdcNr: int
+ mpDMsg: rt_data_msg*
+ UsbDADCIn(id, card, adcNr)
+ Id() : IOLayer::ValueInputId
+ Read() : long
Ioc::UsbADCIn
+ mId: IOLayer::ValueInputId
+ mAdcNr: int
+ mpDMsg: rt_data_msg*
+ UsbADCIn(id, card, adcNr)
+ Id() : IOLayer::ValueInputId
+ Read() : long
Ioc::UsbEncoderIn
+ mId: IOLayer::ValueInputId
+ mEncNr: int
+ mpDMsg: rt_data_msg*
+ UsbEncoderIn(id, card, encNr)
+ Id() : IOLayer::ValueInputId
+ Read() : long

# IO Card


class iolayer
«struct»
loc::UsbCard
+ mUsbBusNr: int {readOnly}
+ mUsbPathName: char* {readOnly}
+ mDigIoOutputMask: __u32 {readOnly}
+ mAmpEnabled: bool ([AMP_NUM]) {readOnly}
+ mAmpDecayMode: __u8 ([AMP_NUM]) {readOnly}
+ mAmpExtMode: __u8 ([AMP_NUM]) {readOnly}
+ mAmpBlankPinMode: __u8 ([AMP_NUM]) {readOnly}
+ mAmpEnableSignalPath: __u8 ([AMP_NUM]) {readOnly}
+ mAmpPwmSrc: __u8 ([AMP_NUM]) {readOnly}
+ mPwmThreshold: double ([AMP_NUM]) {readOnly}
+ mPwmThresholdGainMlt: __u16 ([AMP_NUM]) {readOnly}
+ mPwmThresholdGainDiv: __u16 ([AMP_NUM]) {readOnly}
+ mI2TConfigTsec: __u8 ([AMP_NUM]) {readOnly}
+ mI2TConfigLimit: __u16 ([AMP_NUM]) {readOnly}
+ mSlipAbsEncChannel: __u8 ([AMP_NUM]) {readOnly}
+ mSlipAbsEncNum: __s16 ([AMP_NUM]) {readOnly}
+ mSlipAbsEncDen: __s16 ([AMP_NUM]) {readOnly}
+ mSlipRelEncNum: __s16 ([AMP_NUM]) {readOnly}
+ mSlipRelEncDen: __s16 ([AMP_NUM]) {readOnly}
+ mSlipLimit: __s16 ([AMP_NUM]) {readOnly}
+ mAdcEnabled: bool ([ADC_NUM]) {readOnly}
+ mDadcEnabled: bool ([DADC_NUM]) {readOnly}
+ mDacEnabled: bool ([DDAC_NUM]) {readOnly}
+ mEncoderType: EEncType ([ENCODER_NUM]) {readOnly}
+ mMaxFreq: __u32 ([ENCODER_NUM]) {readOnly}
+ mEncA: __u8 ([ENCODER_NUM]) {readOnly}
+ mEncB: __u8 ([ENCODER_NUM]) {readOnly}
+ mEncZ: __u8 ([ENCODER_NUM]) {readOnly}
+ mDMsg: rt_data_msg
+ mLkmDevNr: __u8
+ Open() : int
+ Write() : int
+ WaitWriteDone() : int
+ Read() : int
+ WaitReadDone() : int
+mDMsg
«struct»
loc::rt_data_msg
+ D_Dev_nr: __u8
+ D_Index: __u8
+ D_Type: __u8
+ D_TimeStamp_PC: __u32
+ D_TimeStamp_EN: __u32
+ D_DigIO_Mask: __u32
+ D_Enc: __s32 ([ENCODER_NUM])
+ D_ADC: __u16 ([ADC_NUM])
+ D_DADC: __u16 ([DADC_NUM])
+ D_DDAC: __u16 ([DDAC_NUM])
+ D_DigIO: __u32
++mpDMsg
IValueOut
loc::UsbDDACOut
IValueIn
loc::UsbEncoderIn
+ mId: IOLayer::ValueInputId
+ mEncNr: int
+ mpDMsg: rt_data_msg*
+ UsbEncoderIn(IOLayer::ValueInputId, UsbCard&, int)
+ Id() : IOLayer::ValueInputId
+ Read() : long

# Appplication (rtApp)

- Starting point for all software in the controller
- Initializes the IOLayer
- Initializes the CORBA interface
- Defines the servos in the system
  (sets the IO channels for each servo and some constant parameters like Kenc, Kact)
- Executes the periodic Real Time loop

# CORBA interface

- CORBA middleware stack:
  http://www.cs.wustl.edu/~schmidt/TAO.html

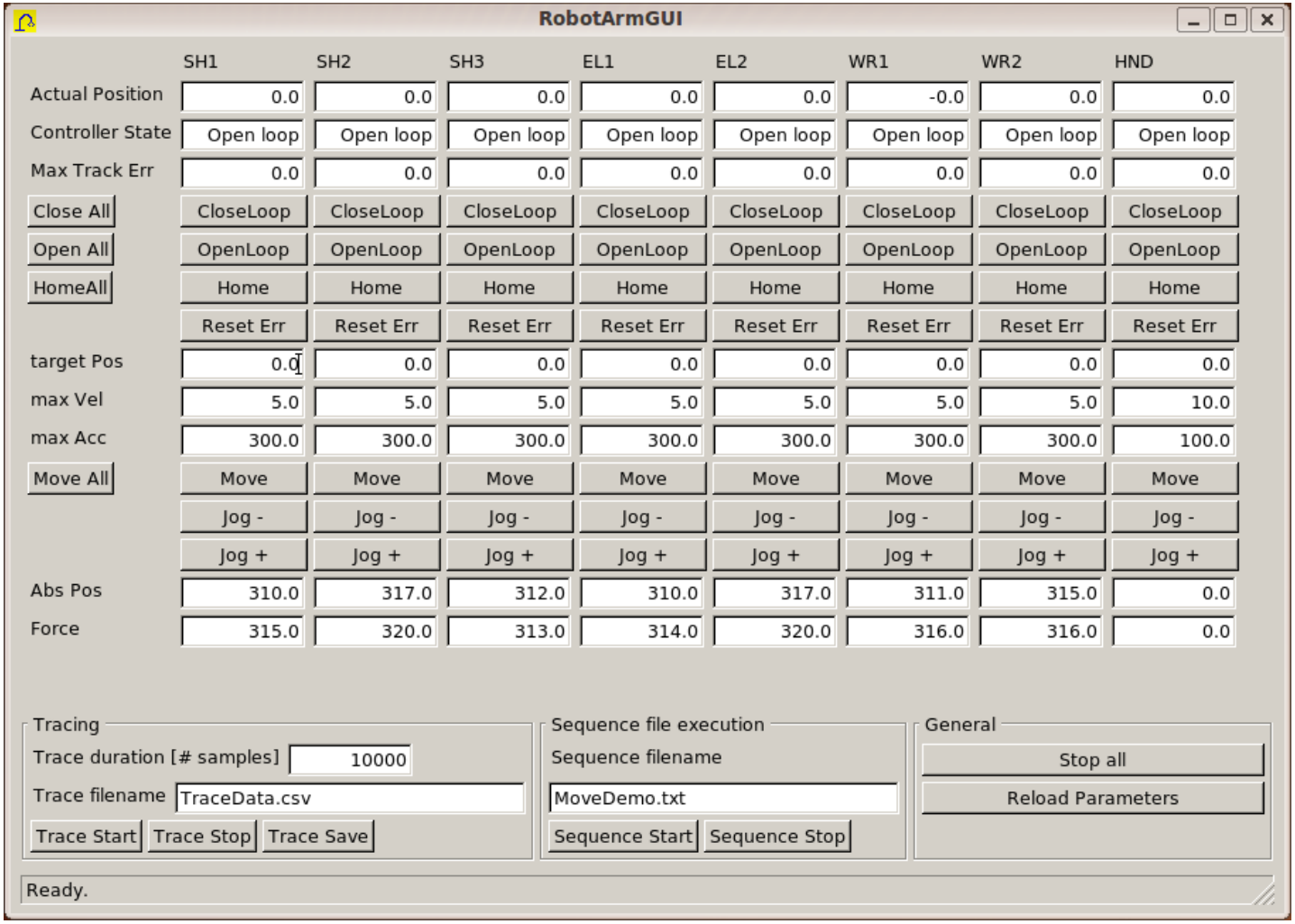
- Interfaces: IServoManager, IServoController
  see file comp\controller\corba\inc\rtServoIF.idl

# User Interface Application

- Features:
  - Control of all eight servo loops, including ptp and jog movements
  - Allow servo parameter reloading (only when open loop)
  - Allow tracing of controller signal values
  - Allow automatic execution of move commands

- Graphical user interface is based on the FOX toolkit (www.fox-toolkit.org). This is a platform independent GUI toolkit (also available for Windows)

# User Interface screenshot



# Build system

- CMake (www.cmake.org) is a open source cross platform build system
- CMake is used to generate makefiles and to build them.

- cdbuild  (alias to jump to the build directory)
- ./xeno_generate.sh
- ./linux_generate.sh
- ./xeno_build.sh
- ./linux_build.sh

# Running the software

- Boot the PC
- Open a shell prompt window
- Execute the commands:
  - ldlkm  (loads the kernel mode driver; run only once after booting)
  - ldxeno  (loads the Xenomai modules; run only once after booting)
  - cdservo (jumps to the directory with controller app)
  - sudo ./RtMotionApp  (run the controller app)
- Open another shell prompt
- Execute the commands:
  - cdgui (jumps to the directory with GUI app)
  - ./RtMotionGui  (run the GUI)
- To stop: exit the GUI, kill the controller app with ^C
- Restart: repeat all except ldlkm and ldxeno

# Questions

- Q: Can I swap the USB connectors on the hub
  A: NO, unless you change ioConfig.cpp
- Q: Can I connect the USB hub to another USB port
  A: NO, unless you change ioConfig.cpp
- Q: Does the PC detect the RtMotion-USB boards
  A: “cat /proc/RtMotionUSB” should list 4 boards
- Q: Can I connect other USB devices
  A: At your own risk, it could disturb real time behavior

- Wvttk


PHILIPS