Simplified instrument/application development and system integration using Libera BASE software framework

Matej Kenda, Tomaž Beltram, Tomaž Juretič, Borut Repič, Damijan Škvarč, Črt Valentinčič, Instrumentation Technologies, Solkan, Slovenia

Abstract

Development of many appliances used in scientific environment forces us to face similar challenges, often executed repeatedly. One has to design or integrate hardware components. Support for network and other communications standards needs to be established. Data and signals are processed and dispatched. Interfaces are required to monitor and control the behaviour of the appliances. At Instrumentation Technologies we identified and addressed these issues by creating Libera BASE, which is a framework composed of several reusable building blocks. Libera BASE simplifies some of the tedious tasks and leaves more time to concentrate on real issues of the application. Further more, the end product quality benefits from larger common base of this framework.

Increasing Role of Software in Reconfigurable Instruments

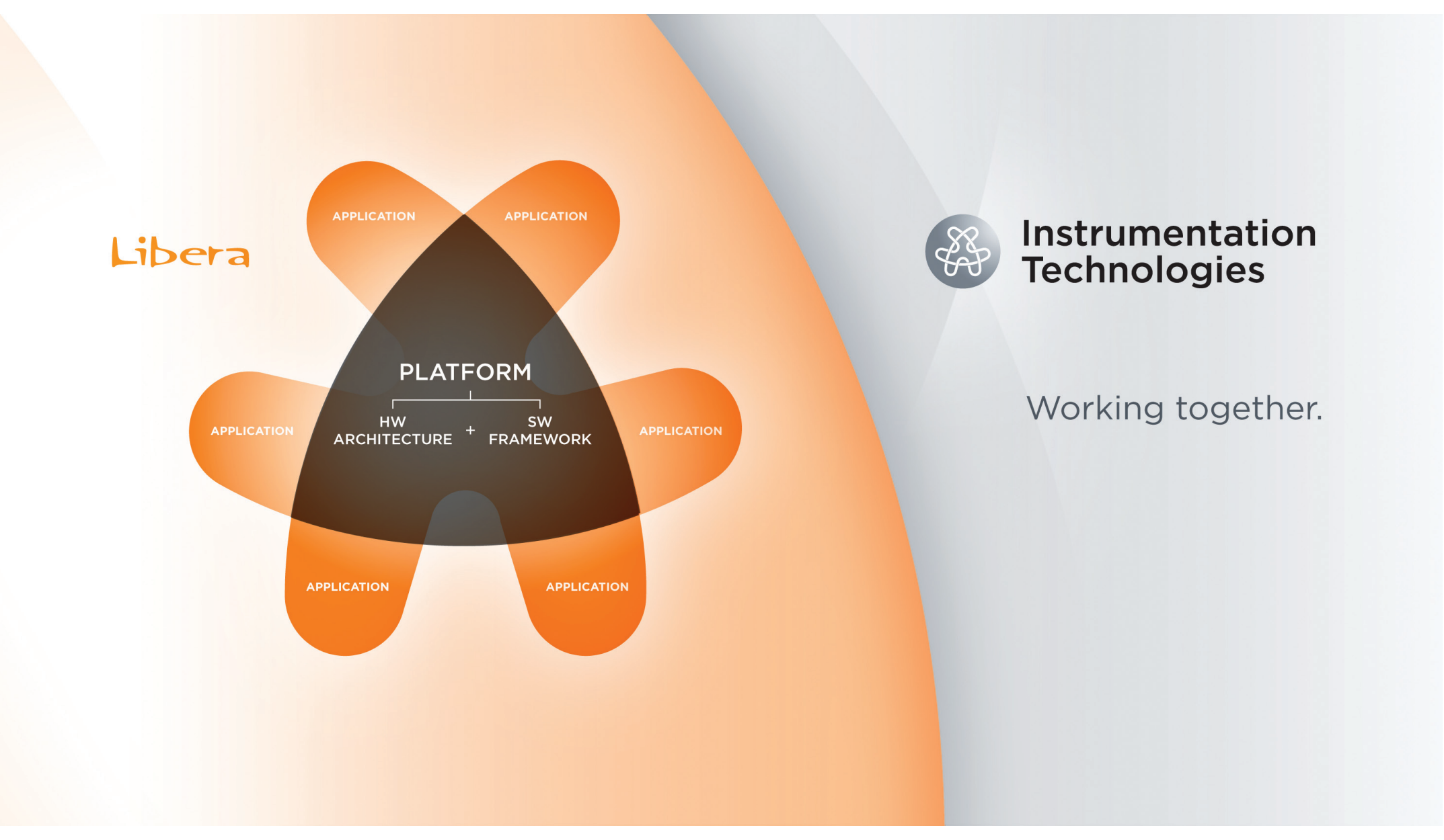
Software became an essential part of electronic devices. Chips are becoming integrated and programmable and software needs to control them:

- FPGA by its nature
- ADCs, VCXOs that are controlled over SPI, I2C buses.

Software interfaces are the points where people communicate with the instrument and define **user experience** through graphical, control, programming and other interfaces.

There is a set of concepts that is occurring repeatedly in measurement instruments:

- **hardware** detection, platform management
- control of **functionality implemented in FPGA**
- configuration **parameters**
- **notification** of changes
- **signal** acquisition, processing and dispatching
- scaffold for running instrument applications
- supporting standard **control system interfaces**



Libera BASE

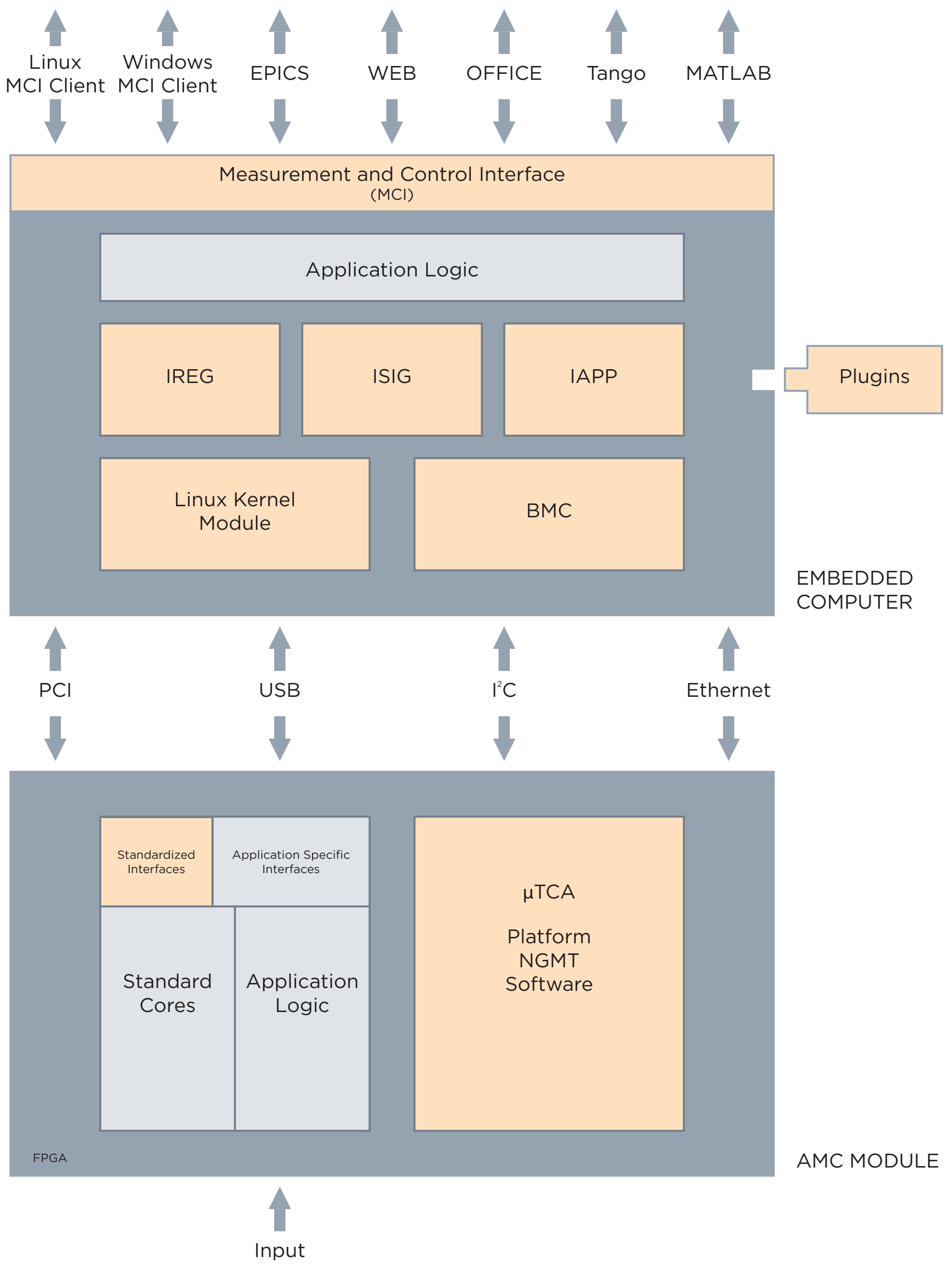
Libera BASE is a **software framework for application development** on different hardware architectures with intuitive structure and programming interfaces. It

- **narrows the gap** between customer's hardware and the machine's control system.
- helps to **focus on the application**, for which the instrument is designed for
- **simplifies integration into various control systems**, however it does not aim to act as a replacement for them.
- ensures higher **reliability**

Concepts and Building Blocks

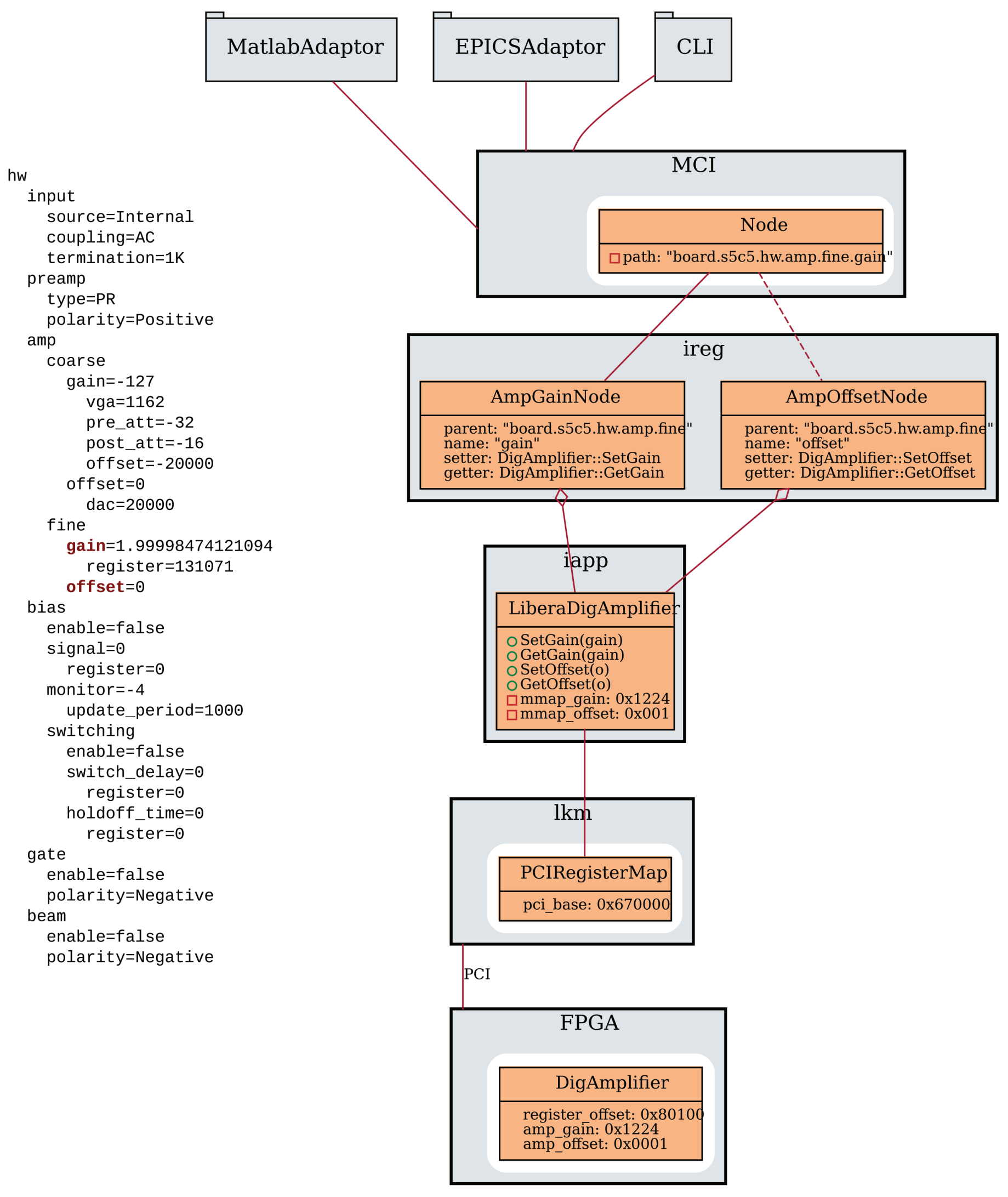
- **fw**: MicroTCA-compliant platform management
- **bmc**: Hardware abstraction layer (uses IPMI, USB, OpenHPI)
- **lkm**: PCI Linux kernel module relies on a set of standardised FPGA registers
- **ireg**: Application parameters as hierarchical tree
- **isig**: Signal acquisition, processing and dispatching
- **iapp**: Application development framework, plugins
- **mci**: Client programming interface (API) for Linux and Windows: exposes registry and access to signals
- **tools**: Simple command line tools for automation and scripting
- **adapters**: for Matlab and LabView scripts, web, EPICS (EDM, pyEpics), Tango CS, FESA

Achitecture of Libera BASE software framework



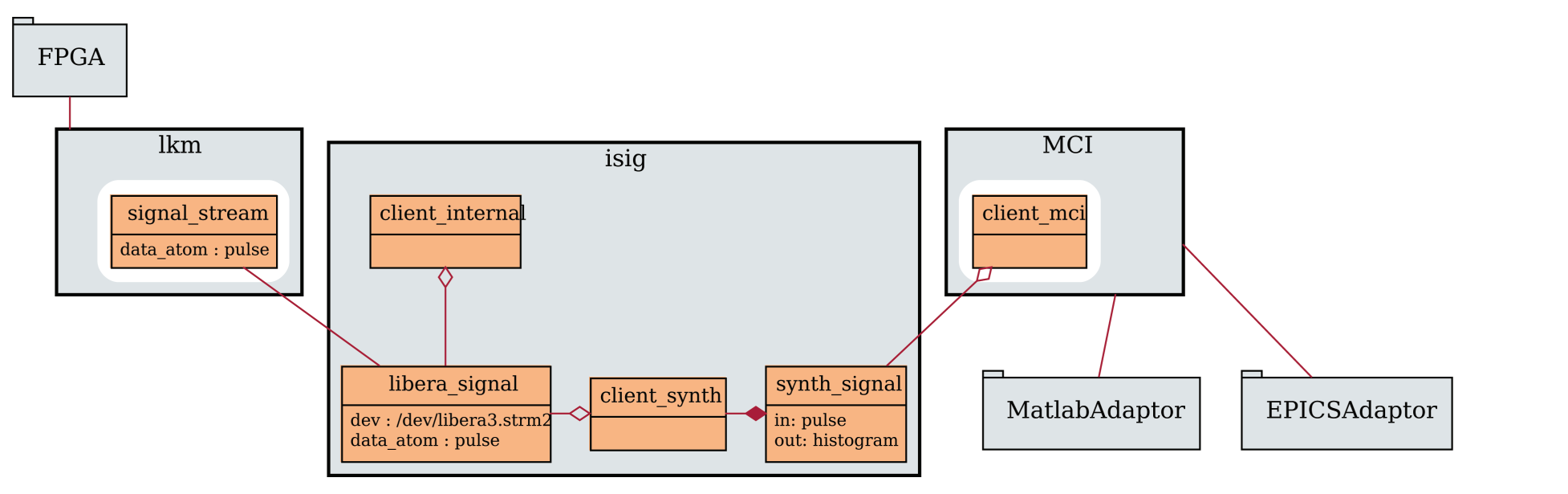
The framework can be easily adapted to support different hardware architectures.

Example of registry parameters of an FPGA digital amplifier



```
hw
  input
    source=Internal
    coupling=AC
    termination=1K
  preamp
    type=PR
    polarity=Positive
  amp
    coarse
      gain=-127
        vga=1162
        pre_att=-32
        post_att=-16
        offset=-20000
      offset=0
        dac=20000
    fine
      gain=1.99998474121094
        register=131071
      offset=0
  bias
    enable=false
    signal=0
      register=0
    monitor=-4
      update_period=1000
    switching
      enable=false
      switch_delay=0
        register=0
      holdoff_time=0
        register=0
  gate
    enable=false
    polarity=Negative
  beam
    enable=false
    polarity=Negative
```

Example of signal acquisition, processing and dispatching

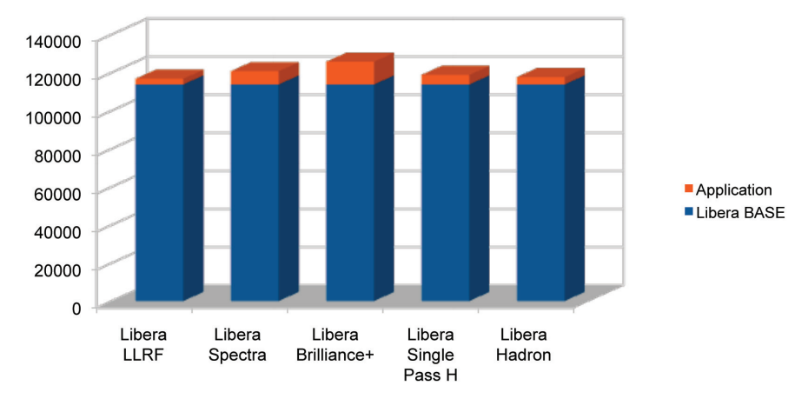


Instruments leverage Libera BASE

Libera BASE accelerates development of instrument applications.

Application defines

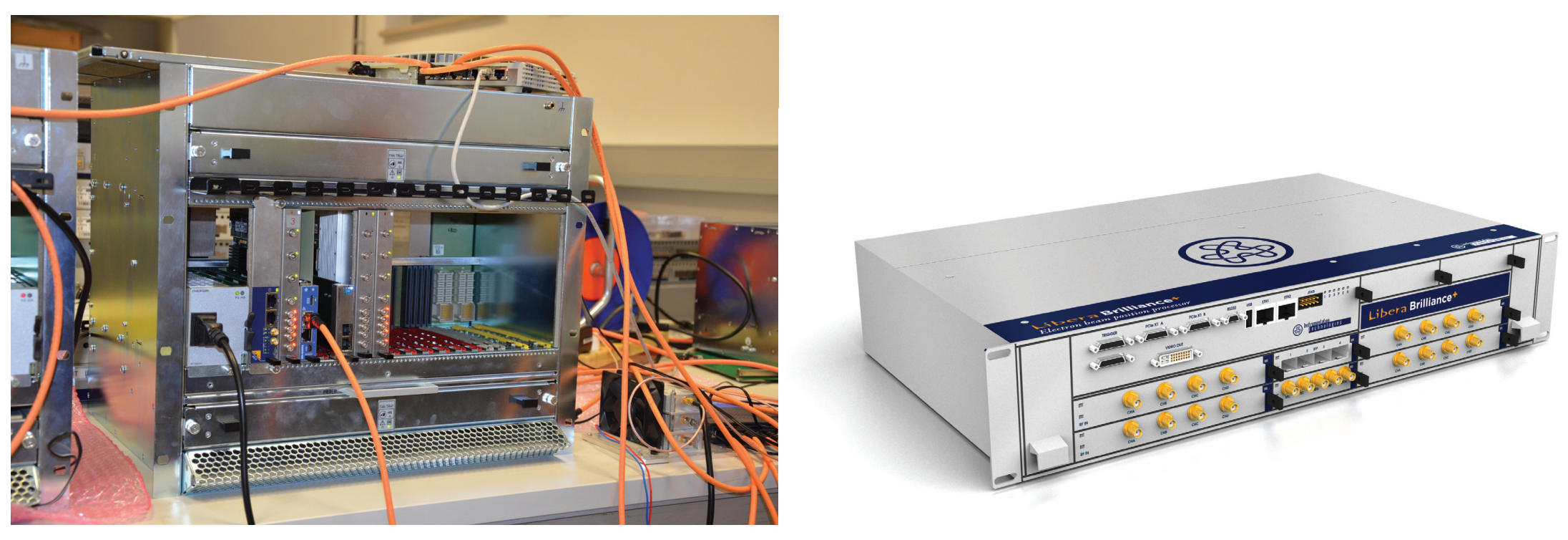
- set of user settable parameters
- signals
- processing and algorithms.



In September 2011, the following instruments use Libera BASE: Libera Brilliance+, Libera Single Pass H, Libera Spectra, Libera LLRF, Libera Hadron.

Measurements of source code size reveal that the share of application-specific software is around 10% of total software; remaining 90% is Libera BASE.

Libera BASE supports different hardware achitectures



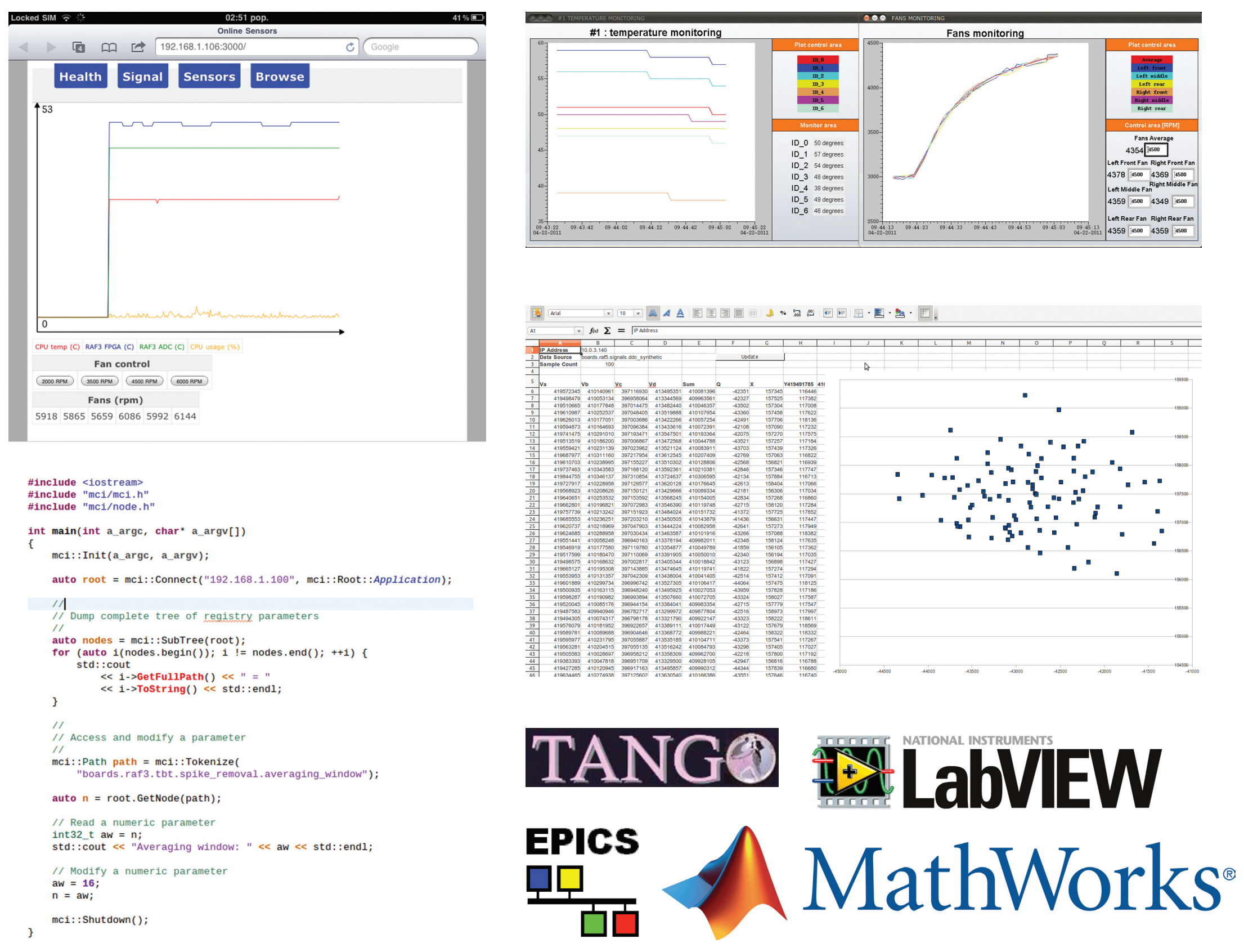
Simplified integration based on MCI API

The MCI programming interface is designed to be simple, intuitive and powerful. **Different instruments expose the same networked API**, which can be used from GNU/Linux and Windows clients. MCI is used to implement different client programs and adaptors.

Different instruments have already been used from the many client applications, for example:

- graphical user interfaces
- command line interface
- Matlab, SCILab
- OpenOffice.org spreadsheet
- mobile devices: iPhone, iPad, Android-based phones
- web browsers
- EPICS EDM

Examples of integration with different applications.



```
#include <iostream>
#include "mci/mci.h"
#include "mci/node.h"

int main(int a_argc, char* a_argv[])
{
    mci::Init(a_argc, a_argv);

    auto root = mci::Connect("192.168.1.100", mci::Root::Application);

    //
    // Dump complete tree of registry parameters
    //
    auto nodes = mci::SubTree(root);
    for (auto i(nodes.begin()); i != nodes.end(); ++i) {
        std::cout
            << i->GetFullPath() << " = "
            << i->ToString() << std::endl;
    }

    //
    // Access and modify a parameter
    //
    mci::Path path = mci::Tokenize(
        "boards.raf3.tbt.spike_removal.averaging_window");

    auto n = root.GetNode(path);

    // Read a numeric parameter
    int32_t aw = n;
    std::cout << "Averaging window: " << aw << std::endl;

    // Modify a numeric parameter
    aw = 16;
    n = aw;

    mci::Shutdown();
}
```

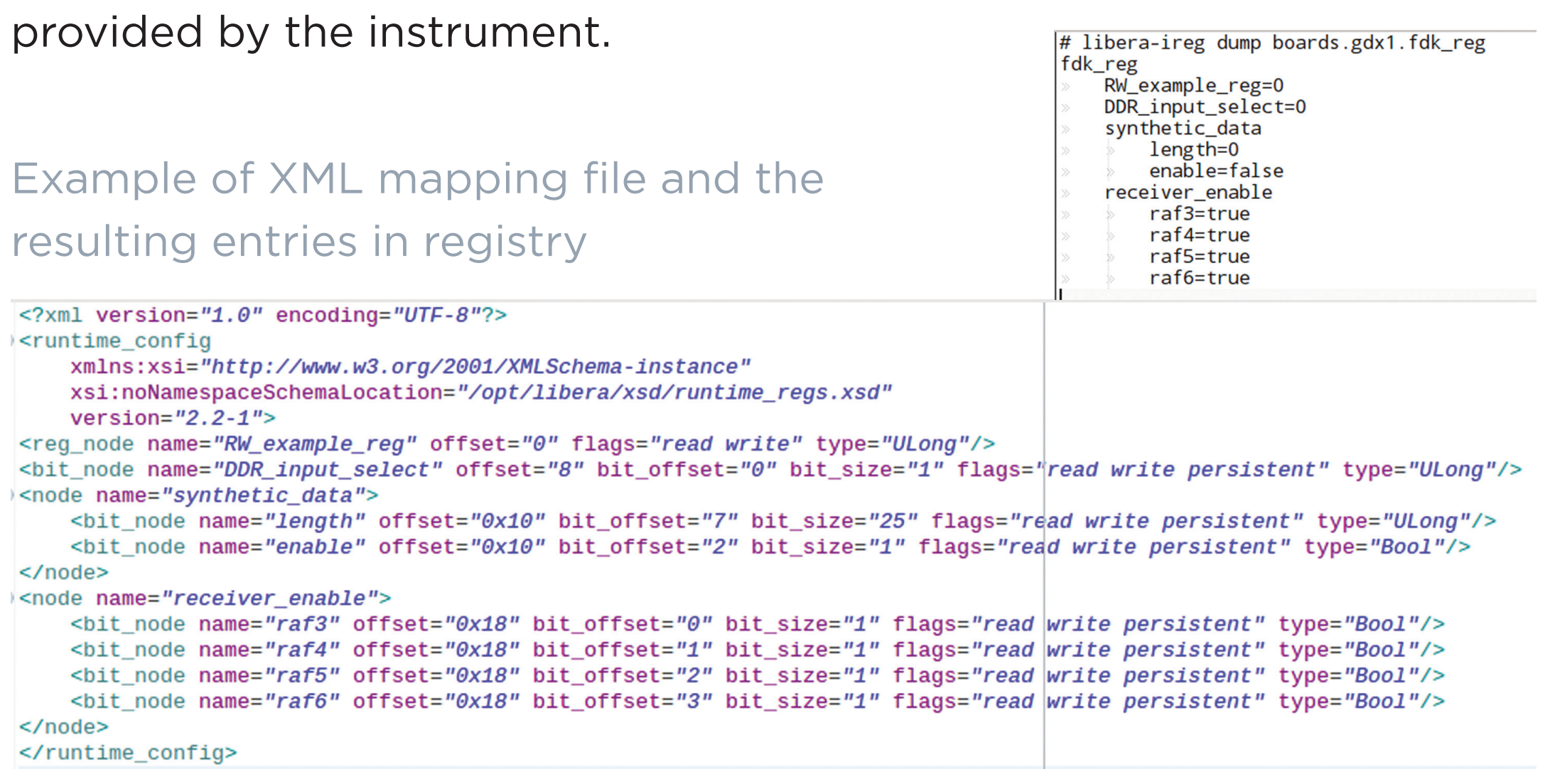
Customisation of Instruments

Several tools are provided to customise Libera instruments:

- FPGA development kit (FDK)
- FPGA to registry map (no programming needed)
- Software development kit (software plugins)

New parameters and signals implemented when extending the instrument are exported through MCI API in the same way as those originally provided by the instrument.

Example of XML mapping file and the resulting entries in registry



```
# libera-ireg dump boards.gdx1.fdk_reg
fdk_reg
    RW_example_reg=0
    DDR_input_select=0
    synthetic_data
        length=0
        enable=false
    receiver_enable
        raf3=true
        raf4=true
        raf5=true
        raf6=true
```

```
<?xml version="1.0" encoding="UTF-8"?>
<runtime_config
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="/opt/libera/xsd/runtime_regs.xsd"
    version="2.2-1">
<reg_node name="RW_example_reg" offset="0" flags="read write" type="ULong"/>
<bit_node name="DDR_input_select" offset="8" bit_offset="0" bit_size="1" flags="read write persistent" type="ULong"/>
<node name="synthetic_data">
    <bit_node name="length" offset="0x10" bit_offset="7" bit_size="25" flags="read write persistent" type="ULong"/>
    <bit_node name="enable" offset="0x10" bit_offset="2" bit_size="1" flags="read write persistent" type="Bool"/>
</node>
<node name="receiver_enable">
    <bit_node name="raf3" offset="0x18" bit_offset="0" bit_size="1" flags="read write persistent" type="Bool"/>
    <bit_node name="raf4" offset="0x18" bit_offset="1" bit_size="1" flags="read write persistent" type="Bool"/>
    <bit_node name="raf5" offset="0x18" bit_offset="2" bit_size="1" flags="read write persistent" type="Bool"/>
    <bit_node name="raf6" offset="0x18" bit_offset="3" bit_size="1" flags="read write persistent" type="Bool"/>
</node>
</runtime_config>
```

Powered by Libera
BASE

Basic Application
Support Environment

Instrumentation Technologies, d. d.,
Velika pot 22, SI-5250 Solkan, Slovenia, **P**: +386 5 335 26 00, **F**: +386 5 335 26 01,
E: info@i-tech.si, sales@i-tech.si, support@i-tech.si, **W**: http://www.i-tech.si

Instrumentation Technologies

When your users demand stability.