Libera BASE

Basic Application Development Environment

Matej Kenda, Libera Workshop 2011, Solkan

mateja.kenda@i-tech.si

Agenda

- Development of Electronic Devices
- Software in reconfigurable devices
- Libera BASE
  - Purpose, structure, benefits
- Scenarios with examples
- Relation to hardware architectures and MicroTCA for Physics
- Summary

How Does An Electronic Device Get Born?

- A challenge, issue or question needs an answer
- A need (or vision) for new type or better device emerges from it
- The device gets defined through the development process
  - Functionality, performance, user experience, …
  - Analysis → design → implementation → validation → production
- Nothing revolutionary, this is our day-to-day job

www.draganfly.com

The Result: A New Electronic Device (Instrument)

Usually a measurement instrument in this community, comprising

- Selected hardware architecture (for example Libera HW Architecture A/B, MicroTCA)
- Instrument specific electronics, RF
- Input signals (analog, digital, timing)
- Platform management
- Imposed communication protocols
  - PCI, IPMI, Ethernet, …
  - Control System protocols, ...
- Generic FPGA cores and application specific processing
- Spplication-specific algorithms, parameters, ...

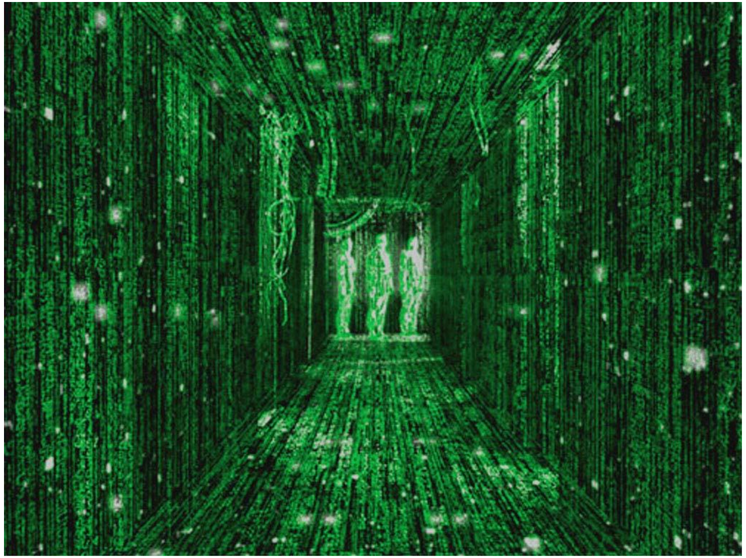
Software in Reconfigurable Electronic Devices

- Share of software in electronic devices is increasing
  - Software is not just an add-on, but an essential part of an electronic device
- Chips are becoming increasingly integrated and programmable
  - FPGA by its nature
  - Controlling ADCs, VCXOs and so on over SPI, I2C
  - As a consequence, more software needs to be written
- Software integrates hardware components, application logic and user interfaces
  - Issues not discovered until the integration pop up then
- Software interfaces are the points where people communicate with the machine
  - Largely defines user experience
  - Human behaviour doesn't comply to standards → needs to be handled in software

Reusable Software: Seeing the Patterns

Concepts occurring repeatedly in measurement instruments:

- Hardware detection, platform management
- Access to functionality implemented in FPGA
- Configuration parameters
- Notification of changes
- Signal acquisition, processing and dispatching
- Scaffold for building instrument applications
- Supporting standard control system interfaces



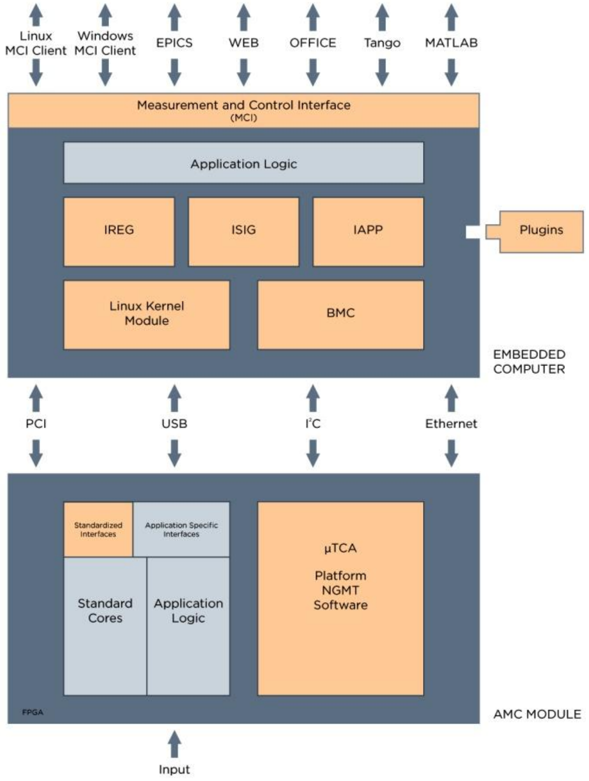
From The Matrix, Warner Bros

Libera BASE

- Libera BASE narrows the gap between your hardware and the machine control system
- Helps to focus on the application with
  - Software framework for application development
  - Intuitive structure and programming interfaces
- Does not intend to replace Control System protocols
- Libera BASE + Libera HW Architecture B = Libera Platform B
- Design started in this form in the beginning of 2010
  - Based on many years of previous experience

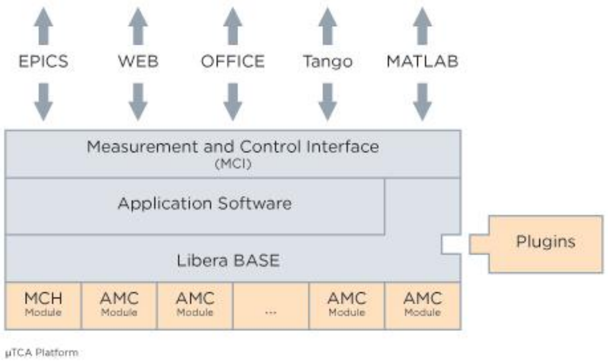
Libera BASE: Concepts and Building Blocks

- **FW**: MicroTCA-compliant platform management
- **BMC**: Hardware abstraction layer (uses IPMI, USB, OpenHPI)
- **LKM**: 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**: Matlab, LabView, web, EPICS, Tango CS, FESA



Libera BASE: Benefits for Customers

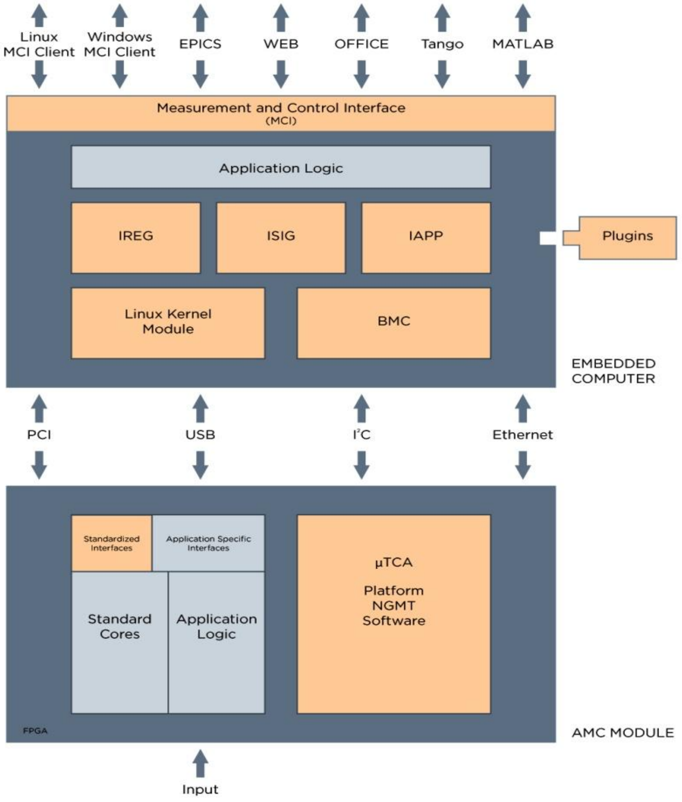
- Enables **immediate use** of your new instrument
- Simple, versatile and effective **instrument interfaces**
- **Ready** to be integrated in **various Control Systems and other applications**
- Suitable for **various hardware architectures**
- Large common base increases **reliability and quality** of instruments
- **Rapid prototyping** and creation of new instruments
- Supports **reconfigurable instruments** with FDK, XML configuration and plugins
  - Easy exchange of solutions between customers and instruments



Libera BASE: Relation to Libera Instruments

- Instrument application software is created on Libera BASE
  - Accelerated development
  - Size ratio of Libera BASE vs application-specific software is approximately 10:1
  - Set of parameters, signals, algorithms are instrument-specific
  - Libera Brilliance+, Libera Single Pass H, Libera Spectra, Libera LLRF
- Common MCI API simplifies integration of multiple types of instruments
- Synergy between Libera BASE and instruments
  - Improved incrementally with each new instrument or its new version
  - Dedicated projects for common functionality
  - Improvements of Libera BASE during development of one instrument get incorporated into other instruments on regular basis

Scenario 1: Development of New Instrument



Instrumentation Technologies

Scenario 2: Integration of Instruments into their Working Environment

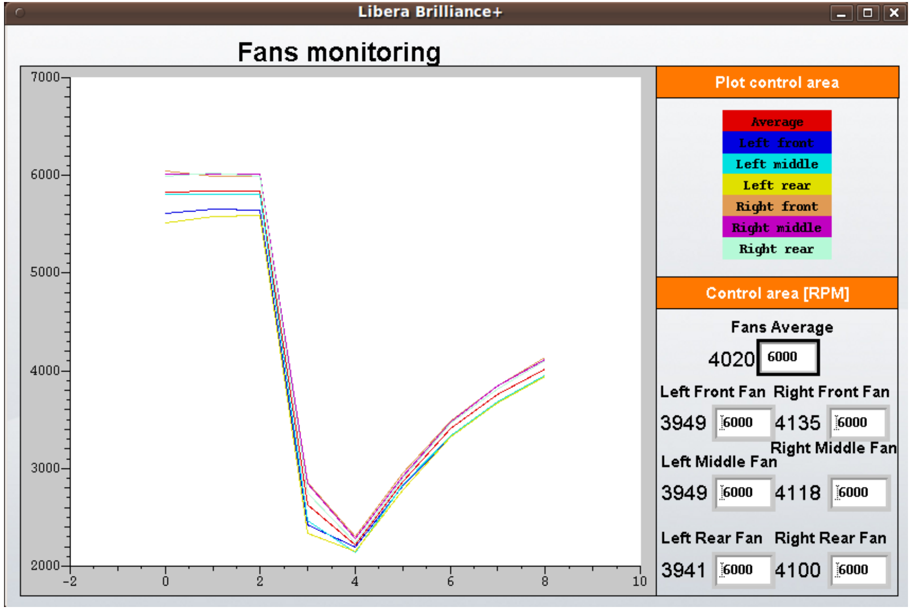
- MCI programming interface
  - Simple, yet powerful and intuitive
  - The same programming interface for all instruments
  - Available for Linux and Windows
  - C++ based API
    - Plans to support other programming languages
- Helps to focus on the application
- Cornerstone for adaptors for integration with
  - Control systems, LabView, Matlab, mobile and web devices, …

Example: Command line

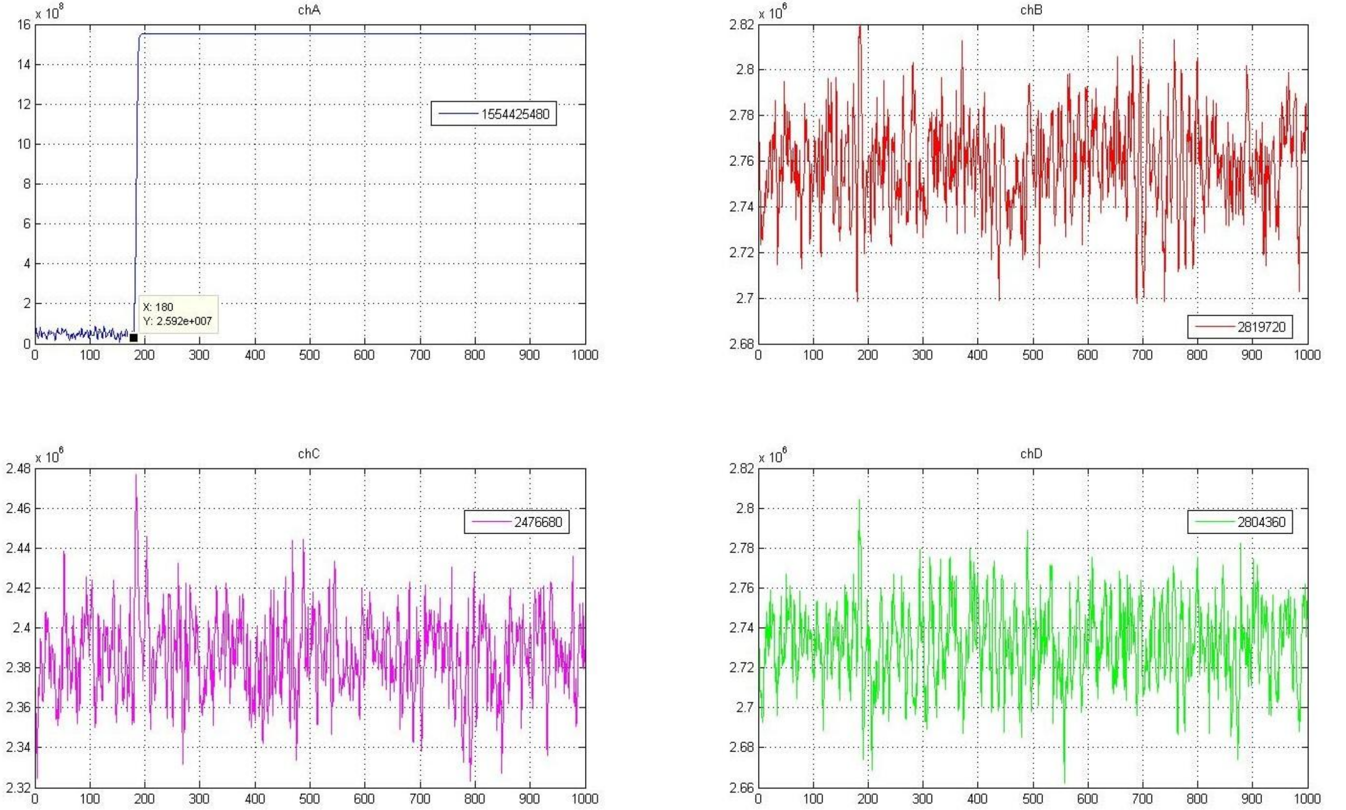
```
$ libera-ireg dump -h 10.0.3.106 -l 3
app-name=libera-ebpm
version=2.2-425-r12548 tupai
boards
  tim2
    info
    clock_info
    pll
    events
    signals
    sc_source=Internal
  raf3
    info
    conditioning
    clock_info
    conf
    tbt
    local_timing
    interlock
    postmortem
    signal_processing
    average_sum
    beam
    events
    signals    application    synchronize_lmt=0
```

```
$ libera-ireg info -h 10.0.3.106 boards.raf3.tbt.spike_removal.averaging_window
--------------------------------------------------
  Registry hostname    : 10.0.3.106
  Node name            : averaging_window
  Value                : 8
  Value type           : ULong ()
  Validator expression : {0,1,2,4,8,16}
  Num of values        : 1
  Num of children      : 0
  Flags                : readable writable persistent
  Root                 : false
  Parent node name     : spike_removal
  Children             : No children for this node
--------------------------------------------------
```

Example: EPICS GUI

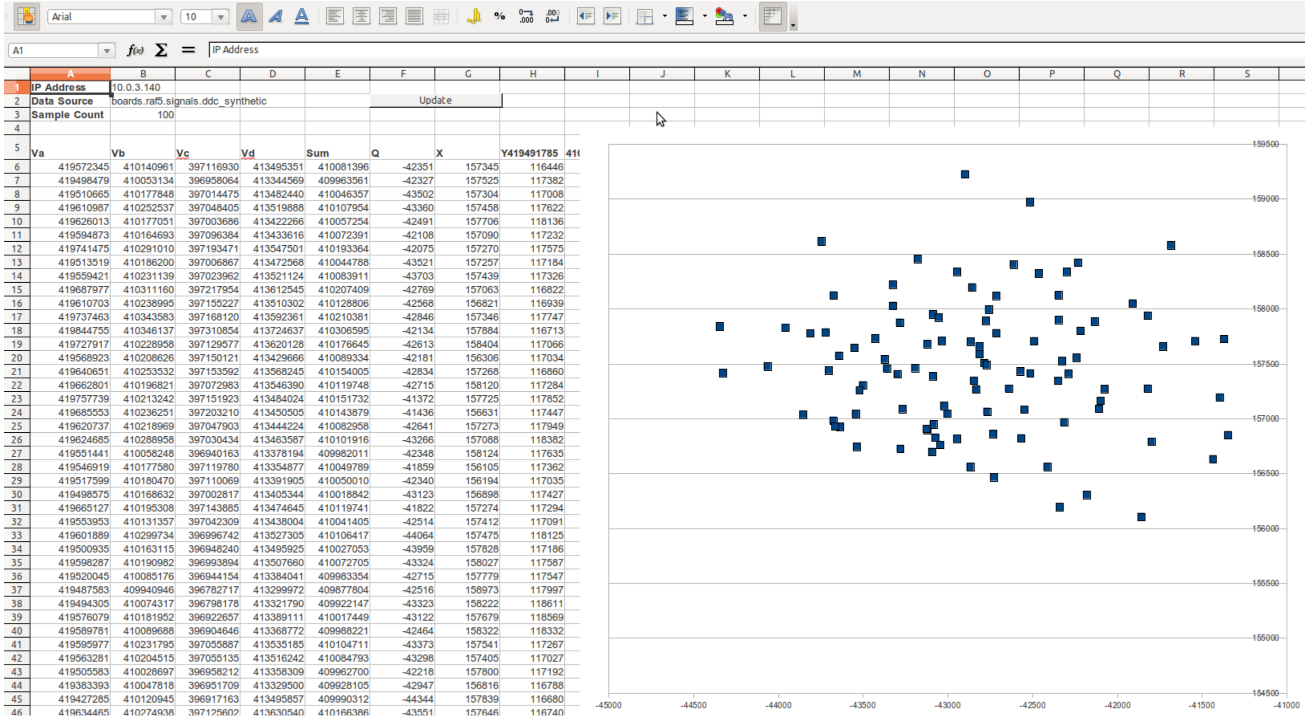


Example: Matlab



Libera

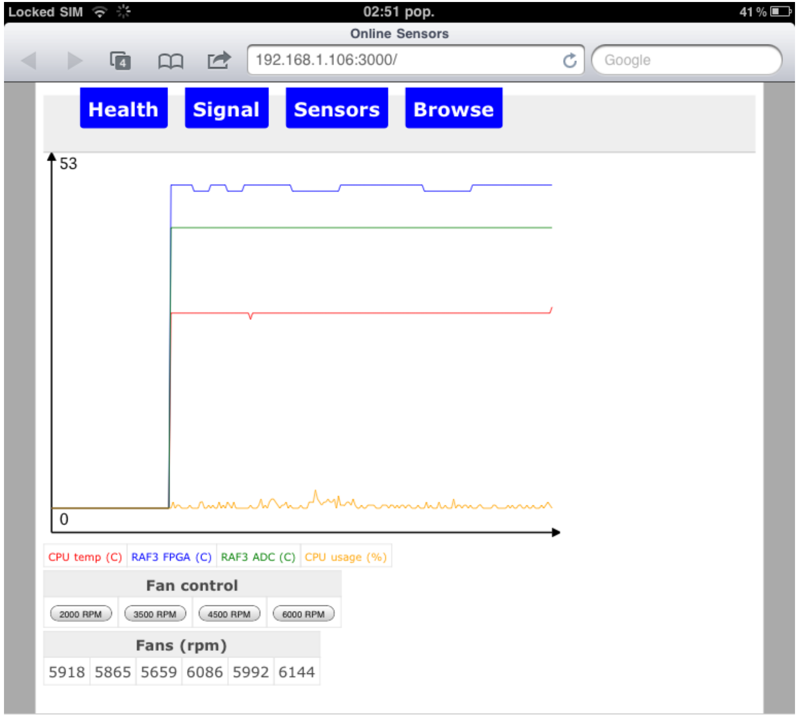
Example: Spreadsheet

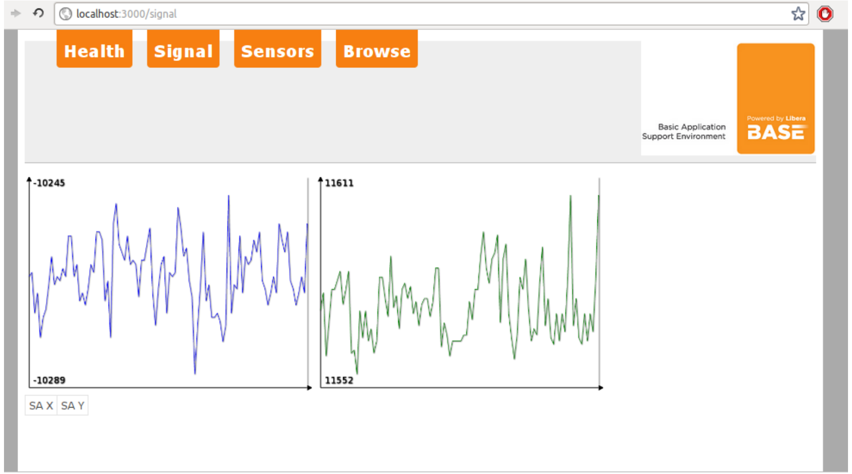


	A	B	C	D	E	F	G	H
1	**IP Address**	10.0.3.140						
2	**Data Source**	boards.raf5.signals.ddc_synthetic						
3	**Sample Count**	100						
4								
5	**Va**	**Vb**	**Vc**	**Vd**	**Sum**	**Q**	**X**	**Y419491785**
6	419572345	410140961	397116930	413495351	410081396	-42351	157345	116446
7	419498479	410053134	396958064	413344569	409963561	-42327	157525	117382
8	419510665	410177848	397014475	413482440	410046357	-43502	157304	117008
9	419610987	410252537	397048405	413519888	410107954	-43360	157458	117622
10	419626013	410177051	397003686	413422266	410057254	-42491	157706	118136
11	419594873	410164693	397096384	413433616	410072391	-42108	157090	117232
12	419741475	410291010	397193471	413547501	410193364	-42075	157270	117575
13	419513519	410186200	397006867	413472568	410044788	-43521	157257	117184
14	419559421	410231139	397023962	413521124	410083911	-43703	157439	117326
15	419687977	410311160	397217954	413612545	410207409	-42769	157063	116822
16	419610703	410238995	397155227	413510302	410128806	-42568	156821	116939
17	419737463	410343583	397168120	413592361	410210381	-42846	157346	117747
18	419844755	410346137	397310854	413724637	410306595	-42134	157884	116713
19	419727917	410228958	397129577	413620128	410176645	-42613	158404	117066
20	419568923	410208626	397150121	413429666	410089334	-42181	156306	117034
21	419640651	410253532	397153592	413568245	410154005	-42834	157268	116860
22	419662801	410196821	397072983	413546390	410119748	-42715	158120	117284
23	419757739	410213242	397151923	413484024	410151732	-41372	157725	117852
24	419685553	410236251	397203210	413450505	410143879	-41436	156631	117447
25	419620737	410218969	397047903	413444224	410082958	-42641	157273	117949
26	419624685	410288958	397030434	413463587	410101916	-43266	157088	118382
27	419551441	410058248	396940163	413378194	409982011	-42348	158124	117635
28	419546919	410177580	397119780	413354877	410049789	-41859	156105	117362
29	419517599	410180470	397110069	413391905	410050010	-42340	156194	117035
30	419498575	410168632	397002817	413405344	410018842	-43123	156898	117427
31	419665127	410195308	397143885	413474645	410119741	-41822	157274	117294
32	419553953	410131357	397042309	413438004	410041405	-42514	157412	117091
33	419601889	410299734	396996742	413527305	410106417	-44064	157475	118125
34	419500935	410163115	396948240	413495925	410027053	-43959	157828	117186
35	419598287	410190982	396993894	413507660	410072705	-43324	158027	117587
36	419520045	410085176	396944154	413384041	409983354	-42715	157779	117547
37	419487583	409940946	396782717	413299972	409877804	-42516	158973	117997
38	419494305	410074317	396798178	413321790	409922147	-43323	158222	118611
39	419576079	410181952	396922657	413389111	410017449	-43122	157679	118569
40	419589781	410089688	396904646	413368772	409988221	-42464	158322	118332
41	419595977	410231795	397055887	413535185	410104711	-43373	157541	117267
42	419563281	410204515	397055135	413516242	410084793	-43298	157405	117027
43	419505583	410028697	396958212	413358309	409962700	-42218	157800	117192
44	419383393	410047818	396951709	413329500	409928105	-42947	156816	116788
45	419427285	410120945	396917163	413495857	409990312	-44344	157839	116680
46	419634465	410274938	397125602	413630540	410166386	-43551	157646	116740

Instrumentation Technologies

Example: Web and Mobile





Powered by Libera BASE

Example: C++ Source Code

```
#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();
}
```


Instrumentation Technologies

Scenario 3: Customising an Instrument

- Tools
  - FPGA development kit
  - FPGA to MCI map
    - No programming: FPGA registers described in an XML file
  - Software plugins
    - full control of the behaviour and access to functionality
- New parameters/signals get exported through MCI API automatically
  - Consequently accessible from any other CS using adaptors

Example: Export FPGA Registers

```xml
<?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>
```

```
# 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
```

Libera BASE and Hardware Architectures

- Currently supported architectures:
  - Libera HW Architecture B
  - Micro TCA
  - Micro TCA for Physics
- Supporting a completely new hardware architecture is not a major effort

Libera BASE and Libera CSPI

- Instruments based on Libera HW Architecture A provide CSPI API
- CSPI is only API, Libera BASE is complete framework
- CSPI was designed for a single hardware architecture and instrument
- Libera BASE can be adapted to support Libera HW Architecture A
  - The opposite would be harder

Libera BASE, Micro TCA for Physics and PICMG

- PICMG (PCI Industrial Computer Manufacturers Group) is working on a reference software implementation for applications on MicroTCA for Physics
- Libera BASE is offering more than what is the goal of the reference implementation
  - Mission is the same: help to focus on the application
- Libera BASE was presented to PICMG to share ideas and will adapt to the reference implementation when available

Status and Plans

- Status
  - Finalising support for MicroTCA and Windows MCI API
- Plans
  - Enhance modularity of instruments
  - Improve connectivity (adaptors)
  - Improve usability (user interfaces, upgrades, ...)

Summary

- Libera BASE
  - Narrows the gap between your hardware and the machine control system
  - Simplifies development of instruments
  - Simplifies integration into control systems
  - Has easy to learn and powerful interfaces
  - Is designed for various hardware technologies
  - Increases reliability and quality of instruments
  - Supports reconfigurable instruments

Development of Electronic Devices

- Multidisciplinary and Synergy:
  - Development of electronics is a multidisciplinary activity
  - A good product is not just a sum of independent parts, but the parts support each other well
- Re-usability:
  - Good design produces reusable parts that can be combined in different ways
  - We get new standard components on the stock
  - Doesn't apply to physical components only, but also particular solutions, placement “tricks”, software
- Stability, quality:
  - Using and extending the same components multiple times increases stability and quality
  - Improvements and fixes found in one device are applied to others