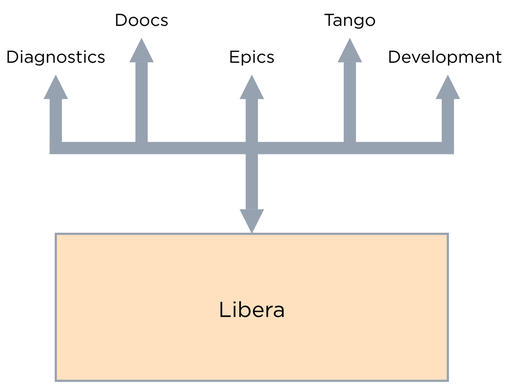
Programming Interfaces for Reconfigurable Instruments

Matej Kenda, Hinko Kočevar, Tomaž Beltram, Aleš Bardorfer, Instrumentation Technologies d.d, Solkan, Slovenia

Abstract

Application Programming Interfaces (APIs) provided by the manufacturers of the instruments for the accelerators are a very important part of the functionality. There are many interface standards (EPICS, TINE, Tango,...) and even same standard can be used in various ways.
Important features of modern instruments are reconfigurability and embedded computing.
The developers of instruments that need to be connected to a control system are facing different requirements: adherence to standard protocols and support of reconfigurable instruments with diverse capabilities with a consistent interface.
Instrumentation Technologies has implemented a well accepted solution with its proprietary Control System Programing Interface (CSPI) layer and adapters for each standard protocol.
There are new challenges like reconfigurability, quality of service, discovery and maintainability that are being addressed with improved Measurement and Control Interface (MCI).

Control System and Software Interfaces



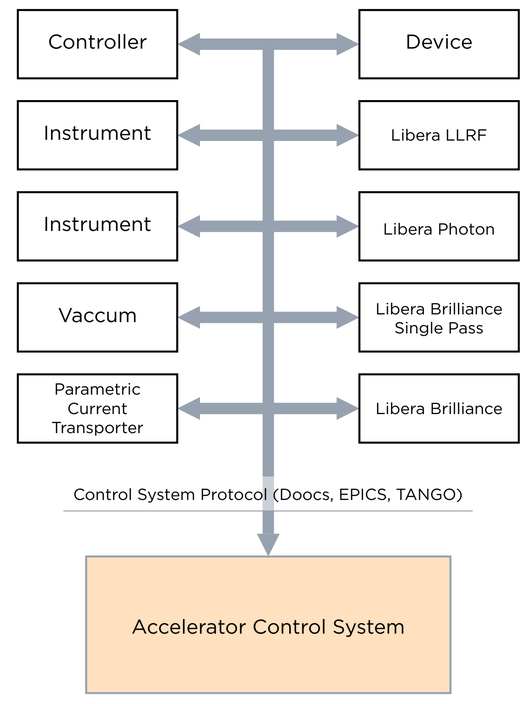
The service that a Control System provides is defined on certain **interface requirements** towards instruments that must cover following areas:

- device discovery, identification and capabilities
- operation mode control and configuration parameters
- events, alarms and health state monitoring
- data acquisition and attributes (data type, size, offset, time-stamp)
- error handling

Instrument Manufacturer's View

From the reverse **point of view**, instrument can be used in different environments. Requests for data can come from different sources for different purposes.

- **Control System**: Different types of control system protocols
- **Other instruments**: Instrument interoperability, multiple instruments working together, clustering, shared processing,
- **Development Lab**: Development, testing of new, updated instruments
- **Maintenance**: Diagnostics, repair



Embedded Computing

Using embedded computers in the instruments enables instruments to behave as **network attached devices** with built-in control system interfaces.

Embedded computer can be used to
- **control** the instrument's operation
- perform a part of **digital signal processing**
- provide **remote access** to the instrument

The embedded computer is one of the important components of an instrument, because it provides convenient way to bring all of the parts (hardware modules, FPGA, software) of an instrument together into a **working application**.
Increasing **computing power** (multi core, SIMD, GPU) can in certain cases be used to replace certain data processing which is usually done by specialised DSP processors and FPGA.

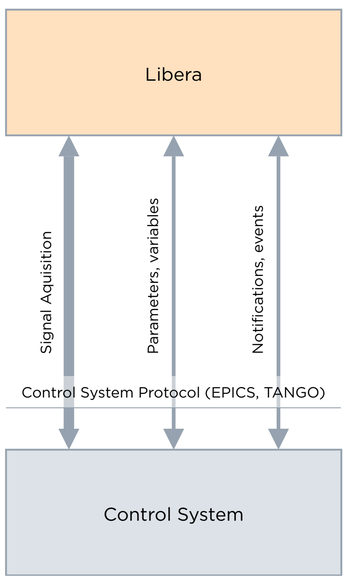
Reconfigurable Instruments

Physical setup and **behaviour** of the instrument is **not completely defined** during manufacturing.

Properties of reconfigurable instruments:
- **Reuse of modules**: Hardware module MOD_A can be used in instrument INS_A, INS_B, ...
- **Behaviour** of the hardware module MOD_A can be altered by loading different FPGA designs
- Instrument INS_A can comprise **variable number of modules** MOD_A, MOD_B, MOD_C, thus defining different **variations of the instrument**.

In general, the responsibilities of the instrument software can be split in several **semi-independent layers**:
- managing hardware platform
- instrument application logic
- external interfaces

Hardware flexibility influences all of the software layers, including external interfaces.



Semantic Types of Information

When defining a data transfer **interface** one there are certain properties that have to be defined:

- Supported **data types**. For example integers, floating point numbers, text, structures and arrays.
- Time considerations in the data transfers: **data rate** and **frequency**.
- **Quality attributes** like QoS or real-time responsiveness.
- **Origin** (data provider, source) and its **destination** (data consumer, sink).
- **Active** or **passive** involvement in the data flow: **data stream** (data provider push) or **data on demand** (data consumer pull) and conversion from one to another.

Signals

	Range	Streaming
Data access	Pulled by user (on demand)	Pushed by instrument (on trigger)
Size	Large	Small
Example	Turn by turn, ADC	Slow acquisition, fast acquisition, events

Programming Interfaces of Libera Family Instruments

Instrumentation Technologies develops families of **specialised instruments** for use in the accelerators. They are all equipped with embedded computers and have network connectivity.
Instruments can be divided in two classes: **Platform A**, **Platform B**. Main difference in hardware is the level of modularity, reconfigurability and computing power.
Platform A instruments contain energy efficient ARM based embedded computer with limited computing power and provide control and signal acquisition through the API called **Control System Programming Interface** (CSPI).
Platform B instruments are **modular and reconfigurable** (QTCA, IPMI and other standards) and comprise powerful embedded computer. Access is provided through the **Measurement and Control Interface** (MCI).
The goal of both programming interfaces is similar: implementation of as much functionality as possible in a common fashion and converting that information to a specific control system protocol as late as possible.
Both types of interfaces provide access to the semantic types of information described above.

Comparison of CSPI and MCI interfaces

	CSPI	MCI
Networked API	Yes (Generic Server)	Yes
Number of signals	Fixed number	Dynamic (based on setup and processing)
Processing	Control loops	Control loops, more DSP algorithms (depends on application)
Configuration	Numeric identifiers	Registry (dynamic structure)
Notifications	Callbacks	Registry (built-in functionality)
CLI	libera	Multiple
Control system interface	EPICS driver (two versions), Tango driver (3rd party)	MCI to EPICS adapter, more planned
GUI	EPICS EDM, Matlab, custom as implemented by CS integrators	Qt GUI, EPICS EDM

CSPI

Hardware configuration of Platform A instruments is **defined at manufacturing**. Available data and the API are coupled together.

CSPI provides interfaces for:
- **Monitoring, controlling the instrument** through a number of parameters. They are all integer numbers and identified by numeric Ids. The set of parameters is fixed for a certain instrument.
- **Acquisition of the signals**. Functions to easily access pre-defined number of signals are available.
- **Change notifications**. A callback function can be registered, which is called with the ID of the parameter that was modified.

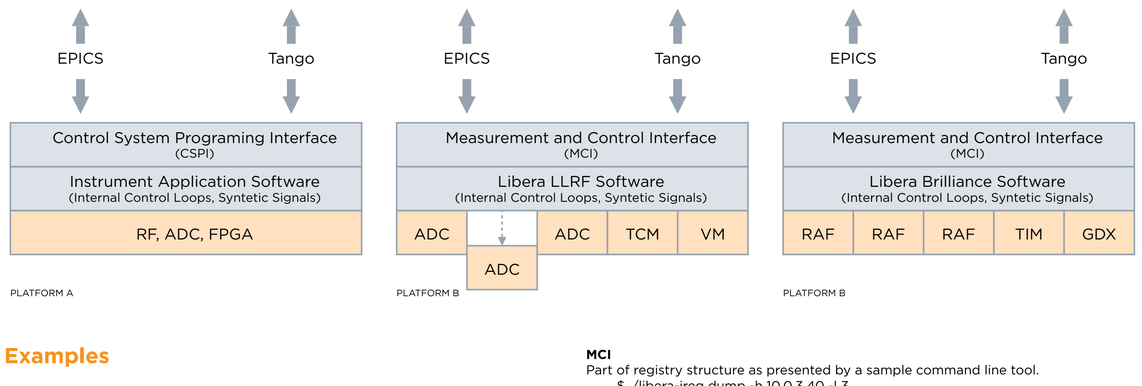
Remote access is provided by:
- Generic server: transparent CSPI API access over the TCP/IP.
- Embedded EPICS driver (Instrumentation Technologies and Diamond Light Source)
- Embedded and External Tango Server (Elettra, Alba, Desy, Elettra, ESRF and Soleil)
- External TINE Server (Desy)

MCI

Dynamic nature of Platform B instrument required different design approach of the software and its API.
MCI has separated classes and functions of the API from the information that they are used to access.
MCI is networked by design.

The following concepts have been introduced in the API:
- **Registry**
- Information is presented in a tree structure, individual nodes are identified by names (similarly as directories and files) and have pre-defined data type
- The tree is **populated by the instrument software dynamically**, depending on the hardware setup and type of the instrument
- Nodes values can be stored **persistently** in **XML** file, can have different **flags** (readable, writeable, constant) that define access to node's value
- Nodes can emit **notifications** (for example: value change). Callbacks functions can be registered to nodes to receive those notifications
- Registry can be used by the instrument application software (**local access**) or remotely (**network access**)
- **Data Streams**
- Available data streams (signals) are enumerated in the registry and defined by instrument application software
- Data stream classes provide access to different types of signals
- The classes simplify processing of the signals (either in the embedded computer or by the client)

Remote access is provided by:
- Directly by **MCI**
- **EPICS adapter**: lightweight server without a database maps MCI registry and signals to EPICS Pvs
- **Tango**, **Tine adapter**: will be developed when needed



Examples

CSPI
Sample command lint tool for reading the Libera unit environment parameters.

```
$ net-libera -i 10.0.0.100 -l
    Temp [C]: 45
    Fans [rpm]: 4590 4560
    Voltages [mV]: 1489 1782 2439 3233 4892 11865 -12020 -5089
    SC PLL: unlocked
    MC PLL: locked
    TRIGmode: 1
    Feature: 0x00000000 0x01000009, Brilliance, GbE, Grouping (RIO)
    Kx [nm]: 10000000
    Ky [nm]: 10000000
```

Example of source code:

```
// Connect to the Libera unit at IP address 10.0.0.100
server_connect ("10.0.0.100", 23271, "224.0.1.240", 0);
// Allocate the environment handle
cspi_allochandle (CSPI_HANDLE_ENV, 0, henv);
// Prepare variables for environment parameter readout
CSPI_ENVPARAMS params;
CSPI_BITMASK mask = -(0LL);
// Acquire the parameter
cspi_getenvparam (henv, &params, mask);
// Release the envirnment handle
cspi_freehandle (CSPI_HANDLE_ENV, henv);
// Disconnect from the Libera unit
server_disconnect ();
```

MCI
Part of registry structure as presented by a sample command line tool.

```
$ ./libera-ireg dump -h 10.0.3.40 -l 3
IP_10-0-3-40
boards
raf5
chassis:0
chassis:1
chassis:2
chassis:5
os
$ ./libera-ireg dump -h 10.0.3.40 -l 3 boards.chassis:1.board_info
board_info
type = VM
status = Running
power_status = Mng + Main
fpga_revision = 7103
fw_version = 17
```

Example of source code:

```
Using namespace mci;

// Connect to instrument 1
RemoteNode h1 = CreateRemoteRootNode("10.0.33.1", 5678, "libera-platformd");
Node r1(h1);
// Connect to instrument 2
RemoteNode h2 = CreateRemoteRootNode("10.0.33.2", 5678, "libera-
platformd");
Node r2(h2);

// Query specific temperature from ins 1
Node tempNode = r1.GetNode( {"boards", "chassis:0", "sensors", "ID_2" } );
float temp = tempNode.GetValue();
```

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