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 Programming 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

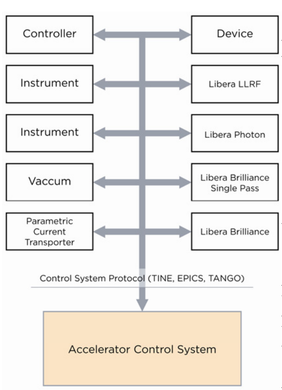

Figure: 1

There are quite some parameters that define environment in which the Control System operates. We can find heterogeneous instruments with different levels of complexity. Beside that the equipment is distributed over large remote regions and needs to provide reliable access regardless of the distance from the control room (see Fig. 1). Another characteristic of such operating environment is that the control is centralized, but the data acquisitions is distributed and to some extent also the data processing.

Based on that we can define interface requirements from the Control System's point 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

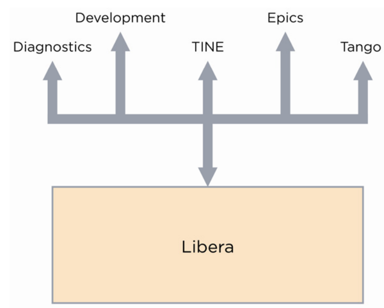

Figure: 2

From the reverse point of view, an instrument can be used in different environments (see Fig. 2). 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

Not all of the access paths are active concurrently.

A great deal of the information access has a common denominator, defined by the type of the information requested.

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 and perform certain digital signal processing.

Software running on the embedded computer can seen as one of the variable parts of a reconfigurable instrument.

RECONFIGURABLE INSTRUMENTS

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

Modern trends in development of instrumentation encourage modularity with many standards for physical dimensions, electrical interconnectivity and data exchange protocols (see Fig. 3).

This leads to the following combinations:

- **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**.

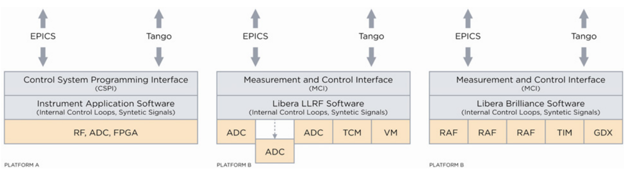

Figure 3

Design of the software, running on such an instrument, must be done in a way to recognise and make use of these combinations.

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

The information transferred between the CS and the instrument can be divided into: digital signal acquisition, alarms (notifications), monitoring and control of the instrument state and behaviour (see Fig. 4).

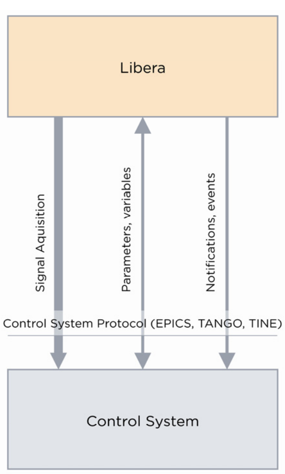

Figure: 4

Time considerations in the data transfers involves data rate and frequency. That is the time that is needed to transfer certain amount of data and the repetition speed how often that transfer happens.

Every data has its origin (data provider, source) and its destination (data consumer, sink). Depending on the active or passive involvement of either side in the data flow we can distinguish between data stream (data provider push) or data on demand (data consumer pull) as depicted in Table 1.

Table:1

	History Buffer	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 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.

Modern trends in instrumentation required Libera instruments to evolve and become more modular and reconfigurable. Platform B instruments comply to μ TCA, IPMI and other standards and comprise powerful embedded computer. Software, developed for these instruments had to be modified as well to support and utilise new hardware platform.

The goal of programming interfaces on both platforms 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 (see Table 2).

Table: 2

	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

Control System Programming Interface (CSPI)

CSPI is available on **Platform A** type of instruments (Libera Briliance, Libera Brilliance Single Pass, Libera Photon, Libera BunchByBunch). These instruments contain energy efficient ARM based embedded computer with limited computing power.

The operating system, used on the computer, is stripped-down distribution of Debian Linux, running on Linux kernel 2.6.20.

The computer is designated for proper operation of the hardware and FPGA from powering the box on to shutting it down and to provide network connectivity.

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**: EPICS IOC driver that utilizes CSPI API. Alternative implementation was developed at Diamond Light Source that by-passes CSPI and communicates with the hardware in more direct fashion.
- Embedded **Tango Server**, developed by Elettra institute.
- External **Tango Server**, developed as a collaborative effort between Alba, Desy, Elettra, ESRF and Soleil institutes.
- External **TINE Server**, developed by Desy institute.

Measurement and Control Interface (MCI)

MCI is the interface of the Platform B instruments (Libera LLRF, Libera Brilliance+, Libera Single Pass H).

Platform B instruments contain various types of i386-based embedded computers. These computers run standard Ubuntu Server edition (Linux kernel 2.6.26 or 2.6.32).

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: tree-structured representation of information, used to monitor and control parameters of an instrument.
    - The tree nodes are **populated by the instrument software dynamically**, depending on the hardware setup and type of the instrument
    - Nodes can emit **notifications** (for example: value change). Callbacks functions can be registered to nodes to receive those notifications
- Data Streams

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

Sample command line tool for reading the Libera unit environment parameters with CSPI

```
$ 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
```

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 ();
```

Structure of MCI registry 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
  ipmi_version = 81
```

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();
```

REFERENCES

[1] CSPI Reference Guide; Instrumentation Technologies d.d.; http://www.i-tech.si
[2] MCI Reference Guide; Instrumentation Technologies d.d.; http://www.i-tech.si
[3] Experimental Physics and Industrial Control System (EPICS); http://www.aps.anl.gov/epics/
[4] TAco Next Generation Objects (TANGO); http://www.tango-controls.org/
[5] Intelligent Platform Management Interface; http://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface
[6] MicroTCA; http://www.picmg.org/v2internal/resourcepage2.cfm?id=5
[7] Gstreamer; http://www.gstreamer.net/