Distributed Data Acquisition and Control by Software Bus

Cecil Bruce-Boye, Dmitry A. Kazakov

Abstract: Increasing global competition forces manufacturers of products from all technical fields to guarantee a high product quality for a long period of time. At the same time it is necessary to minimize production costs. In order to meet all these requirements, on-line data acquisition and processing are of increasing importance in distributed automation systems. A software bus operating on industrial Ethernet has an ability to minimize operating costs by offering easy installation, scalability, high degree of reliability and remote monitoring and control.

Key words: software bus; distributed data acquisition; LabMap

Most modern control processes are such that all input/output operations are not centralized at one particular location. The input/output operations and the corresponding data said to be distributed both physically and logically. Thus both the input/output devices are therefore connected to the control system by a communication network such as industrial Ethernet. A software bus stretching through the whole system provides a distributed access to the data.

Software bus systems in one hand provide abstraction the application level from the hardware specific and decoupling the hardware interface modules from the application level. In the other hand they allow a smooth integration of the software components, supporting a component based software design. Single software components can be separately developed and tested within shorter and more cost efficient cycles, because a software bus architecture natively assists early testing, Simulation without access to the target system, which might be unavailable for time/space/money reasons.

LabMap® [1] is an example of an implementation of the concept of software bus[2]. It is successfully applied the automotive area[3]. LabMap® currently supports a variety of software and hardware protocols, such as OPC, AK, Modbus, CAN etc.

By an application LabMap® is viewed as a set of variables. Each variable has a type, the current value and the current timestamp. No configuration is required. An application may access a value immediately after the system start. The following basic requests are provided for each variable:

- **Get**. The value of a variable can be read in a safe way. It is guarantied that the read value bits and the timestamp are consistent. The application is relieved from the burden of locking the value in presence of concurrent tasks accessing it. The application is unaware of the source of the value and the policy used to actualize or obtain the value.
- **Set.** The value of a variable can be set in a safe way.
- **Request.** A new value of a variable can be requested from the underlying hardware. The application doesn't know which actions are necessary to request the new value. It is the responsibility of the driver. The request is asynchronous and the application is not blocked until input/output completion. However it may enter a non-busy waiting for the input/output completion.
- **Send.** The value of a variable can be sent to the underlying hardware. Like in the case of **request** the application is unaware of the actions the hardware undertakes upon **send.** The application may enter a non-busy waiting for the input/output completion.

LabMap® offers a variety of synchronization mechanisms to the application:

- **Wait for input/output completion.** An application may enter time-limited waiting for completion of input/output involving a variable.
- **Wait for value change.** An application may enter time-limited waiting for the time moment the value of a variable is getting changed. This method is very often used for monitoring system state variables. The main advantage of this approach is that the application developer is relived from the burden of polling the variable value and may concentrate on the domain objectives.
- **Blackboard.** Sometimes it is necessary to trace all changes of a variable. For instance an application visualizing signal waveforms would like to trace the signal. A usual approach for catching value changes involves some kind of notification mechanism between the source of the value and the application. Such point-to-point communication is hard to implement without overstraining the system resources and a danger of resource leaks when an application ends abnormally. The software bus uses an alternative approach. All value changes are written onto the blackboard and remain there for a certain time. Any application may inspect the blackboard contents in order to trace the changes of desired variables.

To achieve better hardware abstraction LabMap® provides integrated support of measurement units. Variables of floating-types have measurement units attached. An application may get the value either in SI units or in the desired units compatible with the units used by the hardware driver.

The hardware abstraction interface uses the messaging mechanism for decoupling the application from the communication software. A hardware driver responsibility is to provide values of the variables. Any variable is a hardware driver attached to it. The driver receives notification messages upon interface start/stop and input / output start / stop. For the values requested from the hardware the following input / output strategies are supported:

- **On request.** A new value is requested from the hardware only on an explicit request of an application.

- **Polling.** A new value is requested periodically.
- **On change.** A new value is requested when the value of the corresponding physical variable is about to change.

The described abstraction mechanisms hide the distributed nature of the software bus from the applications and their components. The software bus itself abstracts networking as a hardware interface. LabMap® has an implementation of the networking interface called LabNet. It implements presentation and application levels (6~7) within OSI model. LabNet may function over different transport layers, but its most compelling use is based on Ethernet.

LabNet is fully transparent for applications using it:

- The data synchronization between distributed partitions is maintained in background and invisible for an application.
- The basic requests of LabMap® are transported by LabNet and executed on remote partitions as if the hardware they control were directly connected to the local host.
- Measurement units are consistently handled, not compromising an ability to deal with different though compatible unit Systems on different network hosts.
- An application may ensure that no other application accesses a variable for write. The software bus distinguishes local and global variables. Local variables are seen by the applications running on the same network host where the variable is declared. Global variables accessible from potentially any are host. Access operations on global variables are also global.
- Values are provided with the time stamp. A hardware driver is responsible for supplying correct time stamps. An application may figure out that the values of several are consistent by inspecting their time stamps.
- LabNet provides peer-to-peer a time synchronization mechanism to ensure consistency of the time stamps. This does not require any synchronization of the partition's clocks, because the time stamps are translated from one clock reading to other transparently. This feature can be used when there is no way to synchronize local clocks. At the same time LabNet supports use of any external time synchronization, such as NTP etc.
- LabNet provides plug and play support. All local variables can be enumerated from remote hosts.

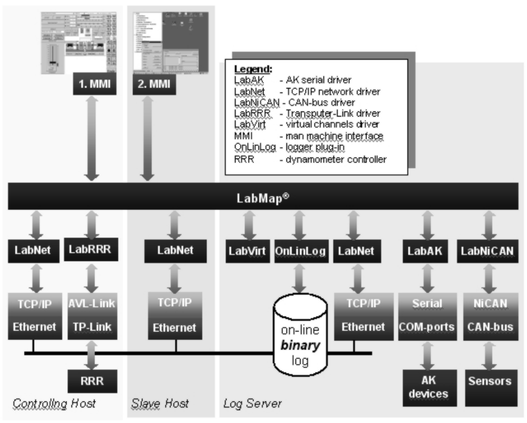

Fig. 1 Software bus system in roller dynamometer automation

The figure 1 represents an example use of the software bus LabMap® in a medium-sized real system. The automation system controls a roller dynamometer used perform various truck tests[4]. Here three permanent hosts are connected by Ethernet and the LabMap® software bus system.

The controlling host runs the primary MMI (man-machine interface) application. It has a real-time controller connected to it.

The slave host is used for secondary MMI application, which does not require a permanent attention of the operator. It has no hardware connected and yet accesses all the necessary values by LabMap®.

The log server is used for on-line logging and also data acquisition using various measurement devices connected to it. The devices use various hardware protocols. The log server also runs various on-line computations using the integrated virtual channel interface of LabMap®. Virtual channels are ones computed from real ones. Another responsibility of the log server is to log the system state. All the data broadcasted by the software bus can be automatically logged. Log writing happens upon demand. No polling is used. Each record is provided with the time stamp. In other words, no data or information is lost. The off-line post processing has an ability to restore the system state to any time moment of the past. Because LabMap® makes all system variables available with the correct time stamps, logging need not to be distributed over the network hosts. LabMap® and Ethernet add a high flexibility to the system configuration.

In addition to the permanent hosts the system has optionally a practically unlimited number of additional hosts running LabMap®. This includes portable computer used for data acquisition within the truck, driver aid devices etc. The whole system can be remotely monitored over WAN. Which allows a cost effective system maintenance. A change in the hardware or its location does not mean a change in the control application but just a reconfiguration in LabMap®.

Conclusion: LabMap® provides an interface to decouple the control application development from the hardware layer. A distributed application communicates with the hardware through and its partitions by variables and LabMap® carries out the actual sending of the data through the appropriate protocol and interfaces for the given hardware. LabMap® offers:

- Very short installation time;
- High grade of extensibility;
- Excellent reliability;
- Minimal engineering effort;
- Reduced maintenance costs.

References:
[1] http://www.cbb-software.de/products/labmap.html
[2] M. Schneider, J. Pisarz, S. Thiebeaut. Ethernet-Newsletter - To whom it concern. IEE automatisierung+datentechnik, No: 10 Oct. 2000
[3] Müller et al / A smooth ride. Testing Technology International. Nov. 2000
[4] D. Kazakov, A. Sechner, C. Bruce- Boye / A smooth ride. Testing Technology International. May 2001