

RACKS - European Initiative for Fieldbus Interface Harmonisation

Jacek Szymanski- CEGELEC SA(FRANCE); Juergen Quade SOFTING GmbH(GERMANY)
John Johansen -PROCES-DATA A/S(DENMARK);

Summary: The paper presents the objectives and some results of ESPRIT IV Project RACKS. The project which started in January 1996 aims at a proposal of a harmonized interface to a set of industrial fieldbus communication solutions including EN50170 (P-NET, PROFIBUS, WorldFIP) and Fieldbus Foundation(FICOMP). The main topics of the paper is CAPI (Common Application Programming Interface), a model of harmonized access to the selected fieldbus stacks.

1. Introduction

In the current state of the art the discipline of distributed manufacturing system design is strongly dependent on the technology of communication networks used in the system. The dependence is so strong that in many cases the details of the communication sub-system employed to federate the embedding system migrate into the user application software. This feature forces the application provider to re-design and re-implement its software packages every time he decides to change the communication sub-system. This very inopportune situation is caused by the use of communication platforms issued from different origins.

The co-existence of heterogeneous fieldbus standards in distributed manufacturing systems is a hard reality dictated by historical, technical and economical reasons . For some large control system providers and field device vendors the part of turnover directly dependent on communication products is too important to let them abandon the solution which they adopted and brought to market. Most evidently this reason is hardly acceptable for application providers for whom the inescapable software tunings caused by this situation void big part of benefits.

In order to reinstate the lost comfort of application provider strategies adopted during the last decade by the producers of computer platforms (PCs, workstations) can be followed. In this currently stable sector every manufacturer has its own implementation of a standardised operating system. The structure of entire platform differs from one provider to another and only the interface can be standardised. In such a situation application providers are free from exploring the details embedded within the platform architecture and should only be aware of the interface

The origins of RACKS project (the abbreviation stands for **R**eusable **A**pplication Interface for **C**ommunicating **R**eal-Time **K**ernels) are linked with the initiative of European fieldbus standardisation which led to elaboration of EN 50170 CENELEC Standard[1]. This standard provides three normalised solution for industrial control which are P-NET, WorldFIP and PROFIBUS. Three European companies : CEGELEC SA, PROCES-DATA A/S and SOFTING GmbH decided in 1995 to create a consortium which had for objective to follow the example of computer providers transposed in the domain of field bus communication equipment. The idea was as follows: if one proposes a common interface to a set of chosen communication standards the applications programs can be developed for wider range of system architectures and the final choice of communication system will be guided only by performance and by economical criteria. The definition of such an interface would hide the details of communication system from the application point of view.

2. Current Status in European Fieldbus Market

Fieldbus is a communication technology widely used in industrial sectors such as process industry, manufacturing, heavy industry, electromechanical applications, energy production, energy transport and distribution, building automation, road and train remote control.

The major segments of automation products are PC based SCADA systems, Programmable Logic Controllers, field devices and VME based systems.

The PC based SCADA (Supervisory Control and Data Acquisition) products can only be successfully and profitably marketed if they target a very wide range of applications. The possibility of making them independent from the communication system linking the supervisory and control parts of the systems will highly contribute to increase the range of their applicability.

Current product lines of Programmable Logic Controllers and process control systems support only one field bus standard chosen by their provider. The opening to other protocols is realised by means of gateways which are almost exclusively customised for specific applications.

A uniform communication interface for different fieldbus systems, represents the benefit for both users and producers of field devices. The users can choose from a larger variety of products knowing that the supervisory control system, maintenance and technical management systems will be independent of the network interface. For the manufacturers, a larger market opens.

VME systems are used in applications where high flexibility and high real-time performance are required. Porting of application software from one platform to other is in this area a very common situation. Even if the current trend of using a standard operating system as POSIX makes this operation easier, the functional dependence of the communication still remains.

The status of the market gives evidence that the industrial actors are willing to converge towards a common fieldbus standard. The technically optimal solution, improbable to emerge for political reasons, would consist in adoption of a unique world-wide accepted standard, federating the advantages of all existing proprietary solutions. The members of RACKS consortium are strongly concerned with the situation on the world market in the field automation products and especially with the market of field bus based systems. They believe that the rational step towards improvement passes through the adoption of evolutionary approach which consists in defining and implementing the user-level common interface to a set of chosen fieldbus systems which follow similar strategies.

3. RACKS Project Objectives

The aim of the RACKS project is to create and prove the validity of a coherent set of communication concepts within the field bus networking domain applied in the automation system design discipline. These concepts should create the baseline for the development of products implementing standardised interfaces between application software and each of chosen field bus standard implementations. The field of interest of the project is limited to the communication systems following either one of three volumes of European standard prEN 50170 (P-NET, PROFIBUS, WorldFIP) or the solution worked out by the FICOMP ESPRIT III project (EP6526) on the basis of the communication model adopted by the Joint Working Group ISA/IEC. The latter solution is close to the standard Proposed by the Fieldbus Foundation[2].

The steps in achieving the objectives led to the construction of an ESPRIT proposal deposited in March 1995. The work plan for the proposed project can be outlined in two points:

- Development of a uniform interface to communication systems. During the first project phase a conceptual model of the common interface to selected field bus communication architectures were developed and specified. This interface , described below in sections 4 and 5 provides the means for accessing physical

objects within a process control system independently of the communication protocol employed.

- Experimental validation of the model. To prove the usefulness of the proposed model four full scale development experiments are proposed. These experiments, led in four independent threads, aim at proving the validity of the proposed model.

The project is working in order to give three types of results : innovating concepts, technical and scientific contributions documenting them and working prototypes of software products which will reinforce the market position of all partners.

The marketing advantages emphasised by the RACKS Consortium as being immediately obtainable from the project results are the following:

- Reduction of development load for Distributed Control Systems. Cost effective and time effective development of further software will be easier to achieve due to the use of standardised platforms.
- Larger field of coverage for existing application programs. Software reusability in different system configurations for different application software packages will be increased.
- Facility of integration of different automation techniques within one system (continuous regulation and discrete control).

The prototype components upgrading the existing communication products are thought as a basis for the future offer of the technology providers. It is important to develop the most appropriate solutions within optimum time scale. For this reason a relatively short project duration time of 18 months was proposed.

4. Conceptual Model Structure

The principal technical idea of RACKS proposal consists in creation of a coherent set of concepts and models which enable the uniform access to physical objects within a distributed process control system. Uniformity of access means in this context the usage of the same interface to the communication network independently from the communication stack employed. The variety of stacks taken into account is limited to four initially chosen by the Consortium :

P-NET, PROFIBUS and WorldFIP stacks which are specified in three volumes of CEN CENELEC standard EN 50170 [1]

FICOMP stack which properties are close to these set-up by Fieldbus Foundation specifications[2]

The results of the works give the conceptual model called Common Application Programming Interface (CAPI). The implementation of CAPI is intended to take in charge mapping of the requests issued by user application programs to the underlying communication stack. This mapping should provide the application process with the same effects independently of the stack used in the system.

The role of CAPI in the architecture of distributed system is twofold. Firstly it introduces the harmonised access to the communication stack by freeing the application provider from burden of taking into account specific properties of different fieldbus standards. Secondly it increases the comfort of application programming by hiding and encapsulation within the interface functionality certain communication oriented programming tasks.

The design process of CAPI was guided by two sub sets of objectives, one linked with the facility of use by the application provider and the other by the demand of conservation of full functionality of the underlying stacks:

The facility of use rely on the following properties:

- *Simplified services*: The definition of the services and the usage of the services is as simple as possible and easy to understand from an application programmers point of view.
- *Simplification of service use*: Some services like download and connection establishment are simplified in order to free the application programmer from handling some protocol sequences and from implementing some state machines.
- *No restrictions on application dynamics*: The CAPI does not impose any restrictions on the application program in relation to time consuming and task shift. The CAPI is able to handle time critical communication in any situation.
- *Multi-User support*: CAPI is able to handle several communicating applications simultaneously, i.e. it can deal with many requests and responses at the same time.
- *Same client access for objects independent of their location and of their transport mechanisms*: The application programmer will not usually see where the manipulated objects are located.

The full functionality of underlying stacks is preserved thanks to the following properties:

- *Complete protocol functionality, no subset*: The CAPI has a simple application programmers interface but from the network point of view provides a complete protocol functionality ; this makes that the CAPI based system is able to communicate with already existing products (see fig.1).
- *Roles*: The CAPI is adapted for all roles (Server, Client, Producer, Consumer).
- *Portable*: The specification of the CAP is made in a way which makes possible to adapt it on several platforms.

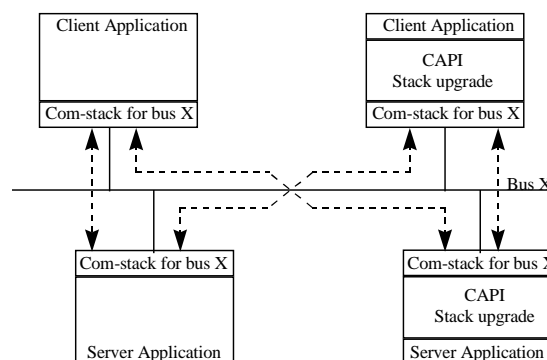


Fig.1. Possible configuration of CAPI based communications

The consortium members wanted the model to represent the current state of the art in the communication domain. For this reason it features the following properties

- *Object based*: The exchange of information is done via objects with methods and properties.
- *Operating System orientation*: The specification do not exclude some emerging technologies like OLE2 automation/OPC servers
- *Easy to integrate*: One concept behind CAPI is to provide a common link between independent suppliers, allowing different companies to develop applications, which

communicate via CAPI. This is achieved by the implementation techniques facilitating the system integration process (in WINDOWS NT environment via Dynamically Linked Libraries).

Some constraints limit the fulfilment off these objectives. The constraints are naturally related to the differences in the four fieldbus systems , such as:

- *Different identification schemes*: Each fieldbus system has its own kind of addresses. This makes it necessary to find a common addressing scheme and the CAPI has to translate these addresses to the fieldbus specific addresses.
- *Different initializing and configuring*: Because the procedure of initializing and configuring is different in each fieldbus system, CAPI provides only general functions for it and the implementation is a communication stack dependent issue.
- *Different capabilities (i.e. segmentation)*: Because of the objective, that CAPI should be usable together with existent applications, it is not possible, to restrict the functionality of any fieldbus system. For this CAPI has to hide some features on the application interface and, in addition, has to provide some of the missing features.
- *Different types of communication channels*: The mechanisms to establish a connections are different in the fieldbus systems. In order to make an application independent of the underlying fieldbus system it is necessary to hide the different connection mechanisms.
- *Different object attributes and service parameters*: Although the bus systems share a common set of services to objects, the services have some different parameters, which influence their behavior.
- *Different data types*: Each fieldbus system has a set of predefined data types. For the common application interface such a set has to be specified as well. Necessary conversions will be made inside CAPI.

In summary, CAPI , a model of harmonised access to four selected fieldbus stacks, is intended to be easy to use, extensible, robust, applicable for both client and for server applications and independent of implementation platform. It is specified as an application interface, and not yet another protocol. That means that it is possible via CAPI to communicate with (existing) applications without deep knowledge of its internal mechanisms.

From the functional point of view CAPI provides the means to create and manipulate collections of objects accessible to user application.

From the architectural point of view CAPI can be modeled as three additional layers between the existent fieldbus protocol stack and the application:

- a) extension (upgrade) of the fieldbus stack,
- b) stack harmonization layer (HL)
- c) comfort layer (CL).

The stack harmonization layer is in charge of mapping the application requests to the communication stack extension, whereas the fieldbus stack extension contains necessary (hidden) state machines as well as functionalities which CAPI adds to the functionalities of the pure fieldbus stack implementation. Because the stack extension resides on top of the (existent) fieldbus interface CAPI has the full use of all services, which a specific bus system provides. The comfort layer provides an easy access to a fieldbus network by hiding communication dependent mechanisms. The comfort layer can be optional for some server applications.

The basic layered model of CAPI is shown in fig.1. Its full description can be found in the report [3].

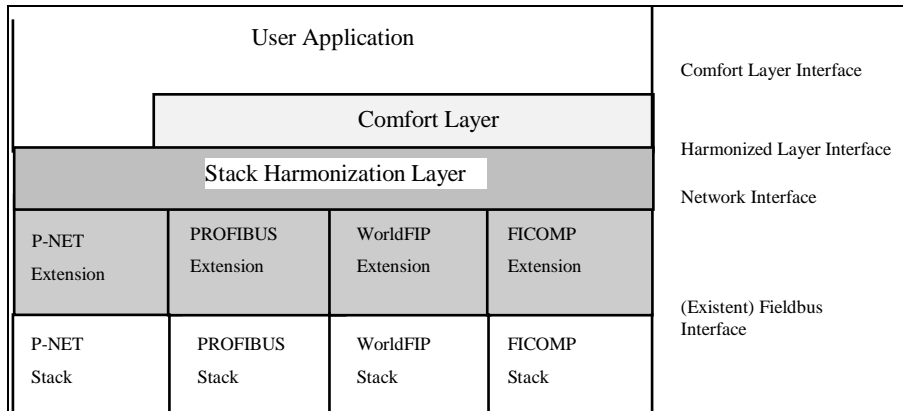


Fig 2: CAPI Model Architecture

5. Programming Interface Structure

As mentioned the application programmer has the choice between two types of interfaces, the h-layer interface and the c-layer interface. Both interfaces provides the functionality of a virtual fieldbus model. This virtual fieldbus model supports MMS-like objects and services.

The h-layer interface resembles to the existent interfaces of fieldbus communication systems, where communication services , composed of primitives (request, indication, response, confirmation) and acting on communication objects are visible on the programming interface. A communication service is activated with the aid of a data structure (called exchange block or shortly x_block) and - as a request or a response - is handed over to the to the h-layer by a function call (hl_write). The h-layer maps this request to the corresponding communication stack. In the other direction the communication stack forwards incoming indications or confirmations to the h-layer, which prepares the exchange block for the application.

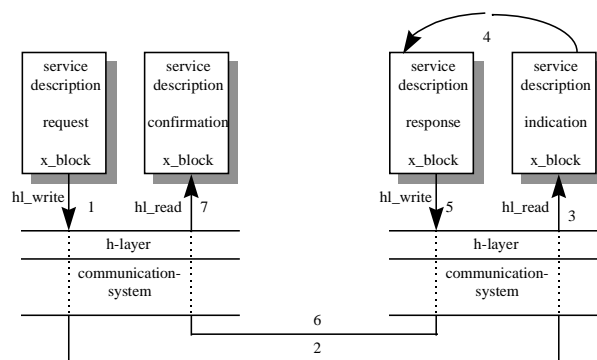


Fig.3. Information flow on the h-layer interface

Because the configuration is completely different in the supported fieldbus systems, it is hidden in a common function. This function has to be called with a stack- and manufacturer specific configuration for each network. The reason for this is, that the RACKS interfaces are specified to give one application at one time access to several communication systems from the same or from different types (multi-user support).

The h-layer interface is very close to the communication systems (virtual fieldbus model) and one of its characteristics is, that it doesn't keep state information any longer than the duration of the request.

The philosophy behind the c-layer interface is application oriented rather than communication system oriented. For an application programmer it means that he is no longer forced to play directly with the fieldbus protocol (even harmonised). The c-layer maps the objects in his communication system to the objects in the application, which can directly be read and written.

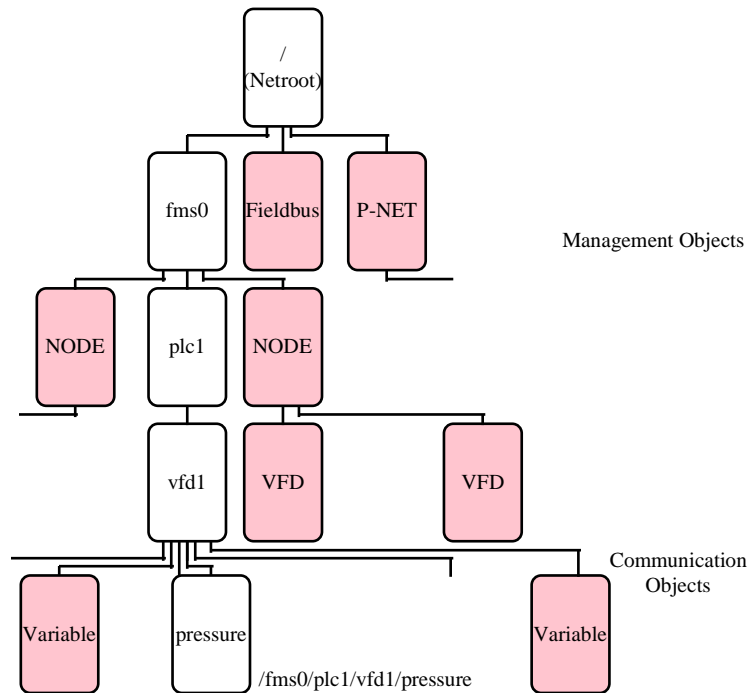


Fig.4.Hierarchical object identification on the c-layer interface.

Connection establishment management, service indication management or data type conversion are handled completely by the c-layer. The goal is to provide an interface which is very close to programming interfaces to other devices especially file system interface, which are well known to programmers and comfortable in use.

The basic c-layer functions are `cl_create`, `cl_delete`, `cl_read`, `cl_write`, `cl_get_attribute` and `cl_set_attribute`.

The c-layer interface gives access to the complete network in a hierarchical way. In opposite to the h-layer, the c-layer keeps object state information. Therefore the application has to create the object in a first step, which is identified by a unique symbolic name (as files are also identified by names and not by magic numbers) similar to a URL. In the second step the application can access (read/write) the object identified by the address of the internal object data structure (object descriptor). This scheme allows very efficient implementations because all necessary information about the object, for example the real fieldbus address, exists when the application wants to read or write the object value.

The c-layer interface does not distinguish between client and server applications. The server application accesses the objects in the same way as the client application is doing it. The only difference is, that the server application has to provide additional code in order to make the link between the communication object and the real object.

Both layers of the interface provide three access modes. In the callback mode an application provided callback function is called if either an indication arrived on a server or if a confirmation arrived on the client. In the blocking mode the application is blocked until the request over the

fieldbus is completed. In the non blocking mode the application has to check (poll) by itself whether a request is completed or not.

6. Example of Application

One of the project results is an experimental application designed to demonstrate the usefulness of concepts developed . The experiment models the application of Common Application Programming Interface (CAPI) in the domain of process control systems distributed on a fieldbus. The basic experiment platform is constructed of four PC work stations running WINDOWS NT linked by four complete fieldbus communication stacks (communication platforms).

The common goal of both experiments is the proof that thanks to functionality of CAPI layer basic application properties remain unchanged and independent of the type of fieldbus stack applied. In order to be exhaustively convincible it merges two classes of application. First of them is a supervisory application VIGO provided by PROCS-DATA A/S. In this experiment VIGO software is running on a PC which communicates with four fieldbus drivers implementing four different stacks. The second part of the experiment is a distributed application which runs on distributed platforms using different fieldbus networks without changing its properties. The PCs simulating the process control system elements execute the same distributed application for each stack.

Most evidently the communication part of validation platform is different for each the stack. The communication hardware is composed of three electronic cards plugged into the PC internal bus. The cards are equipped with appropriate connectors and linked with the appropriate cables. Software components of the communication part are composed of upgraded complete stacks and of implementation of CAPI

The experiments proves that thanks to stack upgrade and CAPI implementation the application provides the same services to its user. For both parts of the experiment the operations which create the functional invariant are as follows :

- start at power-up
- refresh distant objects
- refresh local objects
- detect the disappearance of distant object

The both parts of the experiment show two complementary approaches to the notion of uniform access from the application to the fieldbus stacks.

7. Perspectives

The expected results of RACKS project are the stairway to the technical solutions which may contribute to simplification of the development of fieldbus compatible products. RACKS results may be used by any product or system manufacturer, system integrator to provide an offering in fieldbus related business. RACKS is particularly of interest for those OEMs who have to deliver products to the international market and therefore have to support all relevant fieldbus concepts. In addition RACKS provide an migration path also for numerous proprietary solutions towards the most important open fieldbus systems.

The principal impact of project results is that of development cost reduction. The concepts proposed by the project aim at a proposal of a standard which can reorganise the whole market of automation products and distributed control systems. It is expected that the CAPI implementation will reduce the product development costs of 15% reducing also considerably

time to market. In the system development area the cost reduction seems to be of order of 10%. It can strongly influence the results of automation product providers when one takes into account that direct development costs for high technology products stand for 12% of total turnover.

Acknowledgement

The RACKS Project is partially funded by the European Commission under the Framework Four Programme contract EP 20468.

Reference Documents

- [1] General Purpose Field Communication Systems - CEN CENELEC standard EN50170, 1996
- [2] Fieldbus Message Specifications - Fieldbus Foundation Report FF-94-870
- [3] Specification of Common Application Programming Interface - Deliverable Report D11, RACKS Esprit Project EP20468