A&QT-R 2004 (THETA 14)
2004 IEEE-TTTC - International Conference on Automation
Quality and Testing, Robotics
May 13-15, 2004, Cluj-Napoca, Romania

Design and implementation of industrial communication network interfaces

Authors: Dr. Gheorghe Sebestyen, Dr. Kalman Pusztai

Technical University of Cluj-Napoca
str. G. Bariţiu nr. 26-28, Tel:0264-400476
gheorghe.sebestyen@cs.utcluj.ro
kalman.pusztai@cs.utcluj.ro

Abstract:

Modern distributed control systems require an adequate communication infrastructure. Industrial networks are specially designed to fulfill the requirements of an industrial environment. This paper presents issues and solutions concerning the design and implementation of industrial network interfaces. The first part of the paper summarizes communication requirements and conditions in a distributed control application. As a solution, in the second part of the paper, a generic multilevel and multiprocessor interface model is proposed. This model was used to implement a master interface for the ASi industrial protocol. Tests made on the interface showed that the restrictive specifications of the ASi protocol were successfully fulfilled, demonstrating the feasibility of the proposed model.

Keywords: industrial networks, real-time systems, interface design, distributed control

1. INTRODUCTION

In traditional control systems the information flow between the process automation devices (sensors, actuators) and intelligent control devices (PLCs, regulators, process computers) is assured through a set of dedicated, point-to-point connections. These connections are using analog signals, or in the best case digital serial protocols. This approach limits the scalability of the system, offers a low reliability and implies high cabling costs.

Complex control applications require a more flexible network-based communication [1]. The use of digital networks in control applications offers a number of advantages: lower installation and maintenance costs, more complex data transfer facilities, higher reliability, higher noise immunity, error detection and correction mechanisms, and many others. But in most control applications the use of general-purpose computer networks is not a feasible solution. There are a number of special requirements imposed to the communication infrastructure that derive from the special nature of the control applications [5]; these requirements are not covered by the common computer network protocols. The most critical requirements are as follows:

- higher reliability and security conditions for control data transfers
- real-time message delivery conditions

- deterministic and predictive behavior
- higher immunity to industrial electromagnetic noise and tolerance to wide environment parameters' variations (temperature, humidity, supply voltages)
- small reaction time, adapted to the limitations of local control loops
- efficient transfer of specific data flow patterns (usually small and unstructured control data), which implies small message overheads and explicit support for periodic data transfers
- simple protocols that can be implemented on devices with limited resources

Because of these conditions a number of networks were specially developed for the control field. These networks are known as industrial networks or fieldbuses. Today there is a wide variety of such networks, which sometime cause compatibility and interoperability problems between automation devices.

Design and implementation of interfaces for these networks is a challenging task, mainly because of the previously specified conditions and restrictions. In the next paragraphs a generic interface design model is proposed, which is than used to implement an ASi master interface.

2. DESIGN ISSUES FOR AN INDUSTRIAL NETWORK INTERFACE

The most critical problem in the design of an industrial network interface is the fulfillment of real-time conditions [2]. Most industrial protocols offer explicit mechanisms to specify and control the time restrictions of control messages [4]. Periodic data transfers are typical for control systems (e.g. process data acquisition, command generation, visualization, etc.), because most of the control and supervisory functions are repeatedly activated. The network interface must assure the delivery of such data with high time precision. Any delay or variation in the transmission period may affect the accuracy of the computed control functions. This is understandable if we consider that almost all the control functions are time dependent and their generation implies integral and derivative operations in time domain.

Often the correct behavior of a network interface is directly determined by the fulfillment of time restrictions. If a network node is not able to respond in a predefined time interval to a request, it is considered, by the other nodes connected on the network, as defective and it is excluded from the future data transfers. This condition is present also in some general purpose computer network protocols, but the difference is in the magnitude of the time limit. For instance, in the case of ASi protocol, the magnitude of the time restrictions is around 5-10 νs, which is comparable with the execution time of a single instruction. Therefore many low level protocol functions must be implemented in hardware.

Another issue is the execution of different communication, synchronization and network administration functions in a parallel way [6]. In the same time, the interface must react to the network traffic (e.g. implement the multiple access control mechanisms, identify the network configuration, detect errors, etc.), it must respond to requests coming from the application level, it must prepare data for the outgoing data flow and it must receive data from the incoming flow. Considering the previously mentioned time restrictions, in most cases this issue can be solved only with a multiprocessor architecture and special task scheduling and execution strategies.

An important aspect in the design of a control network is the reliability and security issues. It is not acceptable to have failures or even accidents in the manufacturing process caused by the communication environment. Therefore the communication

protocol includes more complex error detection and data recovery mechanisms, which must be implemented at a low level, in the network interface. Another reason for an error-free transmission is the fact that communication errors and their recovery mechanisms may significantly affect message delivery delays.

3. GENERIC MODEL OF AN INDUSTRIAL NETWORK INTERFACE

The proposed solution for the issues mentioned in the previous paragraph is a multilevel and multiprocessor interface model. Figure 1 shows the general scheme of this solution.

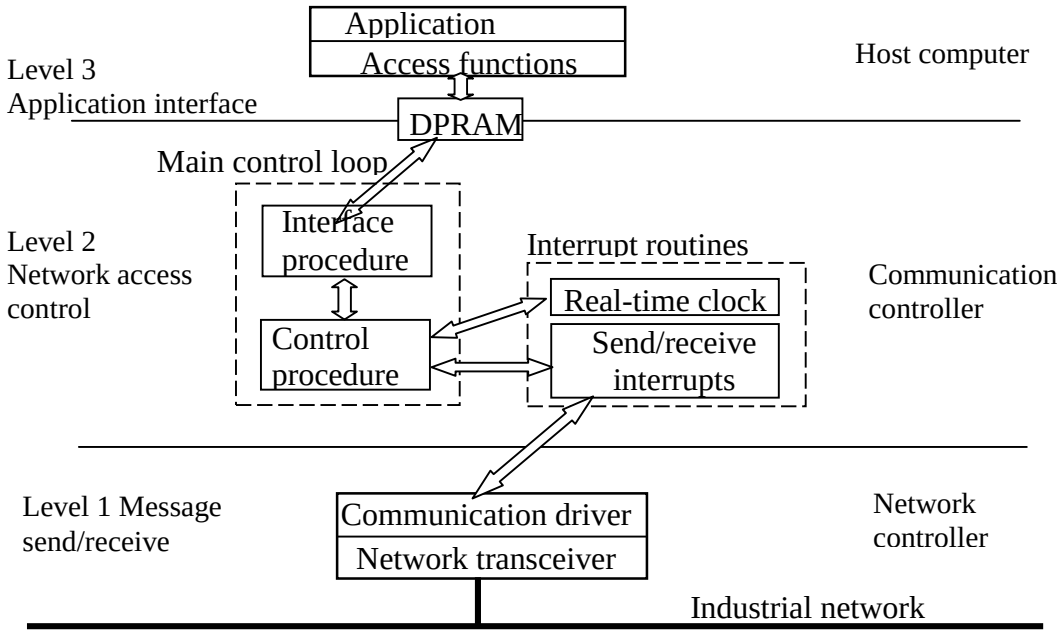

Figure1. Multilevel interface model

The communication functions implemented by the network interface are divided on three logical levels, as follows:

The lowest level (called **Message Transmission/Reception**) is responsible for the transmission and reception of messages on the communication medium. It solves the following problems:

- digital information coding and decoding
- detection of transmission errors (e.g. erroneous data, delayed or lost messages)
- message selection or filtering, based on address of on content
- message buffering
- network traffic monitoring

This level is implemented with specially designed transceiver circuits (for the coding/decoding purpose), with a general-purpose microcontroller (e.g. from the Microchip's PIC family) and a firmware embedded in the non-volatile memory of the microcontroller.

The second level (called **Network Access Control and Administration**) implements the network access mechanism specified in the protocol. In the case of industrial networks deterministic access mechanisms are preferred, such as: token-passing on a bus or ring topology (e.g. Profibus, Interbus-S), circular polling (e.g. ASi or WorldFIP), virtual token (e.g. P-Net), time division and reservation (e.g. Mars), or

even collision-based mechanism but with more deterministic static priorities (e.g. CAN). This level is also responsible for the automatic network identification and reconfiguration functions.

The implementation of this level is the most critical part of the interface design because this level solves a number of complex communication and network administration functions in a concurrent manner and with restrictive time limits. At this level the following functions are implemented:

- network identification and access order establishment
- time parameters' settings (e.g. duration of a token roundtrip cycle, transmission time limits, etc.)
- error detection and correction, node failure detection and recovery
- bidirectional data flow transmission and reception
- synchronization with the lower and the upper levels

A medium-performance microcontroller (e.g. Intel's I8032 microcontroller family) is dedicated to this purpose. The specified functions are implemented through a program written in the microcontroller's internal memory. To fulfill the time restrictions of the concurrent tasks, a special program execution model must be adopted. There are a number of classical solutions for this purpose, such as:

- the main control loop approach - concurrent tasks are sequentially activated in a loop that has a well controlled cycle time
- interrupt-based system - timer and external interrupts determine the execution order of the concurrent tasks
- foreground-background approach - time-critical tasks are executed with a higher priority in a foreground loop and less critical tasks in the time that remains between loops
- the state automaton model - the execution of a given function is planed in a number of steps (states), avoiding wait loops for external events
- real-time scheduler - tasks are activated by a scheduler module based on their time parameters (deadlines, periods)

Analyzing the available computing resources of a microcontroller and the required functional and time conditions, at the end, a mixed solution was adopted, in which the first four methods were combined: There is a main loop executed in foreground that implements the network access control and administration functions, there are a number of routines activated by interrupts (the clock routine and the send/receive routines) and there is an upper interface module executed in background. All program modules are designed as state automatons; if activated, a module will execute at most a single transition between two states. In this way waiting loops are avoided and the reaction time of the interface is significantly improved. This approach allows a precise control of the execution time for the main control loop. The designer can evaluate the execution time by counting the maximum number of instructions contained in a program sequence.

The third level (called **Application Interface**) contains a set of functions used by a control application to access the network's communication facilities. These functions allow: bidirectional data transfers, periodic data acquisition and commands generation, network configuration changes and visualization. This level is implemented by the host processor. The functions are gathered in a dynamically Linked Library (DLL).

In the proposed model a special concern was given to the interface between levels. This interface must solve the data exchange and synchronization between levels. The adopted mechanisms directly influence the reaction time of the network interface.

For the lower two levels synchronization is solved through interrupt signals and flags. Whenever a new message is received (receive buffer full) or sent (send buffer empty) an interrupt is generated to the microcontroller of the second level. Data is transferred through interrupt routines. The data is physically transferred through an input and an output register connected to both microcontrollers. The network controller preserves the sent or received messages in a small circular buffer that may contain the last 2 to 4 messages (its dimension is optimized to the protocol's characteristics). Through this buffer synchronization delays are avoided. The data to be sent is prepared in advance and it is placed in the buffer. In this way the message is sent on the network as soon as the network access mechanism allows it.

The interface between the second and the third level is implemented with a dual-port-RAM memory (DPRAM). This memory can be accessed simultaneously by the host processor and by the second level microcontroller. It contains a section for data received and sent and a section for synchronization flags. To avoid access conflicts to the same data, two zones are defined: one available for the host computer and one for the communication controller. When data and flags are filled-in the zones are logically switched. This dual-memory approach assures a very good transfer time, avoiding unnecessary delays. There is also possible to access this memory directly from the application program shortcutting the third level.

4. CASE STUDY: IMPLEMENTATION OF A MASTER INTERFACE FOR ASi PROTOCOL

The ASi (Actuator Sensor interface) protocol is a very good testbed for the proposed generic network interface model because it has very restrictive time limits and incorporates complex network administration, automatic reconfiguration and error detection mechanisms.

ASi is an industrial protocol meant to connect simple automation devices (sensors and actuators) to a medium complexity device (PLC or process computer). The communication is made through a two-wire bus, which is also used to supply the network nodes. On a bus up to 32 nodes (64 in the extended version) may be connected. The master node controls the traffic on the bus and periodically sends messages and interrogates the slave nodes. A slave node handles four input and four output digital signals. In a bus cycle the master node sends and receives data to and from all the existing and enabled slave nodes. The protocol assures automatic identification of new nodes and reconfiguration in case of defective nodes. The functions of a defective node are automatically transferred to a redundant node that has similar characteristics.

There were two critical aspects in the implementation of the master ASi interface: the complexity of the automatic network identification and reconfiguration mechanism included in the protocol and the fulfillment of time restrictions. The protocol requires solving the network administration functions (e.g. identification of new nodes, detection of defective nodes, transfer of functions to new nodes, etc.) in parallel with normal data transfers. A given function (e.g. identification of a new node) is achieved during a number of complete data transfer cycles (7 in the case of identification). Any consecutive transmission errors affect the evolution of the administrative functions.

The parallel execution was simulated by using the state automaton model: every function, including the data transmission one, was implemented as a sequence of steps; when the main control loop calls a routine implementing a given function only one step is executed. In this way the execution time is strictly controlled and during a single loop every function has the opportunity to evolve to the next stage if the conditions for that step are fulfilled. To increase the responsiveness of the system two priority levels were introduced. Higher priority functions are called in every cycle of the control loop; lower priority functions are called only at a multiple of the control loop cycle. In the higher priority class time-critical functions were included such as low-level data transfer function or the error detection function; the lower priority class contains data access functions implemented for the third level.

Time restrictions are part of the ASi protocol specification. For instance the delay between two consecutive messages must be in the interval 12-18 νs, otherwise it is considered a transmission error. The fulfillment of these sever time restrictions are achieved with a combination of methods and components that derive from the generic interface model: parallel execution of interface function on three processors, a combined execution strategy for the software modules, flexible dataflow support between the logical levels and processors and optimized assembly language programming.

The measurements and tests made on the interface showed compatibility with the protocol specifications regarding functional behavior and time restrictions.

7. CONCLUSIONS

This paper analyzes the main issues and possible solutions concerning the design and practical implementation of interfaces for industrial networks. A generic multilevel and multiprocessor interface model is proposed as a solution to the special requirements present in the specifications of an industrial network. A combined execution strategy is proposed to solve the real-time restrictions imposed by the protocol. The model was used to implement an ASi master interface. The tests and validation procedures showed the feasibility of the generic model.

References

1. Cardeira C., Simonot-Lion F, Bayard M., [1995], “Intelligent Field Devices and Field Buses: Impact on Applications Design Methodology*”, Studies in Informatics and Control,* Vol 4 No. 3 1995, pp 255-262
2. Pasadas F. Cardeira C., [1995], Real-Time Protocols for Industrial LANs, in *Proceedings of the Network of Excellence in Intelligent Control and Integrated Manufacturing Systems*, Portugal
3. Sebestyen Ghe., [1996], A Multistate Model Used in Small Real-Time Systems, in *Automation Computers and Mathematics ACAM 1996*, lito. Technical University of Cluj
4. Sebestyen Ghe., [1998], Real-Time Communications through Industrial Networks, *Proceedings of A&Q'98 International Conference on Automation and Quality Control*, Cluj-Napoca, p A243-249
5. Upender B. Koopman P., [1994], Communication Protocols for Embedded Systems, *in Embedded Systems Programming*, 7(11), pp46-58
6. **** [2000], Actuator Sensor Interface V2.11 Complete specification, *protocol standard*