NAIST-IS-DT 361040

Doctoral Thesis

DESIGN AND ANALYSIS OF FIELDBUS CONTROL SYSTEM

Yan-bin Pang

July 5, 2004

Department of Information Systems

Graduate School of Information Science

Nara Institute of Science and Technology

Doctoral Thesis
Submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Yan-bin Pang

Thesis committee: **Hirokazu Nishitani Professor**
Kenji Sugimoto Professor
Shoji Kasahara Associate Professor

DESIGN AND ANALYSIS OF FIELDBUS CONTROL SYSTEM[*]

Yan-bin Pang

Abstract

A Fieldbus Control System (FCS) is a distributed system composed of field devices and control and monitoring equipments and it is integrated into the physical environment of a plant or factory. The FCS is increasingly being used in the automation arena because of the advantages of the fieldbus. In recent years, more than 50 different names of available fieldbuses have emerged. However, research works for the FCS lag far behind their current practice. This thesis investigates several important issues of the FCS.

In Chapter 1, differences between the FCS and a conventional control system are summarized, and related researches are reviewed with focus on timing analysis, evaluation, control algorithms, modeling and simulation, development and applications. Chapter 2 presents models of different fieldbus protocols from basic to complete ones to lay fundamentals for this study. The main fieldbuses are introduced briefly and their characteristics are discussed. The fieldbus medium access control mechanism is also discussed. As a result, several classification methods of fieldbuses are proposed.

In Chapter 3, the timing characteristics of the FCS are analyzed in details. The control period of a control loop in the FCS is formulated and analyzed. The stability condition for normal operation of the fieldbus control loop is derived. The analysis and experimental results show that the execution time and the margin time are dominant in a control period, whereas the communication time is secondary. It is also shown that the execution time of function blocks depend on the configuration of application software. The effects of the communication time, the computation time, and the jitter of control period on the control performance are evaluated.

In Chapter 4, a complete set of evaluation indices is proposed from the user point of view by analyzing the requirements of data communication and installation environment for fieldbuses. As a case study, experimental and simulation results for FOUNDATION Fieldbus are presented. A general procedure with a complete set of detailed indices for selecting a fieldbus system is also proposed.

Chapter 5 considers how to overcome bad effects caused by the delays of communication and computation time, and the jitter of control period. A modified PID (Proportional-Integral-Derivative) control algorithm and a predictive control algorithm are applied and the effectiveness of these algorithms are analyzed and verified by simulation study.

Chapter 6 proposes an object-oriented, hierarchical, and hybrid modeling approach for the FCS development. Based on this approach, a simulation platform for the FCS including fieldbus devices, fieldbus segments and the plant is developed using the Matlab environment.

[*] Doctoral Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT 361040, July 5, 2004.

In this simulation, both computation times of application software and communication delays are considered at the same time. Also both the control and the communication performances are evaluated through simulation runs. Chapter 7 illustrates the development of two fieldbus systems by describing both the hardware and software of the system. The contributions of this thesis are summarized with directions for future work in Chapter 8.

Keywords: Fieldbus, Networked control system, Timing analysis, Control algorithm, Modeling and Simulation, Evaluation

ACKNOWLEDGEMENTS

First of all, the greatest gratitude goes to my advisor, Professor Hirokazu Nishitani. During the work he has continuously provided me with his guidance, inspiration and enthusiasm. I cannot thank him enough for personal understanding, encouragement and support he has given to me to complete this thesis. I personally feel so proud and fortunate to have had him as my research and academic advisor. I would also like to thank Professor Kenji Sugimoto and Associate Professor Shoji Kasahara for their helpful comments.

I would also like to thank Professor De-xian Huang in the Tsinghua University in China, Dr. S.H. Yang in the Loughborough University in United Kingdom, and Dr. Ling-qi Li in the Kyoto University in Japan for their constructive comments for my research work and the insightful discussions we had on various topics.

Thanks go to all the fellow researchers and students in the System & Control Lab of Nara Institute of Science and Technology, and in the Fieldbus Lab of the Beijing University of Chemical Technology. I have completely enjoyed and enormously benefited from the interaction and collaboration with them, especially Xi-wei Liu, Xiu-xia Fu, Kui Zhao, and Gang Wu. Special thanks go to Choy Wai Cheong for his careful checking of my thesis in English.

Finally, but most importantly, this thesis is dedicated to my family for their continued love and support that enabled me to complete this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

A fieldbus is a real-time local area network dedicated to connect all kinds of field devices, such as sensors, controllers, actuators, display terminals, and workstations. A Fieldbus Control System (FCS) is a distributed system composed of field devices and control and monitoring equipments and it is integrated into the physical environment of a plant or factory. Fieldbus devices work together to provide I/O (Input/Output) and control for automated processes and operations. Fieldbus systems may operate in manufacturing and process control environments that include intrinsic safety requirements. In these environments, devices operate with limited memory and processing power and with networks that have low bandwidth (Fieldbus Foundation, 1996). Since the first research in the early 1980's, fieldbuses have grown to maturity with numerous solutions available. At present, there are more than 50 different names (Thomesse, 1999) of available fieldbuses such as FOUNDATION Fieldbus, Profibus, InterBus, WorldFIP, LonWorks, CAN, DeviceNet, SwiftNet, P-Net and so on. The advantages of using a fieldbus are obvious, namely a reduction in the amount and cost of cabling and in the installation, operation and maintenance costs of industrial plants, better interchangeability of measurement and control signals, truly distributed control, and prompt information on field device fault diagnosis. In the field of automation, there has been a trend away from centralized to distributed for control and from analog to digital for instrumentation. The fieldbus is the only technology doing this. Therefore, fieldbuses based products have been increasingly used in various areas. In automobiles, a fieldbus has to manage the communication of about one hundred nodes. In building automation systems, more than 10,000 nodes are connected by the fieldbus. In China, a large scale of petrochemical enterprise is being built, where about 25,000 fieldbus devices including transmitters and actuators are used. Fieldbuses are bringing many new things including benefits and challenges to measurement and control.

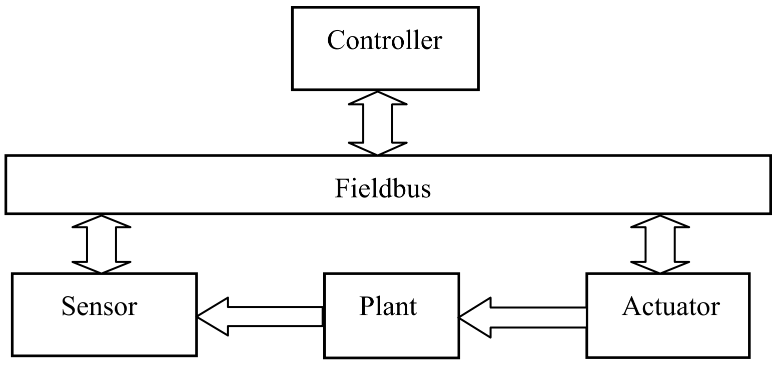


Figure 1.1: Schematic diagram of a fieldbus control system

As shown in Figure 1.1, what differentiates a fieldbus control system from a conventional control system is the presence of a communication fieldbus. The insertion of the fieldbus in the feedback control loop brings many features to FCS and makes it different from conventional control systems as follows (Cavalieri, 1996; Thiele, et al., 1999):

- Different system features

  At the system level, a fieldbus control system can provide a mechanism that allows distributed computation, communication and control to be integrated into a seamless process. It is a representation of ubiquitous computing at the lower field devices level. It is a hybrid system because the signals of analog and digital, continuous and discrete, data value and event coexist in the system. From the communication and software points of view, it is also a network and real-time multitask system.

- Different devices

  We consider the sensor or transmitter, actuator, and controller as discrete devices. The devices in a fieldbus control system all have two important capabilities, namely the computing capability and the communicating capability, which give the devices more functions and make them more intelligent.

- Different information

  In the classical control system, the information content that flows unidirectionally among the sensor or transmitter, actuator, controller, and plant is only control dependent, whereas the information content that flows bi-directionally in a fieldbus control system is richer, such as diagnosis and management information.

- Different services

  Based on the capabilities of distributed computation and communication, more services can be provided in a fieldbus control system, such as functionally and spatially distributed and switched measurement and control algorithms, diagnostic and management services, and beyond the general measurement and control services.

All of these differences make it difficult to study the issues of a fieldbus measurement and control system. Like the history of communication, the researches on the fieldbus control systems have lagged behind the rapid increment of their applications. In the past decade, most researches only focus on the fieldbus itself and many achievements have been made. Although a lot of the problems on fieldbus itself have been solved, none of the existing solutions may claim universality, and many interdiscipline issues, such as the fieldbus evaluation, cross influence between the fieldbus and control applications, and analysis and design methodology in the environment of the fieldbus, deserve research.

Although IEC (International Electrotechnical Commission) approved the IEC 61158 proposal for an international fieldbus standard at the end of 1999, the standard actually comprises eight different fieldbus protocols. Apart from IEC standard, there are many other standards of fieldbus protocols, which have also achieved success on market. Consequently, the fieldbus confusion to users still exists. Facing so many types of fieldbuses, the developers and users of fieldbus based products need to evaluate the fieldbuses. On the other hand, the QoSs (Quality of Services) of the fieldbuses are essential to guarantee the performance of the applications. Therefore, methods to evaluate the fieldbuses need to be researched.

For a fieldbus control system, a time-delay is inevitably introduced into the closed control loop from the control point of view. In order to analyze and reduce the fieldbus delay, the timing characteristic of fieldbus communication and the cross influence between the fieldbus and a control application must be investigated. In addition, new control algorithms, which are suitable for the fieldbus devices and system, should be investigated to overcome the specific problems in the fieldbus control system, such as the time-varying delays caused by communication and computation.

The methods to model and simulate the fieldbus devices and systems need to be researched in order to analyze, evaluate, design and apply the fieldbus devices and systems better. General principles and techniques for designing, implementing, and applying the fieldbus devices and systems are also important issues for the developers and users of fieldbus based products.

1.2 Related work

We briefly summarize previous work related with the following issues of fieldbus, namely the timing analysis, evaluation, control algorithms, modeling and simulation, development and applications.

1.2.1 Timing analysis of fieldbus control system

Several papers have analyzed the temporal characteristics of various fieldbuses. By Lian (2001), the timing components during data transfer are analyzed according to the communication protocol model of a network. By Tovar et al. (2002), a P-NET fieldbus timing analysis model is proposed. For the civil aircraft avionic databus standard ARINC 629, the timing analysis has been done by Audsley and Grigg (1997). The message response time of the CAN (Controller Area Network) is calculated by Tindell et al. (1995). To verify the end-to-end response times for distributed hard-real time systems, the scheduling analysis is done by Henderson et al. (2001) by considering both task processing and message communication. Although much work has been done for the timing analysis of some specific fieldbuses, the results focus on only the communication of the fieldbus, and there is a lack of the timing analysis from the fieldbus system perspective, which will involve some factors related with applications, such as the computation times and the configurations of applications, except the communication times.

1.2.2 Fieldbus evaluation

Issues surrounding the evaluation of fieldbus have been studied in the literature. The message scheduling and delay evaluation are made by Blum and Juanole (1999) by using stochastic time Petri nets. A QoS architecture for a client-server model is presented by Angel and Chavez (2001). However, the important QoSs which should be included in the fieldbuses is not discussed. In the study of Sempere et al. (1999), three performance indices, namely utilization, throughput, and mean response time are used to evaluate two interconnection strategies, and the Petri nets based language SAN is used to model a fieldbus gateway. The formulae to evaluate the upper bound of the end-to-end communication delay in P-NET messages are provided by Tovar et al. (1999). To evaluate the reliability of the fieldbus, by considering the transmission errors in the applications because of the disturbances, the concept of worst-case deadline failure probability is introduced in the study of Navet et al. (2000). The guidelines and general procedure for selecting a fieldbus are presented by Farsi and Barbosa (2000). In addition, many issues for fieldbus characteristics, such as network layout, maximum distance, simplicity and flexibility in network configuration, product availability and openness, interoperability, speed, medium access capability, and configuration possibility, are mentioned. Two general categories of the fieldbus system selection criteria, including technical data and strategic criteria, are classified. However, the concrete and complete criteria and their determining methods are not given. The model-based evaluation and prototype-based evaluation are used to validate the in-vehicle real-time applications (Navet and Song, 2001). Despite the work mentioned above, a complete set of evaluation criteria and methodology should be studied from both the communication capability of fieldbus and the application support of system.

1.2.3 Control algorithms

Many researchers (Sawamura et al., 2002; Walsh et al., 2002; Beldiman and Walsh, 2000; Park and Huh, 2000) have proposed various design and delay compensation methods for networked control systems. However, although the FCS is a kind of networked control systems, these results are not appropriate to the FCS because of its specific features that include the low computing capability of the fieldbus devices and a different delay modeling method. Therefore, specific control algorithms used in the fieldbus devices are required. There is a lack of such research work to our best knowledge.

1.2.4 Modeling and simulation

Hong and Lee (1996) modeled the logical behavior of a simplified data link layer protocol of IEC/ISA (Instrumentation Society of America) fieldbus by using Petri net. The network-induced delay in the data link layer of the IEC/ISA fieldbus is evaluated. For the IEC/ISA fieldbus, the discrete-event simulation model of the data link layer is developed by Hong and Ko (1998). The Petri nets based language SAN is used by Sempere et al. (1999) to model a fieldbus gateway. To specify and simulate the PROFIBUS-DP and FIP network, the Estelle is extended with time and used by Hohwiller and Wendling (2000) as well as Barretto, et al. (1990) as a general modeling and simulation language. The fieldbus applications are simulated with DRUGH simulator (Raja, et al., 1995). In the study of Sevillano et al. (1998), a model for the analysis of the CAN protocol is presented. The model can help to obtain some useful results for the design of CAN-based systems. A simulation research of the CAN application in a tractor is carried out by Hofstee and Goense (1997, 1999). Much work of fieldbus modeling and simulation was done for the fieldbus validation and evaluation by considering only logical behavior but not timing behavior. Also, there is a lack of investigation for the applications related to modeling and simulation.

1.2.5 Development and applications of fieldbus devices and control system

Today, we are at a milestone in the continuing evolution of field instrumentation technology. The process industry is moving from the analog transmission of data to the use of digital communication technology (Lindner, 1990). Advantages of fieldbus are introduced by Zech

(1995). The fieldbus is considered as one of low cost automation techniques (Boettcher and Traenkler, 1990). The fieldbus has been the foundation for field control systems (Andrew, 1994). In recent years, in the field of real-time instrumentation and process control, there has been a trend away from centralized control system. Distributed control by using fieldbus technology is only the first step to application of intelligence in the field devices (Mahalik and Moore, 1997). Therefore, a lot of researches have been done for the development and applications of fieldbus devices and control system. With the development of a CAN fieldbus system, the issues of the analysis and design tools for a fieldbus system are discussed by Rodd et al. (1998). The sensor node and the network system based on LonWorks technology are introduced by Pang et al. (1999). A fieldbus solution that meets the user requirements for interoperability and functionality is described by Glanzer and Cianfrani (1996). In the study of Blevins et al. (1996) and Li et al (1999), some of the technical challenges of integrating fieldbus into existing DCSs (Distributed Control Systems) are discussed, and three approaches to fieldbus integration are examined from the standpoint of impact on the existing DCS and effective use of fieldbus technology. Three applications of the fieldbus approach in the implementation of a local control network are described by Lenhart (1994). The fieldbus-based design and development methodology in order to build a real-time industrial distributed control system is highlighted, and a case study with the LonWorks technology is presented by Mahalik and Moore (1997). An intelligent actuator is developed using LONWORK neuron chips by Xie et al. (1998). An approach to designing the intelligent sensor based on fieldbus is presented by Tian et al. (2000). An application of LonWorks control networks for production line automation is studied by Mahalik and Lee (2002). All in all, the development and applications of fieldbus devices and control system have been a continuing research topic because of different application specifics.

1.3 Research contributions

- Timing analysis for the FCS

The timing characteristics of the FCS are analyzed in details. The control period in the FCS for a control loop is formulated and analyzed. The stability condition for normal operation of the fieldbus control loop is obtained. The effects of communication times, computation times, and the jitter of control period on the performance of control are researched by evaluating the performance criteria IAE (Integral of Absolute Error) and ITAE (Integral of Time multiplied by Absolute Error), the overshoot, and the settling time based on the simulation results. The increases of communication times, computation times, and the jitter of control period will worsen the control performance. The different delay components are commutative for a given control period from the control point of view. There are upper bounds for the delays of communication time, computation time, and the jitter of control period, respectively, for a given performance criteria. In the extreme case, the fieldbus control system will become unstable if the delays of these times and the jitter exceed the upper bound.

- Evaluation of fieldbuses

The requirements of data communication and installation environment for fieldbuses are analyzed from user standpoint. After that, a complete set of evaluation indices is proposed. A general procedure and a complete set of detailed indices for selecting a fieldbus system are also proposed. As a case study, the experimental and simulation results for FOUNDATION Fieldbus were presented. Both experimental and simulation conclusions conform to those obtained by analyzing the mechanism of FOUNDATION Fieldbus protocol.

- Novel control algorithms for fieldbus devices

To overcome the bad effects caused by the delays of communication time, computation time, and the jitter of control period, and to guarantee the control performance of FCS, two approaches can be considered. One is to reduce the dominant delays, i.e. computation times in the control period. Accordingly, a parallel computation method is proposed. The other is to design specific control algorithms for FCS. Therefore, a modified PID (Proportional-Integral-Derivative) control algorithm and a model predictive control algorithm are proposed to overcome the effects of the delays and the jitter on the control performance. The simulation results show that the modified PID control algorithm works well for the common process having the first-order model with dead time. This implies that the stability of the fieldbus control loop can be assured under the unpredictable and varying time delays caused by the fieldbus. The simulation results also tell us the fact that the delay of the plant and the delay of the communication and computation are not commutative. The predictive control algorithm can give better control performance if the plant model is obtainable and the jitter range of the control period is small.

- Modeling and Simulation methods of the FCS

An object-oriented, hierarchical, and hybrid modeling approach is proposed for the FCS. Based on this approach and the detailed analysis of the FCS principles, the simulation models for the FCS including the fieldbus devices, fieldbus segment and the plant are developed using the Matlab environment. Both the application computation and the communication are considered in the models at the same time. It provides us a good platform to study the FCS.

- Development and applications of fieldbus sensor and control systems

Finally, two fieldbus systems are successfully developed and put into practical applications. The development procedure is illustrated by describing the hardware and the software of the systems.

1.4 Outline of the thesis

This thesis is divided into eight chapters. Chapter 1 introduces the fieldbus research issues and provides a brief overview of related work. Chapter 2 introduces the fieldbus technology. We provide an overview of main fieldbuses. The fieldbus protocol models and the medium access control mechanisms are introduced and discussed. Four methods to classify the fieldbuses are proposed. In Chapter 3, we will analyze the message transfer mechanisms. A methodology to do timing analysis of the fieldbus is proposed. The influence of time delay on control performance is analyzed. In Chapter 4, we consider the performance evaluation of fieldbuses. The control algorithms used in the fieldbus devices are addressed in Chapter 5. In Chapter 6, the methods to model and simulate the fieldbus control system are presented. Chapter 7 presents development and applications of the fieldbus sensor and control systems. Finally, in Chapter 8, we summarize the contributions of this thesis. In addition, some directions for future work are given.

Chapter 2

Fieldbus Technology

2.1 Introduction

Until now, there is no standard definition for a ‘fieldbus’. Nevertheless, the following definition is commonly acceptable. A fieldbus is a real-time data communication network dedicated to connect all kinds of “field” equipment, such as sensors, controllers, actuators, display terminals in an automation system. As show in Figure 2.1, the fieldbus environment is the base level group of digital networks in the hierarchy of plant networks. The communication networks with fieldbuses now sustain the new industrial infrastructure.

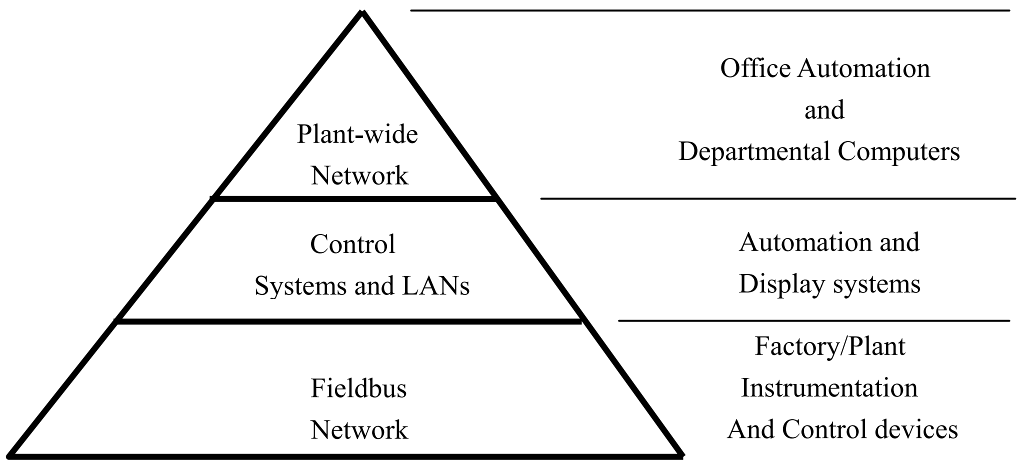

Plant-wide Network
Control Systems and LANs
Fieldbus Network
Office Automation and Departmental Computers
Automation and Display systems
Factory/Plant Instrumentation And Control devices

Figure 2.1: Hierarchy of plant networks (Fieldbus Foundation, 1998)

A large number of different fieldbus protocols have emerged since the eraly 1980s. Thomesse (1998) gives a detailed review for the history of the fieldbuses. The fieldbus, as a dedicated network, has some differences listed in Table 2.1 from other general networks. These differences can answer why Ethernet is not used as the fieldbus. Ethernet cannot satisfy the requirements in real-time, medium length, bus powered, and operating environment. A perspective on Ethernet – TCP/IP as a fieldbus is discussed by Decotignie (2001).

Table 2.1: Differences between fieldbus and other general networks

Differences	Fieldbus	Other network (LAN)
Typical application	Measurement and control	office, Management
Typical user data	Small, variables, messages <100 bytes	Large, files>1kbyte
Typical reaction time	Rapid, <1s	Slow, >1s
Typical type of node	controllers, sensors, actuators	computers, printers
Real-time requirement	Yes	No
Operating environment	hazardous	safe
Bus powered	Possibly required	Not required
Data transmission rate	Low, <5Mbps	High, >10Mbps
Medium length	Long, ≈1000 meters	Short, ≈100 meters
Cost	Low	High

In this chapter, we first introduce different protocol models of fieldbuses, and each layer of the protocol model is described. Then we introduce the main fieldbuses, which are commonly used in practice. A detailed discussion of the typical medium access control (MAC) methods for fieldbuses is provided. According to the classifying methods proposed, the main fieldbuses are classified.

2.2 Fieldbus protocol model

Understanding the architecture model of fieldbus communication protocol (Patzke, 1998) is the fundamental to research a fieldbus. Despite the types of fieldbuses, almost all of the fieldbuses have similar communication architectures because they all take the 7-layer ISO/OSI (Open Systems Interconnection) reference model as a reference, which was devised by the International Standards Organization (ISO) to support the development and implementation of open communication protocols. In the layered model, each layer is used by the layer immediately above it, and using the services of the layer immediately below. All fieldbus protocols only have a reduced layers model, which may have two, three or four layers depending on the fieldbus type except that the LonWorks fieldbus claims to have all seven layers of the reference model. The content and technique contained in each layer of the model depend on a specific fieldbus, and are various even if two fieldbuses have the same number of layers.

2.2.1 Basic fieldbus protocol

If a fieldbus protocol only has the lower two layers of the 7-layer ISO/OSI reference model, i.e. physical layer and data link layer, we call it the “basic fieldbus protocol” because any fieldbus protocol needs these two layers. The model of the basic fieldbus protocol is shown in Figure 2.2. This protocol is considered as a network protocol only for simplifying wiring between devices or for providing the base of networked applications. Higher layer is often needed to support more data transmission services in order to meet the requirements of complex applications. The CAN fieldbus discussed in the following section is an example of the basic fieldbus protocol.

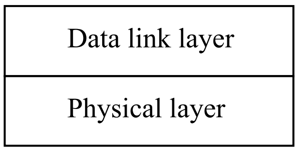


Figure 2.2: Model of basic fieldbus protocol

The functions of the physical and data link layers are briefly described in the following:

- Physical layer

  The fieldbus physical layer receives messages from the upper layer of data link and converts the messages into physical signals on the fieldbus transmission medium and vice-versa. It is composed of the physical medium and the signaling protocol used to transmit data. Physical layer protocols provide the data link layer protocol with ability to send and receive data. Concretely, this layer performs data encoding and decoding operations, generates control signals for proper operation of the Data Link Layer, and defines the medium and physical connections to it.
- Data link layer

  This layer is to deal with data transmission error detection, data framing and deframing. It also deals with link management and medium access control. From the timing point of view, the medium access control is very important because the physical medium is shared by many competitive communication tasks. Medium access control will be discussed in Section 2.4.

2.2.2 Common fieldbus protocol

Most of the fieldbuses protocols have a common fieldbus protocol that has three layers of the 7-layer ISO/OSI reference model: physical layer, data link layer, and application layer. The model of a common fieldbus protocol is shown in Figure 2.3. An application layer is defined based on the basic fieldbus protocol. In general, this layer provides all services required for transmitting and receiving messages and support of the application process, which can satisfy requirements of most user applications. The profiles over the application layer sometimes are necessary to support the compatibility and the interoperability for distributed and complex user applications.

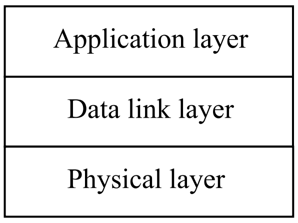


Figure 2.3: Model of common fieldbus protocol

2.2.3 Complete fieldbus protocol

The complex type of fieldbus has a model with four layers and two management blocks of network and system as shown in Figure 2.4. We call it complete fieldbus protocol because almost all of the requirements for distributed and complex user applications has been considered and supported in the model. The user layer and the two blocks added based on the common fieldbus protocol are briefly described in the following. The Foundation Fieldbus introduced later has such complete fieldbus protocol.

- User layer
  User layer mainly includes the computation tasks that implement user application.
- System management
  System management is used to coordinate operation of various devices in a distributed fieldbus system.
- Network management
  Network management supports communication configuration management, performance management of device communications and fault management.

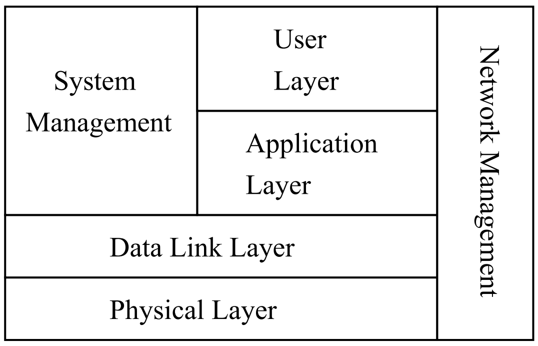


Figure 2.4: Model of complete fieldbus protocol

2.3 Main fieldbuses

There are many fieldbuses in use. Although international standardization effort has been done, a single standard fieldbus protocol is not achieved. Each fieldbus protocol has its characteristic and a particular application area. Several important fieldbus protocols are introduced in the following. We only introduce the low speed fieldbuses (<500 kbit/s), and not the high speed fieldbuses such as the ControlNet and FOUNDATION Fieldbus HSE, which are not used in the simple field device level.

2.3.1 FOUNDATION Fieldbus

The FOUNDATION Fieldbus is developed by the fieldbus Foundation, which is an independent organization to complete development of a single, open, international, and interoperable fieldbus. The FOUNDATION Fieldbus technology is based on the work of the International Electrotechnical Commission (IEC) and ISA (Instrument Society of America). The FOUNDATION Fieldbus protocol has a complete fieldbus protocol shown in Figure 2.4, and is a subset of the international standard IEC 61158. This technology consists of the physical layer, the communication “stack”, and the user application (Fieldbus Foundation, 1996, 1998).

In the physical layer, signals are encoded using the well-known Manchester Biphase-L technique. The signal is called “synchronous serial” because the clock information is embedded in the serial data stream. Its transmission rate is 31.25 kbit/s. This low speed may be a disadvantage of FOUNDATION Fieldbus. The FOUNDATION Fieldbus devices can be powered directly from the fieldbus. This supports intrinsically safe fieldbus with bus-powered devices, which are required in the hazardous area. This is one of the FOUNDATION Fieldbus features. The number of devices on the FOUNDATION Fieldbus is up to 32, the maximum length of the fieldbus is 1900 m. It supports the topology of bus / tree.

The communication stack of the FOUNDATION Fieldbus includes data link layer and application layer. The application layer is further composed of two sub-layers: fieldbus message sub-layer (FMS) and fieldbus access sub-layer (FAS). The FMS provides communication services to fieldbus application processes, system management application processes, and network management application processes. The FAS provides different types of communication channels, called Virtual Communication Relationships (VCRs).

The data link layer (DLL) provides system management and FAS access to the fieldbus medium for transferring messages between applications. The data link layer is divided into two levels of operation, one to provide access to the bus, and the other to control data transfer between data link users. The FOUNDATION Fieldbus provides three DLL mechanisms for transferring data, one is connectionless and the other two are connection-oriented data transfer. Connectionless data transfers support report distribution VCRs. One type of connection-oriented transfer supports publisher/subscriber VCRs, and the other supports client/server VCRs. The medium access control (MAC) method is defined in the data link layer. From the time point of view, this layer is very important because the physical medium is shared by many competitive communication tasks.

In the FOUNDATION Fieldbus system, medium access is provided on a scheduled basis and on an unscheduled basis by a special data link layer entity known as Link Active Scheduler (LAS). Two types of devices are defined in the DLL specification, namely the basic device and link master. The link master devices are capable of becoming the LAS, whereas basic devices do not have the capability to become the LAS. The bandwidth of the fieldbus is divided into two parts: a synchronous window for periodic data transfer and an asynchronous window for asynchronous data transfer as shown in Figure 2.5. Three types of VCRs are defined. Publisher/Subscriber VCRs are used for periodic data transfer. Data producers (publishers) use this type to post data into a network buffer that may be read, on demand, by users (subscribers) of the data. Transfers of this type can be scheduled on a precisely periodic basis. Using system management services, updates to the publisher buffer can be synchronized with the buffer transfers to reduce the delays between data production and data transfer. Report Distribution and Client/Server VCRs are used for asynchronous data transfer. For these VCRs, a token passing mechanism is used to share the unscheduled time between devices. The FOUNDATION Fieldbus is a well-designed real-time communication network.

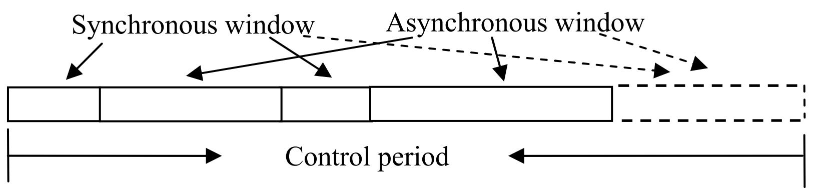


Figure 2.5: Communication windows on the FOUNDATION Fieldbus

The FOUNDATION Fieldbus defined a standard user application based “Blocks”. Blocks are representations of different types of application functions, and include resource block, function block, and transducer block. The resource block describes characteristics of the fieldbus device, such as the device name, manufacturer, and serial number. The function blocks represent the basic automation functions required by a user control application. Therefore, the function blocks provide the control system behavior. The input and output parameters of the function blocks can be linked over the fieldbus. The execution of each function block is precisely scheduled. The function blocks can be built into fieldbus devices as needed to achieve the desired device functionality, and to enable the distributed controls at the device level. For example, a simple temperature transmitter may contain an AI (Analog Input) function block. A control valve might contain a PID (Proportional/Integral/Derivative) function block as well as the expected AO (Analog Output) block. Thus a complete control loop can be built using only a simple transmitter and a control valve. The Fieldbus Foundation defined sets of standard function blocks. The transducer blocks decouple the function blocks from the local input/output functions required to read sensors and command output hardware. The following additional objects are defined in the user application:

<u>Link Objects</u> define the links between the function block inputs and outputs internal to the device and across the fieldbus network.

<u>Trend Objects</u> allow local trending of the function block parameters for access by hosts or other devices.

<u>Alert Objects</u> allow reporting of alarms and events on the fieldbus.

<u>View Objects</u> are predefined groupings of block parameters sets that can be used by the human/machine interface.

Furthermore, the FOUNDATION Fieldbus contains network management and system management. The network management supports communication configuration management, performance management of device communications, and fault management. The system management is used to coordinate the operations such as the function block scheduling of the various devices in a distributed fieldbus system. From the protocol and application points of view, the FOUNDATION Fieldbus is the most complete fieldbus. However this makes the FOUNDATION Fieldbus system more complex. It is costly and difficult to develop.

2.3.2 CAN

The Controller Area Network (CAN) (Robert Bosch GmbH, 1991) was introduced by Robert Bosch GmbH in cooperation with the Intel Company in the late 1980s as a serial network for automobile applications (Egan-krieger at al., 1994; Cena and Valenzano, 2000a), which became the international standard 11898 for the interchange of digital information of road vehicles in 1993 (International Standard Organization, ISO 11898, 1993). The success of the CAN protocol in the automotive applications, together with its low cost and high performance have led to a rapidly increasing interest also in the industrial automation and process control areas, especially in the areas of medicine, industrial and process automation. The CAN protocol implements most of the features of the basic fieldbus protocol shown in Figure 2.2.

The addressing of CAN is not node but message oriented. The CAN protocol uses a message-based data format in which information is transferred from one node to another. Unlike address-based systems, every node in this system listens to every message on the bus to determine if it needs to respond. Every message has an identifier field consisting of either 11 or 29 bits. The message can also contain data, but it's not required. The node uses the identifier to determine if the incoming message should be accepted and acted on or discarded. With the message-based format, you can add nodes to the bus without reprogramming the other nodes to recognize the addition. The new node will start receiving messages from the network immediately.

The CAN protocol defines four types of messages, or frames. The first and most common frame is a data frame, which is used when a node transmits information to any or all other nodes in the system. The second most common frame, a remote frame, is used when one node requests data from another node. The other two frame types are used to handle errors. A node generates an error frame when it detects one of the many protocol errors defined by CAN. And the protocol calls for an overload frame when it requires more time to process messages already received.

CAN uses a nondestructive bitwise arbitration, which means that messages remain intact after arbitration is completed even if collisions are detected. All the arbitration take place without corruption or delay of the message that wins the arbitration. It belongs to the method of the CSMA/CA. The arbitration principle is used for which the values high and low are realized in different technical ways. The results are recessive (value 1) and dominant (value 0) bits. That means, when there are several nodes, trying to access the bus to send their data, this node has to retreat. The first recessive bit is overwritten by a dominant bit of another node. The result is the nondestructive access. The message with the lowest identifier ('000…') has the highest priority and a guaranteed delay time.

The CAN is just a low level communication protocol, and doesn't include the application support. To develop an application system based on the CAN, high layers are needed. The DeviceNet and the CANopen that will be introduced below do such things.

- Advantages

The CAN is a deterministic protocol optimized for short messages. The replacement of a node in the CAN-based system, for example by a newer one, makes it unnecessary to set an address on the spot, which is a very important in the area of automobile manufacturing. The CAN fieldbus also has a good safety. The fault tolerance, the fault recognition and fault removal are multistage organized. The protocol implementation of hardware and low cost are the important aspects of the CAN.

- disadvantages

One of the disadvantages of the CAN is the inability to bound accurately the worst-case response time of a given message transfer. The messages with lower priorities can have large delays if the bus traffic is high. To overcome this disadvantage, much work has been done to improve the response time (Livani et al., 1999), (Cena and Valenzano, 2000b), (Navet and Song,

1999) and (Leen and Heffernan, 2002). Another disadvantage of the CAN is that the maximum extension allowed when bit rates above 500 kbit/s are considered is noticeably lower than other kinds of fieldbuses, such as Profibus or FIP. This drawback depends on particular medium access method of CAN, which is based on an arbitration phase carried out by means of a wired-AND connection scheme, and hence it cannot be overcome in any way (Cena, 2001). Therefore, CAN is not suitable for the applications that require high bit rate above 500 kbit/s.

2.3.3 LonWorks

LonWorks (Local Operating Networks) fieldbus was created by the company of Echelon in the beginning of 1990s. LonWorks fieldbus system, in terms of its scope of definition and realization support together, is far different from other fieldbus systems, although there are similarities in some details. Comparing LonWorks with other systems with respect to the communication layers of the ISO/OSI model, it can be stated that (Dietmar, et al., 2001):

- LonWorks supports directly "differential Manchester coding" as this code is available at the network port of the Neuron chip. This is not a new code and is also used by the IEC fieldbus standard. The advantage of this coding is that no DC (Direct Current) components are incorporated. This allows transformer isolation and coupling and powering of devices over the bus. The LonWorks physical layer supports a lot of transmission rates and media. Currently, the transmission rate of 78 kbit/s, together with transformer coupling, allowing really free topologies as far as the total bus length is limited to 500 m seems to be becoming the most used LonWorks physical layer.
- LonWorks makes use of a CSMA (Carrier Sense Multiple Access) medium access control which, in principle, allows telegram collisions but limits their probability. This causes the disadvantage of the LonWorks that response times of nodes cannot be bounded.
- The LonWorks addressing scheme provides for the subdivision of a network into logical subnetworks. Therefore the network can be segmented by means of routers. Many fieldbuses have defined only the layers 1, 2, and 7. Therefore they are unable to isolate communication loads.
- LonWorks has a transport layer, which provides an acknowledged service, but no segmenting of long data blocks across several telegrams. Acknowledgments at the transport layer are mandatory because of the implemented network layer (in complex networks the acknowledgment has to be performed by the target node rather than by routers between the source and the destination of telegrams), but LonWorks does not use an acknowledged layer 2 service.
- A special service, not provided by other fieldbuses, is the authentication service in the session layer. This service prevents unwanted penetration into communication relationships of nodes inside the network. This protection allows the application of LonWorks in security systems.
- At the application layer, LonWorks provides a very comfortable way of communication by doing this implicitly with network variables. The services themselves are comparable to other fieldbuses; there is an acknowledged service in the application layer with the "poll" instruction for reading access on the data of other nodes. For writing access on other nodes this acknowledgment has to be programmed explicitly in the application.
- A very important feature of the LonWorks fieldbus system is the availability of a

comprehensive set of network management telegrams which themselves are the basis of many tools for configuration, installation, commissioning and maintenance.

The situation concerning the support of concrete realization in LonWorks can be characterized as follows:

- There are essentially two types of chips (the neuron chip) available from two manufacturers. The communication protocol is not only specified but also implemented. This is also the case for other fieldbus systems e.g. INTERBUS_S or CAN.
- These chips run the application programs and are linked to the protocol by the development tool.
- Interoperability plays an important role for all fieldbuses, because no single manufacturer will be able to cover the broad range of fieldbus devices and to deliver all components employed in fieldbus based systems. Numerous obstacles to interoperability are removed by the implementation of the communication protocol on a chip and the development of the application program with a unique tool. Other fieldbus systems such as INTERBUS_S and EIB have a similar background. However it was soon realized that, despite this situation, interoperability could not be guaranteed. Because of this, the so-called "Standard Network Variable Types" (SNVT's) was introduced to achieve at least compatibility of the data types. In addition, the Lonmark association was founded, which is establishing guidelines for application objects and function profiles for those objects, which means a further step in the direction of interoperability. Other fieldbus organizations have acquired profiles for complete classes of applications and recognized the necessity to introduce interoperability tests and certification procedures for fieldbus based products.
- The conceptual development of LonWorks took a path rather different from other fieldbus systems. While most fieldbuses have been drawn up for a specific class of applications and were later utilized for other purposes (CAN, INTERBUS_S, PROFIBUS), LonWorks was conceived for a spectrum of applications as broad as possible. Because of this, LonWorks has been used in various areas, especially in building automation.

2.3.4 Profibus

The PROFIBUS (Process Fieldbus) (European Committee for Electrotechnical Standardization, EN50170/2, 1996) is a German national standard promoted by the German standard organization (DIN) in 1991, one of fieldbus standards EN50170 of Europe in 1996. It is a set of protocols including: FMS (Fieldbus Messaging Specification), DP (Decentralized Peripherals) and PA (Process Automation). PROFIBUS/FMS is a powerful and complex multi-master system. It is often used as a backbone in a system. The DP is a FMS, reduced to one master system, but optimized for very short delay times. For this master-slave system, Siemens has realized an ASIC allowing for low priced nodes. The type PA is defined for process automation and has the IEC standard 61158-2 (intrinsically safe) in physical layer. Therefore, it is qualified for hazardous areas.

The PROFIBUS can connect the number of nodes up to 32 on one segment of the fieldbus, the fieldbus length without a repeater: 200 m to 1200 m, depending on the transmission rate, and the transmission rates 9.6 kbit/s to 500 kbit/s. The maximum transmission rates of the PROFIBUS/DP 12 Mbit/s. PROFIBUS offers several transmission media. At present, the most usable system is based on the RS-485 standard. In this case the cable has two shielded wires and is terminated on both sides of the line. The network topology is based on a common bus.

A PROFIBUS network can include two kinds of stations: masters and slaves. The PROFIBUS medium access control includes the services set out in the IEEE 802.4 MAC standard. However, an additional service has been introduced called cyclic send and request data with reply to deal with cyclical polling of the slave nodes. Therefore, the medium access control method adopted by the PROFIBUS is the token-bus mechanism plus master-slave polling. Masters take part in the circulation of the token while slaves play a passive role and only be polled by the master stations. This hierarchy is typical in field control systems, where sensors and actuators are slave stations while PLCs and NCs are master nodes. The token principle guarantees a deterministic behavior of the system, a decisive advantage of PROFIBUS.

The PROFIBUS application layer has been designed to meet the needs found in real-time factory environments and specifically to support communications among application processes. Some user applications are not contained in the protocol, but defined in separate profiles for common devices.

2.3.5 WorldFIP

The FIP (Field Instrumentation Protocol) is a French national standard fieldbus developed in 1990, and the WorldFIP as one of Europe standard fieldbuses in 1996 (European Committee for Electrotechnical Standardization, EN50170/3, 1996). It adopts a classical reduced protocol profile consisting of three communication layers (physical layer, data link layer, and application layer) and a network management block. In the FIP network, each data exchange is supervised by a special station called the bus arbiter, which is responsible for assigning the right to transmit data on the bus to data producers. When data information is broadcast on the bus, it can be read by each listening station. Thus the number of information exchanges depends only on the quantity of data produced and not on the number of stations that consume the data.

2.3.6 Interbus

The INTERBUS developed by Phoenix Contact Inc., stands in contrast to all other fieldbus systems. It is based on a topological ring structure, functioning by the Shift Register principle. It is a master-slave system. All slaves together are seen by the master as one big peripheral unit (register). The size of this peripheral unit is determined by the number of data points in the slaves. The master shifts the whole frame for all peripheral units through the whole ring, to receive the frame again after the shift operation, delayed (time shift) by the number of data points of the slaves. Before clocking the sum frame through the unit, each slave moves its data into its transmission buffer. Then the master pushes a Loop-Back Word (LBW) and control bits into the ring, until it receives the LBW again. This way it gets all identification information for all nodes, and the first cycle is finished. The second shift cycle starts again with the LBW, followed by the output data for the slaves (Dietmar et al., 2001).

The register shift structure has the logical conclusions that the transferring frame has a minimum of overhead. The addressing of the nodes is easy because the addresses are fixed by the position of the nodes and the priority levels are not applicable. The deterministic behavior is reached by the shift operation. In fact each node has its own clock generator, but they synchronize by the first incoming bit of the frame.

Five wires in one shielded cable are recommended (INTERBUS is also based on the RS-485 standard), two in each direction and one for ground.

The master offers consequently also the function of coupling the network to hierarchical higher situated networks. It serves in such a way for example as a gateway, if the higher network is a PROFIBUS working as a backbone. The data rate is usually 500 kbit/s. With this rate the whole point-to-point connection can reach a total length of 400 m. When using special hardware the maximum speed can be increased to 2 Mbit/s.

A drawback of the Interbus system is that the ring topology is susceptible to failure.

2.3.7 P-Net

P-Net is also a part of the CENELEC standard EN 50170, which was voted in 1996. It is a master-slave system supporting multi masters up to 32 per bus. In opposite to PROFIBUS each master node can also serve as a slave node, thus a master can have a communication relation with another master easily.

P-Net is also based on the standard RS-485 but in a little bit modified way. In order to reduce the power of reflected waves, the shielded twisted pair cable is connected to a ring, which means in the sense of medium access it is still a line. The bus access procedure is based on the virtual token passing principle. The masters get the access cyclically, following the ascending number of their addresses. In detail, this medium access procedure is a counter principle especially defined for the P-NET protocol. Each master can send only one frame to one node, and the slave can also answer with only one frame. If all slaves are addressed (broadcast message), no slave is allowed to answer. The token cycle time depends on the frame length, but it is limited. Therefore P-NET is defined as a deterministic system, because the worst-case scenario can be calculated exactly.

The networking segmentation of P-NET is interesting. According to the OSI model in P-NET, layer 3 is implemented to be able to design independent segments, connected by so-called controllers. These controllers are in the common language routers. The advantage of these routers is their simplicity. To the designer they offer the possibility of static routing or dynamic online routing, according to the program.

The data rate of P-NET is fixed to 76.8 kbit/s, which is the characteristic for the whole system. On the one hand, P-NET offers interesting features and possibilities. On the other hand, the designer of P-NET is restricted to essential aspects, parameters etc., to get a simple and easy to understand system.

Contrasting P-NET to LonWorks, it is important to point out that – similar to PROFIBUS – the P-NET protocol is implemented in common controllers, which raises the problem of interoperability, but no special chips are necessary. The only advantage is that the protocol is easier to handle. P-NET is an easy deterministic system with clear restrictions (Dietmar et al., 2001).

2.3.8 DeviceNet

The CAN we discussed before only has two lower layers, which are the physical layer and the data link layer. The CAN data link layer offers services for transmitting and requesting data units no longer than 8 bytes. These services are sufficient for the original universe of applications in which CAN was to be used. However, in more complex distributed applications, higher layer services are needed. Therefore, an application layer is defined in the DeviceNet (Open DeviceNet Vendor Association，1994).

The advantages and disadvantages of the DeviceNet are summarized by Lian (2001).

2.3.9 CANopen

CANopen is also an extended protocol based on the CAN. The CAN in Automation (CiA) organization specified the CAN application layer over the CAN. Based on the CAN application layer, a set of profiles can be defined. Thus the CANopen reference model is established. The profiles are divided into two categories: CANopen Device Profiles and CANopen Communication Profiles. The objective of the supplement is to make the protocol more complete and support the compatibility and the interoperability required by practical applications. The real-time data communication requirement can be satisfied in the CANopen (CAN in AUTOMATION, 2000), (Farsi and Barbosa, 2000).

2.4 Fieldbus medium access control

Fieldbus medium access control is one of the aspects that affect the time performance of a fieldbus. In any fieldbus, a medium access control (MAC) mechanism handles when and how a node on a fieldbus gets to use the medium, and decides what to do when there is competition for it. The purpose of the MAC is to allocate the medium in an equitable manner to all nodes on the fieldbus. There are two kinds of mechanisms: random-access and deterministic-access. The random-access mechanism means that all nodes on the fieldbus can send their messages whenever they want to do, but must contend for access to the fieldbus in the collision case. In the deterministic access techniques, fieldbus access is controlled in some way all the time during operation. Most fieldbuses make use of the deterministic access techniques, which satisfy the requirements of real-time applications better.

This section discusses several medium access control mechanisms commonly used for the fieldbuses and what performance they have. The performance indices are the efficiency and the responsiveness used by Cena et al. (1995). The characteristics and performances of the medium access mechanisms used by several fieldbuses including the Interbus, PROFIBUS, and CAN are compared for three kinds of fieldbus traffic (Cena and Valenzano, 2000a).

2.4.1 CSMA

The basic principle of the CSMA (Carrier Sense Multiple Access) is as following (Andrew, 1997). When a node has data to send, it first listens to the medium to see if anyone else is transmitting at that moment. If the medium is busy, the node waits until it becomes idle. When the node detects an idle medium, it transmits a frame. If a collision occurs, the node waits a random amount of time and starts all over again. The CSMA mechanisms used for fieldbuses are some variations based on the basic CSMA, and have been improved using various techniques. In CAN fieldbus, for example, each frame is assigned a priority identifier. When collision takes place, the contention is solved by means of a network-wide arbitration phase which is based on the identifiers values that the frames involved have. In addition, CAN uses a nondestructive bitwise arbitration, which means that frames remain intact after arbitration is completed even if collisions are detected. All the arbitration takes place without corruption or delay of the message that wins the arbitration. The priority technique is also used by the protocol of LonTalk using predictive p-persistent CSMA in the LonWorks in order to reduce the probability for collisions (Dietmar et al., 2001). The advantages of CSMA are the equalities of the nodes, truly distributed, and high responsiveness. Its disadvantage is that there is no upper bound to guarantee the time of message transmission in the case of heavy traffic load on the fieldbus, which is not satisfactory for real-time applications.

2.4.2 TDMA

When the times to transmit all the data in a system are known in advance, such as cyclic exchanges, the time division multiple access (TDMA) technique is surely the most efficient solution to medium access. TDMA combines all the data produced or consumed by the different nodes in a single summation frame and requires that the protocol control information be added only once per frame in order to achieve bit and frame synchronization and guarantee appropriate error controls.

Since all the information from or to the different nodes is collected together, TDMA ensures the highest efficiency among the different MAC techniques. For example, in the Interbus, its behavior is similar to TDMA though in reality it is not true. The communication efficiency of the TDMA network can be as high as 60%. On the contrary, TDMA is not suitable for spontaneous data exchanges and high level communications, in that it is necessary for the whole system bandwidth to be allocated in advance to the different nodes, thus leading to poor flexibility. Moreover, in this case, the bandwidth which is not used by a node cannot be reallocated to other nodes, so that it is effectively wasted, thus reducing the system’s overall efficiency.

2.4.3 Token Passing

In the token passing mechanism, the nodes on the fieldbus are logically organized into a ring, with each node knowing the address of the node to its “left” and “right”. The highest numbered node passes permission to its immediate neighbor by sending the neighbor a special control frame called a token. The token propagates around the logical ring, with only the token holder being permitted to transmit frames. Since only one node at a time holds the token, collisions do not occur. Therefore, the medium access technique of token passing can guarantee a deterministic behavior of the system although it requires more overhead than CSMA/CD. In other words, the token passing technique can grant an upper bound to the time needed to access the fieldbus channel and hence to the time required to send a message. The bound depends heavily on the target token rotation time, which is an important parameter in the token mechanism. The Profibus uses the token mechanism for message transmission among masters in a system. For long message transmissions, a token passing technique can offer more satisfactory performances with respect to both TDMA and CSMA, at least from the point of view of bandwidth sharing in terms of flexibility, fairness and efficiency. In this case, however, responsiveness is reduced and jitters are increased for real-time process data.

2.4.4 Polling

The polling technique is used in a master-slave system, in which only one master is permissible. The master has the right to control the fieldbus access. The master polls each slave node in turn to see if it has a frame to transmit. If it does, the slave node transmits its frame to the master as a response. Thus communication is implemented by each poll/response sequence. The slave nodes cannot communicate with each other. In the PROFIBUS, the master node uses the polling technique to communicate with its slave nodes. The polling technique is the simplest medium access technique, and it is only suitable for networks having a small number of nodes. It has a low efficiency and responsiveness.

2.4.5 Scheduling

In the scheduling technique, all of the communication activities on the fieldbus are scheduled based on some principles by a bus controller, which may have the names, such as arbitrator, scheduler, and master. In this meaning, the polling technique can be viewed as the simplest one of the scheduling techniques. The scheduling principles are often designed according to the kinds of traffic on the fieldbus, such as periodic, aperiodic, urgent, and non urgent. The ideas of reservation, priority, and best-effort are combined together to form the scheduling principle. The examples of fieldbuses using the scheduling technique include the FOUNDATION Fieldbus, FIP and etc. The bus controller is a key to the reliability of the fieldbus using the scheduling technique. Therefore, the bus controller in many fieldbuses is designed to be redundant as done in the FOUNDATION Fieldbus, FIP. Most fieldbuses use the scheduling techniques since good time performances can be achieved to satisfy the requirements of distributed real-time applications. The efficiency and the responsiveness of scheduling technique are medium.

2.5 Fieldbus types

At first glance to the models of fieldbus protocols, all of the fieldbuses look quite similar. However, each one has a unique set of features and is designed to move a specific type of data between certain kinds of equipment. The different methods to classify fieldbuses from various points of view will lead to different types of fieldbuses.

2.5.1 Classifying method from design objective

According to Thomesse (1999), by analyzing the first objectives of fieldbus designers, there are two main types of fieldbuses: type of signal transmission and type of computation distribution.

The type of signal transmission is considered as the network only for simplifying wiring between devices. The design objective of signal transmission type primarily is to substitute networked digital way of signal transmission for analog way of signal transmission. The application computation in the FCS (Fieldbus Control System) based on the type of signal transmission is centralized or limited decentralized only in several master nodes, most of nodes in such system don’t have the capability of distributed computation. The representative system for this is the Profibus- DP fieldbus system.

The computation distribution only represents the algorithms contained in fieldbus nodes that can be configured by users for their application. It doesn’t include the necessary processing algorithm for the operation of the fieldbus node itself. The type of computation distribution is designed to be a spinal column of distributed real time system. And the capability of computation distribution is considered as the main design objective while using a networked digital way of signal transmission. Hence, the type of computation distribution is based on and comprises the type of signal transmission. The FCS based on the type of computation distribution, for example, the FOUNDATION Fieldbus system, totally distributes its application computation into each node. But it does not mean that the type of computation distribution is advantageous over the type of signal transmission. Which one is better depends on the requirements and characteristics of the concrete application case.

2.5.2 Classifying method from protocol model

By analyzing the model of fieldbus protocol, we can have three types of fieldbuses: the basic protocol type, the common protocol type, and the complete protocol type as mentioned in Section 2.2. If a fieldbus only has the physical layer and data link layer in its protocol model, it belongs to the basic protocol type. The CAN fieldbus is an example of this type. It has the features of low cost and simplicity, but weak services to user application. If a fieldbus has the physical layer, data link layer, and application layer in its protocol model, we call it the common protocol type. The P-Net fieldbus is an example of this type. If a fieldbus protocol architecture contains all three layers of the physical layer, the data link layer, and application layer, and there is a user layer in the protocol or a separate profile outside the protocol to define user application to support the interoperability, this fieldbus protocol is referred to as the complete protocol type. The FOUNDATION Fieldbus and LonWorks are typical examples of the complete protocol type. In addition, the DeviceNet and CANopen can be classified to be the complete protocol type although they both use CAN fieldbus as its lower two layers.

2.5.3 Classifying method from medium access mechanism

According to the MAC mechanism that a fieldbus uses, the fieldbus protocols can be categorized into deterministic type and random type. The CSMA mechanism belongs to the random access mechanism, hence the fieldbus protocols that use this mechanism are the random type. The random type of the fieldbus protocols includes LonWorks and CAN.

Most fieldbus protocols belong to the deterministic type because they use the controlled access mechanism. It is either centralized or decentralized. The centralized mechanism uses polling or scheduling technique, whereas the decentralized mechanism also uses a token passing technique. Both polling and token passing mechanisms are used in the Profibus-FMS. The representatives of the centralized and controlled access type are the FOUNDATION Fieldbus, the P-Net, the WorldFIP, the Interbus-S, and the Profibus-PA.

The random type of fieldbus can not bound the response time of messages requested depending on the traffic load on the fieldbus. Collisions may happen when transmitting messages on the fieldbus. In order to satisfy the requirement of real-time communication, additional technique is needed for the random type of fieldbus. However, the deterministic type of fieldbus can guarantee the response time of messages transmitted, and meet the requirement of the hard real-time communication.

2.5.4 Classifying method from message mode

According to the mode of message that a fieldbus uses, fieldbuses can be classified into two types: address-oriented type and message-oriented type. The two modes of message are shown in Figure 2.6. The address-oriented type of fieldbus corresponding to the source-destination model has the source and destination addresses bits in the message structure. Most fieldbuses such as the FOUNDATION Fieldbus and LonWorks belong to the address-oriented type. The message-oriented type of fieldbus corresponding to the producer-consumer model doesn’t have the source and destination addresses bits, but the identifier bits for the message. The CAN fieldbus is a typical example of the message-oriented type. The message-oriented type has less message overhead than the address-oriented type, hence has a higher message efficiency.

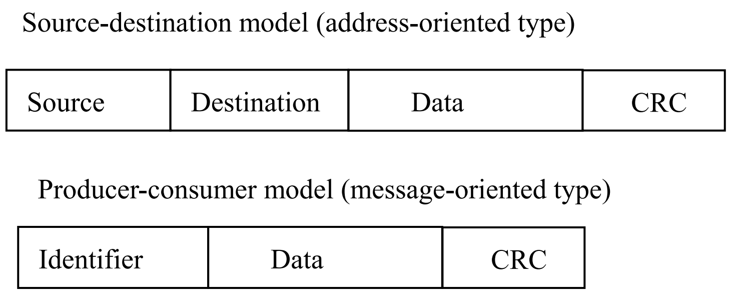

Source-destination model (address-oriented type)
Source
Destination
Data
CRC
Producer-consumer model (message-oriented type)
Identifier
Data
CRC

Figure 2.6: Two message models

2.6 Summary

In this chapter, we presented the models of different fieldbus protocols from basic to complete ones in detail. The model of a fieldbus protocol can reflect the functionality and complexity of the fieldbus. Main fieldbuses were introduced briefly, and their characteristics were discussed. The fieldbus medium access control mechanisms, which have a dominant effect on fieldbus time performance, were also explained. Finally, four methods to classify fieldbuses were proposed. An overview of main fieldbus issues was presented to lay a fundamental for later chapters.

Chapter 3

Timing Analysis and Delay Effect in FCS

3.1 Introduction

Real-time requirement is essential to the FCS. A real-time system must guarantee its performance in both the logical and the temporal domains. A correct answer provided late is as useless as the wrong answer delivered at the right time. To this end it is essential that any real-time computing system must provide predictable and explicit processing in terms of both correct data handling and execution timing. This processing performance must be end-to-end from the lowest-level, isolated sensor, right through all layers of the hardware and software, and back to an isolated actuator (Rodd at al, 1998). The computation tasks and the communication tasks in the FCS are all time consuming, and the times required by these tasks function as time delays in the control loop of the FCS. It is well known that the time delays will degrade the control performance. Therefore, timing analysis is essential to the analysis of FCS. Also, it is the foundation for other research including evaluation of fieldbus and control algorithm design through modeling and simulation.

According to the system model at the physical level of IEC61499 (Lewis, 2001), a fieldbus control loop consists of a set of field devices, such as transmitters, controller and actuators, interconnected through the fieldbus to form a co-operating application for plant control, which requires the interoperation of software running in these devices. Figure 3.1 illustrates a typical setup and the information flow of a fieldbus control loop.

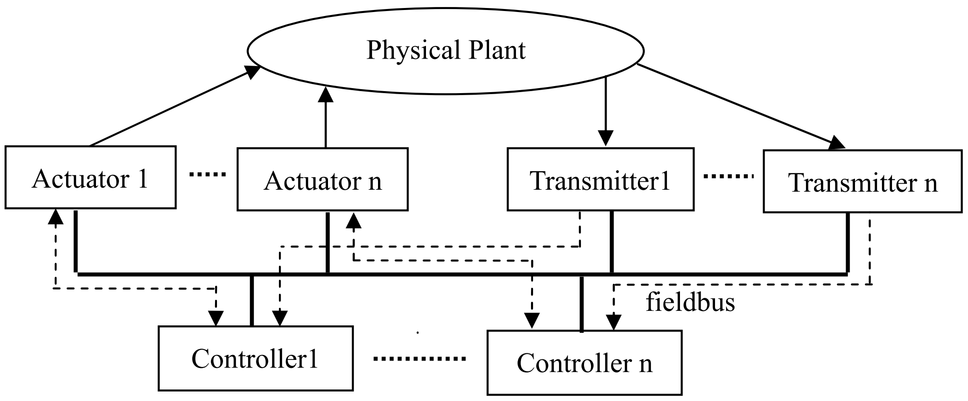


Figure 3.1: A typical FCS setup and information flow

The transmitter nodes will measure the controlled variables of the plant and transmit measured data to the controller node via the fieldbus. The controller node read data from the transmitter nodes, compute the control command for the actuator nodes in terms of the embedded control algorithms, and then send these control commands to the actuator nodes via the fieldbus to maintain the controlled variables at the desired values. The fieldbus is a shared medium for exchanging data among devices. There are two types of fieldbuses. One is only a network for simplifying the wiring between devices, and the other is the spinal column of a distributed real-time system (Thomesse, 1998). The latter supports distributed computation in fieldbus devices, and the term FCS used hereafter only refers to this type.

Tasks carried out in the FCS are divided into three categories: time-critical, periodic, and time-available tasks. Time-critical tasks, such as alarm and event notification, must be done within a very short interval of time. Time-available tasks include management services, and are randomly generated. The periodic tasks are for monitoring and feedback control, which must be completed within a given control period in order to satisfy the real-time requirements or constraints for FCS. In principle, the shorter the control period is, the better the control performance.

In the FCS, the elementary functions are implemented in individual function blocks in the form of software algorithms, and a measurement and control task is specified through the configuration of these function blocks. These function blocks are distributed in fieldbus devices and interconnected through fieldbus communication links and/or internal links in a single device. In this way, a measurement and control task is implemented using distributed devices and algorithms embedded in them. Therefore, a measurement and control task can be partitioned into computation tasks distributed in different field devices and communication tasks among these devices. Moreover, the configuration for an application determines the numbers of the computation tasks as well as communication tasks and the locations where these tasks are completed. The controller can be implemented in the form of hardware or software, depending on the configuration of the system. In a control task, computation tasks are used to execute algorithms embedded in the form of function blocks in fieldbus devices, and communication tasks are responsible for exchanging data among these function blocks located in the different fieldbus devices. Both tasks take a period of time, and the control period will depend on the length of time the two tasks required. In some high speed networked control system, data are transmitted in a package, which contains data of many sampling periods. However, due to the low speed and short message of the fieldbus, only one data is transmitted during each communication in the FCS.

By a fieldbus control loop, we mean a control loop with a feedback loop closed through a fieldbus network. For the fieldbus control loop, a time-delay is inevitably introduced in the loop from the control point of view. In order to reduce the fieldbus delay, much research work has been done on fieldbus design (Tovar and Vasques, 1998, Martins and Fonseca, 2001). There are also a number of studies that address the communication scheduling problems (Almeida et al., 1999, Cavalieri and Monforte, 2001, Franco and Henriques, 2000, Franco, 1996, Hong, 1995).

Although much work has been done for the timing analysis of some specific fieldbuses, the results focus on only the communication time of the fieldbus. None of the papers explicitly consider the timing analysis of the periodic control task in a fieldbus control system. In this chapter, we will analyze the time characteristics of the closed control application from the fieldbus system point of view, which not only involve the communication times but also the computation times and the configurations of applications. The various time-delay components in the control application will be formulated and discussed. The influence of the time-delays on control performance will be analyzed. We will investigate how the different configurations affect the control period, what the best configuration is, and how to determine the control period. The formula and analysis results will be validated by simulation and experiments.

3.2 Timing analysis of fieldbus control system

3.2.1 Periodic task in fieldbus control system

In a fieldbus control loop, time is an important resource. Here, we analyze these time-delay components according to the way they are produced. From the real-time system point of view, the operation of the fieldbus control loop can be abstracted as a periodic task to complete a user control application. This periodic task performs the following three subtasks in each cycle, where each subtask involves many jobs.

(1) Transmitter subtask, S_t

Acquire dynamic data from the plant, complete the data processing or computation such as filtering and calculation, and then transfer the resulting data to the controller via the fieldbus.

(2) Controller subtask, S_c

Receive the data from the transmitter via the fieldbus, compute the control algorithm, and send the resulting data as a control command to the actuator via the fieldbus.

(3) Actuator subtask, S_a

Receive the control command data from the controller via the fieldbus, complete some of the processing, and convert it for execution into the plant.

We can depict the periodic control task in the fieldbus control loop with a task graph (Liu, 2002) as shown in Figure 3.2. Each node represents a subtask. (t_t, t_{td}), (t_c, t_{cd}) and (t_a, t_{ad}) are the release times and deadlines of subtasks S_t, S_c, and S_a, respectively. S_p is a virtual subtask completed by the plant without time consumption, which changes the dynamic output of the plant. Each subtask in a node is dependent. S_t is the immediate predecessor of S_c, S_c is the immediate predecessor of S_a, and so on. Therefore, the precedence graph of the periodic control task is a loop.

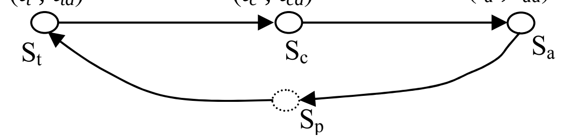


Figure 3.2: Periodic control task graph

3.2.2 Timing components

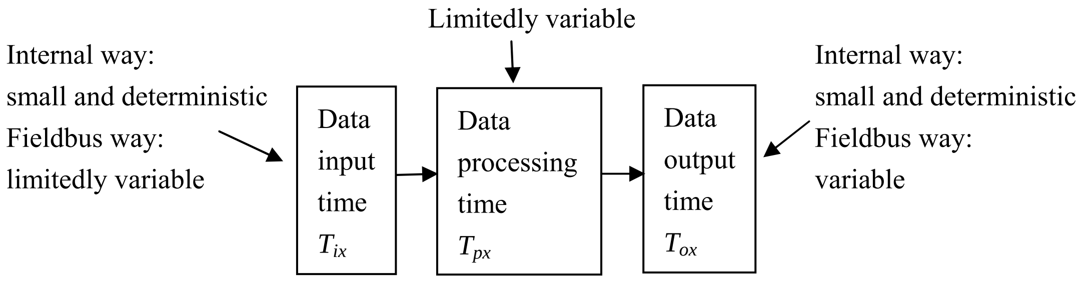
In a fieldbus control loop, there is a cyclic data flow among the subtasks shown in Figure 3.2. The required loop time T_L is determined by Eq. (3.1) to complete one cyclic data flow.

$$T_L = T_{st} + T_{sc} + T_{sa} \ , \qquad (3.1)$$

where T_{st}, T_{sc} and T_{sa} are the subtask times required by transmitter subtask S_t, controller subtask S_c , and actuator subtask S_a, respectively. For each of the above subtasks, there are three timing components from the data-flow point of view as shown in Figure 3.3. To complete the data flow, the subtask time T_{sx} from input to output required by node x is given by Eq. (3.2).

$$T_{sx} = T_{ix} + T_{px} + T_{ox}, \tag{3.2}$$

where T_{ix}, T_{px} and T_{ox} are the data input time, data processing time, and data output time of node *x*, respectively. By the data, here we mean only the real-time control data in the fieldbus control loop. These data do not include any other communication data such as parameters of algorithms, management data and so on.



Limitedly variable
Internal way: small and deterministic
Fieldbus way: limitedly variable
Data input time T_{ix}
Data processing time T_{px}
Data output time T_{ox}
Internal way: small and deterministic
Fieldbus way: variable

Figure 3.3: Timing components of subtask in node x

Here we analyze the timing components in the loop time T_L. First, we discuss the timing components of subtask time T_{sx} in node *x*.

(1) Data input time T_{ix}

Different nodes have different ways to get data input, and different data input methods consume different lengths of time. A transmitter node has two ways to get data input. One way is to get data input via the input circuit to an external sensor such as the A/D converter connected to the sensor, which measures the output of the plant. We call this way the internal way. The time needed to input data by the internal way is small and almost always determined as constant. Therefore, it is negligible or can be added to the time of algorithm execution. The other way is to get data input via a fieldbus. This way is called the fieldbus way. The data input time of the fieldbus way is included in the data communication time, which is discussed later. The transmitter data input by the fieldbus way are not loop control data but parameters or management data. In this study, for the purpose of control performance, we only need to consider the internal way for the transmitter node. For the controller node and actuator node, they only have the fieldbus way to obtain data input.

(2) Data processing time T_{px}

The length of data processing or computing time in node *x* depends on the CPU computing power, number and complexity of algorithms, task load, and the task scheduling method. In general, a fieldbus device often only has one CPU, which divides time into chips to do many tasks including control-dependent data processing, so the algorithm's scheduling method for the CPU is very important in satisfying the real-time requirement. The property of data processing time is determined but not constant. Even for the same data processing, the processing time always varies within a limited range.

(3) Data output time T_{ox}

Like the data input time, the data output time is also dependent on the different data output methods that different nodes have. The transmitter node and controller node only have the fieldbus way for data output, whereas the actuator node has both the internal way and the fieldbus way. But for the loop control data, the actuator node only use the internal way to send data out through an output circuit such as a D/A converter to affect the plant. Similar to the data input time of the transmitter node, the data output time of the actuator node is also small, determined and negligible. The data output time of the fieldbus way is also included in the data communication time.

(4) Data communication time T_{dcxy}

The data communication time is the time required for a data transfer from one node to another node via a fieldbus. The data transfer involves the data being transferred from one data processing application to another one. Therefore, it includes both the data output and the data input, and both are completed continuously during the data transfer. From the communication-protocol point of view, the data communication time via the fieldbus can be classified into various parts as shown in Figure 3.4 (Lian, 2001). At the source node, delays occur due to executing the protocol of the data message, queuing at source node *x*, and blocking due to fieldbus traffic. Once the data message is sent, there is a transmission time delay as discussed below. At destination node *y*, there are again delays of protocol execution before the data can be used. Therefore, the data communication time from node *x* to node *y* T_{dcxy} is determined by Eq. (3.3).

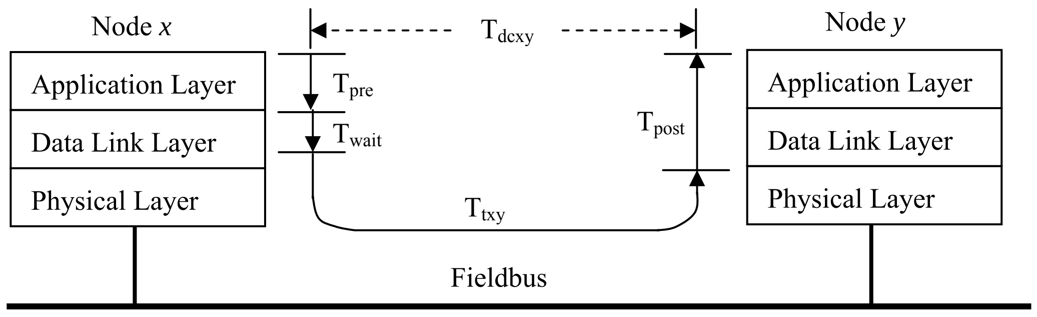


Figure 3.4: Timing diagram showing data communication from source node *x* to destination node *y* via fieldbus

$$T_{dcxy} = T_{pre} + T_{wait} + T_{txy} + T_{post} \quad , \tag{3.3}$$

where T_{pre} is the preprocessing time at the source node, which is the protocol execution time for data transmission, that includes encoding time, frame control time and so on. T_{wait} is the waiting time that includes queuing time and block time before data transmission to the fieldbus. T_{pre} and T_{wait} belong to the data output time. T_{txy} is the transmission time of the data message from node *x* to node *y*, and it is shared by the data output and input times. $T_{post,}$ which belongs to the data input time, is post-processing time at the destination node; it is also the protocol execution time except for data reception, such as decoding time and so on. T_{pre}, T_{wait} and T_{post} are randomly variable times that cannot be calculated but can be measured through experimental methods.

According to the principle of the network data transmission (Leon-Garcia and Widjaja, 1999), if the fieldbus at its physical level uses synchronous transmission mode and no error occurs during transmission, the transmission time T_{txy} of the data message from node *x* to node *y*, which depends on the physical length of the fieldbus, message size, and data rate of the fieldbus, can be calculated by Eq. (3.4).

$$T_{txy} = N_{mxy} \times 8 / V_f + L_{xy} / C_f \, , \tag{3.4}$$

where N_{mxy} (byte) is the byte length of the data message transmitted from node x to node y. V_f (bit/s) is the data rate of the fieldbus. L_{xy} (m) is the length of the fieldbus medium from node x to node y. C_f (m/s) is the speed of signal propagation in the fieldbus medium. From Eq. (3.4), we can see that the transmission time of a data message includes two components of time, where one is the frame time and the other is the data signal propagation time. Generally, the frame time is on the order of millisecond and dominant in the data transmission time, while the data signal propagation time is on the order of microsecond and negligible. In other words, the transmission time is fieldbus-dependent and determined by the network parameters of the fieldbus such as the data length and rate.

3.2.3 Control loop time

Substituting the respective terms in Eq. (3.1) with Eq. (3.2), we can rewrite the control loop time T_L as the following equation.

$$T_L = T_{it} + T_{pt} + T_{ot} + T_{ic} + T_{pc} + T_{oc} + T_{ia} + T_{pa} + T_{oa} \tag{3.5}$$

According to the previous analysis, T_{it} and T_{oa} can be negligible or included in their respective data processing times T_{pt} and T_{pa}. Also, we have $T_{ot}+T_{ic}=T_{dctc}$ and $T_{oc}+T_{ia}=T_{dcca}$. Thus,

$$T_L = T_{pt} + T_{dctc} + T_{pc} + T_{dcca} + T_{pa} \quad , \tag{3.6}$$

where T_{pt}, T_{pc} and T_{pa} are the data processing times of transmitter node, controller node, and actuator node, respectively. T_{dctc} and T_{dcca} are the data communication times from transmitter node to controller node and from controller node to actuator node, respectively.

Similarly, for complex control applications, the general control loop time of a control task in the FCS can be represented by Eq. (3.7).

$$T_L = \sum_{i=1}^{N} T_{comp}^{i} + \sum_{j=1}^{M} T_{comm}^{j} \; , \tag{3.7}$$

where T_L is the control loop time. T_{comp}^{i} is the time taken by the i th computation task and T_{comm}^{j} the time taken by the j th communication task. N and M are the numbers of computation tasks and communication tasks included in the control task respectively. These numbers depend on the complexity and configuration of a user control application. In the case of Figure 3.2, N=3 and M=2.

If T_L is constant, the loop control task becomes accurately periodic. However, based on our previous analysis, T_L practically cannot be constant and has more or less variation due to many factors. We can use the period jitter J_T defined by Eq. (3.8) as an index to measure the magnitude of the variation.

$$J_T = T_L - T_{\min} \quad , \tag{3.8}$$

where T_{min} is the minimum period occurring in the control loop. J_T is a random variable, which has some kind of distribution.

The period jitter degrades the performance of the control loop. The jitters are mainly caused by program synchronization and communication and to a lesser degree by transmitters and actuators (Dietmar et al., 2001). The medium access control method of fieldbus and the traffic load on the fieldbus are the main causes of jitters. The sampling and control period of the fieldbus control loop T_{cp} is closely related to the control loop time T_L. The stability condition for normal operation of the fieldbus control loop is as follows:

$$T_L \le T_{cp} \, . \tag{3.9}$$

It should be noted that here the stability is for the communication and computation of the fieldbus control loop, not for the control. Equation (3.9) gives the lowest bound for the control period of FCS. The bound is obtained by considering communication and control computation at the same time. The communication tasks and the computation tasks are not independent, but have a precedence relationship with each other as shown in Figure 3.2. Therefore, T_L is the feasible and minimum control period. In some FCSs such as the FOUNDATION Fieldbus-based FCS, a constant T_{cp} that overly satisfy the condition given by Eq. (3.9) is selected manually during configuration in order to avoid the period jitter. Thus a margin time is produced, and according to Eq. (3.7) the control period

$$T_{cp} = \sum_{i=1}^{N} T_{comp}^{i} + \sum_{j=1}^{M} T_{comm}^{j} + T_{m\arg in} \quad , \tag{3.10}$$

where T_{margin} is the margin time that can guarantee the completion of the computation tasks and communication tasks within a control period. The margin time sometimes includes the time cost for tasks scheduling. The margin time acts similarly as a buffer to avoid the effect of period jitter but makes the control period longer.

3.3 Determination of control period in FCS

From Eq. (3.10), the control period T_{cp} depends on two components: computation time and communication time. These times further depend on the design principles and the type of fieldbus used in the FCS. Therefore, a common formula of control period that is suitable for any FCS cannot be obtained. The FOUNDATION Fieldbus-based FCS is chosen as a study example of the timing analysis since it is the most typical of FCSs used in process control.

3.3.1 Time taken by computation tasks

Computation tasks compose the execution of all algorithms contained in the function blocks in a user control application. The architecture of the FOUNDATION Fieldbus-based FCS is designed to support the function block model in its user layer. The function blocks implement elementary field device functions, such as analog input (AI) functions, proportional-integral-derivative (PID) control algorithm, and analog output (AO) functions.

As shown in Figure 3.5, the function blocks are composed of a complete complement of parameters, algorithms, and events. Their algorithms perform input, output, calculation, and control functions for the application system. For process input and output functions, they are locally linked to transducer blocks that interact with device-specific I/O hardware.

The control function block can be implemented in any fieldbus device. For example, the function block of a PID controller can be embedded in a transmitter, a valve positioner, or a hardware controller. Therefore, control functions can be distributed in the factory floor level and in different field devices.

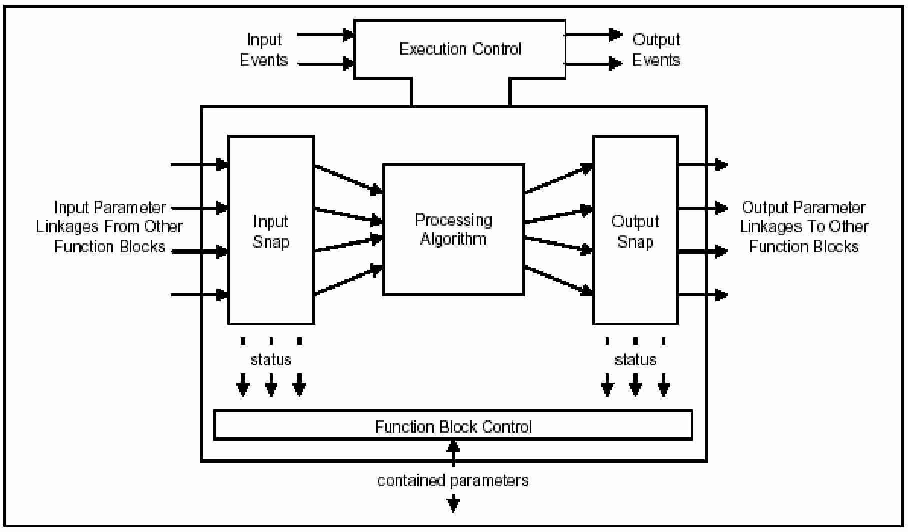


Figure 3.5: Compositions of function block (Fieldbus Foundation, 1996)

Each function block takes an execution time, which depends on the complexity of the task algorithm and the power of the microprocessor used in the device. The execution time of a function block includes the computation time required by the computation task and the time required by a function block to produce data for its successor function block. Table 3.1 lists typical execution times for common function blocks, including AI, PID and AO in the FOUNDATION Fieldbus devices. The execution time of a function block ranges widely from 15 ms to 160 ms because different vendors and devices use different algorithms and different microprocessors, even for the same type of function block. The temporal characteristics of computation tasks can be described by the execution times of function blocks provided by the device vendor.

Table 3.1: The execution time of function blocks (DeltaV^{TM} website)

Function blocks / Vendors& Devices	AI Execution Time T_{AI}	PID Execution Time T_{PID}	AO Execution Time T_{AO}
Rosemount, Pressure Transmitter (3051)	30 ms	45 ms	----
Rosemount, Temperature Transmitter (3244)	45 ms	100 ms	----
Yokogawa, Mag Flow Meter (ADMAG AE100)	100 ms	160 ms	----
Rosemount, Mag Flow Meter (8742)	15 ms	25 ms	----
Fisher Controls, Digital Valve Controller (DVC5000f)	----	160 ms	80 ms
Smar, Fieldbus to Pneumatic Signal Converter (FP302)	----	67 ms	106 ms

3.3.2 Time taken by communication tasks

In the FOUNDATION Fieldbus, the bandwidth of the fieldbus is divided into two parts: a window for periodic data transfer and a window for asynchronous data transfer as shown in Figure 3.6. The number of the two types of windows in a control period depends on the configuration of the user control application.

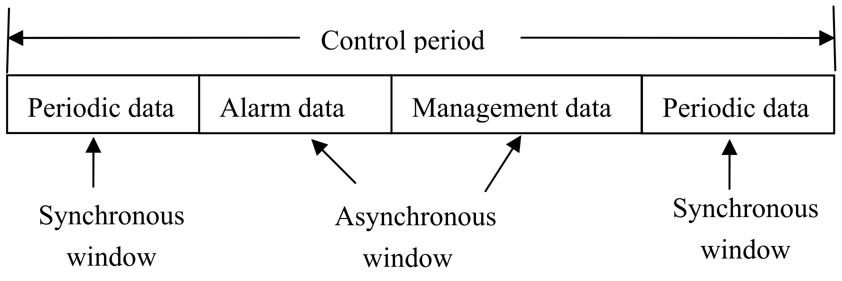


Figure 3.6: Communication window within a control period

The FOUNDATION Fieldbus uses the scheduling technique as discussed in Section 2.4.5 to transmit the periodic data. Each periodic communication task contains the transfer of two frames: the CD (Compel Data) frame and the DATA (Data) frame. The CD frame is used to schedule the transfer of periodic data. Each transfer of a periodic data frame requires a transfer of the corresponding CD frame. Each CD that arrives to the fieldbus has an associate bandwidth requirement. In the synchronous window, all transfers are real time data and require equal bandwidth. However, for the asynchronous window, the non real time data may require different amounts of bandwidth. The time spent for each periodic communication task is represented as the sum of the two transferring times of a CD frame and a DATA frame as given in Eq. (3.11).

$$T_{comm} = T_{frame}^{CD} + T_{frame}^{DT} \quad , \qquad (3.11)$$

where T_{comm} is the time required by each periodic communication task, and T_{frame}^{CD} and T_{frame}^{DT} are the transferring times required by a CD frame and a DATA frame, respectively. According to Eq. (3.3), the frame time is

$$T_{frame} = T_{pre} + T_{wait} + T_{txy} + T_{post} \quad , \qquad (3.12)$$

where T_{frame} can be T_{frame}^{CD} or T_{frame}^{DT}. T_{pre} and T_{post} depend on the complexity of the protocol and the power of the microprocessor used in the device. For example, a CD frame goes through fewer protocol layers than a DATA frame. Consequently, the transfer of a CD frame through the filedbus protocol layers should take less time than that of a DATA frame. In practice, T_{pre} and T_{post} are difficult to measure separately. However, for the FOUNDATION Filedbus, the fieldbus idle time T_{idle} between two consecutive frames is approximately equal to $T_{pre}+T_{wait}+T_{post}$ because there is no preemption during this interval of time. However, for the fieldbus using the random medium access control method, possible preemptions must be considered in the measurement of $T_{pre}+T_{wait}$

$+T_{post}$. Therefore, $T_{pre} + T_{wait} + T_{post}$ is a random length of time depending on the traffic load and must be estimated through experiments using a specific fieldbus analysis tool. T_{wait}, the waiting time, is random for any fieldbus using the random medium access control method. For the FOUNDATION Fieldbus using the scheduling method, T_{wait} can be considered to be zero.

For the FOUNDATION Fieldbus using the scheduling method to control the medium access, considering Eqs. (3.4) and (3.12), and ignoring the propagation time in T_{txy}, Eq. (3.12) can be rewritten as Eq. (3.13).

$$T_{frame} = N_{mxy} \times 8 / V_f + T_{idle} \quad , \tag{3.13}$$

where N_{mxy} is the byte length of a frame transmitted from node x to node y, which includes the frame overhead and the frame data. V_f is the transmission bit rate of the fieldbus (bits/second).

3.3.3 Configuration of function blocks

In the FCS, any user application is specified through a configuration of function blocks. The configuration is set up to assign the computation tasks and the communication tasks to fieldbus devices. The types and numbers of function blocks and communication links are determined by the configuration, and therefore the time spent on computation tasks and communication tasks is determined. Any user application can be implemented through different configurations, which take various times for computation tasks and communication tasks. The control period for these configurations will be different.

Consider the simple closed loop control shown in Figure 3.7, which is mostly used in process control. In the FOUNDATION Fieldbus-based FCS, we have the configuration for the control loop shown in Figure 3.8. There are three function blocks, AI (analog input), PID (proportional-integral-derivative) and AO (analog output), and three links in the closed control loop. The function blocks and three links are distributed in the fieldbus devices. The AI function block is located at a transmitter and provides the measurement of the controlled variable to the PID control function block through the link between the AI and PID function blocks. The AO function block is located at an actuator, receives the control command from the PID control function block, and outputs the manipulated variable to the plant through the link between the PID and AO function blocks. Once the transmitter and the actuator are determined, the function blocks of AI and AO are located. However, the PID function block can be located at a controller, a transmitter, or an actuator. Each configuration will cause a different traffic load on the fieldbus. The feedback link from ‘bk_out’ to ‘bk_in’ between the AO and PID function blocks is for output tracking, which is used for controller mode transfer between ‘Auto’ and ‘Manual’.

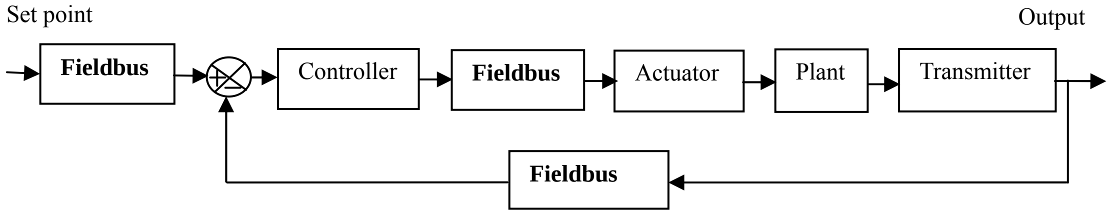


Figure 3.7: Block diagram of fieldbus control loop

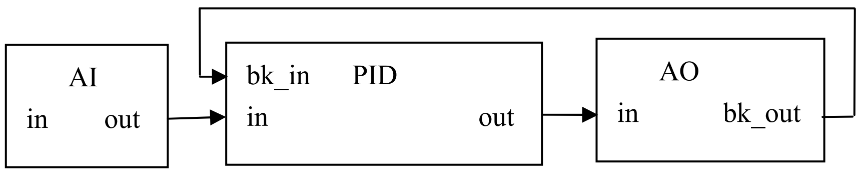


Figure 3.8: Configuration of function blocks for a simple control loop

All the links between the function blocks can be implemented internally or externally. If two linked function blocks are located in a single device they can be linked internally. Otherwise, they have to be externally linked through the fieldbus. External link will increase the traffic load on the fieldbus. A configuration having the most internal links and the less external communication links will produce the least traffic load on the fieldbus and therefore take a minimal communication time. Furthermore, less communication links mean higher reliability. Table 3.2 illustrates the number of links for three different configurations for the simple control loop shown in Figure 3.7. In configuration 1 the PID function block is located at the actuator with the AO function block. In configuration 2 the PID function block is located at the transmitter with the AI function block. The configuration 3 locates the PID function block at the controller. The configuration 1 only has one external link and will take the shortest communication time. The configuration 3 has three external links and will take the longest communication time.

Table 3.2: Three different configurations for a simple control loop

Configuration	**Place of AI**	**Place of PID**	**Place of AO**	**Number of internal links**	**Number of external links**
Configuration 1	Transmitter	Actuator	Actuator	2	1
Configuration 2	Transmitter	Transmitter	Actuator	1	2
Configuration 3	Transmitter	Controller	Actuator	0	3

3.3.4 Control period analysis

To implement a closed control loop, the communication and computation tasks have to be executed periodically at precious instants. They must be accurately scheduled within a control period. The scheduling of these tasks includes determining what tasks should be executed and at what time instant. Different FCSs have different scheduling mechanism. The FOUNDATION Fieldbus uses a clock-driven approach for the scheduling. There is a globally synchronized timer in the system for this purpose. Figure 3.9 illustrates the scheduled execution times for the configuration 1 of the simple control loop shown in Figure 3.7. The only external communication link in the configuration 1 is scheduled at the time slot 20 to 30 with the label 'Synchronous window'. The rest time slots are available for asynchronous data. The function block AI is executed at the time slot 0 to 20, the PID and AO at the time slot 30 to 50 and 50 to 70. The margin time is at the time slot 70 to 90.

For the simple closed control loop shown in Figure 3.7 with the configuration 1, the control period T_{cp} is determined by Eq. (3.14):

$$T_{cp} = T_{comm} + T_{AI} + T_{PID} + T_{AO} + T_{margin} \quad , \tag{3.14}$$

where T_{AI}, T_{PID} and T_{AO} are the execution times of the AI, PID and AO function blocks, respectively. T_{margin} is the margin time. T_{comm} is the time required to complete one communication link task between two function blocks over the fieldbus. It includes the transfer times for both a CD frame and a DATA frame as shown in Eq. (3.11).

For multiple control loops, if these loops have different control periods a multi-cycle scheduling method is required. If these loops have the same control period, a monocycle scheduling method is used. The monocycle scheduling method is much simpler than the multi-cycle method. The FOUNDATION Fieldbus uses the monocycle scheduling method, and the control period is formulated as Eq. (3.15).

$$T_{cp} = M \cdot T_{comm} + T_{AIMAX} + T_{PIDMAX} + T_{AOMAX} + T_{margin} \quad , \tag{3.15}$$

where T_{AIMAX}, T_{PIDMAX} and T_{AOMAX} are the maximum execution times of the AI, PID and AO function blocks used in the fieldbus devices. M is the total number of the communication links over the fieldbus for the multiple control loops, and is determined by the configuration of each control loop. Each communication link takes a same communication time because the byte length is identical for all data transferred among the function blocks.

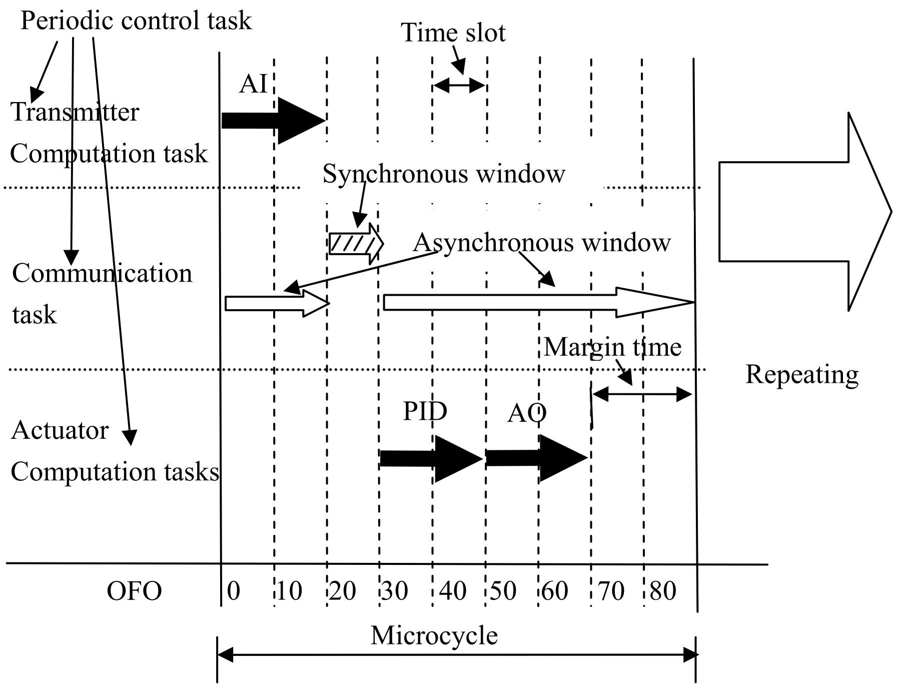


Figure 3.9: Scheduling of a periodic control task

3.4 Experimental study

3.4.1 Experimental setup

The layout of the FCS used in our experiments is shown in Figure 3.10. The system consists of an analysis station, two operator stations, a control station, a water tank, and three fieldbus devices: a temperature fieldbus transmitter, denoted as TT; a differential pressure fieldbus transmitter, denoted as LT; and a fieldbus control valve, denoted as LV. The three fieldbus devices were provided by the Emerson group, and are connected with each other and with the control station via a fieldbus segment using a tree topology. The control station and the two operator stations are linked together via an Ethernet network. The analysis station is connected to the fieldbus segment with a fieldbus interface card provided by NI Inc. to monitor the fieldbus communication activities.

Using the analysis station, all the communication activities on the fieldbus can be monitored and recorded. Figure 3.11 shows a snapshot of a monitoring window on the fieldbus. It contains all of the information sent on the fieldbus over a period of time. The information includes the message type such as CD frame and DATA frame, byte length, the starting time and the ending time of each activity, and the fieldbus idle time between two consecutive messages, and the control period.

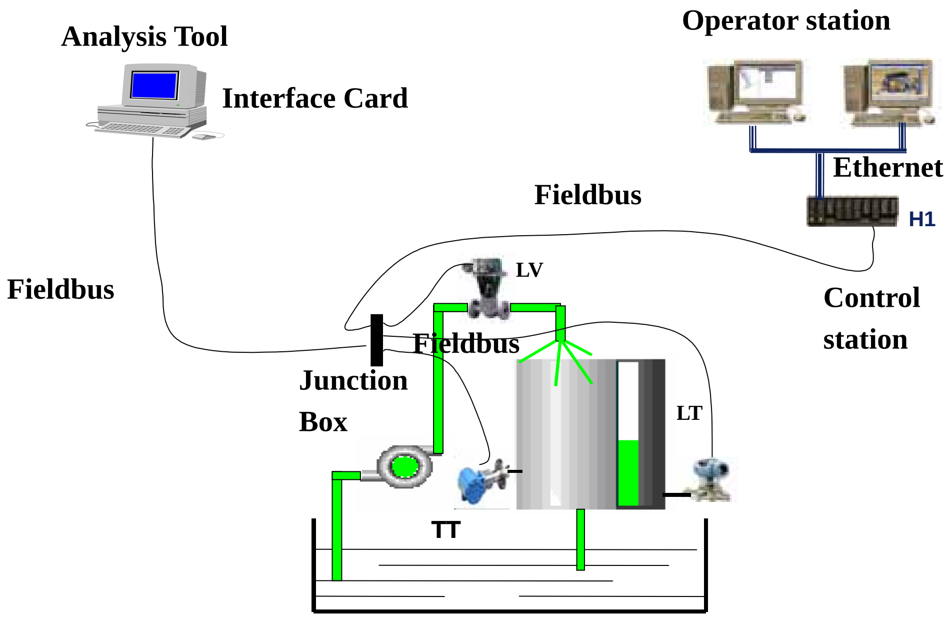
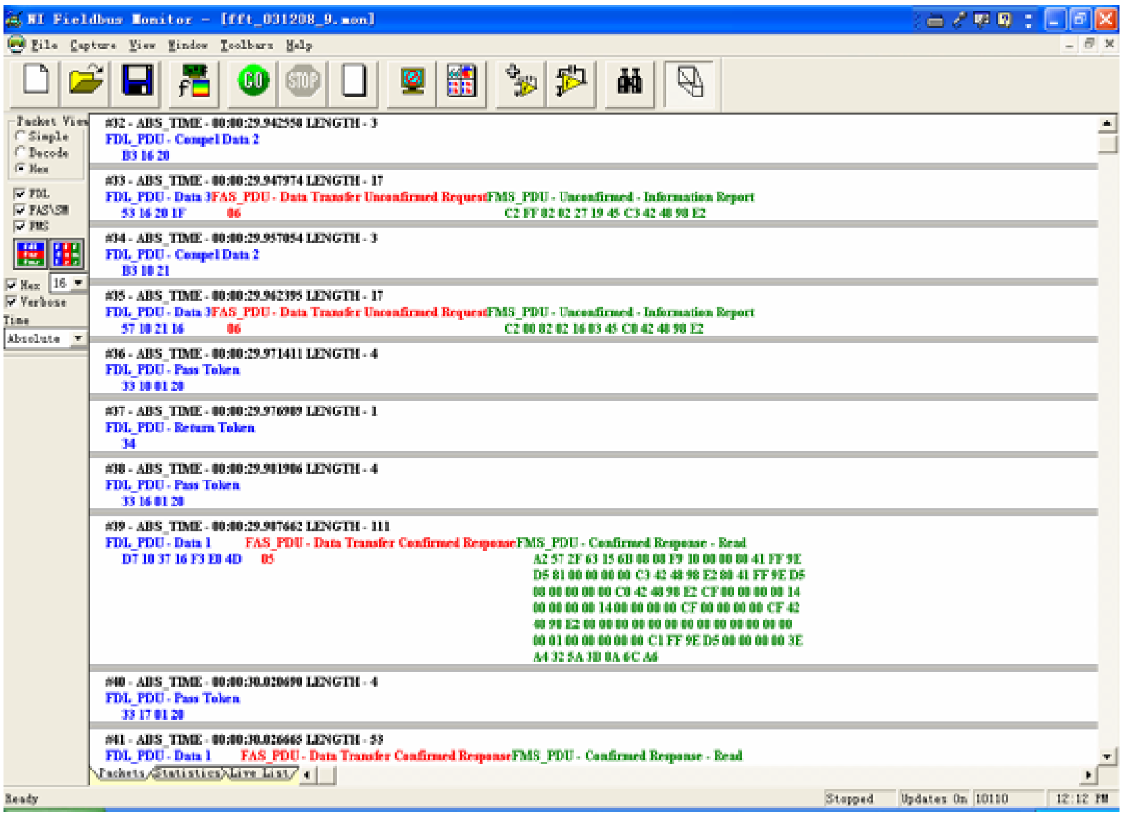


Figure 3.10: Layout of FCS



```
#32 - ABS_TIME - 00:00:29.942550 LENGTH - 3
FDL_PDU - Compel Data 2
    B3 16 20

#33 - ABS_TIME - 00:00:29.947974 LENGTH - 17
FDL_PDU - Data 3FAS_PDU - Data Transfer Unconfirmed RequestFMS_PDU - Unconfirmed - Information Report
    53 16 20 1F          06                              C2 FF 02 02 27 19 45 C3 42 40 98 E2

#34 - ABS_TIME - 00:00:29.957054 LENGTH - 3
FDL_PDU - Compel Data 2
    B3 10 21

#35 - ABS_TIME - 00:00:29.962395 LENGTH - 17
FDL_PDU - Data 3FAS_PDU - Data Transfer Unconfirmed RequestFMS_PDU - Unconfirmed - Information Report
    57 10 21 16          06                              C2 00 02 02 16 03 45 C0 42 40 98 E2

#36 - ABS_TIME - 00:00:29.971411 LENGTH - 4
FDL_PDU - Pass Token
    33 10 01 20

#37 - ABS_TIME - 00:00:29.976989 LENGTH - 1
FDL_PDU - Return Token
    34

#38 - ABS_TIME - 00:00:29.981906 LENGTH - 4
FDL_PDU - Pass Token
    33 16 01 20

#39 - ABS_TIME - 00:00:29.987662 LENGTH - 111
FDL_PDU - Data 1        FAS_PDU - Data Transfer Confirmed ResponseFMS_PDU - Confirmed Response - Read
    D7 10 37 16 F3 E0 4D    05                              A2 57 2F 63 15 6D 00 00 F9 10 00 00 00 41 FF 9E
                                                            D5 81 00 00 00 00 C3 42 48 98 E2 80 41 FF 9E D5
                                                            00 00 00 00 00 C0 42 40 98 E2 CF 00 00 00 00 14
                                                            00 00 00 00 14 00 00 00 00 CF 00 00 00 00 CF 42
                                                            40 98 E2 00 00 00 00 00 00 00 00 00 00 00 00 00
                                                            00 01 00 00 00 00 00 C1 FF 9E D5 00 00 00 00 3E
                                                            A4 32 5A 3D 0A 6C A6

#40 - ABS_TIME - 00:00:30.020690 LENGTH - 4
FDL_PDU - Pass Token
    33 17 01 20

#41 - ABS_TIME - 00:00:30.026665 LENGTH - 53
FDL_PDU - Data 1        FAS_PDU - Data Transfer Confirmed ResponseFMS_PDU - Confirmed Response - Read
```

Figure 3.11: Snapshot of message activities on fieldbus

3.4.2 Experimental results

Experiments were conducted in three different configurations for the liquid level control loop as shown in Figure 3.10. In configuration 1, the PID function block is located in the fieldbus control valve LV, and there is only one communication link for the data transfer of the liquid level from the transmitter LT to the valve LV over the fieldbus. In configuration 2, the PID function block is placed in the transmitter LT, and there are two communication links between the transmitter LT and the valve LV. The feed-forward link is from the transmitter LT to the valve LV over the fieldbus. The other link is from the valve LV to the transmitter LT, which is used for tracking the AO output. There are three communication links over the fieldbus in configuration 3, where the PID function block is placed in the temperature transmitter TT. The three links in configuration 3 are one from LT to TT, one from TT to LV, and one from LV to TT. These three configurations are summarized in Figure 3.12.

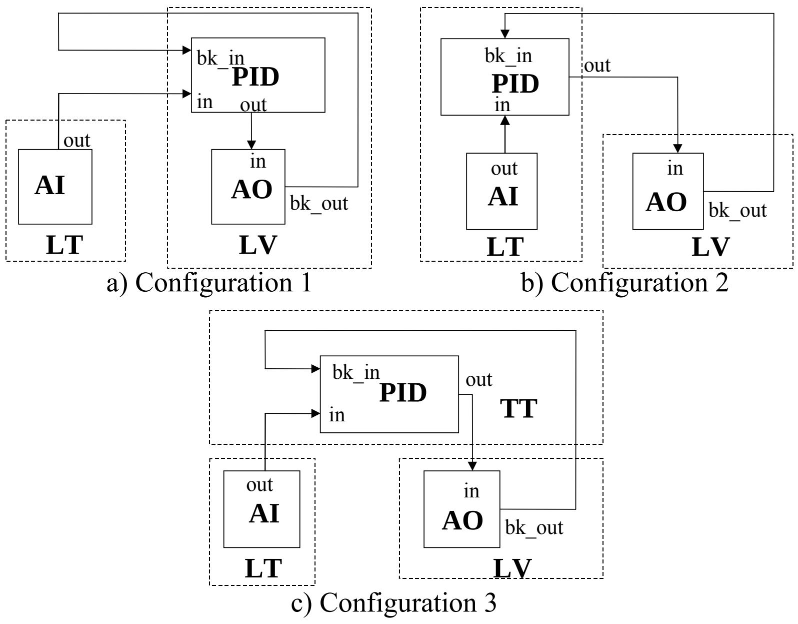


Figure 3.12: Three configurations of liquid level control loop

Table 3.3 illustrates the time spent on each periodic communication task. The experimental results are highlighted in bold. The data transferred among function blocks have an identical number of data bytes, i.e., 23. The number of the CD frame has 9 bytes. The transmission time T_{txy} and the fieldbus idle time T_{idle} for the CD frame and the DATA frame are recorded in Table 3.3. The calculated results of T_{txy} according to Eq. (3.4) by ignoring the propagation time are also given in Table 3.3. There is no error between the calculated results and the experimental results in T_{txy}. The reason should be that the results presented by the monitoring tool are also calculated based on the same formula. This validates partially the correctness of our analysis. The time spent for transferring a frame, T_{frame}, is given out in Table 3.3 by Eq. (3.13). The time spent for each periodic communication task, T_{comm}, is calculated based on Eq. (3.11), and the result is 14.42 ms.

Table 3.3: Results of communication time between function blocks (V_f=31.25 kbits/second)

Frame	$N_{overhead}$ (bytes)	$N_{message}$ (bytes)	N_{mxy} (bytes)	T_{txy} (ms)		T_{idle} (ms)	T_{frame} (ms)	T_{comm} (ms)
				Calculated	Measured			
CD	6	3	9	2.304	**2.304**	**3.097**	5.401	14.42
DATA	6	17	23	5.888	**5.888**	**3.131**	9.019	

Table 3.4 illustrates the control period and its time component in three different configurations for the liquid level control loop. Only the control periods T_{cp} are the measurement results. The communication time T_{comm} is obtained from Table 3.3. The execution times of the AI, PID, and AO function blocks, denoted as T_{AI}, T_{PID}, and T_{AO}, are given in the device specifications. The margin time T_{margin} is calculated by using Eq. (3.10).

Table 3.4: Time components of a control period for three different configurations

Different configurations	AI, T_{AI} Device,(ms)	PID, T_{PID} Device ,(ms)	AO, T_{AO} Device ,(ms)	T_{comm} (ms)	T_{margin} (ms)	T_{cp} (ms)
Configuration 1	LT, 30	LV, 160	LV, 80	14.42	215.552	**499.972**
Configuration 2	LT, 30	LT, 45	LV, 80	28.84	316.197	**500.037**
Configuration 3	LT, 30	TT, 100	LV, 80	43.26	246.709	**499.969**

3.4.3 Discussions

Table 3.4 shows the following facts:

- The execution times of the function blocks and the margin time are dominant, taking about 90% of the control period, whereas the communication time is secondary, taking less than 10%;
- The execution time of the PID function block will be different if located in different fieldbus devices. For example, if the PID function block is located at the transmitter LT the execution only takes 45 ms; if located at the fieldbus control valve LV the execution will take 160 ms;
- Different configurations take different communication times;
- The margin time shows the potential of reducing the control period for each configuration.

In order to reduce the control period the PID function block must be located at a fieldbus device where the execution time is minimal. Also an appropriate configuration must be chosen to reduce the number of the communication links. The minimum control period can be estimated by taking the margin time being zero for each configuration. The most commercial FCSs have a minimum control period of 500 millisecond (ms). For most process plants, which often have a minute-level order of time constant, 500 ms is an acceptable control period. But for a fast plant with a second-level order of time constant, this is too long to achieve a good control performance. The control period analysis should be carried out to reduce the margin time, the communication time, and the execution times of the function blocks. In addition, the communication time occupying less than 10% of a control period indicates that the control period is not sensitive to the fieldbus transmission speed.

3.5 Effects of time delays and jitter of fieldbus on performance criteria

Delays in a control loop are widely known to degrade system performances of a control system, so are the delays of computations and communications in the FCS. In order to evaluate the performance of control system, we select some performance criteria, which include IAE and ITAE. These two criteria are often used in process control. IAE is the integral of the absolute value of the error, and ITAE is the integral of the time multiplied by the absolute value of error. These are represented as follows:

$$IAE = \int_{t_0}^{t_f} |e| dt, or \sum_{k=k_0}^{k_f} |e_k| \text{, and} \tag{3.16}$$

$$ITAE = \int_{t_0}^{t_f} t|e| dt, or \sum_{k=k_0}^{k_f} k|e_k| \tag{3.17}$$

where t_0 (k_0) and t_f (k_f) are the initial and final times of evaluation period in continuous (discrete) time and *e* is the error between the actual and reference trajectories. ITAE weighs later errors heavier and discounts the transient response, whereas IAE weighs all errors equally.

In the FCS, there are two kinds of time delays, which are the computation time delay and the communication time delay. We study the effects of these two delays on the performance criteria mentioned above. The effect of the control period jitter is also investigated.

In order to visualize the effects of the time delays and the jitter on the performance criteria, we use the most common PID control law and the most common plant model of the first order plus time delay in process control. The open-loop transfer function of the plant model and the discrete control algorithm of the PID controller are given by Eqs. (3.18) and (3.19).

$$G(s) = \frac{0.01}{20S+1}e^{-S} \quad , \tag{3.18}$$

$$C_o(t_k) = K_c[e(t_k) + \frac{1}{T_i}\sum_{i=1}^{k} e(t_i)(\Delta t_i) + T_d \frac{e(t_k) - e(t_{k-1})}{\Delta t_k}] \quad ,$$

$$e(t_k) = SP(t_k) - PV(t_k)\,,\ \ \Delta t_i = t_i - t_{i-1}(i = 1,....,k)\,, \tag{3.19}$$

where $SP(t_k)$ and $PV(t_k)$ are the setpoint and process variables of the controller, respectively, $C_o(t_k)$ is the output of the controller, and K_c, T_i, and T_d are the proportional gain, integration time and derivative time of the controller, respectively. Δt_i or Δt_k is the sampling period of *i* th or *k* th sampling instance, and it is variable with time on the fieldbus. The parameters *SP*=2, K_c=1, T_i=0.2, and T_d=0.

Figures 3.13, 3.14 and 3.15 show the simulation results of the effects of the time delays and the jitter on the performance criteria. The results shown in the figures include the effects of following changes: the increase of the control period including computation times and communication times, the proportion changes of the communication times to the control period, and the changes of the jitter range, respectively. The details for modeling and simulation will be described in Chapter 6. Only the simulation results are presented here. During simulation, the range of control period is changed by considering the lowest bound given in Eq. (3.9). That is, the stability of communication and computation is guaranteed during simulation. We don't allow the free change of the period independently unlike some researches (Lian, 2001; Walsh et al., 2002). We think that the period change without considering the control application will result in unrealistic results.

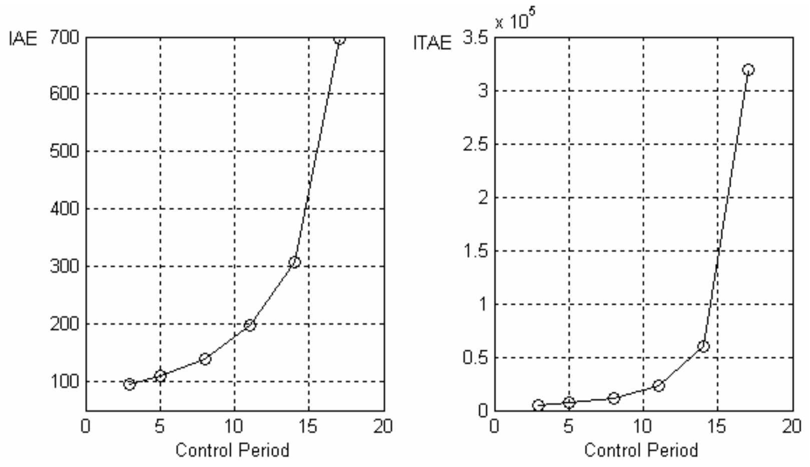


Figure 3.13: The effects of control period changes on IAE and ITAE when jitter of the control period is zero.

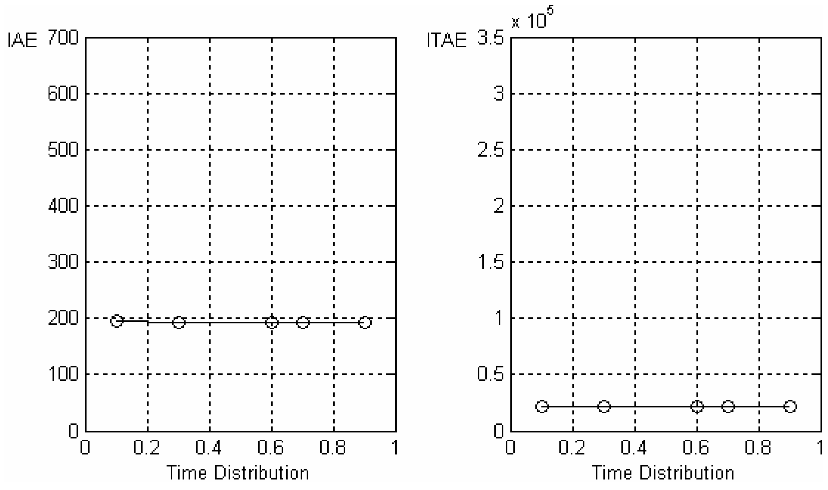


Figure 3.14: Effects of proportion changes of communication or computation times to the control period on IAE and ITAE when the control period is constant and $\Delta t_i = 11$.

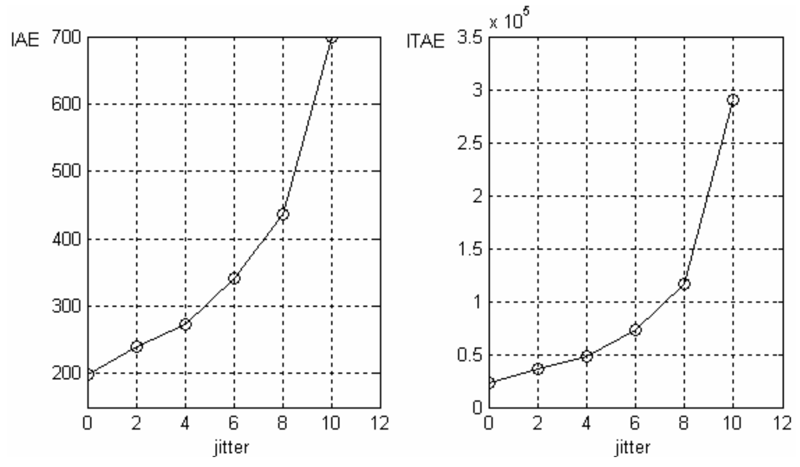


Figure 3.15: Effects of changes of jitter range on IAE and ITAE when the minimum control period $T_{\min} = 11$.

Figures 3.13, 3.14, 3.15 show the following facts:

- The system performance degrades with the increase of the control period including the computation and communication time delays. The larger the delay, the worse the performance.
- For a given control period, the proportion changes of the communication or computation times to the control period have no effects on performance criteria IAE and ITAE. It says that the different time delay components in the control loop, including the communication delays and the computation delays in different fieldbus devices have the similar influence on the control performance. This result tells us the fact that the different delay components are commutative for a given control period from the control point of view.
- The jitter has a bad effect on the performance. However, a small range of period jitter doesn't result in a severe influence on the control performance. Therefore, a small range of period jitter is allowed in the FCS.
- There is an upper bound for the time delay and the jitter of fieldbus. The control system will become unstable if this upper bound is exceeded.

Except the performance criteria of IAE and ITAE, other performance criteria such as the overshoot and the settling time can also be used to evaluate the effects of the time delays and jitter in the FCS. In the study of Tipsuwan and Chow (2003), obvious system performance degradations are the higher overshoot and the longer settling time when time delays exist in the control loop of the FCS. Figures 3.16 and 3.17 show the results of our simulation for the effects of the time delays and jitter on overshoot and settling time in the FCS.

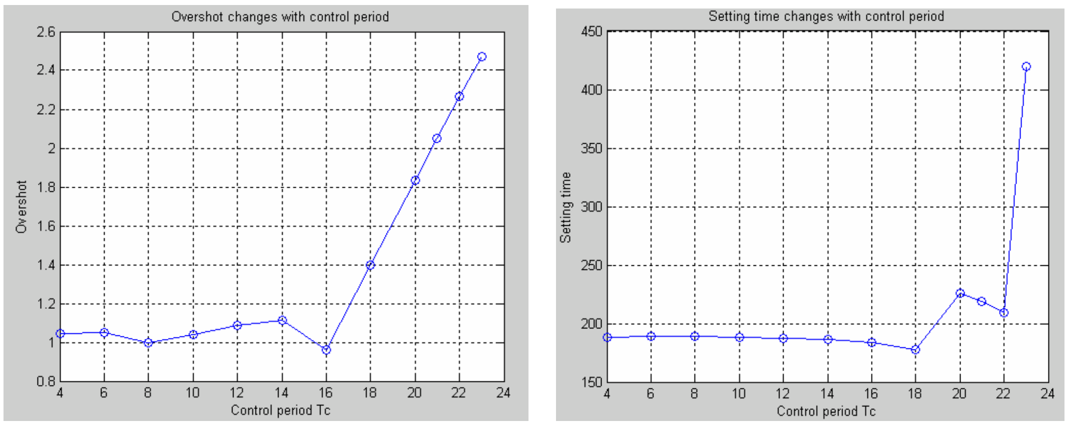


Figure 3.16: Effects of control period changes on overshoot and settling time when jitter of the control period is zero.

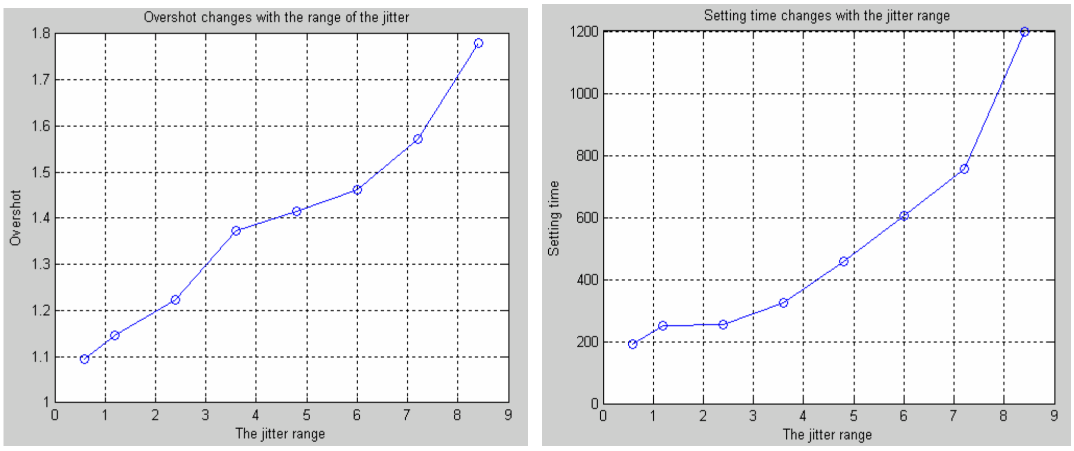


Figure 3.17: Effects of changes of jitter range on overshoot and settling time when the minimum control period $T_{\min} = 4$

3.6 Summary

The timing characteristics of the FCS are analyzed in details. First the control period in the FCS for a control loop is formulated and analyzed. It is equal to the sum of the execution times required by the function blocks distributed in fieldbus devices, the communication times between these function blocks, and the margin time reserved to guarantee all of the periodic tasks to be completed within a control period. The methods of determining these times are given. The stability condition for normal operation of the fieldbus control loop is obtained. This condition gives the lowest bound for the control period of FCS. The bound is obtained by considering the communication and the control computation at the same time. The stability condition also represents that the communication tasks and the computation tasks are not independent from time point of view, but have a precedence relationship with each other. The analysis and experimental results show that the execution times and the margin times are dominant in a control period, whereas the communication time is secondary, and also the execution time of the PID function block might be different if located at different fieldbus devices. Therefore, in order to shorten the control period the PID function block must be located at a fieldbus device where the execution time is minimal. Also an appropriate configuration must be chosen to reduce the number of the communication links.

The effects of communication times, computation times, and the jitter of control period on the performance of control are discussed by evaluating the performance criteria IAE, ITAE, the overshoot, and the settling time based on the simulation results. In general, the increases of communication times, computation times, and the jitter of control period will worsen the control performance, and the larger these times and jitter, the worse the control performance. For a given control period, the distribution changes of computation or communication times in the control period have no effects on the control performance. This result tells us the fact that the different delay components are commutative for a given control period from the control point of view. There are upper bounds for the delays of communication time, computation time, and the jitter of control period, respectively, for a given performance criteria. In an extreme case, the fieldbus control system will become unstable if the delays of these times and the jitter exceed the upper bound.

To overcome the bad effects caused by the delays of communication time, computation time, and the jitter of control period, and to guarantee the control performance of FCS, two approaches can be considered. One is to reduce the dominant delays, i.e. computation times in the control period. The other is to design specific control algorithms for FCS. These two topics will be studied in Chapter 5.

Chapter 4

Evaluation of Fieldbuses

4.1 Introduction

At present, there are more than 50 different names (Thomesse, 1999) of available fieldbuses such as FOUNDATION Fieldbus, Profibus, Interbus, WorldFIP, Lonworks, CAN, DeviceNet, SwiftNet, P-net and so on. Although IEC approved the IEC 61158 proposal for an international fieldbus standard at the end of 1999, the standard actually comprises eight different fieldbus protocols. So the fieldbus confusion to users still exists. On the other hand, the fieldbus technology and the products based on it are getting more and more applications in the instruments and control systems of different areas because of their unique advantages. Thus many issues arisen from fieldbuses need to be researched. One of them is the evaluation when users face so many types of so-called fieldbuses.

Some comparison and evaluation work for the performance of five fieldbuses that are FOUNDATION Fieldbus, Profibus-DP, DeviceNet, WorldFIP and SwiftNet has been done by Crowder (1996). And the following evaluation indices: bus capacity, scan rate, bus and segment length, cycle error and jitter were used. Durante and Valenzano (1999) studied how the protocol parameters in IEC 61158 fieldbus affect the average response time and the throughput for acyclic data exchanges. The differences of data link layer of Profibus and IEC fieldbus were examined by Vitturi (2000) and two indices of evaluation including the cycle time and medium access efficiency were analyzed. There are two lacks for the evaluation research of fieldbuses. One is that the evaluation is only done from one or two aspects of fieldbus, most on its time performance, lacking complete evaluation. The other is that the evaluation is considered less with the user standpoint and the specific requirements of fieldbus, most only from communication point of view as done in general network research. In this chapter, we will first analyze the requirements of fieldbuses. Then we will propose a hierarchy performance criteria based on the hierarchy model of the fieldbus protocol. A complete set of evaluation metrics, both quantitative and qualitative, will be given while considering the special requirements of fieldbus. Finally, The FOUNDATION Fieldbus (Fieldbus Foundation, 1996) will be studied through experiments and simulation as a case. The measurement technique for the dynamic indices of fieldbuses is discussed. In order to support users's judgment, the evaluation results of the static indices for several fieldbuses are showed, and some guidelines for fieldbus selection are given.

4.2 Requirement analysis of fieldbus

Typically, fieldbuses are used in distributed applications to monitor and control a manufacturing process such as automotive industry, textile machinery, factory automation, semiconductor fabrication, electronics manufacturing, food and beverage, chemical processing, and so on. To effectively evaluate fieldbuses, the requirements for fieldbus should be understood clearly. There are some special requirements that fieldbus must satisfy. We will discuss these requirements from both the communication and control application points of view in the following.

4.2.1 Requirement of data communication

Industrial information needing to communicate between devices can be put into the following three classes, depending on the nature, size and the timing requirement of the information (Cena at al., 1995).

- Urgent data

The data related to important events that happen at unpredictable times, such as alarm data, often need to be transmitted as soon as possible. Such data are called urgent data.

- Cyclic data

The data that are produced and need to be transmitted periodically are considered as cyclic data. Cyclic data transmission via fieldbus requires real time communication. This means that data delivery delay must be bounded. For example, the data from sensor to controller and from controller to actuator in a closed-control-loop based on fieldbus are all cyclic data.

- Sophisticated operation data

This class of data that generally has a larger size of message than the two classes of data above is related to the sophisticated operation of user, such as parameter changing and program downloading as well as device diagnosis and etc.

These three classes of data have different features and different communication requirements. The cyclic data are deterministic because the transmission time of them can be known in advance. However, the urgent data and sophisticated operation data are random because their producing times are unpredictable. The cyclic data and urgent data all have the requirements with tight time constraints that are called real time or time critical requirements. The transmission deadlines for these two classes of data should be guaranteed. The sophisticated operation data has no particular timing constraints, but high throughput is often desirable. The data communication requirements of manufacturing and control applications have been summarized by Thomesse and Chavez (1999).

4.2.2 Environmental requirement

The environment in which fieldbuses have to operate is very different with that of LAN, which is usually installed in the office building, whereas the environment of fieldbus is generally the outdoor field of factory or manufacturing where noise usually, and even explosive atmosphere exist. And the physical area of the environment is usually very large. The cost and complexity of cabling become important because of the long distance often more than 1 kilometer. Due to these reasons, the physical layer of fieldbus protocol often needs to satisfy the following requirements:

- High reliability

The reliability is always the first position issue for any device with which the fieldbus interlinks due to the safety of factory production. To improve the reliability and fault tolerance of the system, some measures for error detection, noise immunity and redundancy technique should be taken.

- Long distance transmission

Long distance transmission is necessary for fieldbus to connect devices in a large factory area. This requirement is often achieved by reducing the data transmission rate and using special signaling method.

- Intrinsic safety (MTL Instruments Group, 2001)

Intrinsic safety is a technique that is based on the principle of restricting the energy available in hazardous area circuits such that any sparks or hot surfaces that may occur as a result of electrical faults are too weak to cause ignition. To achieve this requirement, low-power technique in electronic circuits is necessary.

- Bus powered

Bus powered means that the same cable with a pair of twisted-wires is adopted for transmitting data and also for supplying power to the device at the same time. Thus the cost of cable installation and maintenance can be reduced effectively.

4.2.3 Requirements of distributed functions

As we all know, one of the important advantages for a network system is its ability of distributed calculation. Such ability normally behaves in the form of configurable and different function blocks, which are implemented by the various algorithms embedded or downloaded in fieldbus devices, such as the PID control algorithm in a fieldbus transmitter. Different types of fieldbus devices contain different function blocks. So a user can make use of all the function blocks by configuration to work together via fieldbus in order to achieve the whole function of the system. Therefore, distributed user function is an important requirement when the fieldbus is considered as the backbone of a control system, not only a signal transmission network.

4.2.4 Requirements of interoperability

Interoperability concept has been discussed detailed by Thomesse (1999). Here it represents the application interoperability. Interoperability is a property that ensures that several items of equipment from different vendors can be integrated into a single network to ‘work’ together for a given goal, or the capability of one piece of equipment to work within an existing system, without requiring custom equipment or tool development. Interoperability requirement is higher than the openness and the interchangeability that allows failed equipment could be replaced by similar equipment, even from another vendor.

Of course, there are also other general requirements for fieldbus, such as low cost and maintainability and etc., apart from those discussed above.

4.3 Fieldbus performance criteria

4.3.1 General items of evaluation

To evaluate the fieldbuses, some general items must be considered the same way as we often do, such as reliability, cost and maintainability, etc.

4.3.2 Evaluation criteria of physical layer

- Maximum data rate (MDR)

  The maximum data rate is closely related with the type of medium the fieldbus uses. The maximum data rate should be given for the twisted pair wire, which is the most common medium of the fieldbus. Some fieldbuses use a fixed data rate, others allow a range of data rates.
- Maximum segment length (MSL)

  The length is inverse proportional to the data rate. It also depends on the medium of the fieldbus. Thus the maximum segment length must be specified with a given data rate and the medium of twisted pair.
- Maximum number of segment nodes (MSN)

  This index reflects the node capacity of a filedbus, i.e. the maximum number of devices that can be connected on one fieldbus segment without any repeater.
- Topology (TOPLG)

  More topology means more flexibility of fieldbus wiring. The topology of bus and tree are the most common for fieldbuses.
- Medium types (MT)

  The communication media include twisted pair wire, coaxial cable, fiber optic cable, RF, power line, IR and others. Some fieldbuses only support single medium, others support more.

Table 4.1: Characteristics of typical medium (Reza, 1994)

Characteristics	Twisted paire	Radio	Power line	Coaxial	Infrared	Fiber optics
Typical range, meters	1-1000	50-10000	10-5000	10-10000	0.5-30	10-10000
Typical data rate, kbit/s	0.3-2000	1.2-9.6	0.06-10000	300-10000	0.05-20	1-100000
Relative node cost, US$	\$10-\$30	\$50-\$100	\$50-\$150	\$30-\$50	\$20-\$75	\$75-\$200
Typical installation cost	Low	---	None-low	Medium	----	Medium-high

- Intrinsic safety support (ISS)
- Bus powered support (BPS)
- Transmission mode (TM)

  Signal transmission in the physical layer can be implemented in two modes: synchronous or asynchronous. Synchronous transmission needs to use components designed and developed specially. It has high efficiency, but at high cost. Asynchronous transmission can be implemented cheaply and easily, but with less efficiency.

4.3.3 Evaluation criteria of data link layer

- Cyclic data support (CDS)

  This is to evaluate the medium access mechanism used in data link layer whether it supports hard real time communication for cyclic data. The centralized scheduling method and pass token method meet such requirement, but the CSMA method without any modification doesn't. The cycle jitter index can be used to evaluate this performance quantitatively. Generally speaking, cycle jitter is acceptable if real time transmission is supported.
- Message priority support (MPS)

  This is also to evaluate the medium access mechanism. Priority is a mechanism to transmit messages according to their importance.
- Large size message support (LMS)

  This index reflects the ability of a fieldbus to support the sophisticated operation. If the sophisticated operation data consist of several separate short frames, the transmission efficiency will be low.
- Efficiency of message (frame)

  Efficiency of message E_M is defined the ratio of user data length to the total length of message transmitted as indicated in Eq. (4.1). Different types of messages for different user functions have various efficiencies.

$$E_M = \frac{N_{UD}}{N_{SUM}} \quad , \tag{4.1}$$

where $$N_{SUM} = \sum_{i=1}^{n} N_{Li} + N_{UD} \quad , \tag{4.2}$$

and where N_{UD} is the length of user data for an application, N_{Li} is the length of protocol control data or overhead in sublayer i of fieldbus protocol profile. n is the number of fieldbus protocol layers and depends on types of fieldbuses. The length is the number of bytes or bits.
- Synchronization support (SS)

  Synchronization is an ability to achieve synchronous data and operation among different devices on fieldbus.
- Fault tolerance

4.3.4 Evaluation criteria of application layer

- Response time

  This index specifies the time needed from data requirement to data reception. It is not static, but varies with practical applications. It should be guaranteed to a limited time interval for time critical data.
- Period jitter

  This index specifies the variance of period for cyclic data transfer via the fieldbus.
- Acknowledged service support (ASS)

  For important data transmission, acknowledged service is necessary to be most reliable.

- Efficiency of fieldbus

  The efficiency of fieldbus E_F is defined by Eq. (4.3).

$$E_F = \frac{T_{DM}}{T_{SUM}} \quad , \tag{4.3}$$

where $T_{SUM} = T_{DM} + T_{OM} + T_W$, (4.4)

and where T_{DM} is the time used to send the user data message, T_{OM} is the time spent to send the necessary operation messages such as management or control messages to complete the transmission of the user data message, T_W is the sum of all waiting times from the start of data requirement to the end of data reception. The efficiency of fieldbus is closely related with both the fieldbus protocol and the practical application configuration. Hence, it is an index representing the working efficiency of fieldbus.

4.3.5 Evaluation criteria of user layer

- Distributed functions support (DFS)

  The distributed functions here specifically refer to the configurable functions implemented by the algorithms distributed in fieldbus devices. The ability to support distributed functions is necessary when the distribution calculation is required in the fieldbus system. This index is a measure for the ability of fieldbus protocol provided for device user.

- Interoperability support (IS)

  Interoperability is one of the most important issues concerned by device end user. This ability can be specified by either the protocol profile or a separate application profile. Apart from the specific profile, the interoperability is also guaranteed through the specific test carried out by a specified organization. More details about interoperability are discussed by Thomesse (1999).

4.4 Evaluation techniques

4.4.1 Evaluation techniques

The complete set of evaluation items and indices for fieldbus performances is determined as above from various aspects of fieldbus. Sequentially, we need to find answers for these evaluation items and indices. Some of them are easy, but some are difficult. Some only need Yes or No. Some must be calculated comprehensively based on experimental data or a specific model. Some are static and only depend on the protocol itself. Others are dynamic and vary with the design and running of the application case. The static indices can be known directly from the protocol specification or by some analysis and calculation.

But for the dynamic indices, they are difficult to calculate because they vary with the application configuration. To get the dynamic indices, there are two main approaches: measurement techniques and simulation techniques. The measurement techniques can be applied only when a real system or a prototype of it is available. And some platform or tool of hardware and software are needed. It costs more money and time. However, the measurement results are realistic, reliable and believable. The simulation techniques need to define a model that represents the behavior of a system. Then the model is solved to get the performance indices of the model, which in turn are the estimate of the indices of the system The simulation model allows us to study the system in each phase of its life cycle, or rather in each design, development, set-up and modification stage (Gelenbe et al., 1999). The correctness and accuracy of the model are important and must be verified by the measurement techniques. In this chapter, we will use both the measurement technique and the simulation technique.

4.4.2 The static evaluation indices of some fieldbuses

We summarized the static indices of some fieldbuses in Table 4.2.

Table 4.2: Static indices of fieldbuses

Name / Indices	FOUNDATION Fieldbus (H1)	Profibus -DP	Profibus -PA	FIP	DeviceNet	CAN	LON
MDR(kbps)	31.25	12000	31.25	2500	500	1000	1250
MSL(m) @(kbps)	1900 @31.25	1200 @93.75	1900 @31.25	1900 @31.25	500 @125	10000 @5	2700 @78
MSN	32	32	32	32	64	110	64
TOPLG	Bus, Tree	Bus	Bus, Tree	Bus, Tree	Bus	Bus	Bus, Free
MT	Wire	Wire, Fiber	Wire	Wire	Wire, Fiber	Wire, Fiber	Wire, Fiber, etc.
ISS	Yes	No	Yes	Yes	No	No	Yes
BPS	Yes	No	Yes	Yes	No	No	Yes
TM	Syn.	Asyn.	Syn.	Syn.	Syn.	Syn.	Syn.
CDS	Yes	Yes	Yes	Yes	Yes	No	No
MPS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
LMS	Yes	Yes	Yes	Yes	No	No	Yes
SS	Yes	Yes	Yes	Yes	No	No	No
ASS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DFS	Yes	No	Yes	Yes	No	No	Yes
IS	Yes	Yes	Yes	Yes	Yes	No	Yes

4.4.3 Calculation of some evaluation indices

- Efficiency of message (frame)

In order to calculate the efficiency of message, we need to analyze the structure of message (frame) in details. The analysis can be carried out manually or using specific tools. Through such analysis, the length of user data and the length of protocol overhead can be determined. Then the efficiency of message can be calculated using Eqs. (4.1) and (4.2). Following is an example of calculating the efficiency of message for CAN fieldbus. For the CAN fieldbus (Navet and Song, 2001),

$$N_{UD} = 8d \ ,$$

$$N_{SUM} = 47 + 8d + \left\lfloor \frac{34 + 8d - 1}{4} \right\rfloor , \tag{4.5}$$

where *d* is the number of user data bytes, and ranges from 1 to 8. Therefore, the efficiency of message for the CAN is calculated according to Eq. (4.1). It ranges from 12.3% to 47.4% depending on the length of user data. The longer the user data, the higher the efficiency of message.

The structure of a P-Net frame is shown in Figure 4.1 (Tovar, et al, 2002). With the number of the user data and the overhead shown in Figure 4.1, the efficiency of message for the P-Net is easily calculated to be 0% - 91.3%. It is concluded that the P-Net can have a high efficiency of message even more than 90%.

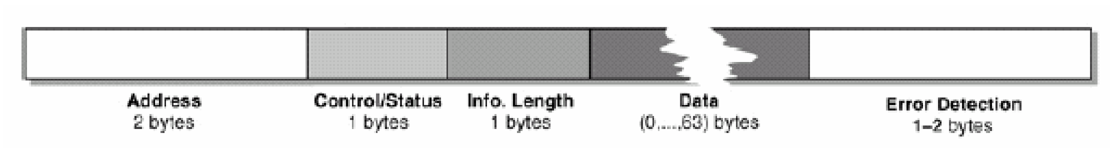


Figure 4.1: Structure of a P-Net frame

The LonWorks fieldbus has a complex structure of message PDUs (Protocol Data Units) with 7 layers and several address formats. Therefore, it needs detailed analysis in order to calculate its efficiency of message. The details for the structure of Lontalk message PDUs are shown by Dietmar (2001). The efficiency of message for FOUNDATION Fieldbus depends on the types of message, periodic and non-periodic. It is calculated in Section 4.5.

- Response time

Media access delay is the main factor in determining the network response time, given a network topology (Reza, 1994). Therefore, different fieldbuses using different medium access methods will have different response times. We discuss the calculation problem of the response times for the CAN fieldbus as an example.

For the CAN fieldbus, the worst-case response time with a reliable medium is studied by Navet et al. (2000). The worst-case response time R_m, which is defined as the longest time between the start of the task queuing *m* and the latest time that the message arrives to the destination processor(s), must be bounded for each frame by D_m, otherwise the timing constraint can not be guaranteed and the set of messages of the application is said to be non-schedulable.

To calculate R_m, as the transmission time C_m and the jitter J_m can be upper bounded, we just have to compute the maximum time needed by the message to gain the arbitration (termed the "interference" time). A message m can be delayed by higher priority messages and by a lower priority message that has already obtained the bus (this time denoted as B_m is the transmission time of the biggest lower priority message). Thus, we have (Tindell et al., 1995)

$$R_m = C_m + J_m + I_m , \tag{4.6}$$

where I_m is the interference time, i.e., B_m plus the longest time that all higher priority messages can occupy the bus. I_m is calculated as follows:

$$I_m^n = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{I_m^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil C_j \quad , \text{(n=1, 2, ······)} \tag{4.7}$$

where τ_{bit} is the bit time, and *hp(m)* is the set of messages of higher priority than *m*. C_j is the transmission time of message *j* with d_j data bytes.

$$C_j = \left(47 + 8d_j + \frac{34 + 8d_j}{4} \right) \tau_{bit} \quad , \tag{4.8}$$

where 47 is the size of the fixed-form bit fields of the CAN frame and $((34+8d_j)/4)$ is the maximum number of stuff bits. I_m is computed starting with $I_m^0 = 0$ until it converges.

The formula of cycle time for the FIP and Profibus are given by Cena at al. (1995). The response time of the WorldFIP has been analyzed by Tovar and Vasques (2001).

4.5 Experimental study

A FOUNDATION Fieldbus is a type of computation distribution fieldbuses. The FOUNDATION Fieldbus computation distribution is implemented by s set of standardized function blocks, which consist of s set of data and algorithms that are intended to provide a given functionality to the user application. These blocks are linked together by local or remote communication relationship.

Here the tank level control system by the DeltaV (trademark of Rosemount Co.) FCS is studied again as an application case. The monitor software and hardware of NI (National Instruments Inc.) for the FOUNDATION Fieldbus was used as a monitor and an analysis tool. There are three different FOUNDATION Fieldbus nodes, which are pressure difference transmitter, temperature transmitter and valve positioner. In this case, two AI's, one PID and one AO function blocks were configured. More details about the FCS and the monitor software tool are described in Chapter 3. Here we only present the experimental results investigating the following indices: efficiency of message, response time and efficiency of fieldbus.

- Efficiency of message of the FOUNDATION Fieldbus

According to Equation (4.1), we calculate the efficiency of message E_M of FOUNDATION Fieldbus.

$$N_{SUM} = N_{PH} + N_{FDL} + N_{FAS} + N_{FMS} + N_{UD} \, , \tag{4.9}$$

where N_{PH}, N_{FDL}, N_{FAS}, N_{FMS} are the lengths of protocol control data in physical layer, data link layer, FAS sublayer and FMS sublayer of the FOUNDATION Fieldbus protocol, respectively. Based on the analysis of the FOUNDATION Fieldbus protocol and the experimental results obtained by the monitor tool, E_M can be calculated as shown in Table 4.3. In this case, three typical messages: cyclic, short acyclic and long acyclic messages are selected to calculate the average efficiency of the FOUNDATION Fieldbus message.

Table 4.3: Results of the FOUNDATION Fieldbus message efficiency

	Cyclic Message	Short acyclic Message	Long acyclic Message
N_{UD}	5	41	99
N_{FMS}	7	4	4
N_{FAS}	1	1	1
N_{FDL}	6	9	9
N_{PH}	4	4	4
N_{SUM}	23	59	117
E_M	0.217	0.695	0.846
Average E_M		0.586	

- Response time of the FOUNDATION Fieldbus

Figure 4.2 and 4.3 are the experimental results of response time for cyclic and acyclic messages of the FOUNDATION Fieldbus, respectively. The response time of cyclic message is much shorter than that of acyclic message, and its variance is limited in a narrow time interval less than 0.5 ms. But the response time of acyclic message is long and has a large undetermined time variance.

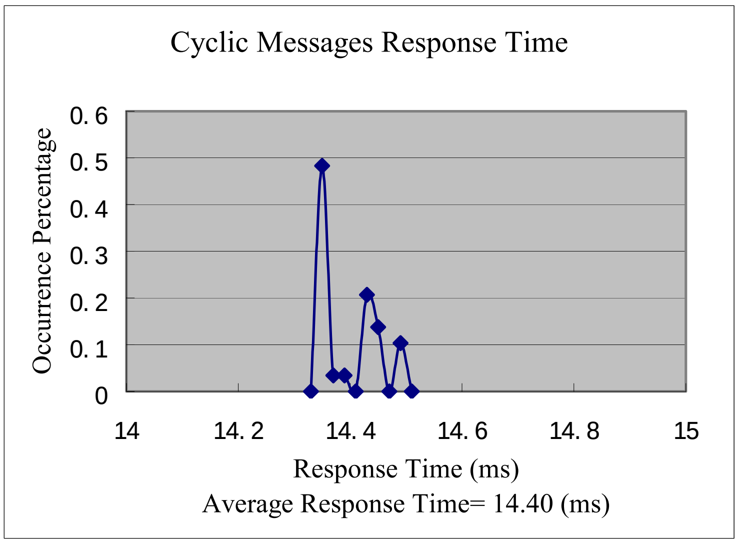


Figure 4.2: Response time of the FOUNDATION Fieldbus cyclic message

- Efficiency of the FOUNDATION Fieldbus

In the FOUNDATION Fieldbus, two different medium access methods are used for two types of messages: pre-scheduling method for periodic messages and centralized pass token method for aperiodic messages. By analyzing the mechanisms of these two methods, the efficiency of the FOUNDATION Fieldbus for cyclic messages should be higher than that for acyclic messages. The experimental results shown in Figures 4.4 and 4.5 prove this fact.

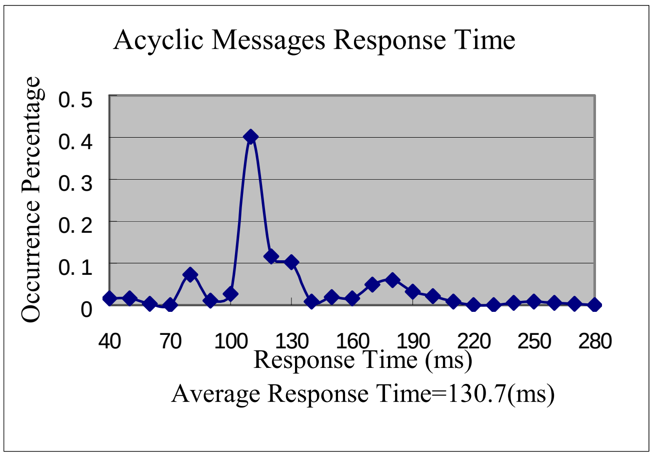


Figure 4.3: Response time of the FOUNDATION Fieldbus acyclic message

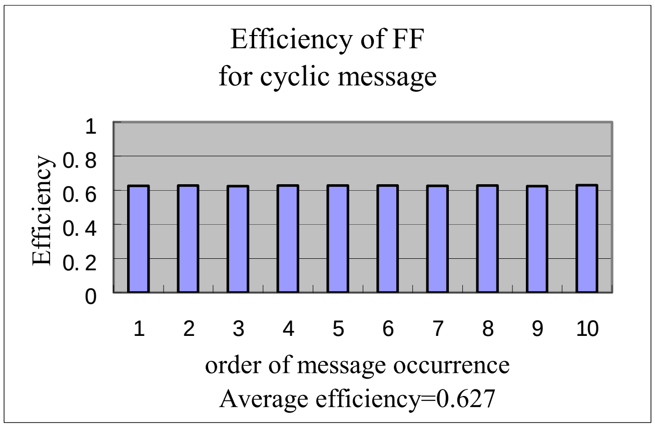


Figure 4.4: Efficiency of the FOUNDATION Fieldbus for cyclic message

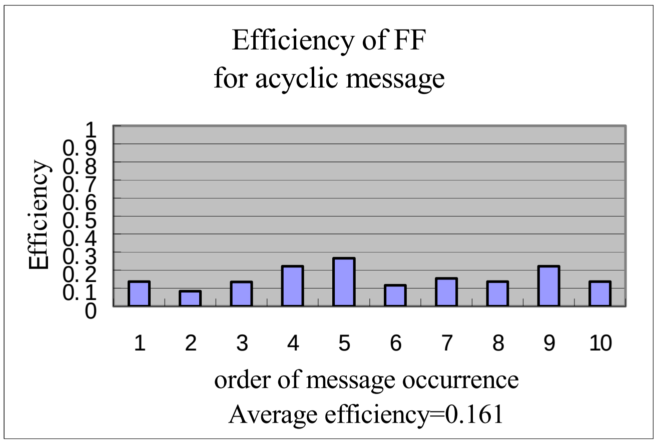


Figure 4.5: Efficiency of the FOUNDATION Fieldbus for acyclic message.

4.6 Simulation study

When some cases such as large variation of traffic load on fieldbus are difficult or impossible to be implemented in a practical FCS, the simulation technique can be used. Figures 4-6 to 4-9 are the simulation results for the response time of acyclic message of the FOUNDATION Fieldbus control system by using the simulation models, which are proposed in Section 6.6.3. For the simulation examples shown in Figures 4-6 to 4-9, the control period is 0.5 second, and the parameter *T* represents the fieldbus traffic load. The smaller the *T* is, the higher the fieldbus traffic load is.

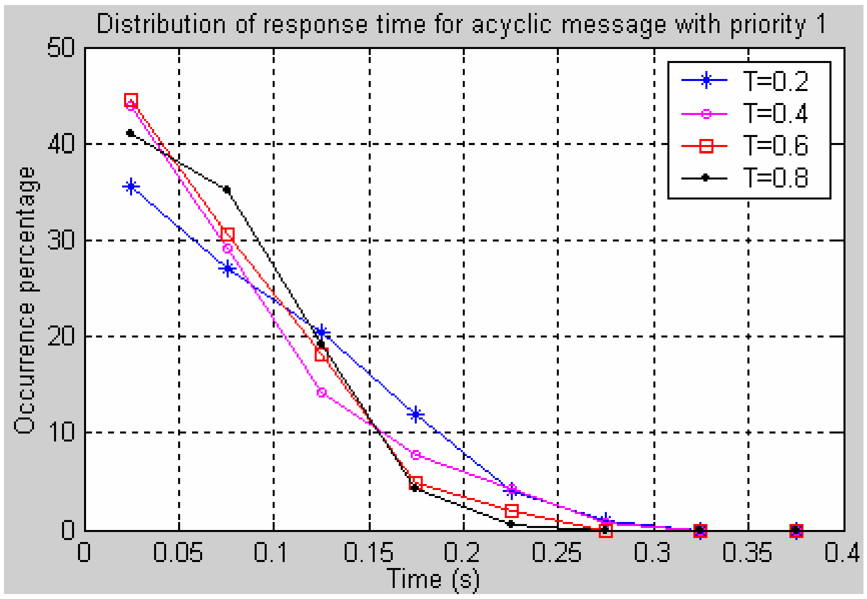


Figure 4.6: Distribution of response time for acyclic message with priority 1 when fieldbus traffic load is low

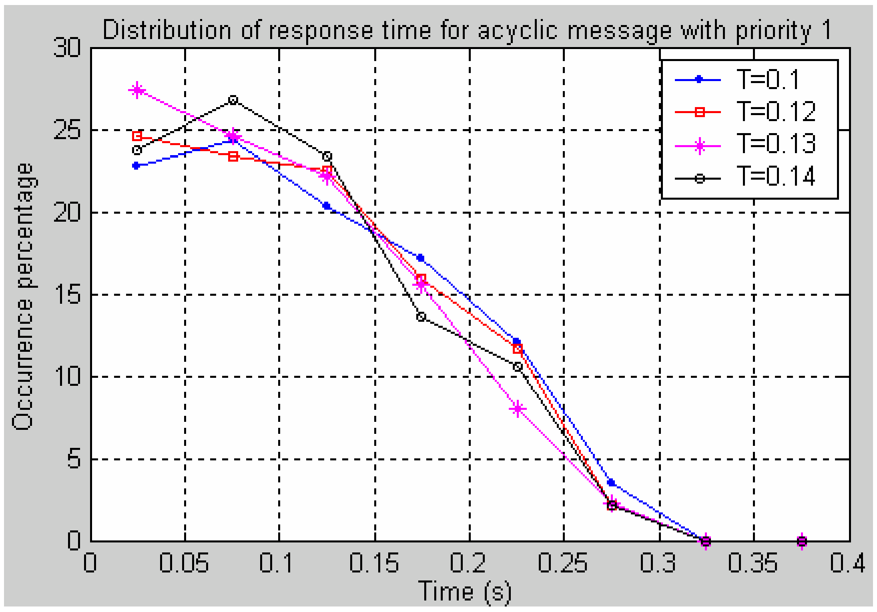


Figure 4.7: Distribution of response time for acyclic message with priority 1 when fieldbus traffic load is high

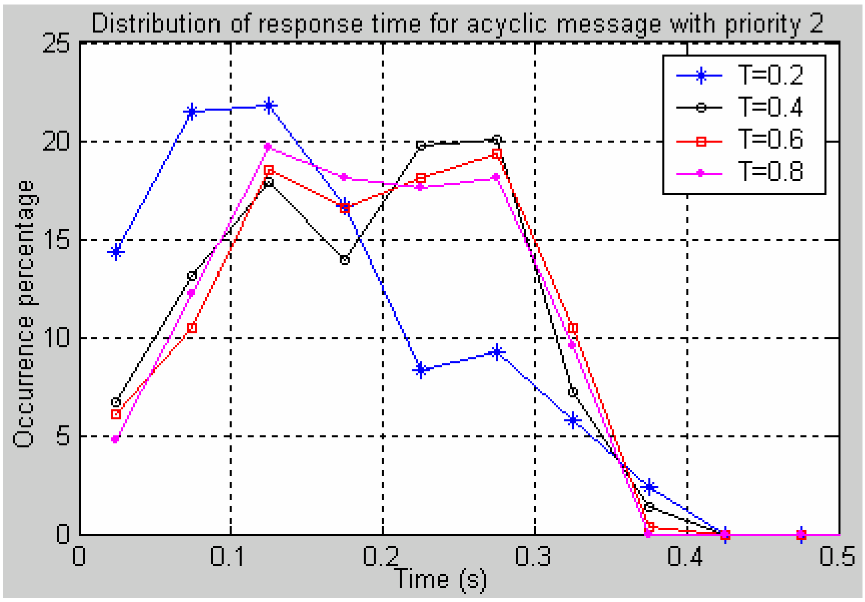


Figure 4.8: Distribution of response time for acyclic message with priority 2 when fieldbus traffic load is low

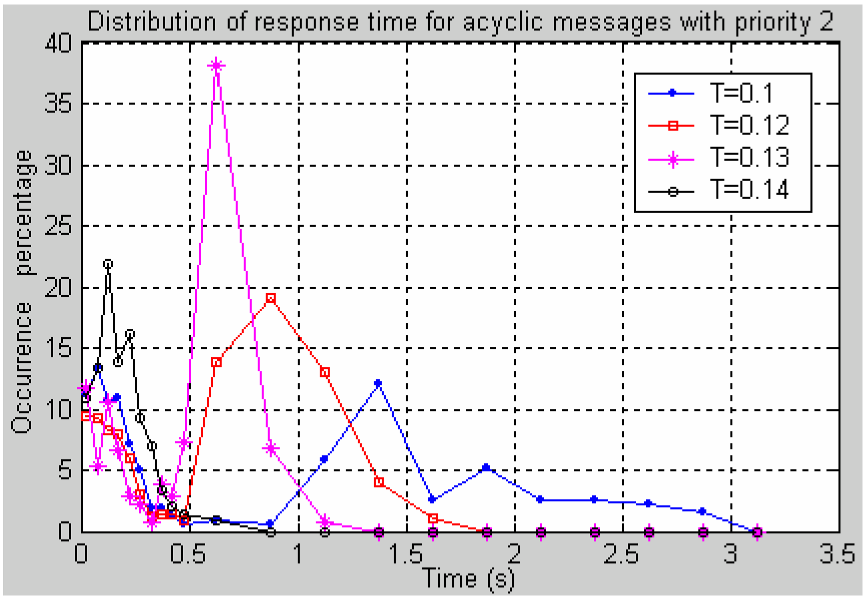


Figure 4.9: Distribution of response time for acyclic message with priority 2 when fieldbus traffic load is high

Figures 4-6 to 4-9 show the following facts:

- The priority mechanism for the acyclic message transfer on the FOUNDATION Fieldbus works effectively. The acyclic message with priority 1 has shorter response time, within the control period 0.5 second, than that with priority 2 even when the fieldbus traffic is high.
- High traffic load leads to longer response time for acyclic messages, especially for those with low priority 2.

The simulation results conform to the expectation based on the analysis for the FOUNDATION Fieldbus.

4.7 Selection of fieldbuses

How to select a fieldbus is an important issue for a particular distributed control application. Evaluation criteria and procedure must be considered. For selection of a fieldbus, Figure 4.10 shows a general procedure developed by Gruhler (Farsi and Barbosa, 2000).

The procedure shown in Figure 4.10 begins on one hand with the specification of the application requirements and on the other hand with the gathering and evaluation of data of the fieldbus systems. Then the requirements of applications are classified according to the importance of each item of requirements, and a weighing factor is assigned to each item or criterion. The characteristics of the evaluated fieldbus systems are compared with the requirements. Finally, a choice is made by assessing the results of the classification and comparison. By Farsi and Barbosa (2000) the fieldbus system selection criteria are divided into two categories: technical data and strategic criteria. However, concrete criteria are not given. Based on the performance indices described in Section 4.3 and comprehensive consideration, we propose a set of detailed indices for technical data and strategic criteria which are listed in Table 4.4. Most indices can be obtained from the data sheet of the evaluated fieldbus systems. The other indices can be estimated by using the measurement and simulation evaluation techniques. In order to select a fieldbus for a particular application, the technical data and the strategic criteria shown in Table 4.4 can be tailored and added to meet the application requirements.

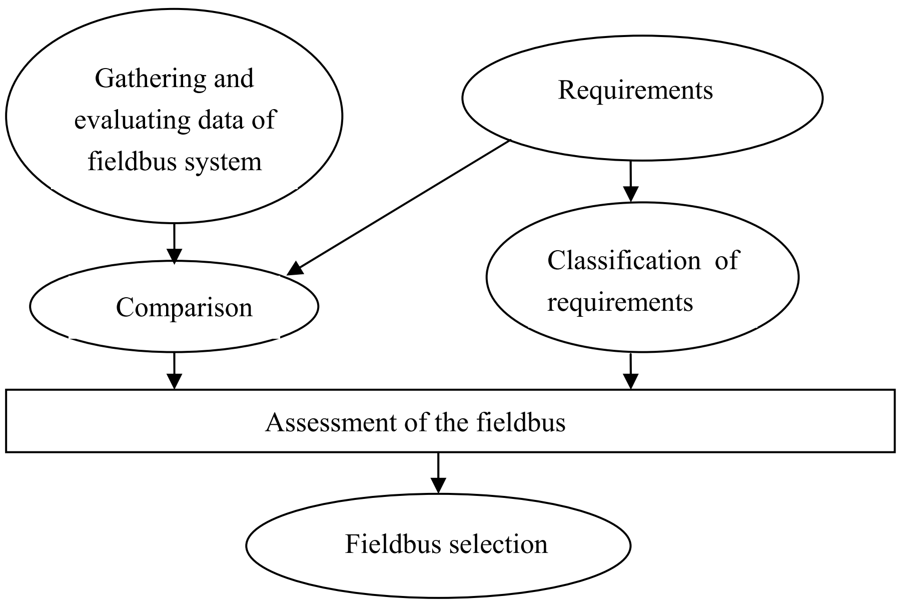

Gathering and evaluating data of fieldbus system
Requirements
Comparison
Classification of requirements
Assessment of the fieldbus
Fieldbus selection

Figure 4.10: General procedure for selecting a fieldbus system

Table 4.4: Technical data and strategic criteria for selecting a fieldbus system

No.	Technical data	Strategic criteria
1	Geographical and scale criteria	Standards
	● Maximum segment length ● Maximum number of segment nodes ● Expandability (length & nodes) ● Topology	● Openness ● Interoperability test ● Conformance test ● Distributed function support
2	Time criteria	Availability
	● Maximum data rate ● Response time ● Period jitter	● Components ● Software ● Services ● Number of vendors
3	Protocol service criteria	Cost
	● Cyclic data support ● Message priority support ● Large size message support ● Synchronization support ● Acknowledged service support	● Purchase ● Configuration ● Installation ● Maintenance
4	Efficiency criteria	Reliability
	● Efficiency of message (frame) ● Efficiency of fieldbus	● Fault detection and control ● Fault tolerance
5	Management criteria	Easy to use
	● Network management ● System management	● Configuration support ● Development support
6	Environmental criteria	Flexibility
	● Intrinsical safety support ● Weather proof	● Multiple fieldbus medium ● Interconnectivity

In the procedure for selecting a fieldbus system, the essential issue is to assign the weight factors to various criteria based on their importance according to the application requirements. The hypothetical example to illustrate how to select a fieldbus is given by Farsi and Barbosa (2000).

4.8 Summary

The requirements of data communication and installation environment for fieldbuses are analyzed from the user standpoint. After that, a complete set of evaluation indices is proposed according to the layer architecture of fieldbus protocol model. The measurement and simulation evaluation techniques are discussed. The calculation formula of the efficiency of message and the response time for the commonly used CAN fieldbus are shown as an example. The static indices of 6 types of fieldbuses are also evaluted. As a case study, the experimental results for the FOUNDATION Fieldbus are presented, including the indices of efficiency of message, response time, and efficiency of fieldbus. The results indicate that the FOUNDATION Fieldbus cyclic messages have shorter and guaranteed response time, higher efficiency of fieldbus than the FOUNDATION Fieldbus acyclic messages. Also, the simulation results show that the priority mechanism for the acyclic message transfer on the FOUNDATION Fieldbus works effectively. The experimental and simulation conclusions conform to those obtained by analyzing the mechanism of the FOUNDATION Fieldbus protocol. Finally, a general procedure for selecting a fieldbus system is presented, and a complete set of detailed indices for the technical data and the strategic criteria are proposed.

Chapter 5

Control Algorithms of Fieldbus Control System

5.1 Introduction

Based on the analysis in Chapter 3, we have known that the delays of computation and communication and the jitter of control period in the FCS are two important factors to degrade the control performance of the FCS. The objective of this chapter is to design effective control algorithms to overcome the effects of delays and jitter on the control performance. In order to understand the characteristics of the delays and the jitter, we first model delays and jitter in the FCS from the control point of view. Then control algorithms are designed to overcome the effects of delays and jitter on the control performance of the FCS. A modified PID control algorithm and a single predictive control algorithm will be proposed. The effectiveness of the control algorithms designed is verified through simulation. A way to implement the control algorithms in the FCS is another issue to deserve research especially to shorten the control period of the FCS. A parallel computation method will be proposed in this chapter.

5.2 Characteristics and modeling of delays in FCS

From the control point of view, we can depict the interaction of any controlled plant and the FCS as shown Figure 5.1 despite the types of fieldbus systems.

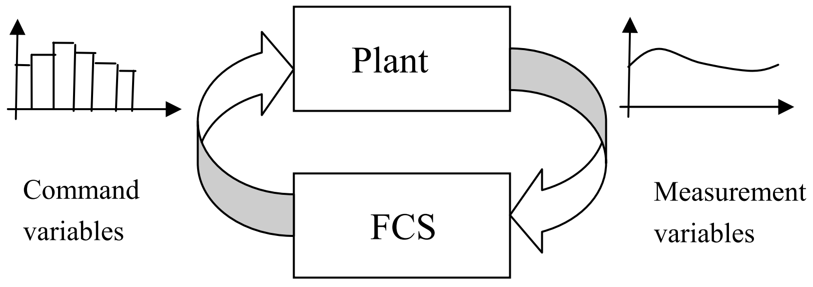


Figure 5.1: Interaction of plant and FCS

The system shown in Figure 5.1 has the following characteristics:

- The controlled plant is continuous and can be described as a continuous model.
- The FCS is discrete and can be described as a discrete model. The delays and jitter caused by computation and communication in the FCS just exist between the inputs and the outputs of the FCS (i.e. measurement variables and command variables).
- The output of the plant is sampled periodically, and the sampling results are the

measurement variables as the inputs of the FCS, which are time-discrete.

- The output of the FCS is also sent periodically to the plant as its input. It is kept to be constant during an interval of the period. It behaves as a zero-order hold.

From the function perspective, the FCS can be divided into three components including sensor or transmitter, controller, and the actuator, which are shown in Figure 5.2.

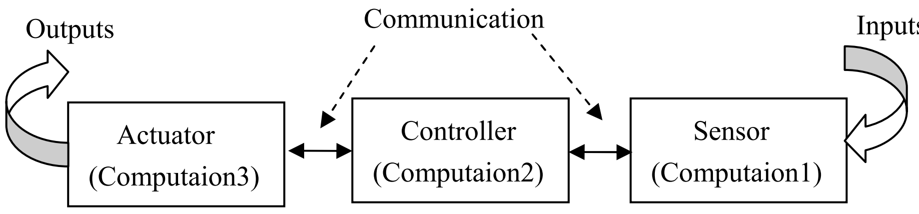
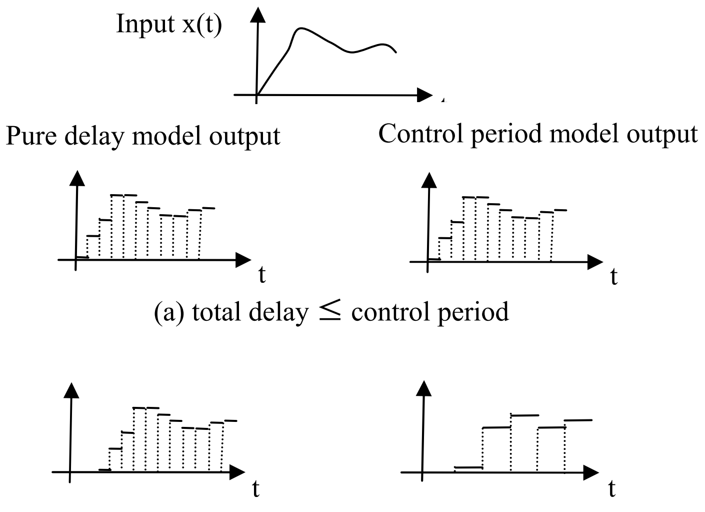


Figure 5.2: Composition of control loop in FCS

It is well known that the communication and the computation in the FCS lead to delays in the control loop, and delays degrade the control performance. In order to analyze the effects of the delays on the control performance, it is necessary to model the delays from the control point of view. However, a resulting problem is how to model the delays in the FCS. It is the most common way for many researchers (Sawamura et al., 2002; Walsh et al., 2002; Beldiman and Walsh, 2000; Park and Huh, 2000) to model the delays as a pure delay unit $e^{-\tau_t S}$. But this model is not appropriate for most cases of FCS as described below. Therefore, we use the control period model based on the true operation way of FCS. There are differences between the two ways of the pure delay model and the control period model. Figure 5.3 shows that the two models give different outputs for the same input and delays. From Figure 5.3, it is not difficult to find that the two models are equivalent only if the total delay is less than the control period.



(b) total delay > control period

Figure 5.3: Differences between pure delay model and control period model

It is important to distinguish the difference because different models lead to different designing problems for the control algorithms of FCS. For the control period model, the problem is how to select an appropriate control period and how to compensate the period jitter. For the pure delay model, the problem is how to compensate the pure delay. In the pure delay model, the delays and the control period are independent as shown in Figure 5.3. However, in the control period model, the delays are included in the control period.

The pure delay model cannot model the following three phenomena, which may occur:

(a) Within a control period, there may be no sample data arriving (when congestion occur);

(b) Within a control period, more than one sample data may arrive (after congestion);

(c) Data loss may occur within several consecutive control periods (when interference occurs).

But for the control period model, the period is dependent with the delay. Therefore the above first two phenomena are impossible to occur. The congestion can be reflected by a large control period jitter. The data loss can be also included in the control period model, it will behave as an extremely large period jitter, which is larger several times than the normal period. However, for the theoretical analysis, the pure delay model is easier than the control period model because of the period jitter. When using the simulation method, there is no difficulty for both models, but the control period model coincides more accurately with industrial practice of FCS. If the output of the plant is sampled at an instant, then communicated, and computed, and only one the computed result or the command variable is outputted to the plant during each period, the delays caused by the computations and the communications are all included in the control period each time, and can be represented by the control period model. In such occasion, the jitter will cause the control period to be variable. The following assumptions are made to formalize the delay problem.

- The plant is a linear time invariant system.
- The output of the plant is sampled periodically. Also the command to the plant is outputted periodically by the actuator. One sample of the sensor always corresponds to one command of the actuator. The sample of the sensor and the command of the actuator keep unchangeable in one period. They only change depending on the control algorithm at the sampling or control instants. The sampling and the output are synchronized. The period may be variable randomly, or period jitter exists. However, the jitter is bounded.
- The delays caused by the computations and communications in the FCS are randomly varying. All time delays are independent.
- The total time delay is always less than one sampling or control period, which is variable and large enough to include any total time delay.

According to the conclusion of Chapter 3, the distribution changes of computation or communication times in one control period have no effect on the control performance for a given control period. Only the control period determines the control performance for a given control algorithm regardless of the distribution of computation or communication times. Therefore, the model of the FCS can be described in Figure 5.4 and the following Eqs. (5.1) to (5.8). The plant in Figure 5.4 includes the continuous-time parts of the sensor and the actuator.

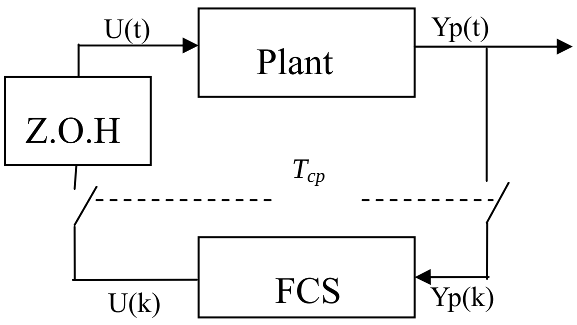


Figure 5.4: Equivalent model of FCS

The plant dynamics are:

$$\dot{x}_p(t) = A_p x_p(t) + B_p u(t) \quad , \tag{5.1}$$

$$y_p(t) = C_p x_p(t) \quad , \tag{5.2}$$

where $x_p \in \Re^{n_p}$ is the plant state, u and y_p are the input and the output of the plant, respectively. Discretizing the plant yields

$$x_p(t+T_{cp}) = \phi_p(T_{cp}) x_p(t) + \int_t^{t+T_{cp}} \phi_p(t+T_{cp}-\tau) B_p d\tau \cdot u(t), \tag{5.3}$$

$$y_p(t) = C_p x_p(t), \tag{5.4}$$

where ϕ_p is known as the state transition matrix of A_p, and $\phi_p(T_{cp}) = e^{A_p T_{cp}}$. T_{cp} is the control period and time-varying. However, its variation should have a boundary that is achieved by the design of FCSs. Therefore, according to the analysis in Chapter 3, T_{cp} can be modeled as

$$T_{cp} = T_{\min} + J_r Rand(t), \tag{5.5}$$

where T_{min} is the minimum of the control period, J_r is a parameter for the jitter range of the control period, and *Rand(t)* is a random variable with time that ranges from 0 to 1.

Because of the control period variation, the dynamics of the FCS is described as:

$$x_c(t+T_{cp}) = A_c x_c(t) - B_c y_p(t) \quad , \tag{5.6}$$

$$u(t) = C_c x_c(t) \quad , \tag{5.7}$$

where $x_c \in \Re^{m_p}$ is the digital controller state in FCS. All computation algorithms distributed in fieldbus devices can be combined into one algorithm, which is the digital controller algorithm. The control algorithm determines the numerical relationship between $Y_p(k)$ and *U(k)* shown in Figure 5.4 during each period. The effects of time delays caused by computation and communication on the control performance can be reflected by the sampling or control period. Therefore, all of the research results for the sampling period in the digital control system can be utilized to analyze and design the control algorithms for the FCS. The specific feature of period problem in the FCS is the existence of jitter. The consequent problem is that the effects on the control performance caused by the jitter of period should be considered and compensated during the design of control algorithm.

According to the analysis in Chapter 3, the normal operation condition for fieldbus loop control is that

$$T_{cp} > D_t \ , \tag{5.8}$$

where T_{cp} is the control period, D_t is the total delay including all of computation delays and communication delays during each period. The vacant sampling and multiple output commands will occur if the condition is not satisfied. The result is that the control performance will be degraded. Therefore, the condition shown in Eq. (5.8) should be satisfied during the design of the FCS.

The period should be selected by considering three factors: the characteristics of the plant, the control algorithm, and the arising computations and communications. According to the sampling theorem, a fast plant requires a shorter sampling period. Table 5.1 gives typical sampling periods for few process variables.

Table 5.1: Choice of sampling period for digital control systems

Type of variable	Sampling period (seconds)
Flow rate	1-3
Level	5-10
Pressure	1-5
Temperature	10-45

Rules of thumb for choosing the sampling period for a digital PID regulator indicate that there is a significant difference between PI and PID regulators. Significant shorter sampling period are required for controllers with derivative action (Chidambaram, 2002). If the characteristics of the plant and the control algorithm tell us what period should be satisfied, computations and communications in the FCS can tell us what period can be implemented. For the design of the FCS, one of the objectives is to implement a shorter period. For the design of the control algorithm, one of the objectives is to tolerate a long period and its long jitter for a given plant without significant degradation of the control performance.

5.3 Design of PID control algorithm

Through the analysis in the Chapter 3, we have known that the jitter worsens the control performance, even the stability of the fieldbus control loop. When the jitter becomes larger than a certain value, the system will become unstable. The first major consideration of any control system design is, of course, stability. Therefore, we need to find a new control algorithm to overcome the bad effect of the jitter in the FCS on the control performance and the stability of the control loop.

Considering the low computation capability of the fieldbus devices in the FCS, based on the principle of simplicity and common use, it is natural to think of the conventional three-mode PID control algorithm. It has been the most common control algorithm used in the industry for a long time due to its effectiveness, robustness and simplicity.

5.3.1 A modified PID control algorithm

To research the possibility by revising the PID control algorithm to compensate for the jitter of the control period, the PID algorithm is analyzed below in detail.

$$C_o(t_k) = K_c[e(t_k) + \frac{1}{T_i}\sum_{i=1}^{k} e(t_i)(\Delta t_i) + T_d \frac{e(t_k) - e(t_{k-1})}{\Delta t_k}] \ , \tag{5.9}$$

where

$$e(t_k) = Sp(t_k) - Pv(t_k) \ , \qquad \Delta t_i = t_i - t_{i-1} (i = 1, ..., k) \ ,$$

and where $Sp(t_k)$ and $Pv(t_k)$ are the setpoint and process variables of the controller, respectively, $C_o(t_k)$ is the output of the controller, and K_c, T_i, and T_d are the proportional gain, integration time and derivative time of the controller, respectively. Δt_i or Δt_k is the sampling period of the *i* th or *k* th sampling instance, which is variable with time on the fieldbus. The first term of Eq. (5.9), i.e. the proportional action, has no relation with the variable sampling period, whereas the second and third terms, i.e. the integral and derivative actions, are both affected by the variable sampling period. Substituting Δt_i and Δt_k in Eq. (5.9) with a constant Δt, when the practical sampling period increases, i.e. $\Delta t_i \rangle \Delta t$, the integral action becomes a partial integral action. The larger Δt_i is, the weaker the integral action is. This is depicted in Figure 5.5. The complete integration should be the area under the *e(t)* curve. However, under the condition mentioned above, the integral action in Eq. (5.9) only represents the shadow area under the *e(t)* curve, i.e. a partial integration. Otherwise, if $\Delta t_i \langle \Delta t$, the extra integration will occur. For the derivative action, if substituting Δt_k with Δt , the effect is reverse to the integral action. Accordingly, by just letting Δt_i take the minimum value of all variable periods, i.e. $\Delta t_i = T_{min}$, a modified PID control algorithm is obtained:

$$C_o(t_k) = C_o(t_{k-1}) + \Delta C_o(t_k) \ , \tag{5.10}$$

where $\Delta C_o(t_k) = K_c[e(t_k) - e(t_{k-1})] + K_I e(t_k) + K_D[e(t_k) - 2e(t_{k-1}) + e(t_{k-2})]$,

and $K_I = \frac{K_c T_{\min}}{T_i}$; $K_D = \frac{K_c T_d}{T_{\min}}$.

T_{min} can be estimated based on the Eq. (3.7) in Chapter 3 and by neglecting random waiting time for communication. Experimental measurement is a better way if possible.

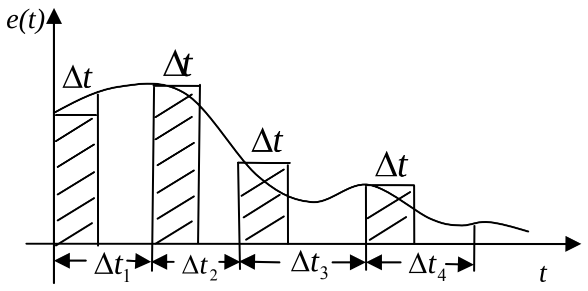


Figure 5.5: Partial integral action

When the control period jitter becomes larger, the integral action becomes weaker due to the effect of the partial integration. This result is equivalent to K_I in Eq. (5.10) decreasing with the increment of the jitter. Therefore, this effect gives the PID control algorithm a self-adaptability to cope with the increment of the control period jitter. This ability will be helpful to maintain the stability of the fieldbus control loop when the loop suffers from the increment of the control period jitter or the loop time delay. On the other hand, the derivative term in Eq. (5.10) should be deleted or tuned through the parameter T_d to have only a small portion of the integral action for keeping the control stability. This observation will be validated by the following simulation study because of the difficulty of theoretical analysis.

5.3.2 Simulation Study

5.3.2.1 Purpose and environment

We used the simulation approach to research the issues of the fieldbus control system because it is difficult to use a theoretical analysis due to its complexity. In order to evaluate the effects of fieldbus-induced delay on the performance of the control loop, especially on the control stability of the loop, and to validate the modified PID control algorithm in Eq. (5.10), we developed simulation models and software for a typical fieldbus control loop in process control by using the Simulink and Stateflow of MATLAB as simulation tools. The simulation software structure is presented in Chapter 6.

In this study, a first-order model with dead time in the form of transfer function is taken as the plant model because the dynamic responses of most plants in process control can be mathematically described by such a model.

To simulate the fieldbus devices including the transmitter, the controller, and the actuator, a state flow machine is developed, which consists of simulation models of fieldbus devices and scheduling. Structure of simulation software of the fieldbus control loop is depicted by Figure 5.6. The model of fieldbus devices and scheduling has three groups of inputs. The first group's inputs are the model time-parameters for communication and algorithm execution such as the data communication time, algorithm execution time, and period jitter range. The second group's inputs are the parameters of the PID controller algorithm. The third group's inputs are the input of the second part of the transmitter and the set-point of the controller. The model has two outputs, namely the state values indicating the real time states of the fieldbus control loop and the output of the actuator. The period jitter caused by the variable times of communication and algorithm execution is implemented by a random function, which generates random numbers whose values are uniformly distributed at the interval (0, J_r). All initial parameters of models and algorithms are included in Figure 5.6.

In the simulation, the control period is time-varying, and the sampling period in the transmitter is also time-varying. Therefore, how to determine the sampling time instance in the transmitter is an issue for implementation in practice. This issue can be resolved by establishing communication and synchronization between the transmitter and the actuator.

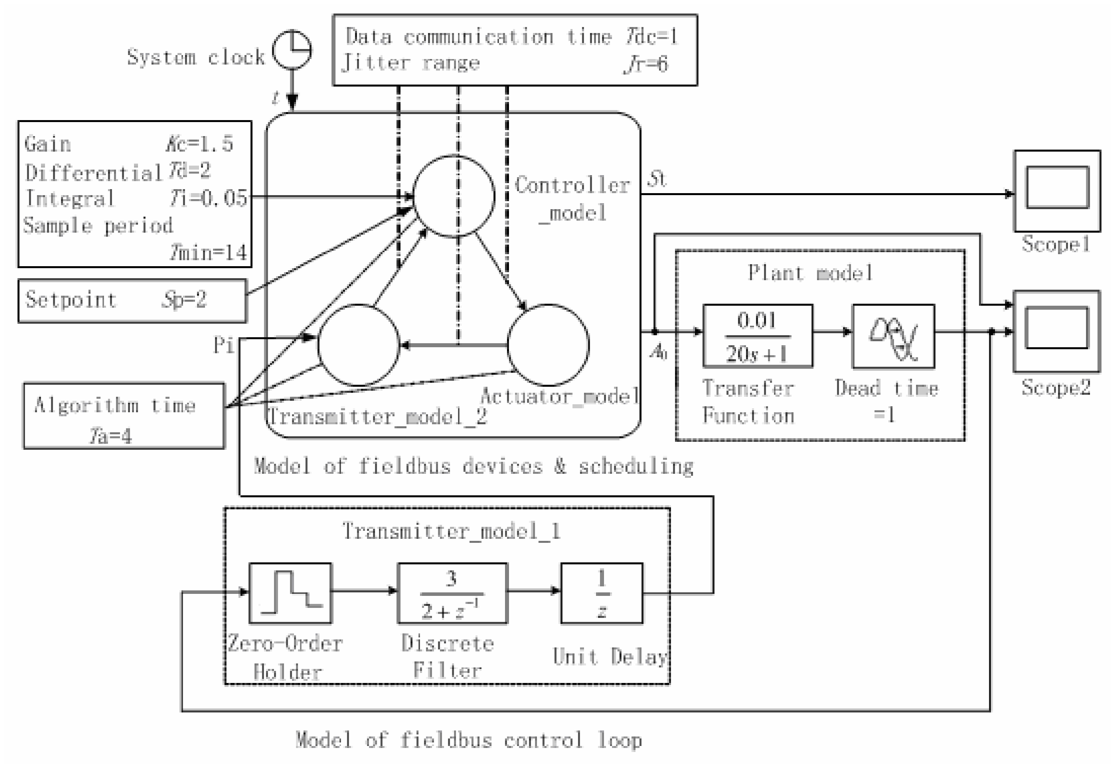


Figure 5.6: Structure of simulation software of fieldbus control loop

5.3.2.2 Results and discussions

The state changes in the state flow machine and the step response of the plant output are shown in Figures 5.7 and 5.8. The parameters of the PID controller were tuned in the interactive mode. Jitters of sampling and control periods are shown in Figure 5.7. Figure 5.8 shows the acceptable control performance under a specified jitter range by the tuned PID parameters.

Figures 5.9 and 5.10 show the simulation results for the case of extending the range of the control period jitter to simulate the long communication delay while keeping other parameters unchanged. We can see clearly that the control period becomes longer as a result of extending the range of the control period jitter to simulate a long communication delay, but the control loop becomes more stable, i.e. the larger the delay, the more stable the loop.

State value=1 Transmitter state

State value=2 Controller state

State value=3 Actuator state

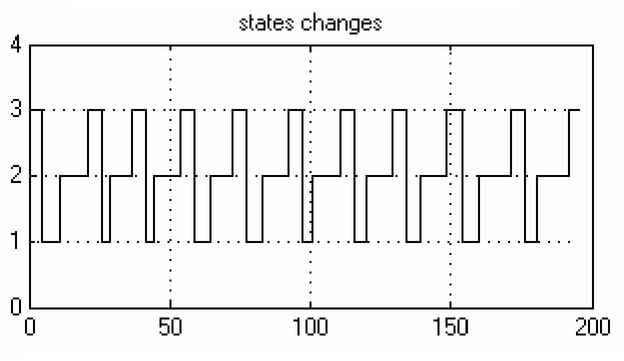


Figure 5.7: State changes of fieldbus control loop

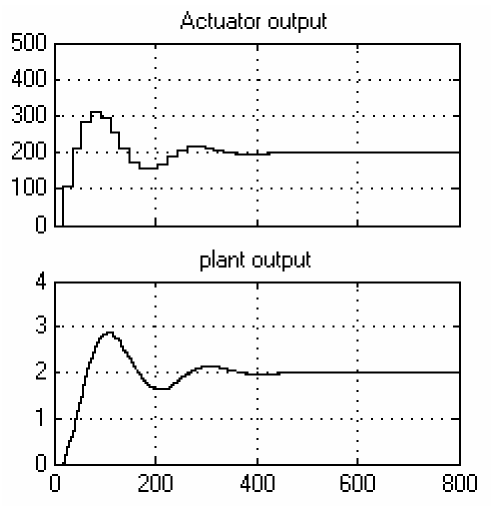


Figure 5.8: Step response of fieldbus control loop

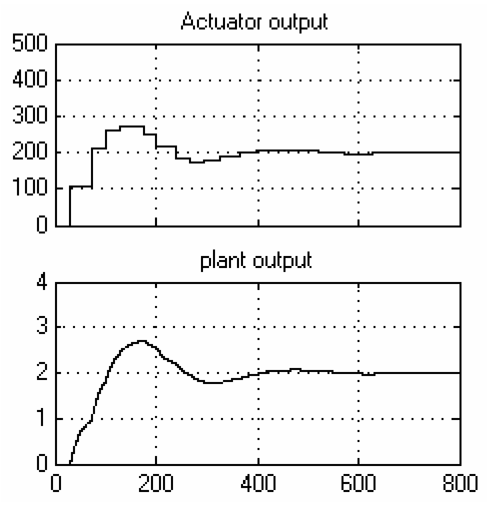


Figure 5.9: Step response after jitter range changes from 6 to 20

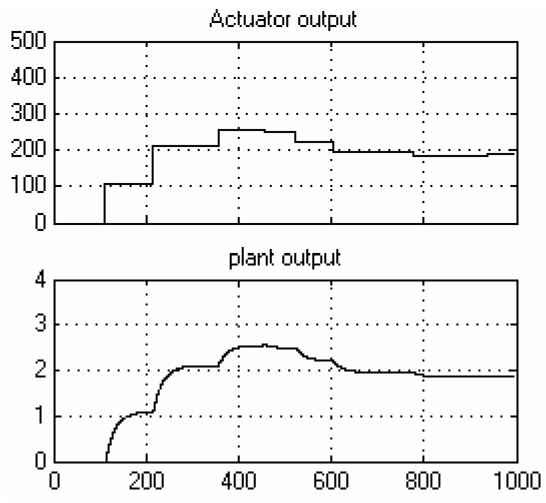


Figure 5.10: Step response after jitter range changes from 6 to 100

The simulation results shown in Figures 5.9 and 5.10 prove that the modified PID control algorithm proposed in Eq. (5.10) works well for common processes having the first-order model with dead time. This implies that the stability of the fieldbus control loop can be assured under the unpredictable and varying time delays caused by the fieldbus. Therefore, the objective of designing the control algorithm for the fieldbus control system is achieved. The delay caused by communication and algorithm execution can be compensated by using the modified PID control algorithm because of the effect of partial integration. Hence, it is possible for a fieldbus control loop to maintain control stability while suffering from long communication delays by using this PID control algorithm. This is an easy and practical approach to maintaining stability when designing a controller for a fieldbus control loop.

Figures 5.11 and 5.12 show the simulation results when the dead time of the plant model is increased while the other parameters are kept unchanged.

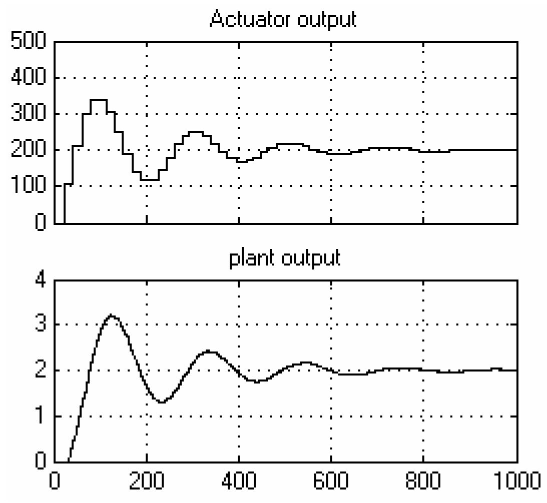


Figure 5.11: Step response after plant delay changes from 1 to 10

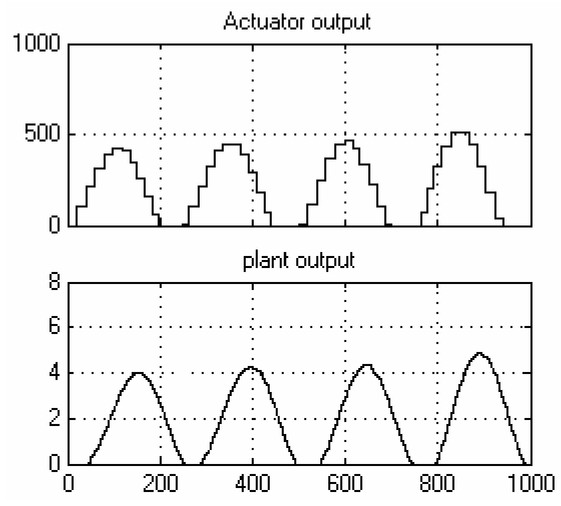


Figure 5.12: Step response after plant delay changes from 1 to 25

The simulation results show the larger the delay, the worse the stability of the loop. We can see that stability of the control loop becomes worse when the delay of the plant model becomes longer and that the control period remains unchanged. The simulation results indicate that the stability of the loop cannot be maintained because there is not any effect of partial integration to compensate for the plant delay in this case. The results also tell us the fact that the delay of the plant and the delay caused by the communication and computation are not commutative.

5.3.2.3 Performance comparison of a conventional PID and the modified PID control algorithms

Figures 5.13 and 5.14 show the comparison results between a conventional PID and the modified PID control algorithms when the control period increases. For the conventional PID control algorithm, the performance criteria was kept almost unchangeable during the initial increase of the control period, but the performance criteria became worse drastically when the control period is larger than a certain value. For the modified PID control algorithm, the performance criteria can be kept to be satisfied during a wide range of the control period.

Figures 5.15 and 5.16 are the comparison results between a conventional PID and the modified PID control algorithms when the jitter range of the control period increases. The results show that the modified PID control algorithm indeed has a better control performance than the conventional PID algorithm even when the jitter range of the control period increases.

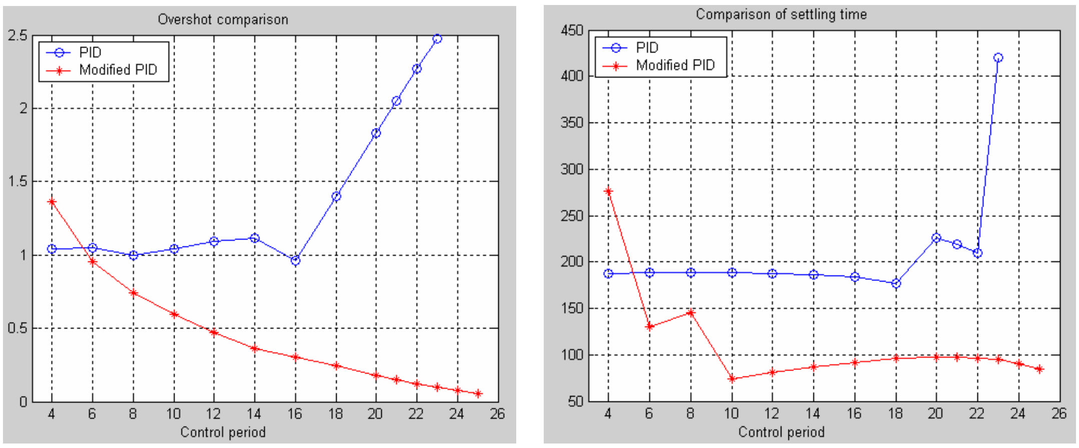


Figure 5.13: Comparisons of overshoot and settling time for PID and the modified PID control algorithms when control period increases.

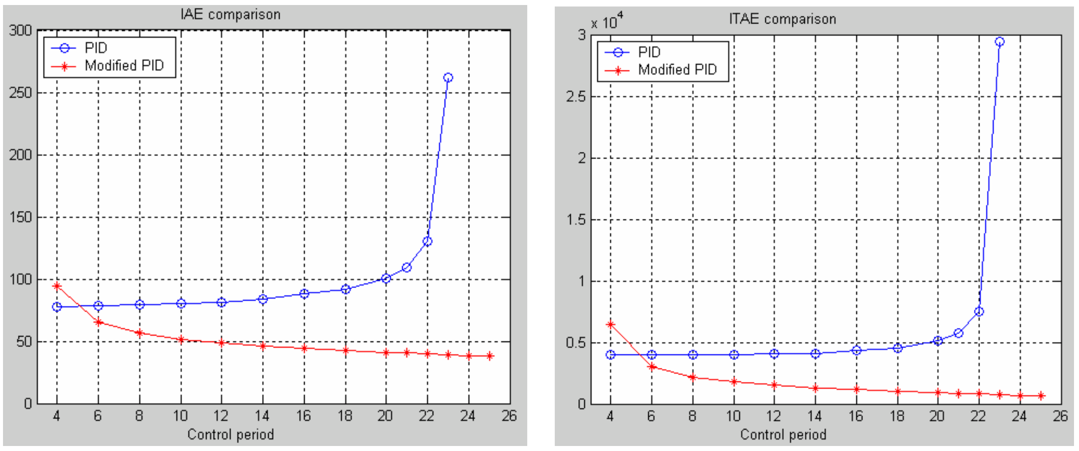


Figure 5.14: Comparisons of IAE and ITAE for PID and the modified PID control algorithms when control period increases.

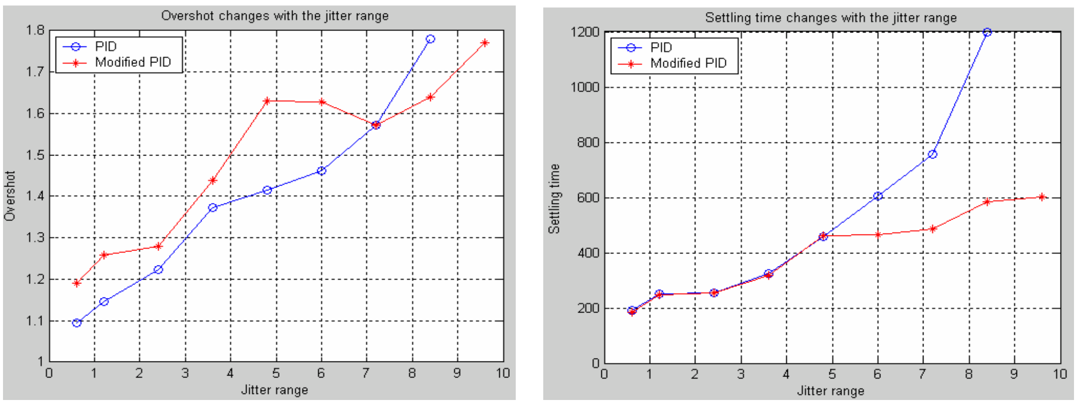
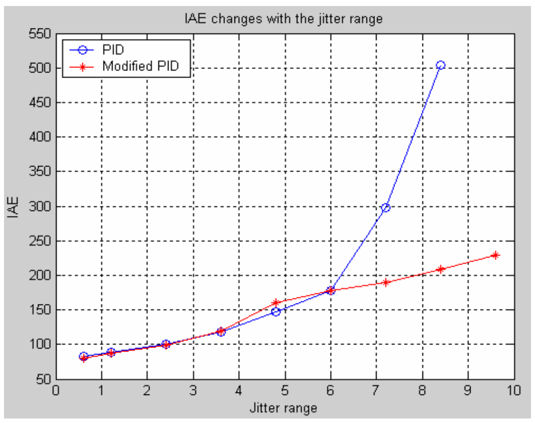


Figure 5.15: Comparisons of overshoot and settling time for PID and the modified PID control algorithms when the jitter range of the control period increases.



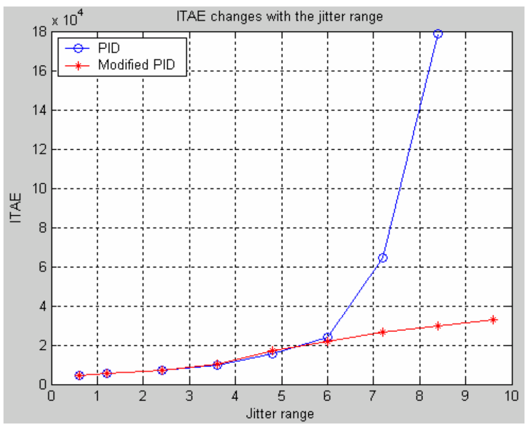


Figure 5.16: Comparisons of IAE and ITAE for PID and the modified PID control algorithms when the jitter range of the control period increases.

5.4 Predictive control

Over the last two decades, predictive control has proven to be a very successful controller design strategy, both in theory and practice. The main reason for this acceptance is that it provides high performance controllers that can be applied to difficult high-order and multivariable processes. The use of finite-impulse response models and finite-step response models, which are easily obtained for open loop stable processes, partly explains the acceptance in the process industry.

Predictive control was pioneered simultaneously by Richalet et al. (1978) and Cutler and Ramaker (1980). Predictive control is a class of control strategies based on the explicit use of a process model to generate the predicted values of the output at future time instants, which are then used to compute a sequence of control moves that optimize the future behavior of a plant. The explicit use of a model is the main difference between predictive control and the classical PID controller. Its advantage is that the behavior of the controller can be studied in detail, simulations can be done and performance can be evaluated. One of the drawbacks is the need of an appropriate model of the process. Another drawback is that although the resulting control law is easy to implement and requires little computations, its derivation is more complex than that of the PID. In the following, a plant model is first described. Then a single predictive control algorithm that is suitable to fieldbus system is derived.

5.4.1 Plant model

Assume a linear, stable, SISO plant with finite gain, described to sufficient accuracy by a finite impulse response (FIR) model *H (i)* with *N* coefficients:

$$\hat{Y}(k) = \sum_{i=1}^{N} H(i)u(k-i), \tag{5.11}$$

where $\hat{Y}(k)$ is the output of the plant, *u(k)* is the input of the plant, *k* represents kT_{cp} (T_{cp} is control period.). *N* is the number of the products, which depends on the control period T_{cp}. NT_{cp} corresponds to the settling time of the plant. Cutler (1980) suggests that N is between 20 – 60 if T_s is selected appropriately. Alternatively, the plant dynamics can be described by a finite step response *S(i)* with N coefficients:

$$\hat{Y}(k) = \sum_{i=1}^{N-1} S(i)\Delta u(k-i) + S(N)u(k-N). \tag{5.12}$$

Here the output $\hat{Y}(k)$ depends on past values of changes in the input $\Delta u(k-i) = u(k-i) - u(k-i-1)$ and the absolute input *N* samples in the past. The plant impulse *H(i)* and step response *S(i)* are related as follows:

$$S(i) = \sum_{j=1}^{i} H(j) \ , \ H(i) = S(i) - S(i-1). \tag{5.13}$$

The prediction of the future output needs to be corrected because of the discrepancy existing between the real plant and the model. The difference between the current measured *Y(k)* and the prediction $\hat{Y}(k)$ can be used to correct the prediction of the future output at *j* th instant. The corrected prediction is as follows:

$$Y_c(k+j) = \hat{Y}(k+j) + Y(k) - \hat{Y}(k). \tag{5.14}$$

Rawlings et al. (1994) showed that this method of feedback correction removed steady-state offset, providing a theoretical support for its use.

5.4.2 Predictive control algorithm

In order to make the control algorithm as simple as possible, only a single future control signal is calculated (control horizon=1) by optimizing a given objective function to keep the plant as close as possible to the reference input. The objective function usually takes the form of a quadratic function of the error predictions. The control effort is also included in the objective function in most cases. However, the solution to this problem is complex and needs more computations, which is not suitable for the fieldbus controller with low computation ability. If we only consider the error prediction at *P* th instant during the prediction horizon in the objective function, the solution will become simple and easy to implement in the fieldbus controller. If the reference input is Y_s *(k+j)*, the error prediction of the output response shown in Eq. (5-15) can be used to be the objective function.

$$e(j) = Y_s(k+j) - Y_c(k+j). \tag{5-15}$$

According to Eqs. (5.12) to (5.15), assume $Y_s(k+j)=Y_s(k)$,

$$e(j) = e_0 - \sum_{i=1}^{j} S(i)\Delta u(k+j-i). \tag{5.16}$$

and

$$e_0 = Y_s(k) - Y(k) - \sum_{i=1}^{N-j} [S(j+i) - S(i)]\Delta u(k-i)\,, \tag{5.17}$$

Note that $\Delta u(k+j) = 0, j > 0$, when control horizon=1.

$$e(P) = Y_s(k) - Y(k) - \sum_{i=1}^{N-P} [S(P+i) - S(i)]\Delta u(k-i) - S(P)\Delta u(k)\,. \tag{5.18}$$

To minimize square of $e(P)$ or let $e(P) = 0$, the optimal control signal is obtained as shown in Eq.(5.19).

$$\Delta u(k) = S^{-1}(P)[Y_s(k) - Y(k) - \sum_{i=1}^{N-P} B_s(i)\Delta u(k-i)]\,, \tag{5.19}$$

where $B_s(i) = S(P+i) - S(i)$. This is the single predictive control algorithm. For the SISO system, $S(P)$ is a scalar variable. Therefore, the single predictive control algorithm is unnecessary to calculate a matrix and easy to implement for the fieldbus devices. The single future control point at P th instant is known as the coincidence point, which is the only tuning parameter in the control algorithm. At this future point the error is brought to zero. The stability analysis of the control algorithm can be referenced to the study of Thomas (1997).

5.4.3 Simulation results

Figure 5.17 is the simulation comparison of three control algorithms including PID, modified PID, and predictive control algorithms for the same first-order plant model with dead time. It is shown that both PID and modified PID control algorithms have the same performance, the predictive control algorithms has better performance when there is no control period jitter. However, both predictive and modified PID control algorithms have better performance than the PID control algorithm when a period jitter exists.

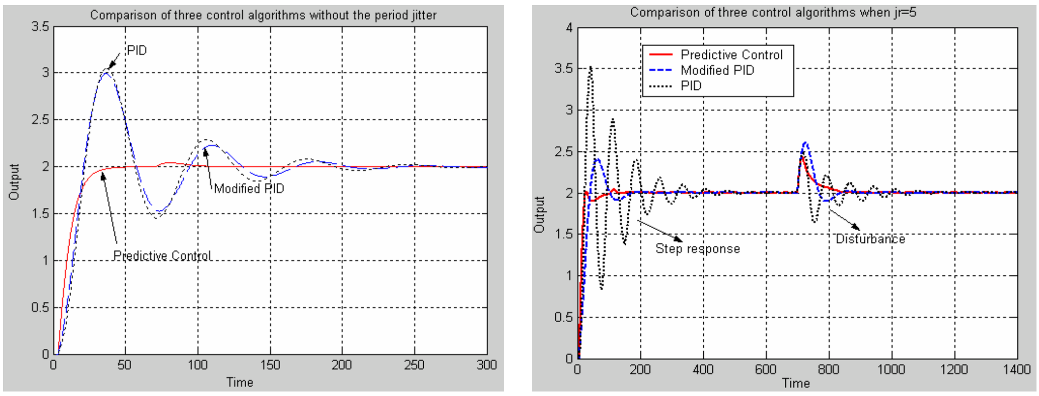


Figure 5.17: Comparisons of the PID, modified PID, and predictive control algorithms without jitter and with jitter

Figures 5.18and 5.19 are the comparison results between the modified PID and the single predictive control algorithms when the control period increases for the same first-order plant model with dead time. It is shown that the single predictive control algorithm has far better control performance than the modified PID algorithm. The single predictive control is one of the optimal control algorithms.

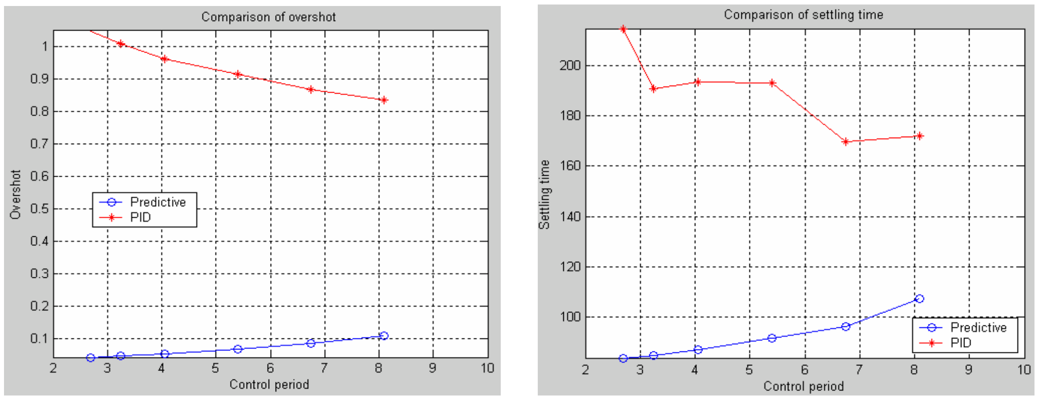


Figure 5.18: Comparisons of overshoot and settling time for the modified PID and the predictive control algorithms when control period increases.

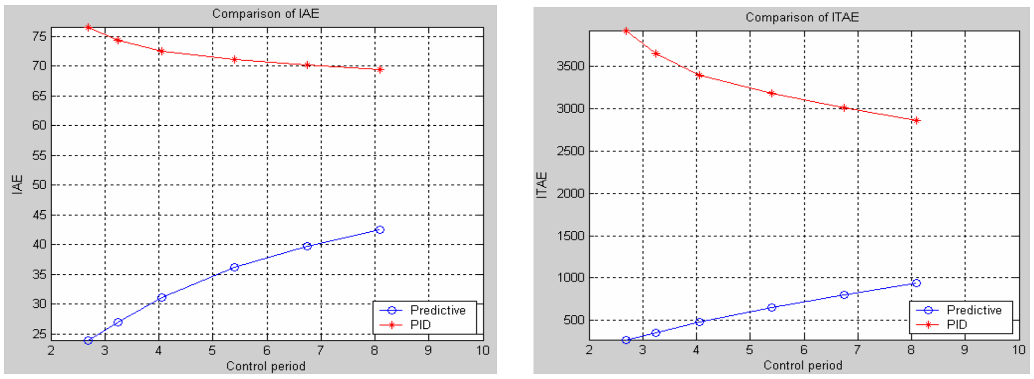


Figure 5.19: Comparisons of IAE and ITAE for the modified PID and the predictive control algorithms when control period increases.

5.5 Parallel computation of control algorithm

A shorter control period means a better control performance for the same plant and the same control algorithm in the FCS. There are two straightforward approaches to shorten the control period in the FCS. One is to enhance the computing power of the microprocessors used in the fieldbus devices, but there are limits on cost and power consumption. The other is to optimize the algorithms used for function blocks, but its effect is limited because most of the work has already been done. We propose a new approach, in which a parallel computation structure of function blocks is used to shorten the execution time of function blocks. For this approach, we are motivated by the fact that the computations in the fieldbus devices in a control loop are serial during each period. While one fieldbus device is in computation, the other is idle. The execution time of function block should be shortened if we can make best use of the computation capability of each fieldbus device all the time during each period. As shown in Figure 5.20, we design two parallel computation lines for a PID fieldbus control loop. Two outputs of the two lines are used for the outputs of two consecutive periods of the control algorithm, respectively. Therefore, the original control period T_{cp} is shortened and divided into two parts, a large period T_{cpl} and a small period T_{cps}. The large and small periods will be alternated during operation. In Figure 5.20, the function blocks of AI1, AI2 and PID1 are located in a fieldbus transmitter, and AO1, AO2 and PID2 in a fieldbus actuator. This approach can be implemented by redesigning the scheduling and configuration software.

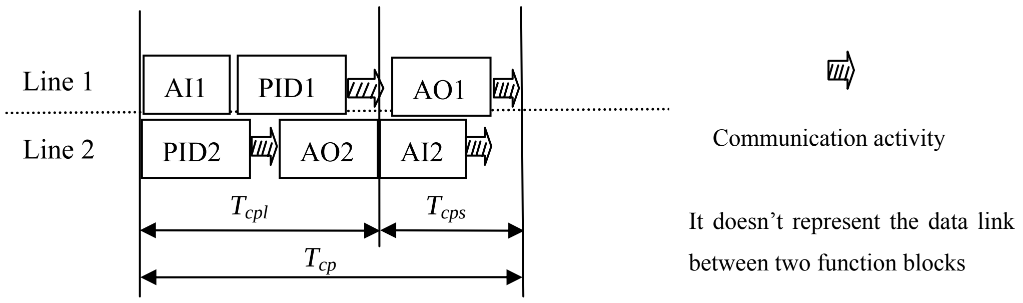


Figure 5.20: Parallel computations of function blocks

From control point of view, the two parallel computation lines for a PID fieldbus control loop cause a switched controller system (Savkin and Evans, 2001) as shown in Figure 5.21. It implies that a switched control scheme can be implemented in the FCS by parallel computation technique if two controller algorithms are different. That will be another interesting research topic.

To validate the feasibility of parallel computation, the two parallel computation lines for a PID fieldbus control loop shown in Figure 5.20 are simulated. Figure 5.22 is the simulation model for fieldbus devices with the parallel computation. The modeling method will be described in Chapter 6.

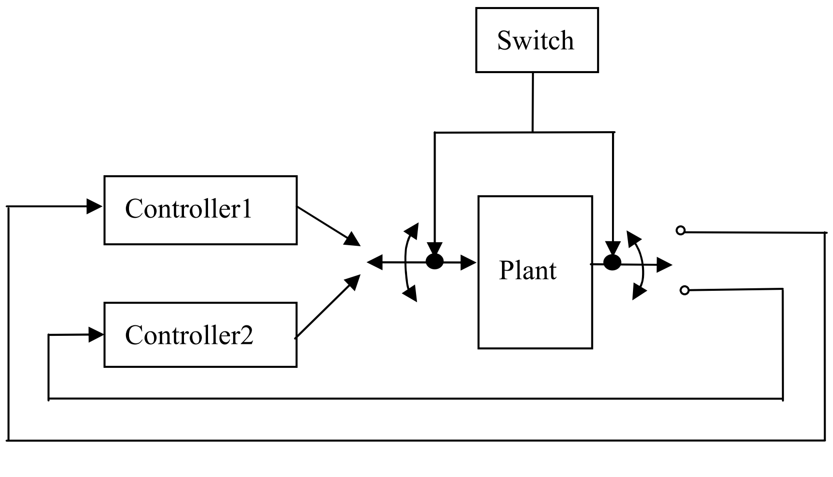


Figure 5.21: Switched control for parallel computation

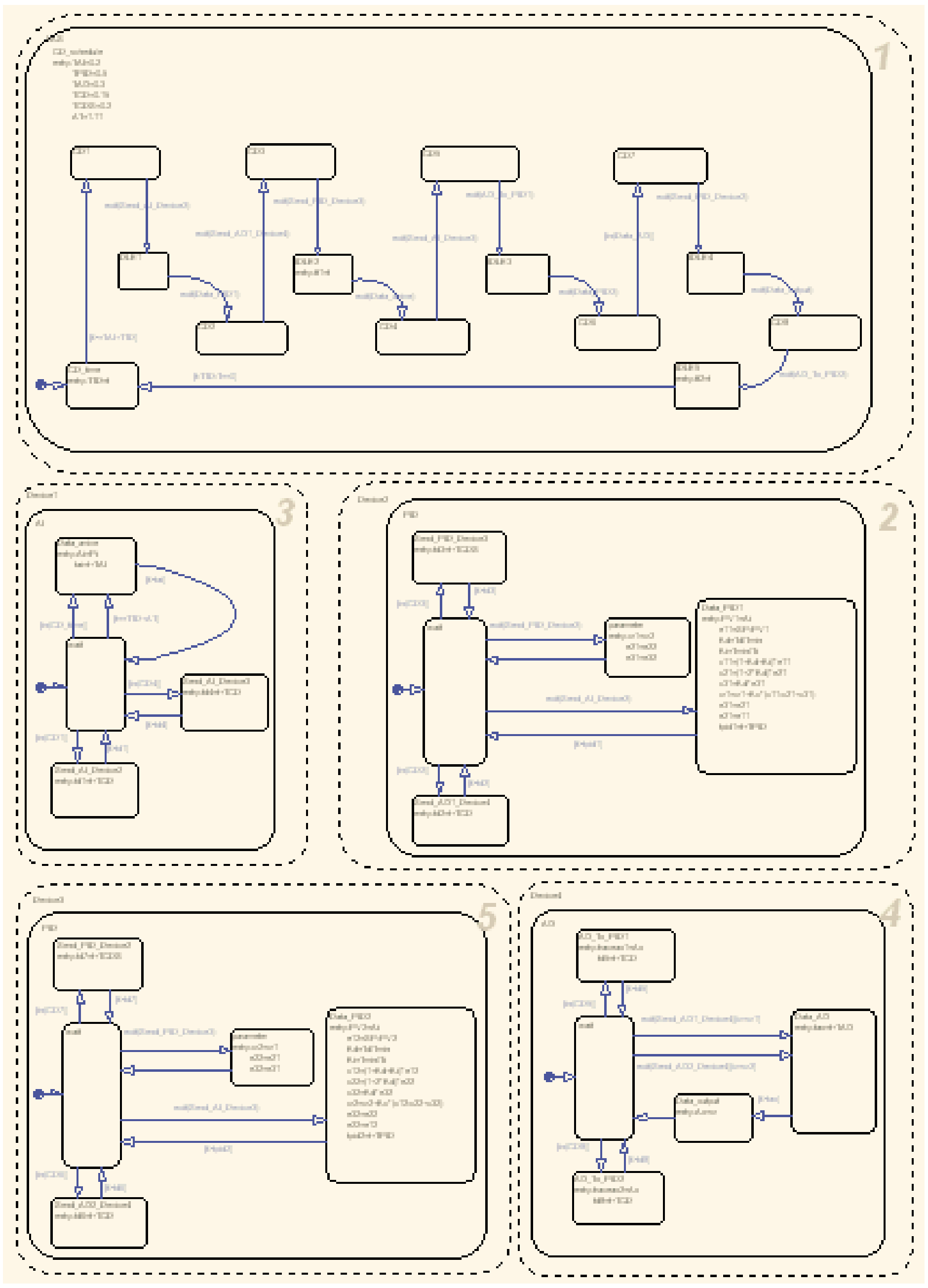


Figure 5.22 Simulation model for fieldbus devices with the parallel computation

5.6 Summary

First, the delay problem in the FCS is analyzed and categorized from the control theory point of view. There are two models: the pure delay model and the control period model to model the delays in the FCS. For the theoretical analysis, the pure delay model is easier than the control period model because of the period jitter. When using the simulation method, there is no difficulty for both models, the control period model is more accurate. The differences between the two models lead to different designing problems for the control algorithms of the FCS. For the control period model used in our work, the problem is how to select an appropriate control period and how to compensate the period jitter. Therefore, a modified PID control algorithm and a predictive control algorithm are proposed to overcome the effects of the delays and the jitter on the control performance. The effectiveness of the two control algorithms are analyzed and verified by the simulation results. The simulation results show that the modified PID control algorithm works well for the common process having the first-order model with dead time. This implies that the stability of the fieldbus control loop can be assured under the unpredictable and varying time delays caused by the fieldbus. The simulation results also tell us the fact that the delay of the plant and the delay of the communication and computation are not commutative. It is shown that the single predictive control algorithm as one of optimal control algorithms has a far better control performance than the modified PID algorithm. If the plant model is not known, the modified PID control algorithm can be used. It can give an acceptable control performance even in the case of a large jitter range of control period. The predictive control algorithm can give better control performance if the plant model is obtainable and the jitter range of the control period is small. Furthermore, the parallel computation way for the modified PID control algorithm is proposed and implemented by simulation to shorten the control period in the FCS.

Chpter 6

Modeling and Simulation of Fieldbus Control System

6.1 Introduction

Modeling and simulation is an important approach in system design and performance evaluation. This chapter is devoted to the modeling and the simulation for the fieldbus control system. Modeling is the fundamental of the simulation. First, the fieldbus devices and systems including transmitter, controller, actuator, and a fieldbus control loop are analyzed and modeled. The models for both computations and communications in the FCS are established. Based on the models, the simulation software for the FCS is developed using an open programming platform Matlab/ Simulink^{TM} and Stateflow^{TM}. The simulation software is used to study and validate the timing characteristics, evaluation, and the control performance of the FCS presented in Chapters 3, 4 and 5, respectively. On the other hand, functional and performance validation is an important issue addressed by fieldbus system design. Simulation support is needed for early functional validation and performance evaluation. The fundamental models of fieldbus devices and the closed-loop of the FCS are described in details in Sections 6.2 to 6.5. The methods and models to simulate the communication and the control in the FOUNDATION Fieldbus control system are shown in Section 6.6.

6.2 Modeling approach for FCS

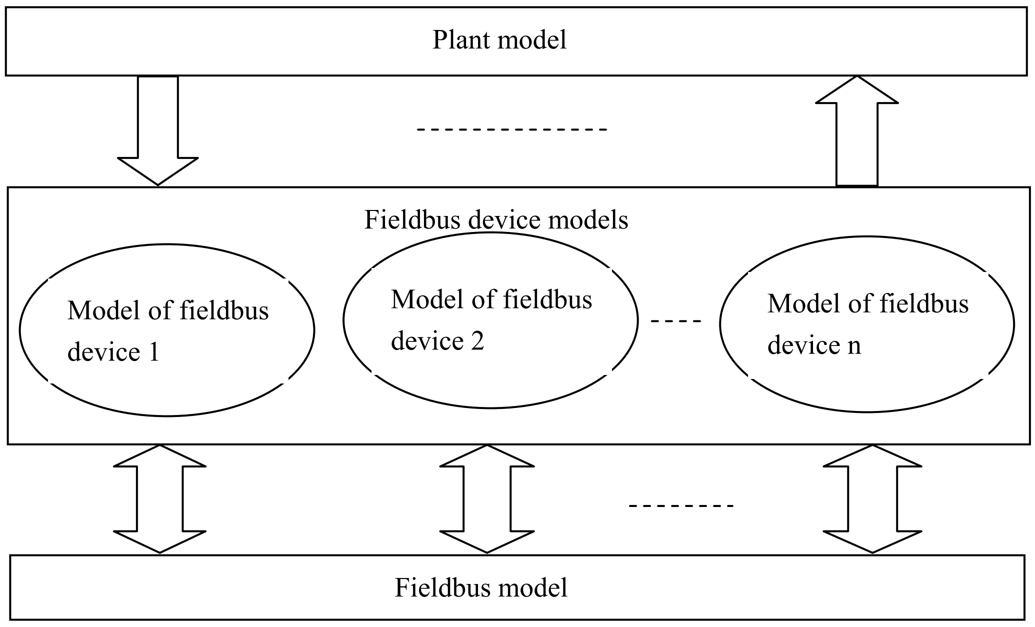
6.2.1 Modeling ideas

An FCS is a real-time distributed system, which consists of many fieldbus devices. There are two kinds of activities including the communication activity and the computation activity in the FCS. Therefore, we take the divide-and-conquer technique to model the FCS. First three kinks of fieldbus devices including the transmitter (or sensor), the controller and the actuator are modeled respectively by considering both the communication and the computation in the device. Then the models of these devices, fieldbus, and plant are integrated together for both the communication and the computation or the related events that occur when the system runs. Figure 6.1 shows the modeling strategy for the FCS. The communication and the computation activities in a device, different devices in a system should be required to be modeled modularly. Modular and hierarchical modeling (Walrand et al., 1998) is our fundamental idea. Parallelism has to be considered during modeling because many activities in different devices are independent and concurrent.

There are many functions and behaviors such as the configuration, control, management, display, operation, and trending, etc. in the FCS. However, we focus on the modeling of the control behavior and the interaction of the control and the communication of the FCS because the control is the most important and complicated function in the FCS. The control is dependent on the plant controlled. Hence, the plant model will be included in the models of the FCS. Control and communication oriented modeling is our second idea.

The FCS is a hybrid system, which includes event, discrete-time and continuous-time objects. For these different objects, different modeling methods should be used. As a result, the discrete-event model, discrete-time model and continuous-time model must be integrated together to represent the behavior of the FCS. Therefore object oriented and hybrid modeling is our third idea.

For the purpose of modeling of control behavior, we address the timing characteristics of these objects, but ignore other unimportant details during the modeling process. All things brought by fieldbus communication and distributed computation can be reflected by the varying and unpredictable time delays in the fieldbus control loop, and these delays sequentially affect the control performance. Therefore time is one important clue for modeling the FCS.



Plant model
Fieldbus device models
Model of fieldbus device 1
Model of fieldbus device 2
Model of fieldbus device n
Fieldbus model

Figure 6.1: Modeling strategy for FCS

6.2.2 Modeling and simulation tools

According to the modeling ideas mentioned above, the Matlab and its Simulink and Stateflow tools are selected to model and simulate the FCS. In the last few years, the Matlab and its tools have become the most widely used software package in academia and industry for modeling and simulating dynamic systems.

Simulink is a software package that enables us to model, simulate, and analyze dynamic systems whose outputs change over time. The model depicts the time-dependent mathematical relationships among the system's inputs, states, and outputs. We can model virtually any real-world dynamic system by selecting and interconnecting the appropriate Simulink blocks. A Simulink block diagram is a pictorial model of a dynamic system. It consists of a set of symbols, blocks and lines. Each block represents an elementary dynamic system that produces an output either continuously or at specific points in time. The lines represent connections of block inputs to block outputs. A block comprises one or more of the following: a set of inputs, a set of states, and a set of outputs. A block's output is a function of time and the block's inputs and states. The specific function that relates a block's output to its inputs, states, and time depends on the type of block.

Stateflow is used together with Simulink running on top of MATLAB as follows:

- MATLAB provides access to data, high-level programming, and visualization tools.
- Simulink supports development of continuous-time and discrete-time dynamic systems in a graphical block diagram environment. A Simulink model consists of combinations of Simulink blocks, toolbox blocks, and Stateflow diagrams.
- Stateflow diagrams enhance Simulink with complex event-driven control capabilities.

Stateflow is a tool to visually model and simulate complex reactive systems based on *finite state machine* theory. A *finite state machine* is a representation of an event-driven (reactive) system. In an event-driven system, the system makes a transition from one state to another prescribed state, provided that the condition defining the change is true. Stateflow provides clear, concise descriptions of complex system behavior using finite state machine theory. A Stateflow diagram is a graphical representation of a finite state machine, where *states* and *transitions* form the basic building blocks of the system. The Stateflow diagram consists of a set of graphical objects (states, boxes, functions, notes, transitions, connective junctions, and history junctions) and non-graphical objects (events, data, and targets). The Stateflow machine is the collection of Stateflow blocks in a Simulink model. The Simulink model and Stateflow machine work seamlessly together. Each Stateflow machine has its own object hierarchy. The Stateflow machine is the highest level in the Stateflow hierarchy. The object hierarchy beneath the Stateflow machine consists of combinations of graphical and non-graphical objects. Additionally, Stateflow enables the representation of hierarchy, parallelism, and history. Hierarchy enables us to organize complex systems by defining a parent/offspring object structure. For example, we can organize states within other higher-level states. A system with parallelism can have two or more orthogonal states that are active at the same time. History specifies the destination state of a transition based on historical information. These characteristics make the Stateflow suitable as a modeling and simulating tool for the FCS.

6.2.3 Modeling approach

Hierarchical modeling (Walrand et al., 1998) is used to define the FCS model through a top-down stepwise refinement technique based on the characteristics of the FCS. Main idea of hierarchical modeling approach is as follows. A system is represented by a succession of models, each of which is defined at a different level of detail. By applying this top-down approach, we start from the least detailed model of the system defined at the highest level of abstraction. By specifying with more detail some components of this first model, we define a new more detailed model of the system at a second level of abstraction. This procedure can be iterated to obtain a model at an appropriate level of detail. The set of models obtained by this top-down approach is hierarchically related in terms of model components. According to this idea, the Figure 6.1 is the model of the FCS at the highest level of the abstraction. In this model, the plant, fieldbus devices and the fieldbus can be further detailed to obtain the models at a second level of abstraction. Modeling plant has been a mature practice in the control area. The plant can be modeled in various forms including transfer functions, state-space equations, etc. Therefore the plant modeling is not discussed. Our focus is on the further modeling of the fieldbus devices and the fieldbus by using the hierarchical modeling approach.

6.3 Fundamental models of fiedbus devices

The modeling process is based on the detailed analysis of the timing characteristics as presented in Chapter 3, the internal structure and operation of the fieldbus devices. The modeling of the fieldbus transmitter is taken as an example to show the details of the modeling process for fieldbus devices.

6.3.1 Modeling of fieldbus transmitter

In the classical feedback control system, the modeling of a transmitter is almost always negligible because the transmitter has a much smaller time constant than the plant or can be included in the model of the plant. But in the fieldbus control system, we must take into account the transmitter dynamics because it has distributed algorithms and time delays caused by the algorithm's execution and data communication between the transmitter and other field devices. The distributed algorithms change its output, whereas the time delays affect the control performance of a fieldbus control loop.

Figure 6.2 depicts the typical scheme of a transmitter's hardware and software for the FOUNDATION Fieldbus fieldbus control system. The CPU is the brain of the fieldbus transmitter, since it controls both the hardware and software. The function blocks are the distributed algorithms in the transmitter. From the standpoint of IEC 61499 (Lewis, 2001), the abstract model of a fieldbus device is shown in Figure 6.3.

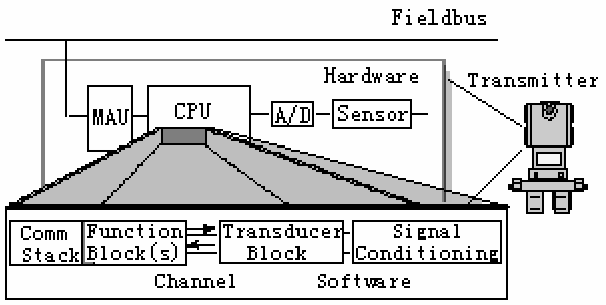


Figure 6.2: Scheme of transmitter hardware and software

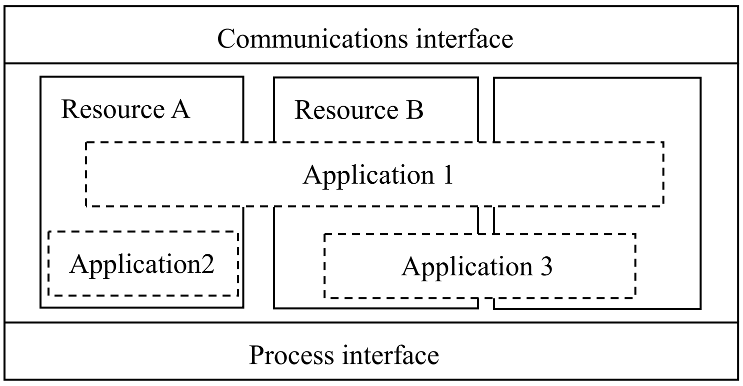


Figure 6.3: Abstract model of distributed fieldbus devices

A fieldbus device supports one or more resources, which provides independent execution and control of algorithms. The fieldbus device model has two interfaces, 'process interface' and 'communications interface'. The former provides the services for resources to exchange data with the input and output points to the plant on the physical device. The latter provides communications services that enable resources to exchange data via the fieldbus with resources in remote devices. In Figure 6.2, the process interface corresponds to the sensor and A/D converter as hardware and to the signal conditioning and transducer block as software. The communications interface corresponds to the media access unit (MAU) as hardware and to the communication stack as software.

Based on the explanation of the transmitter subtask and the timing components analysis in Chapter 3, and the descriptions for Figure 6.2 and Figure 6.3, the transmitter subtask has three jobs, including the sampling and filtering job, the computing job, and the communication job. For the loop control application, the three jobs are completed in sequence during each cyclic execution of the transmitter subtask from its input to its output. Therefore, we derive a model of the fieldbus transmitter shown in Figure 6.4. The three jobs should be executed under the scheduling of the real-time operating system in the device because they often share one CPU (Central Processing Unit). To model each of the three jobs, the following analysis is necessary.

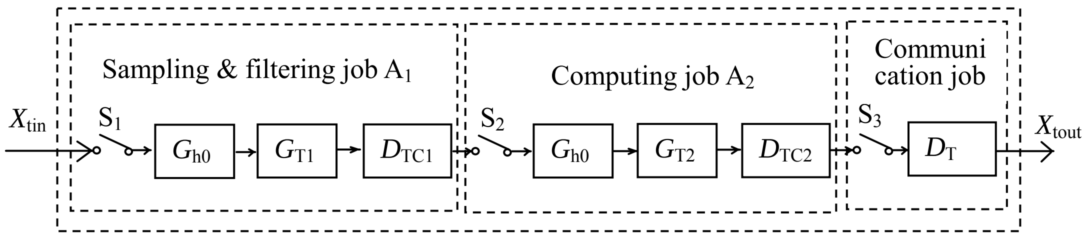


Fig 6.4: Fieldbus transmitter model

First, in the sampling and filtering job, the sampling is done to enable the A/D converter to cyclically convert the signal from a sensor to a digital value. The sampling process belongs to input operation, and it is often modeled by an internal sampling switch S_1 and a zero-order holder $G_{h0.}$ The filtering is done to execute the filtering algorithm to get a noise-free value of measurement, and of course it is modeled by the filtering algorithm G_{T1}. Strictly speaking, the filtering should belong to the computing operation or data processing. The reason that we put the sampling and filtering together as one job is as follows: in practice, the sampling period here is an internal sampling period of the transmitter or instrument sampling period but not the sampling period for control algorithms or control sampling period. The instrument sampling period is usually fixed and far smaller than the control sampling period. Therefore, the filtering has a different execution period from the computing job discussed below. On the other hand, the filtering algorithm $G_{T1}$is fixed, whereas the algorithm of the computing job is configurable. The sampling and filtering job is executed periodically through an internal timer interrupt. D_{TC1} is a time-delay unit used to model the time delay caused by the execution of the sampling and filtering algorithms, and the data wait in a buffer. In fact, D_{TC1} represents the data input time by the internal way analyzed in Chapter 3. The time is very small. Therefore, D_{TC1} is often negligible in the control loop time.

Secondly, the computing job is a part of the distributed control algorithms in a fieldbus control loop and corresponds to the function block (s) of the FOUNDATION Fieldbus transmitter. It is modeled by a scheduling switch S_2, a zero-order holder $G_{h0,}$ a computing algorithm G_{T2}, and a time-delay unit D_{TC2}. S_2 is the scheduling switch controlled by the scheduler of the distributed algorithms in the fieldbus control system. Its period is the control sampling period of the fieldbus control loop. G_{T2} is the scheduled and executed algorithm embedded in the fieldbus transmitter, often in the form of a function block. D_{TC2} is a time-delay unit representing the execution time of the algorithm, and it is the data processing time described in Chapter 3. The computing job can be considered as a part of a conventional controller.

Finally, the communication job is the data output operation and corresponds to the communication stack of the FOUNDATION Fieldbus transmitter. It is modeled by a communication switch S_3 controlled by the fieldbus access method and a time-delay unit D_T representing the data output time by the fieldbus way analyzed in Chapter 3. The period of S_3 is the same as that of S_2, but the both switches are not synchronous.

The algorithms included in fieldbus devices, such as G_{T1} and G_{T2} in the transmitter, can be described in a discrete-time model. If the execution period of the algorithm is constant, the algorithm can be modeled by the transfer function in the form of a Z-transform. Otherwise, i.e. if the period is variable, the output of the algorithm must be calculated during each variable period. In such a case, the algorithm output can be described by Eq. (6.1).

$$y(t_k) = y(t_{k-1}) + \Delta y(t_k) \quad , \qquad (6.1)$$

where $\Delta y(t_k) = f(t_k, t_{k-1}, t_{k-2}, ..., u(t_k), u(t_{k-1}), u(t_{k-2}), ...)$,

and where $u(t_k) \in R^n$ is the algorithm input, $y(t_k) \in R^m$ is the algorithm output, t_k is the k th sampling instance, and *f* is the function of the algorithm.

The fieldbus transmitter model shown in Figure 6.4 is obtained by abstraction for more intention of control or from the control theory point of view. However, the fieldbus transmitter model can also be described in Figure 6.5 for more intention of communication or from the network theory point of view. *A*1 and *A*2 represent the sampling and filtering job and the computing job, respectively. It is shown that the model in Figure 6.5 includes the model in Figure 6.4. Therefore the model in Figure 6.5 is more complete, in which other application *A*n such as alarm and trending can be included, and the communication job can be further detailed.

We will take the fieldbus transmitter model in Figure 6.5 but with more intention of control. This means that we will model the fieldbus devices without considering the applications and the communication details that do not affect control performance. Therefore application A_n will be neglected during simulation. The communication component in the model will be further detailed separately. It has identical model for all fieldbus devices with same protocol.

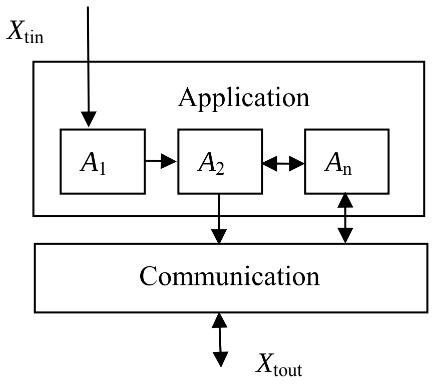


Fig 6.5: Fieldbus transmitter model from network point of view

6.3.2 Modeling of fieldbus controller

Similar to the fieldbus transmitter, a fieldbus controller model can be obtained as shown in Figure 6.6. S_4 and S_6 are the communication switches. D_R and D_T represent the data input time and output time by the fieldbus way described in Chapter 3, respectively. S_5 is the scheduling switch controlled by the scheduler of distributed algorithms in the fieldbus control system. G_{h0} is the zero-order holder. G_c is the control algorithm executed in the fieldbus controller, for example the modified PID algorithm or the single predictive control algorithm proposed in Chapter 5. D_{CC} represents the time delay caused by the execution of the control algorithm, i.e. the data processing time described in Chapter 3.

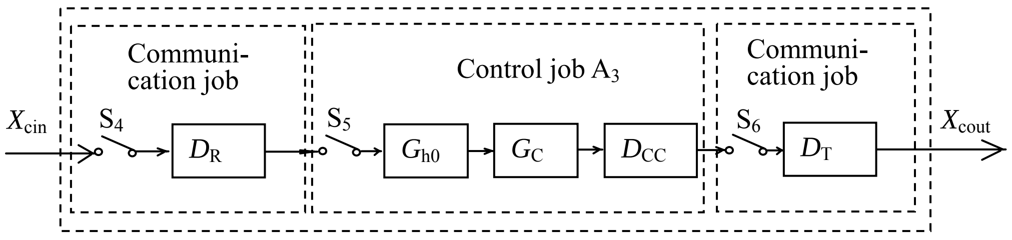


Figure 6.6: Fieldbus controller model

The difference between the controller and the transmitter or actuator is that there is no ‘process interface’ because the controller doesn’t have direct contact with the plant. Therefore, the controller can only be implemented by software in the form of a function block and it is embedded in the fieldbus transmitter or actuator. In fact, the PID controller in the FOUNDATION Fieldbus control system is implemented in this way. In such a case, there is no separate controller device in the fieldbus control loop, and the traffic on the fieldbus and communication time delay are reduced. Accordingly, a separate model of the controller is not necessary, and it can be integrated into the model of the fieldbus transmitter or actuator. The control algorithm can be included in the computing job of the fieldbus transmitter or actuator.

The fieldbus controller model can also be represented by Figure 6.7 from the network point of view. Only the application A_3 for the control job is considered in the application component of the model.

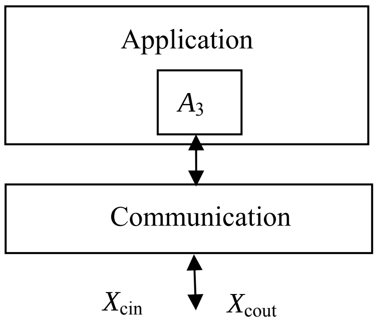


Figure 6.7: Fieldbus controller model from network point of view

6.3.3 Modeling of fieldbus actuator

As done in the modeling process of the fieldbus transmitter, we can obtain the model of a fieldbus actuator as shown in Figure 6.8. S_7 is the communication switch. D_R represents the data input time by the fieldbus way. S_8 is the scheduling switch controlled by the scheduler of distributed algorithms in the fieldbus control system. G_{h0} is the zero-order holder. G_A is the actuator algorithm executed in the fieldbus actuator. D_{AC} represents the data processing time needed to execute the actuator algorithm. S_9 is the output switch. D_C represents the data output time by the internal way described in Chapter 3.

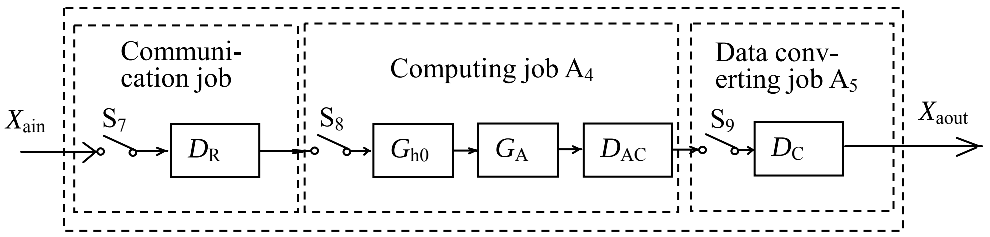


Figure 6.8: Fieldbus actuator model

The fieldbus actuator model can also be shown in Figure 6.9 from the network point of view. The application A_4 for the computing job and the application A_5 for the data converting job are considered in the model.

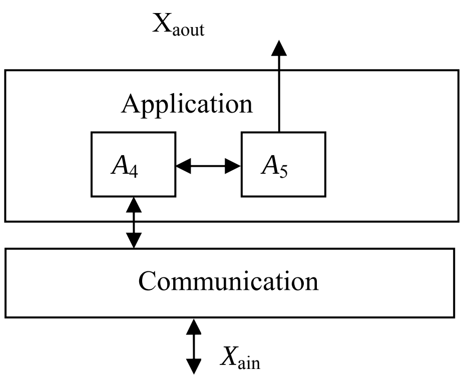


Figure 6.9: Fieldbus actuator model from network point of view

6.4 Communication model of the fieldbus

The communication of the fieldbus is real-time communication of which model can be shown in Figure 6.10 (Liu, 2002).

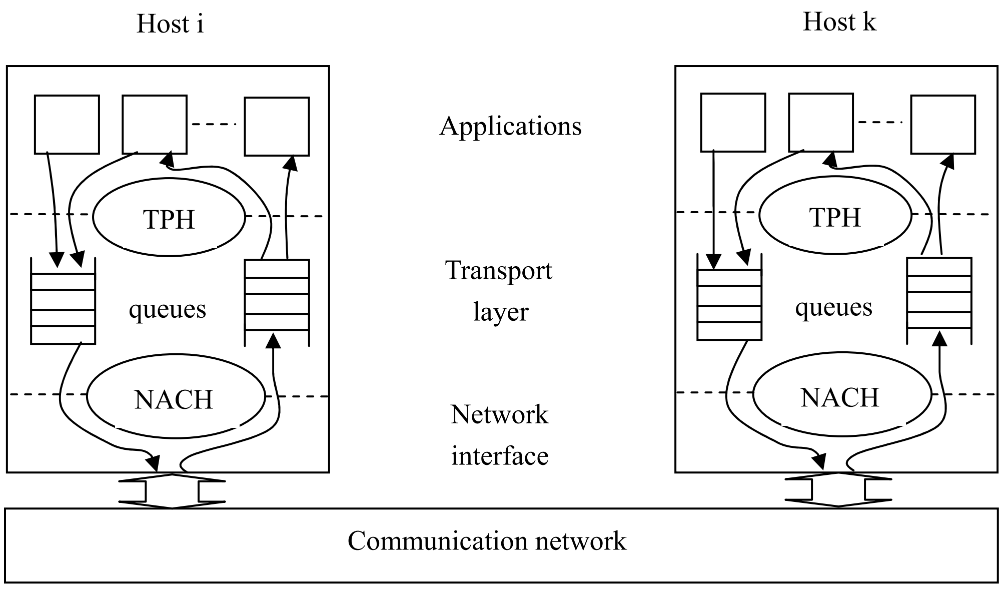


Figure 6.10: Real-time communication model

Figure 6.10 shows data paths traversed by messages in and out of hosts, which are device nodes of the FCS. The circles marked TPH (Transportation Handler) and NACH (Network-Access-Control Handler) are handlers of the transportation layer and the network-access-control. Applications shown in Figure 6.10 are the computing jobs in the fieldbus models shown in the Figures 6.4 to 6.9, which can be described in the discrete-time models. The input and output queues in the transport layer represent the transfers of data. The time characteristics of the communication are determined by the NACH, which depends on the fieldbus access control technique.

When we consider only the control performance for simplicity, the time delay component

D_i related to communication in the device models can be abstractly modeled as

$$D_i = D_{\min}^i + J_r^i Rand(t) , \tag{6.2}$$

where $D_{\min}^i$ is the minimum of the time delay component, J_r^i is the jitter range of the time delay component and *Rand(t)* is a random variable with time that ranges from 0 to 1. Therefore, J_r^i can be considered as a parameter representing the fieldbus access control technique at the highest level abstraction. The clock-driven scheduling technique such as that used by the FOUNDATION Fieldbus has a very small jitter range, whereas the priority-driven scheduling technique such that used by the CAN fieldbus has different jitter ranges for different messages with different priorities. The random access control technique such as that used by the LonWorks fieldbus has a large jitter range, which depends on the traffic load on the fieldbus. Several time delay components can be integrated together into one equivalent delay in some cases.

If we consider the fieldbus performances apart from the control performance, the communication model should be further detailed. In order to obtain the detailed model, the data transferring procedure should be understood clearly. To study the control performance, we only consider the periodic data transfer. We take the FOUNDATION Fieldbus as an example to illustrate the communication modeling.

As discussed in Chapter 2, the periodic data communication in the FOUNDATION Fieldbus is based on the scheduled way by LAS (Link Active Scheduler). Figure 6.11 (Cavalieri, 1996) depicts the operating procedure of periodic data communication between data sender and receiver through the respective three layers.

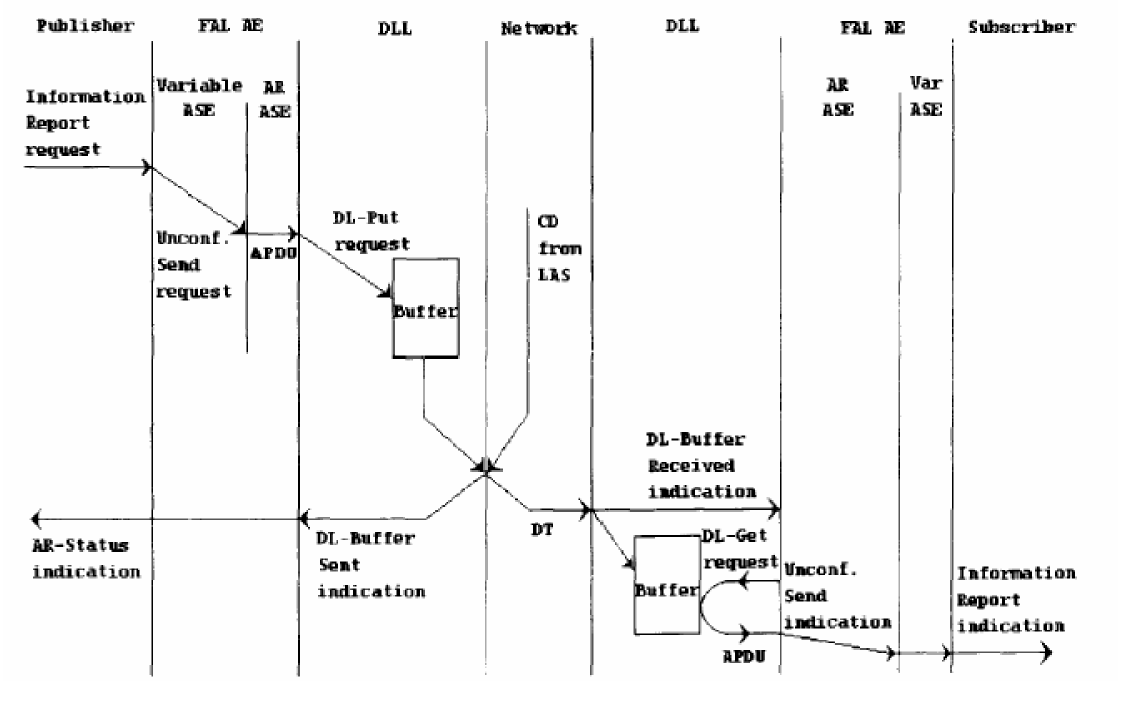


Figure 6.11: Interaction between layers of sender and receiver

For a periodic data transfer, for example, the periodic data transfer from a fieldbus transmitter to a fieldbus actuator, a publisher/subscriber VCR is pre-defined and included in a scheduling table managed by the LAS. As shown in Figure 6.11, the LAS, on the basis of the information contained in its scheduling table, periodically sends the publisher a CD (Compel Data) to compel the value contained in its transmission buffer. The publisher submits an Information Report request primitive to its Variable ASE (Application Service Element), containing the value of the variable to be published. The Variable ASE builds an Information Report Request APDU (Application layer Protocol Data Unit) body and conveys it on the AR ASE using an AR Unconfirmed Send request primitive. The AR ASE builds an unconfirmed send request APDU, which is written in the transmission buffer at the interface between the AL and DLL (Data Link Layer) by a DL-Put request. On transmission of the contents of the buffer which occurs on reception of a CD, a DL-Buffer Sent indication is sent to the AR ASE which notifies the user that the buffer content has been sent by means of an AR Status indication. Likewise, in all the subscribers, reception of information by means of a DT (Data) and overwriting of the reception buffer cause a DL-Buffer-Received indication to be generated. This primitive notifies the local AR ASE of reception. The AR ASE uses a DL-Get request which supplies the contents the buffer, i.e. the Unconfirmed Send request APDU, as its output parameter. Upon receipt of this APDU, the AR ASE delivers an Unconfirmed Send indication to the Variable ASE, which sends an Information Report indication service primitive to its user, delivering the value produced by the publisher.

From the above analysis, we conclude that the data transfer procedure is too complex to model all details of each layer. However, from the time point of view, we can simplify the data transfer procedure. Hence, we will only consider the main factors that affect the time performance during data transfer. The others are combined or not considered at all. The algorithm used by the LAS is shown in Figure 6.12. The LAS operates on CD schedule, which contains a list of activities that are scheduled to occur on a cyclic basis. At the scheduled time instance, the LAS sends CD message to a specific data buffer in a fieldbus device. The device immediately broadcasts or “publishes” a message to all devices on the fieldbus. This is the highest priority activity performed by the LAS. The remaining operations including pass token for non-periodic data transfer are performed between scheduled transfers. For simplicity, TD and PN sending that does not affect the control performance is neglected during modeling and simulation. However, it can also be implemented if necessary. Therefore, the LAS algorithm is simplified and modeled as shown in Figure 6.13.

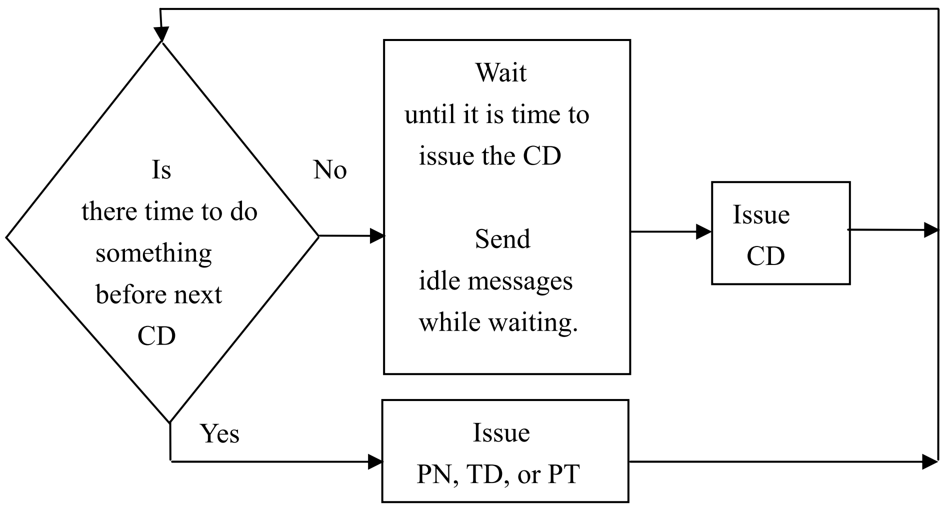

Is there time to do something before next CD
No
Wait until it is time to issue the CD
Send idle messages while waiting.
Issue CD
Yes
Issue PN, TD, or PT

Figure 6.12: LAS (Link Active Scheduler) algorithm (Fieldbus Foundation, 1998)
CD=compel data; PN=probe node; TD=time distribution; PT=pass token

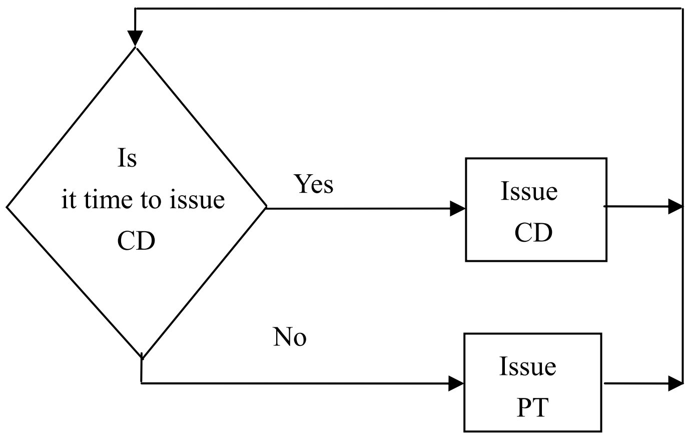


Figure 6.13: Simplified LAS algorithm

The detailed abstract model for the FOUNDATION Fieldbus communication component in the fieldbus devices can be found in Figures. 6.24 to 6.27.

6.5 Closed-loop model of fieldbus control system

As shown in Figure 6.14, the fieldbus control loop is represented by a hybrid model consisting of the discrete-time model for fieldbus devices including transmitter, controller and actuator, the discrete-event model for fieldbus scheduling, and the continuous-time model for the plant. The operation of the fieldbus control loop needs to schedule the communication interconnections among fieldbus devices and the co-operation of algorithms in each device. Such scheduling processes are the event processes. Thus, the activities of the scheduling can be viewed as an event-driven system. Different scheduling methods have different characteristics and ways of implementation. We only focus on the time characteristics of scheduling methods without taking into account other details related to them. Therefore, we developed a clock-driven system to model the behavior of the scheduling processes for communications and algorithm execution by describing it in terms of transitions among working states.

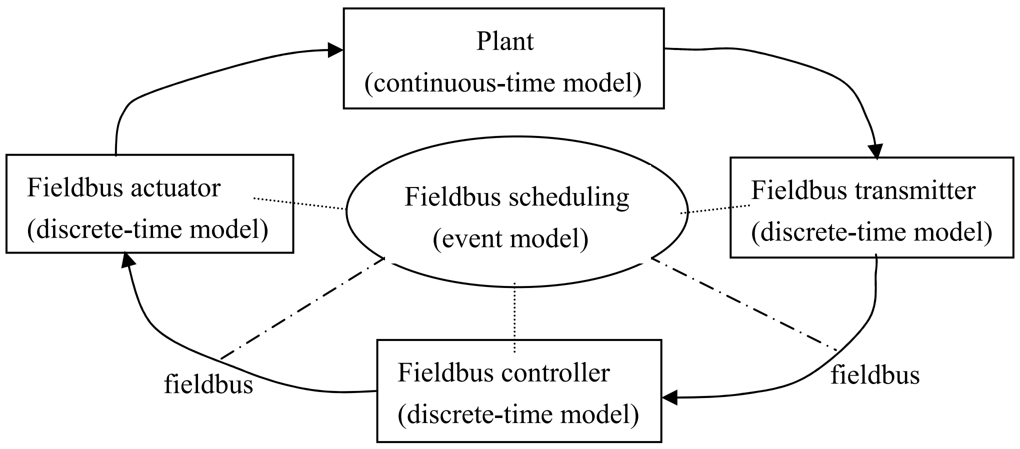


Figure 6.14: Communication and algorithm scheduling model of fieldbus control loop
┈┈┈: Algorithm scheduling; —·—·— : Communication scheduling

The finite state machine (FSM) shown in Figure 6.15 represents the event-driven system designed for fieldbus scheduling of the simplest case – single control loop. This scheduling system has a number of operating states: transmitter, controller and actuator, and the system transitions from one state to another in sequence, provided that the condition defining the change is true. *t* is the time of the clock shared in the fieldbus control loop, and t_1, t_2 and t_3 are varying instances when clock events occur such as the ends of algorithm execution and communication. The models of fieldbus devices described above are included in each state. The active state is determined based on the occurrence of clock events. They are controlled by a clock shared by all fieldbus devices in the fieldbus control loop according to the time-delay units in the models shown in Figures. 6.4 to 6.9. When one state is active, the algorithm in the fieldbus device corresponding to the state is executed and the related communication is completed. The scheduling model becomes complex when more control devices and control loops exist in the FCS.

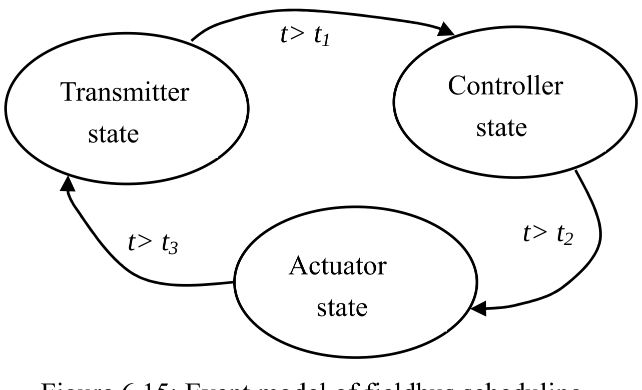


Figure 6.15: Event model of fieldbus scheduling

6.6 Simulation of FCS

6.6.1 Components of simulation software

The conceptual structure of the simulation software for the FCS is shown in Figure 6.16. The simulation software includes four parts:

- FCS simulator to simulate the dynamical behavior of the FCS including the fieldbus communication and the control;
- Fieldbus performance calculator to calculate the fieldbus performance criteria;
- Control performance calculator to calculate the control performance criteria;
- Plotter to plot all the output or performance curves.

The simulation software is developed by using the Simulink and Stateflow of MATLAB as simulation tools, which are described in Section 6.2.

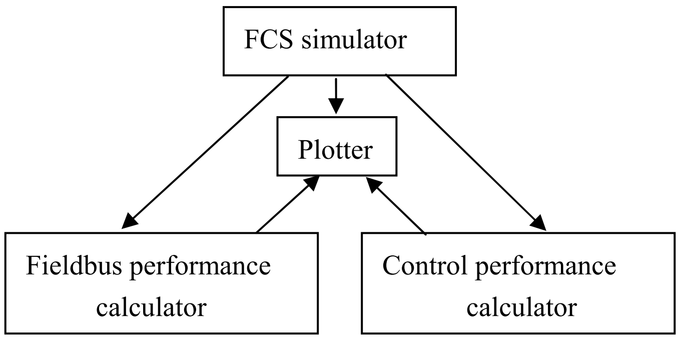

FCS simulator
Plotter
Fieldbus performance calculator
Control performance calculator

Figure 6.16: Components of simulation software

6.6.2 Simulation of the fieldbus control loop

An outline of the FCS simulator for a typical fieldbus control loop in process control is presented in Figure 6.17.

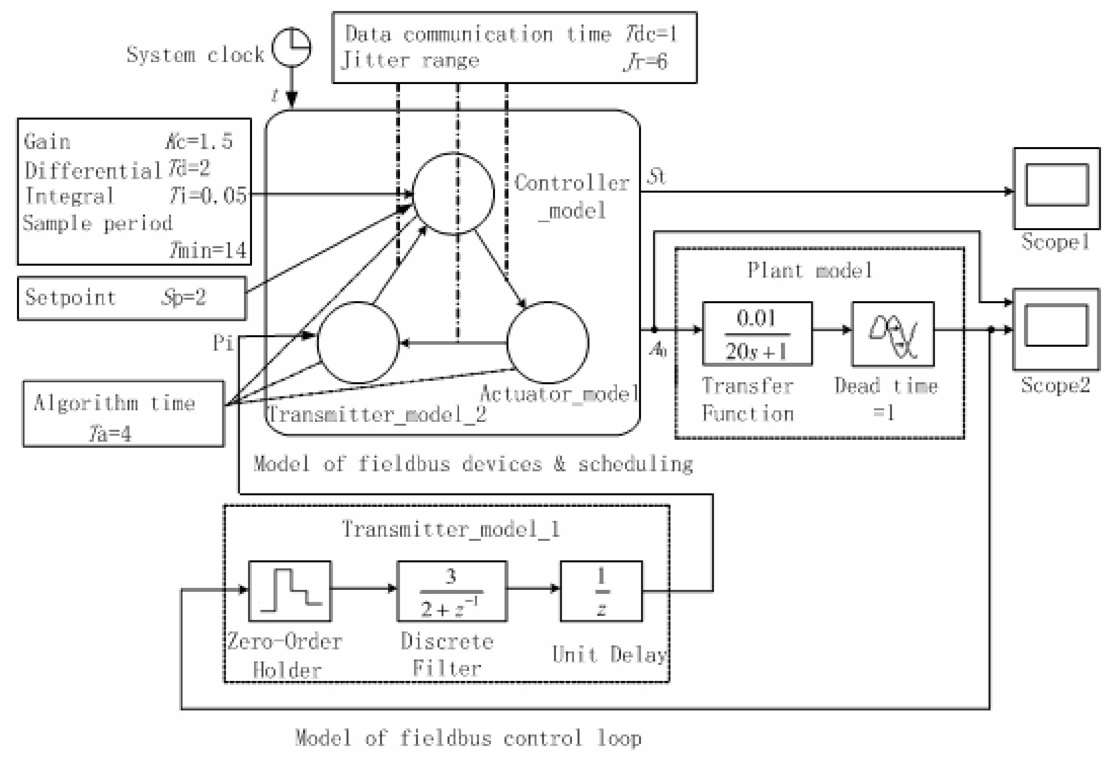
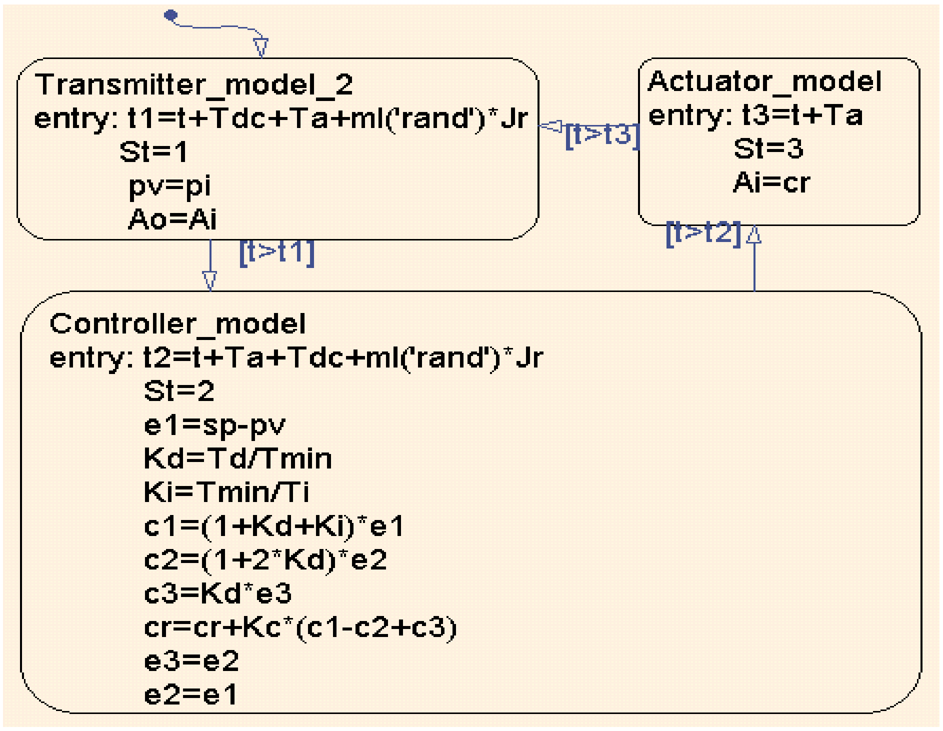


Figure 6.17: Structure of simulation software of fieldbus control loop

In this study, a first-order model with dead time in the form of transfer function is taken as the plant model because the dynamic responses of most plants in process control can be mathematically described by such a model. To simulate the fieldbus transmitter, its model shown in Figure 6.4 is divided into two parts. The first part, i.e. the sampling and filtering job, is implemented by the transmitter model_1 in Figure 6.17. The second part, including the computing job and the communication job in Figure 6.4, is implemented by the transmitter model_2 in Figure 6.18. The modified PID control algorithm proposed in Chapter 5 is implemented by the controller model as shown in Figure 6.18. The fieldbus actuator in Fig 6.8 is implemented by the actuator model shown in Figure 6.18. In this figure, the data communication time T_{dc} is the sum of D_T, the data output time by the fieldbus way in Figures. 6.4 and 6.6, and D_R, the data input time by the fieldbus way in Figures. 6.6 and 6.8. The data output time by the internal way in Fig 6.8 (D_C) is neglected. It is also assumed that D_{TC2}, D_{CC}, and D_{AC} in Figures. 6.4, 6.6, and 6.8 have the same execution time *Ta*. It is noteworthy that for simplicity, $G_{T2}=G_A=1$, and zero-order holders G_{h0} in the computing jobs in Figures. 6.4 to 6.6 are implemented implicitly as a variable memory. It is easy to add any algorithm model for the fieldbus transmitter or actuator if necessary.

The state flow machine consists of simulation models of fieldbus devices and scheduling, and its details are depicted in Figure 6.18. The state flow machine has three groups of inputs. The first group's inputs are the model time-parameters for communication and algorithm execution such as the data communication time T_{dc}, algorithm execution time T_a, and period jitter range J_r. The second group's inputs are the parameters of the PID controller algorithm. The third group's inputs are P_i as the input of the second part of the transmitter and S_p as the setpoint of the controller. The state flow machine has two outputs: S_t as the state value indicating the real time state of the fieldbus control loop and A_o as the output of the actuator. The period jitter caused by the variable times of communication and algorithm execution is implemented by a random function, which generates random numbers whose values are uniformly distributed at the interval (0, J_r). All initial parameters of models and algorithms are included in Figure 6.17.



```
Transmitter_model_2
entry: t1=t+Tdc+Ta+ml('rand')*Jr
       St=1
       pv=pi
       Ao=Ai
```

```
Actuator_model
entry: t3=t+Ta
       St=3
       Ai=cr
```

```
Controller_model
entry: t2=t+Ta+Tdc+ml('rand')*Jr
       St=2
       e1=sp-pv
       Kd=Td/Tmin
       Ki=Tmin/Ti
       c1=(1+Kd+Ki)*e1
       c2=(1+2*Kd)*e2
       c3=Kd*e3
       cr=cr+Kc*(c1-c2+c3)
       e3=e2
       e2=e1
```

Figure 6.18: Structure of state flow machine

6.6.3 Simulation of the FOUNDATION Fieldbus control system

Figure 6.17 is an abstract model for a fieldbus control loop. This model only considers the timing behavior of the fieldbus communication. Its advantage is that it can be used to model the control performance of a system using any fieldbus. However, it lacks a one-to-one mapping between objects in the fieldbus system being modeled and their abstraction in the simulation models. To get such mapping, three levels of hierarchical models are established for the FOUNDATION Fieldbus control system. Figure 6.19 shows the first level models of four subsystems represented by the Simulink tool. One fieldbus segment connects three fieldbus devices including a transmitter, a controller, and an actuator. A plant having a first order model with dead time are controlled by these devices. The communication between devices is modeled as input and output connections shown in Figure 6.19. The block with notation 'Memory' is a trick to avoid 'algebraic loop' problem during simulation under the Matlab environment. It can be considered as a straight through path. This model is an open architecture that means more devices can be easily added. The fieldbus device, the fieldbus segment, and the plant are the three classes of components of the model. They are encapsulated as separate and independent objects. The lines between their input/output ports establish their connection relationship.

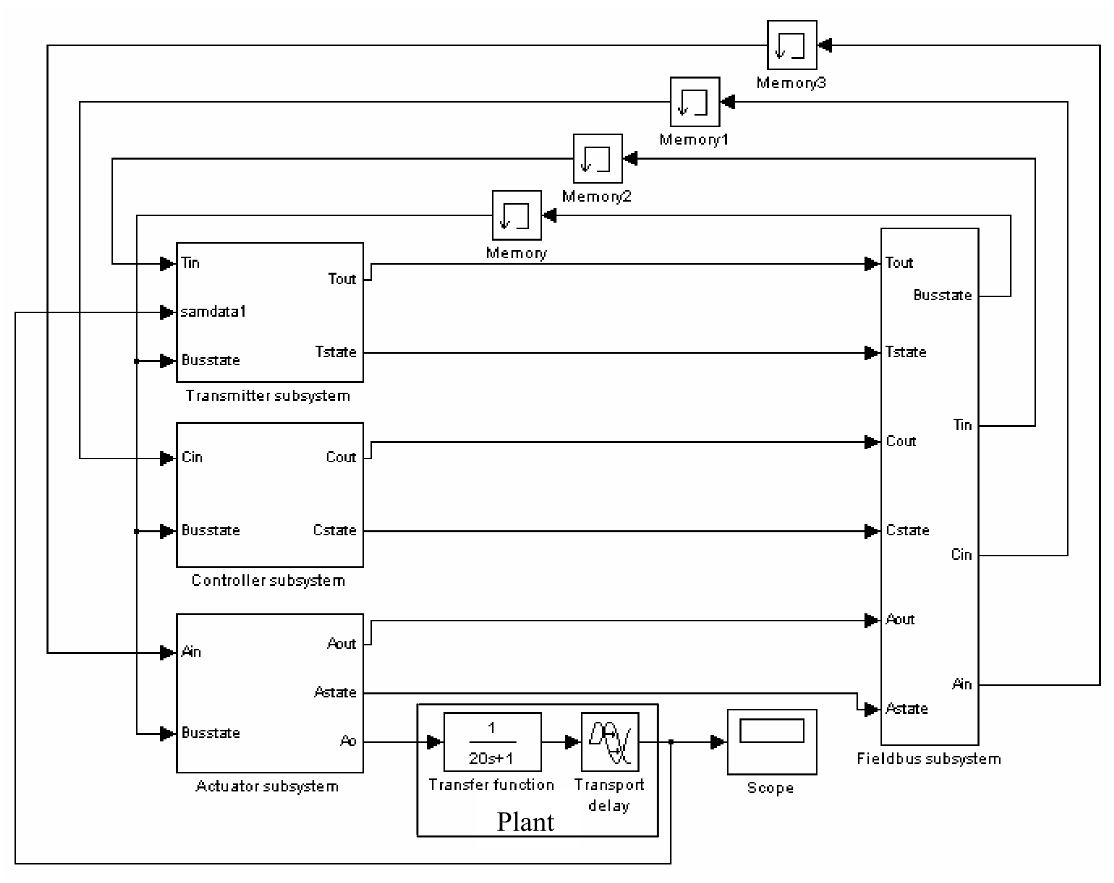


Figure 6.19: Model of the FOUNDATION Fieldbus control system

The subsystems shown in Figure 6.19 can be further detailed into a second level of models, which are shown in Figures. 6.20 to 6.23, respectively. In the model of the transmitter shown in Figure 6.20, the zero-order hold and the discrete filter are the model components for the input process interface of the transmitter. The model components of the communication and application are detailed and explained in the third level of models. The bidirectional communication channels are represented by two arrays of *Tin* and *Tout*, which are models of frames on the fieldbus. The array elements are used to represent the fields in the frame, including the type ID, addresses, and data. CD frame and CD time are the parameters representing the CD scheduling time and communication relationships. *Tstate* is the device communication state used to ignite the fieldbus segment communication. In the second level models of other devices and the fieldbus segment, only the outside ports for communication and parameters setting are shown. The internal details are described in the third level models of them. For the second model of the controller shown in Figure 6.21, the controller parameters such as T_d, T_i, and K_p are included except the parameters mentioned above. The second level model of the actuator shown in Figure 6.22 has an output process interface with a label *Ao*. In the second level model of the fieldbus segment shown in Figure 6.23, *Xstate*'s represent the abstraction of the physical layers of devices and the fieldbus medium, the *Xin*'s and *Xout*'s represent the bidirectional communication channels between the devices through the fieldbus medium. The blocks with notation 'Scope *x*' or 'Display *x*' in Figures 6.20 to 6.23 are used for monitoring and recording simulation states and results.

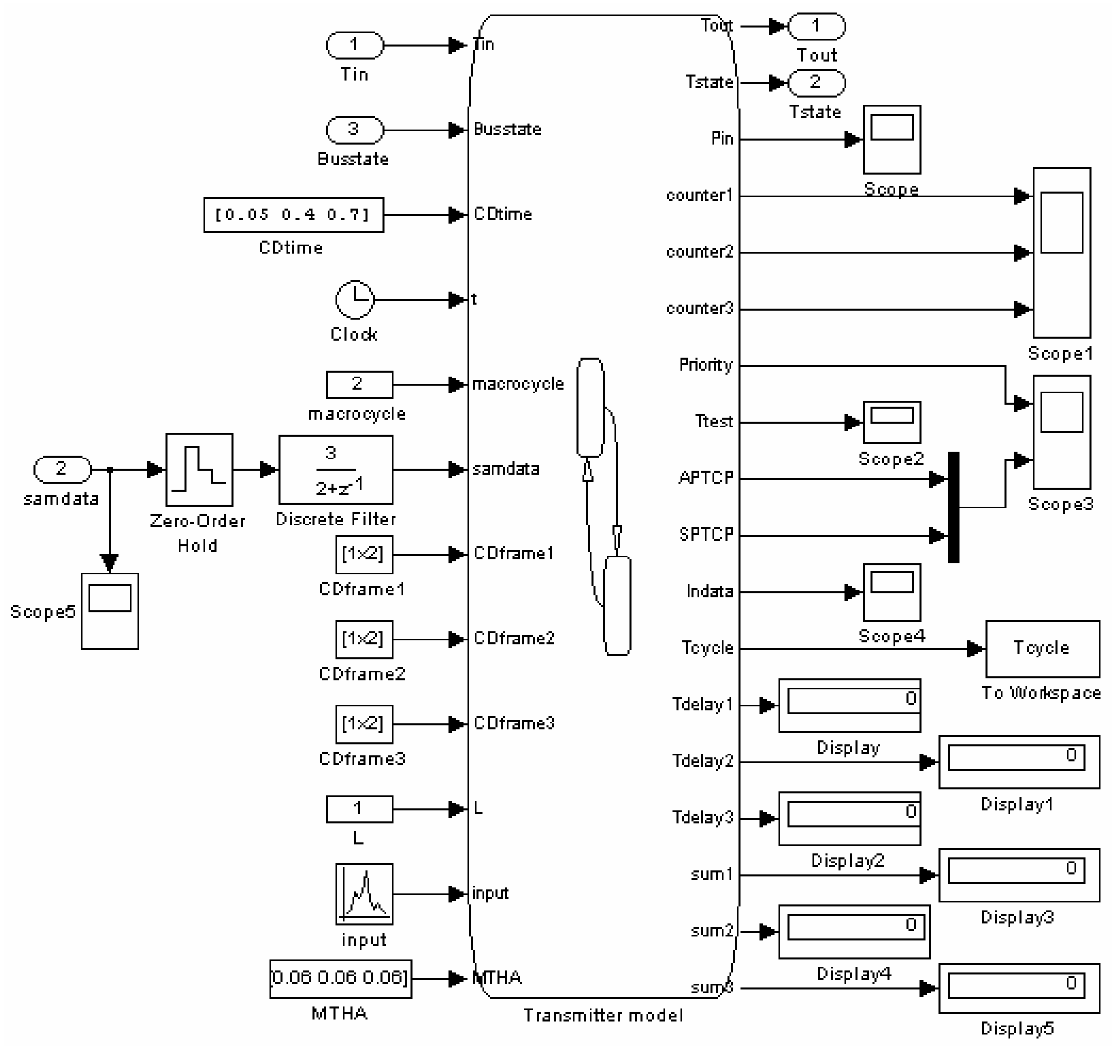


Figure 6.20: Model of transmitter subsystem

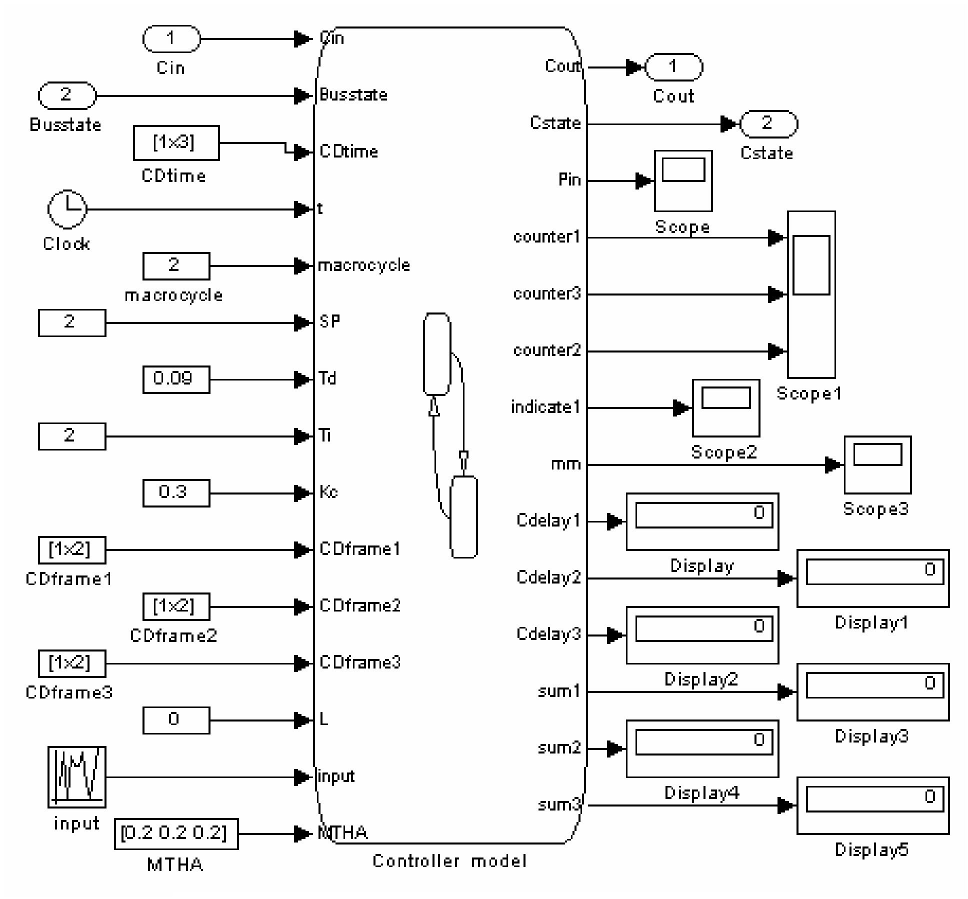


Figure 6.21: Model of controller subsystem

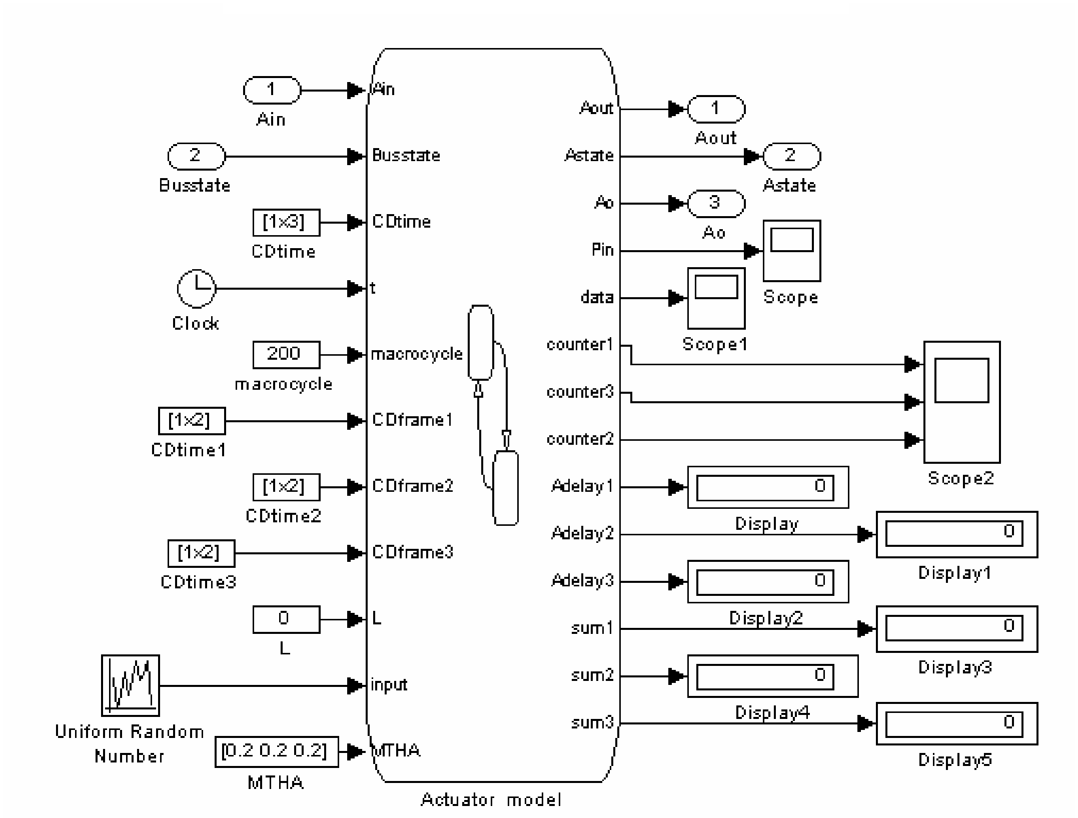


Figure 6.22: Model of actuator subsystem

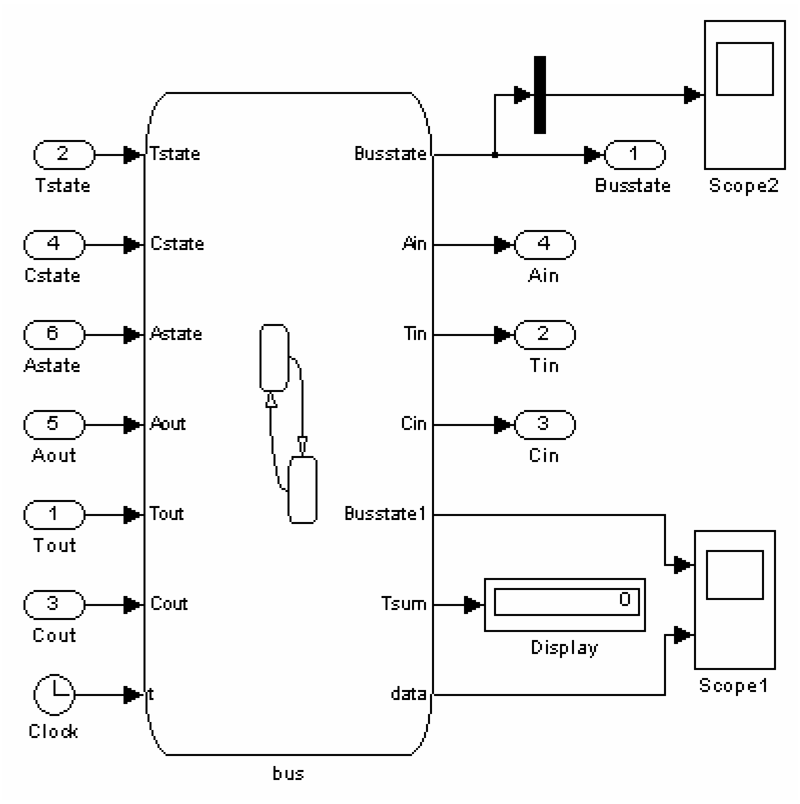


Figure 6.23: Model of fieldbus segment subsystem

The third level models of the fieldbus system are shown in Figures. 6.24 to 6.27. The fieldbus devices have a similar model structure. There are three components in transmitter models: LAS, Communication, and Function. For the controller and actuator models, LAS component is neglected. They can be considered as basic devices. The two components of the LAS and the Communication represent the fieldbus communication protocol, and the Function represents the device application. They are modeled separately since they are independent entities running asynchronously and concurrently. The LAS control the communication activities on a fieldbus segment as described before. In the FOUNDATION Fieldbus system, only the link master device has the LAS capability. If the device is a basic device, the LAS component in the model does not exist. Only one enable LAS is allowable on one fieldbus segment depending the parameter with notation *L*. The Communication component in the model has a receive_frame and a send_data_frame. Their contents depend on the relationship of the Function components, which represent the function blocks distributed in the devices, for example, the PID control algorithm in the controller model. The relationship of function blocks further depends on the control application configuration of the system. Figure 6.27 is the detailed model of the fieldbus segment, in which the CD frame for scheduling and the Data frame are treated separately to give an explicit visualization.

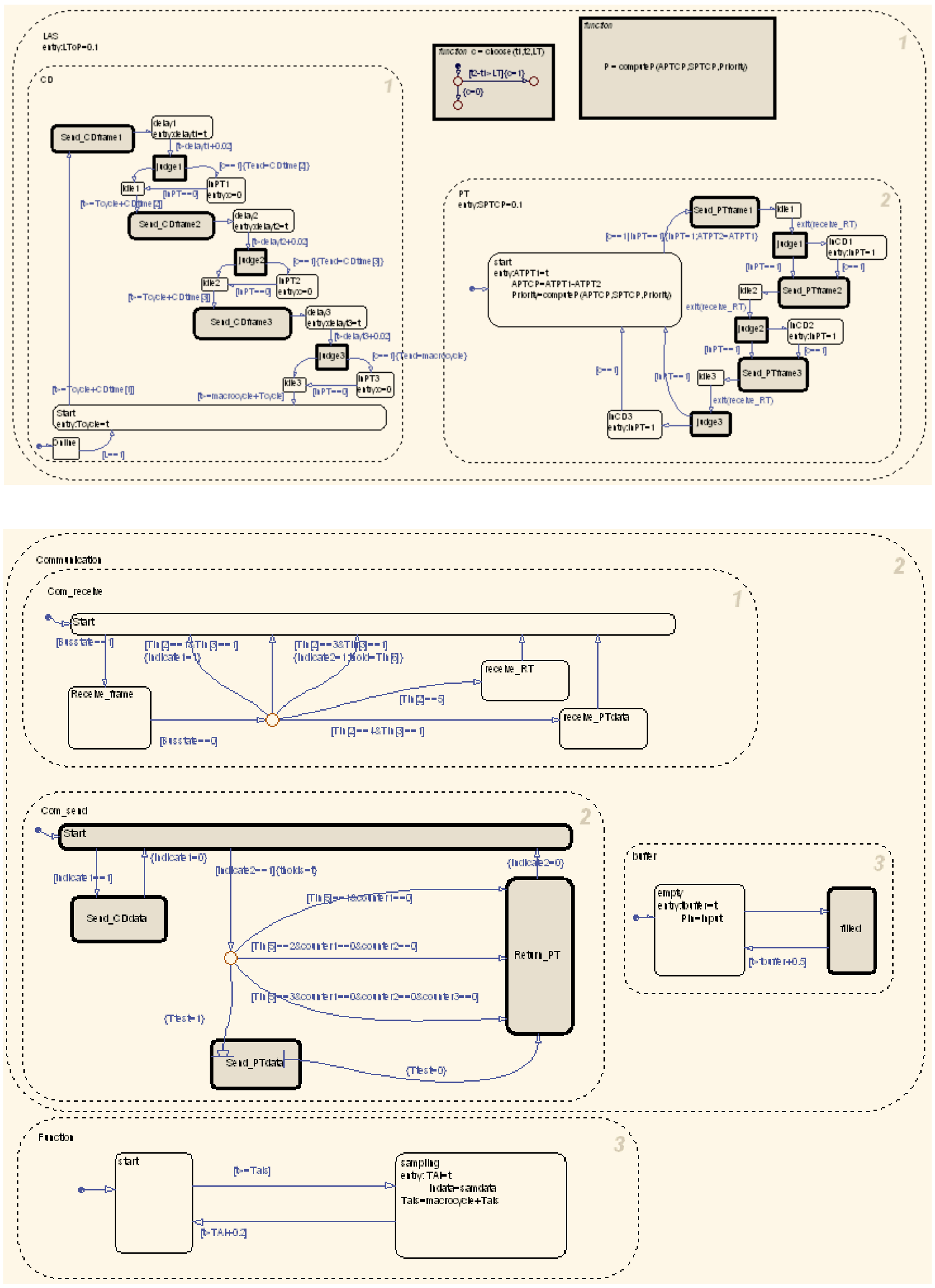


Figure 6.24 Model of fieldbus transmitter

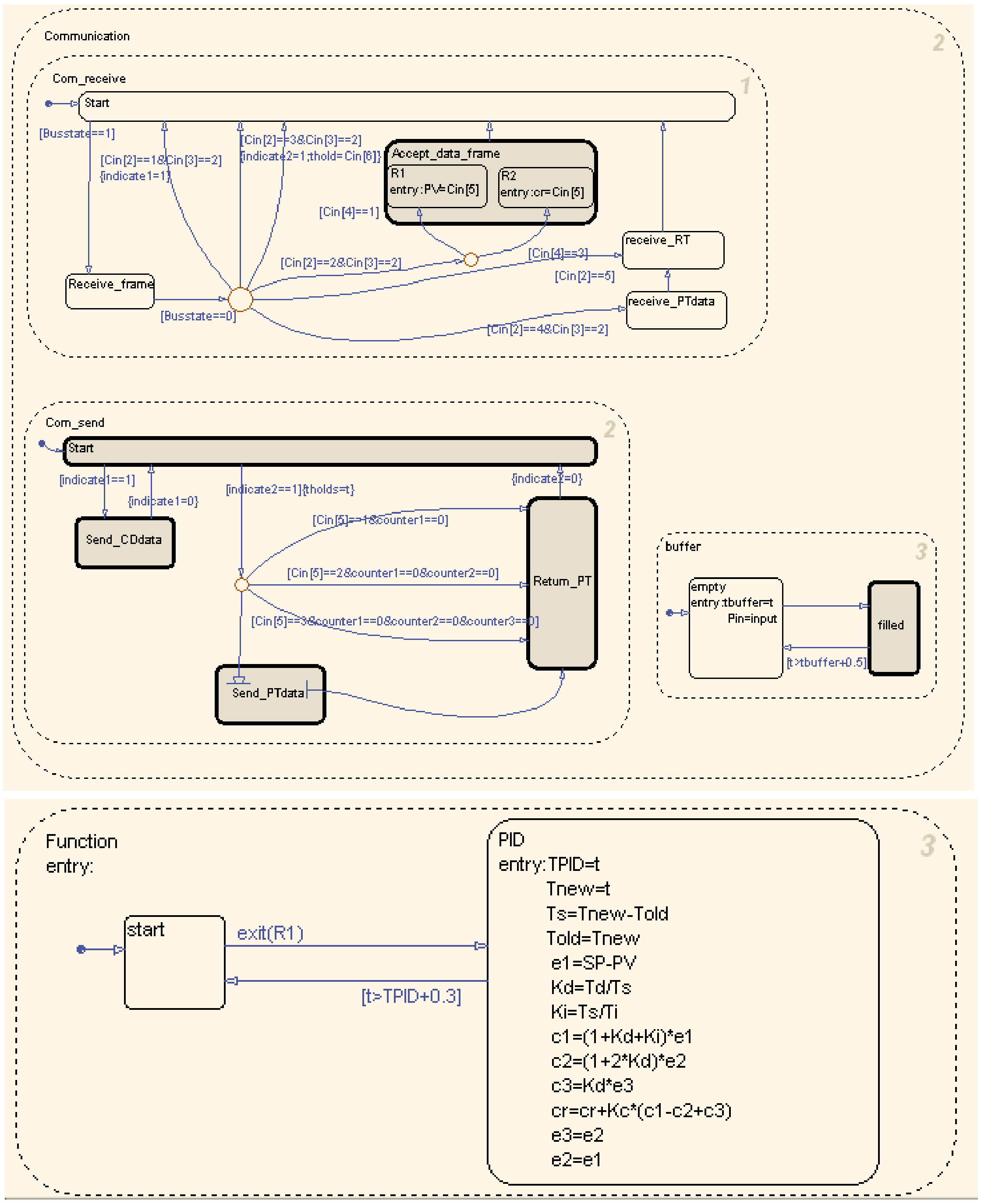


Figure 6.25: Model of fieldbus controller

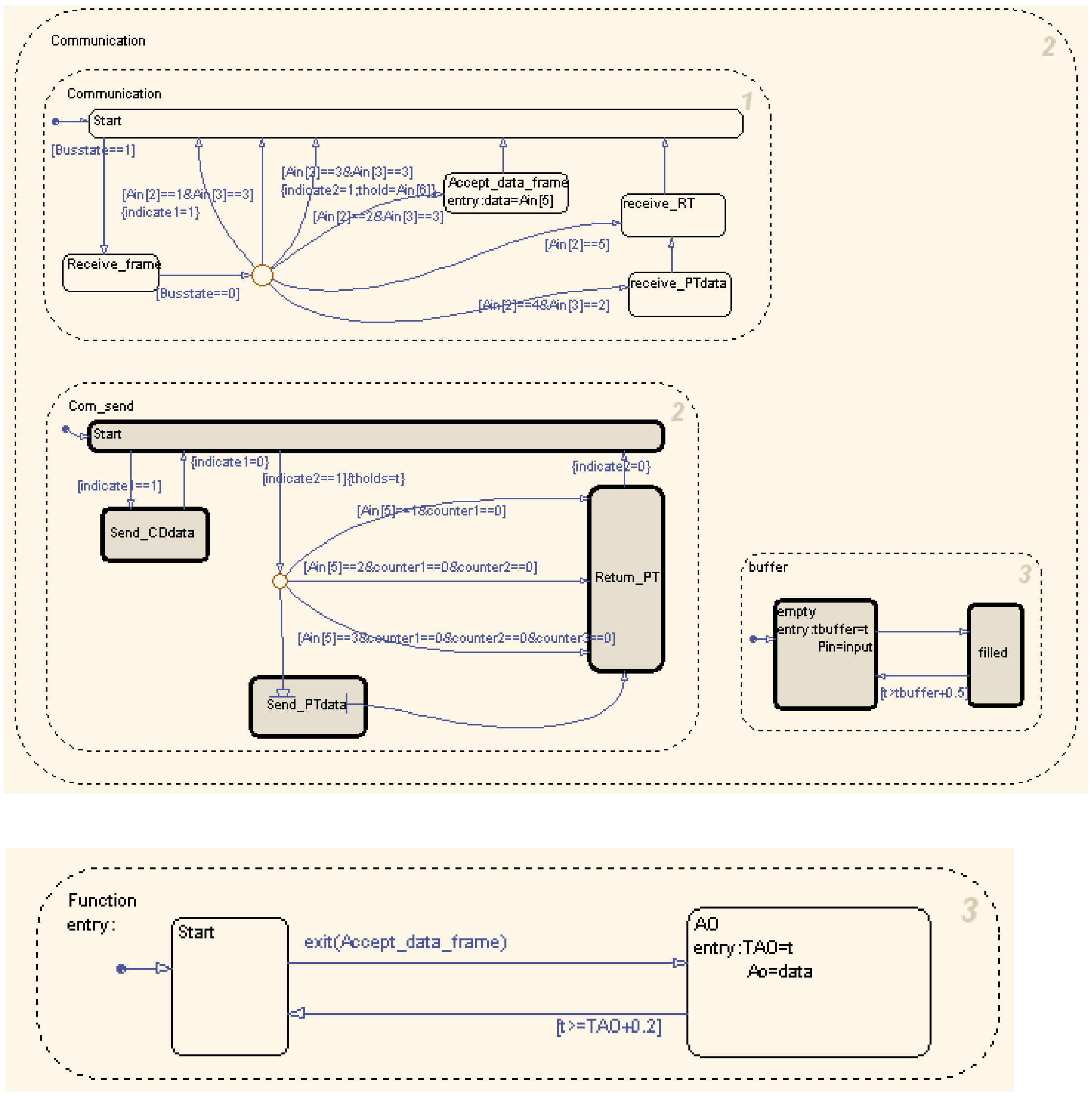


Figure 6.26: Model of fieldbus actuator

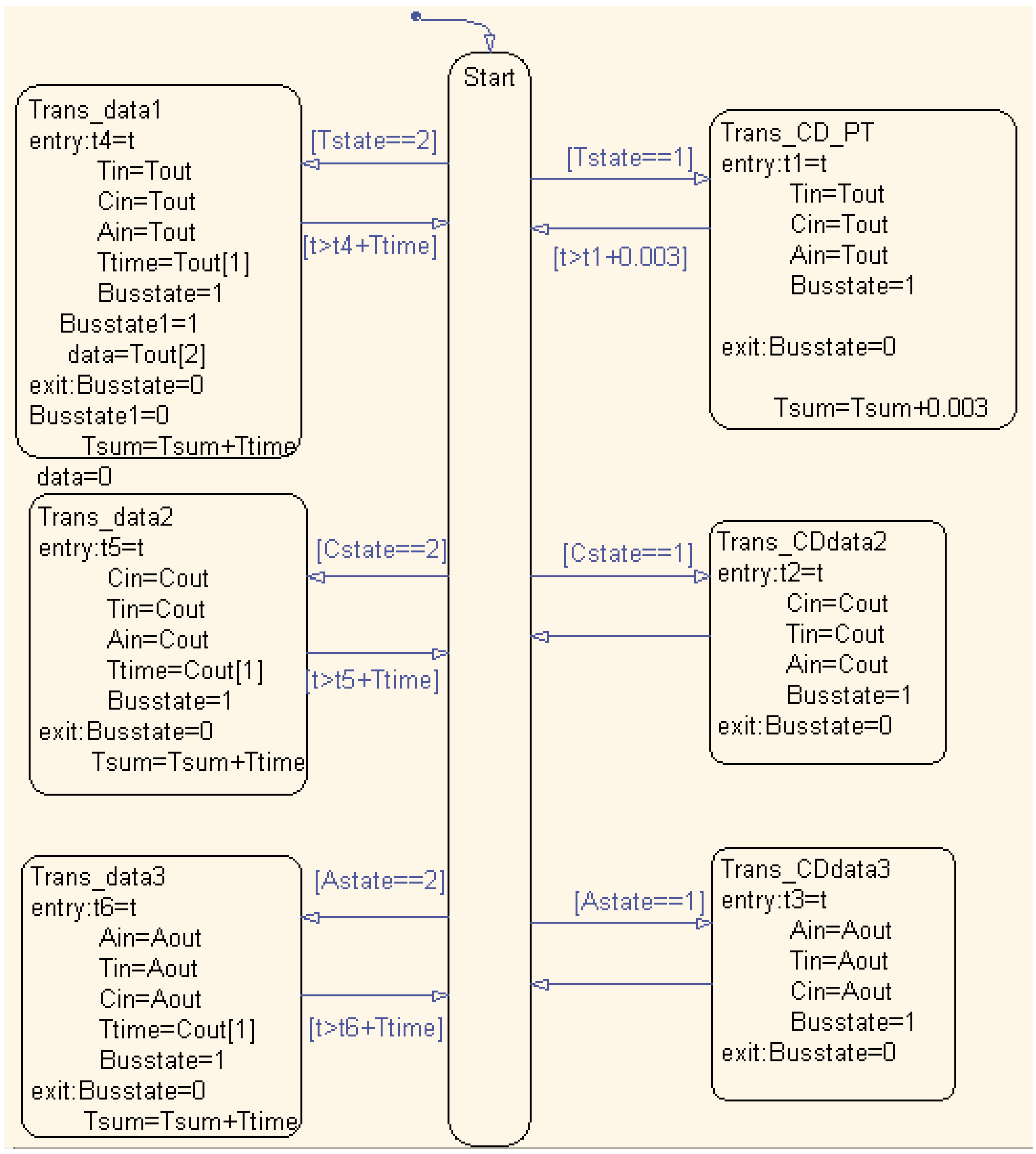


Figure 6.27: Model of fieldbus segment

6.7 Summary

An object-oriented, hierarchical, and hybrid modeling approach is proposed for the fieldbus control system. Based on this approach and the detailed analysis of the FCS principles, the simulation models for the FCS including the fieldbus devices, fieldbus segment and the plant are successfully developed using the Matlab environment. Both the application computation and the communication are considered in the models at the same time. Also both the control and the communication performance can be calculated in the models. It provides us a good platform to study the FCS, for example, to design, validate, analyze or evaluate a control algorithm and the fieldbus performance etc.

Chapter 7

Development and Applications of Fieldbus Systems

7.1 Introduction

This chapter describes two examples of the development and applications based on the fieldbus technology. One is a low-cost control system for process control based on the LonWorks technology. The other is a fieldbus sensor system to monitor the temperature and the level of the underground water well for earthquake forecast. With the two examples, the development procedures are discussed. The implementing technologies of the system hardware and software are illustrated. Finally actual successful applications are presented.

7.2 Development and application of a low-cost control system

7.2.1 Background

In the process control area, distributed control systems (DCS) and control systems consisting of PLC + industrial PC (Agostino et al., 2000, Jimenez et al., 2000) are commonly used for plants with a lot of measurement and control points. The DCS is expensive and often used for complex plant with a large amount of I/O points. The PLC is a centralized control unit. In both PLC systems and in the DCS systems, the PLC or the DCS computer directs the actions of instruments, sensors, and actuators; polls for any results; and manages the resulting data. The essential determination of the system's behavior resides in the PLC or DCS controller. Communication is one-to-one between the controller and each instrument, sensor, or actuator (Eidson et al., 1996). From a technical perspective, centralized control architecture will serve with advantage in application systems where time-critical closely coupled synchronization and high data flow are required.

When devices and/or systems are loosely coupled and the synchronization is less time-critical, the advantages in adopting a distributed control architecture are likely to become apparent (Xie, 1998). Today, the fieldbus technology may migrate control to field devices such as the sensor with greater distribution of function than the DCS technology. On the other hand, in practice for plants with few but distributed geographically measurement and control points, control and recorder instruments are still used. The control system based on instruments has the problems including inefficient mimic visualization and operation, problems with information flows and management, and poor visualization of trending and alarm of process variables. To solve these problems, a kind of low cost control system with distributed architecture and good human-machine-interface (HMI) like the DCS needs to be developed. The advance of the fieldbus and the industrial PC technology makes this idea feasible.

7.2.2 System design goal and architecture

Considering the requirements of general industrial control system, the design goal of the control system to be developed is summarized as follows.

- Distributed and open architecture
- Distributed sensing, monitoring, and control abilities
- Flexible I/O interface to sensors and actuators
- Good and powerful HMI
- Good software interface with third-party software package
- Low cost

According to the system design goal, the system architecture is designed as shown in Figure 7.1. The whole system consists of two layers of networks. The lower layer of the network uses the fieldbus, which connects various smart nodes that are distributed remotely. The upper layer of the network uses the Ethernet, which connects all PCs as operator stations. The fieldbus network is connected with a PC together through an adapter card inserted in the PC. The smart nodes shown in Figure 7.1 have various I/O interfaces to connect sensors and actuators. Distributed computations such as control and data processing algorithms can be implemented in these nodes. In all, there are 10 kinds of smart nodes in our system design. The operator stations will provide powerful and user-friendly HMI for supervision and control. In the system, the analog signals and fieldbus can be mixed. This mixture is an optimal selection for present requirements of control applications.

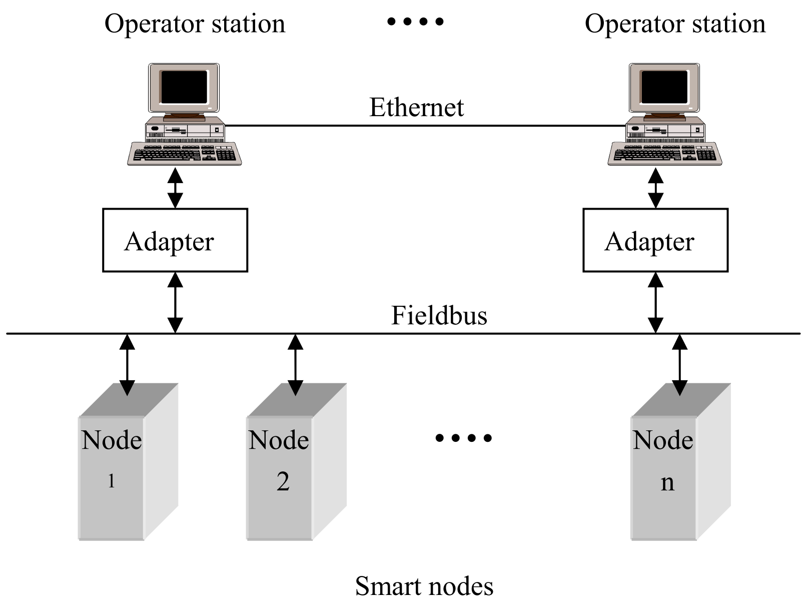


Figure 7.1: Control system architecture based on fieldbus

The system functions of measurement and control depend on the smart nodes. By analyzing the requirements of a general process control, the following ten kinds of smart nodes are designed and developed:

- AI node

AI (Analog input) node is designed to accept the 4-20 mA current or 0-5 VDC voltage analog signals that various process instruments output. It has 8 input channels.

- DI node

DI node is a node that can accept 16 input channels of discrete signals such as dry contact. Logical control algorithms can be implemented locally in this node. Control output is transferred through the fieldbus.

- DO node

DO node is a node that can output 8 output channels of discrete signals. Logical control algorithms can be implemented locally in this node. Input signals are obtained through the fieldbus from other nodes.

- AO node

AO node is a node that can output 4 channels of the 4-20 mA current or 0-5 VDC voltage analog signals. Four of PID control algorithms are embedded in this node. Input signals are obtained through the fieldbus from other nodes.

- PID node

PID node is a controller node with embedded proportional-integral-derivative control algorithms, which are commonly used in process control. It has 4 input channels and 2 output channels of the 4-20 mA current or 0-5 VDC analog signals. Therefore, two closed control loops can be implemented locally and separately in this node.

- DIO node

DIO node in fact is a programmable logic controller with fieldbus communication capability. It has 8 input channels and 4 output channels of discrete signals.

- RTD node

RTD node is designed to accept the signals of standard resistance temperature device such as Pt100 platinum resistance. It has 6 input channels and transforming algorithms from resistance to temperature.

- RTC node

RTC node is same as the RTD node except with two 2 output channels of the 4-20 mA current or 0-5 VDC analog signals and accompanying embedded PID control algorithms for temperature.

- TC node

TC node is designed to accept the signals of standard thermocouples. It has 9 input channels and transforming algorithms from millivolt to temperature depending on the type of thermocouple.

- TCC node

TCC node is the same as the TC node except it has two 2 output channels of the 4-20 mA current or 0-5 VDC analog signals and accompanying embedded PID control algorithms for temperature.

7.2.3 The fieldbus selection

LonWorks fieldbus has been discussed in Chapter 2, and also been evaluated in Chapter 4. It is a general purpose control system technology that can be used to monitor sensors, control outputs, and display system status in a wide variety of applications. LonWorks fieldbus is selected for our development of the low-cost control system because it has the following features:

- Peer-to-peer architecture
- Low node cost
- Inexpensive development cost
- Complete development tools
- Input/Output integration
- Open architecture
- Good compatibility with Windows (DDE, LNS)
- Free topology and long distance (2700m@78kbps)

LonWorks technology includes all of the elements required to design, deploy and support control networks, specifically the following components:

- Neuron chips
- Lontalk protocol
- LonWorks transceivers
- LonBuilder and NodeBuilder development tools

These components make the LonWorks technology be a complete platform for implementing control network systems, and the system designers efficiently and rapidly develop their application systems.

7.2.4 Node hardware development

The heart of a smart node is a microcontroller called the Neuron chip whose internal structure is shown in Figure 7.2. Designed by Echelon Corporation and manufactured and distributed worldwide by Motorola and Toshiba, the Neuron chip is a VLSI device that incorporates communications, control, scheduling, and I/O support needed by a smart node. The Neuron chip enables nodes to communicate with one another using the 7-layer Lontalk protocol which is embedded in every Neuron chip. This protocol supports distributed, peer-to-peer communication that enables individual nodes, such as actuators and sensors, to communicate directly with one another. A central control system is not required.

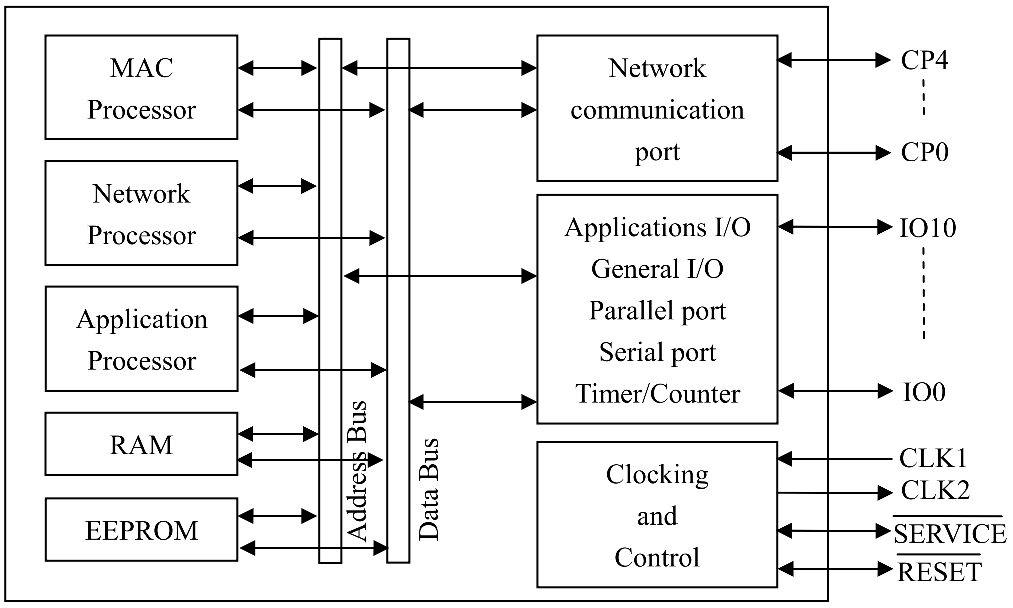


Figure 7.2: Neuron chip block diagram (Motorola, 1996)

The smart nodes we designed have a common block diagram as shown in Figure 7.3. The Neuron 3150 chip is used as the heart of the node with an expansion of 58K bytes flash memory. The transceiver selected is the FTT-10, which supports the twisted pair medium and the free topology. The length of the twisted pair can be up to 2700 meters. Ten kinds of different nodes designed only have different I/O circuitries on the same node structure, for example, the I/O circuitry of the PID node includes an A/D and D/A converters. The Neuron chip connects to application-specific external hardware such as sensor or actuator via eleven pins named IO0 through IO10. These pins may be configured in numerous ways to provide flexible input and output functions with a minimum of external circuitry.

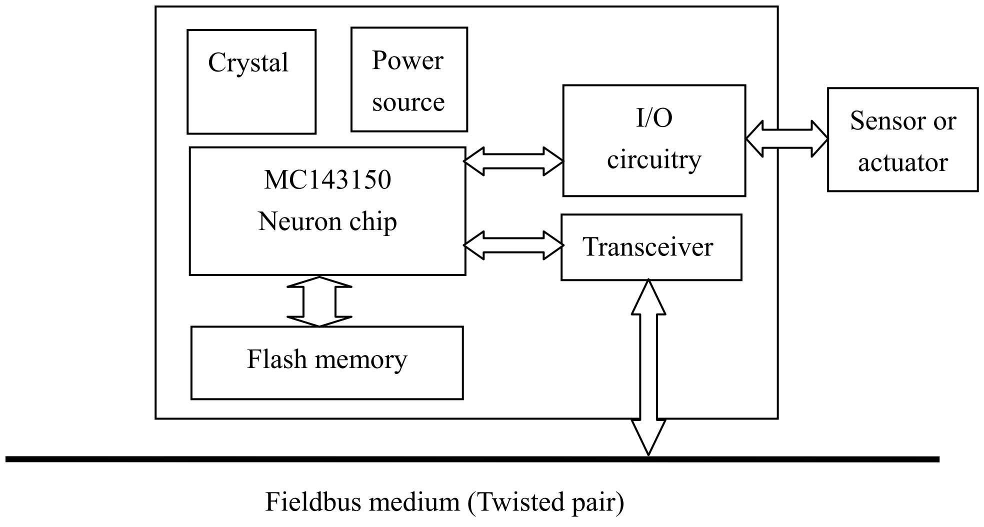


Figure 7.3: Node common block diagram

7.2.5 Node software development

The distributed functions are mostly implemented by node software, which is developed by using the Neuron C programming language. The software of each node to be developed includes three components: I/O interfaces, network communication, and user applications, as shown in Figure 7.4.

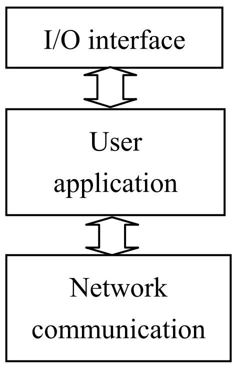


Figure 7.4: Node software components

For the I/O interfaces, the Neuron C programming language provides 34 different I/O objects that support a wide variety of I/O devices. The programmer may declare one or more pins as I/O objects. These I/O objects range from simple bit I/O to complex serial I/O. An object is simply an input or output waveform definition. They can be thought of as prewritten firmware routines in ROM which are accessed by the user's application program.

For the network communication programming, the concept of the network variable is used. It greatly simplifies the programming of complex distributed applications. Network variables provide a very flexible view of distributed data to be operated on by the nodes in a system. The programmer does not need to deal with message buffers, node addressing, request/response/retry processing, and other low-level details. The application program can declare a special class of static objects called network variables, which may be of class *input* or *output*. Assignment of a value to a network variable causes propagation of that value to all nodes declaring an input variable that is connected to the network output variable. For example, an RTD node that acquires an RTD temperature sensor signal could declare an output network variable which contains the current temperature acquired by the node. Every time the node acquires a new value for the temperature, it updates the output network variable. Another nodes or nodes needing to know the current temperature, such as an AO node connecting a control valve, can then declare an input network variable for current temperature. At installation time, the output variable on the RTD node is connected to the input network variable on the AO node. Whenever the AI node acquires a new temperature value, it sends the new value on the network, and the AO node will receive it from the network because of their connection relationship of the network variables. If the AO node has an embedded control algorithm, a closed loop of the temperature control can be built. It can be depicted in Figure 7.5.

For the user application, there are three implementing ways:

- Direct programming using the Neuron C language supported by the development tool:
  It can meet any flexible application.
- Embedding the prewritten application into nodes:
  This way suits to simple and fixed applications.
- Application configuration using third-party configuration software:

This way can satisfy the requirements of complex and flexible applications at the cost of buying the configuration software, which contains a library including common application functions.

Using the second way, we have developed and embedded different prewritten applications for and into different kinds of nodes as shown in Table 7.1. The modified PID control algorithm proposed in Chapter 6 is used in the nodes.

Table 7.1: Embedded application functions in nodes

Node type	Embedded application functions
DI, DO, DIO	Logical operation, Alarm
AI	Filtering algorithm, Totalizer, Alarm
AO	PID control algorithm, Alarm
RTD, TC	Filtering algorithm, Temperature calculation, Alarm
RTC, TCC	Filtering algorithm, Temperature calculation, PID control algorithm ,Alarm
PID	Filtering algorithm, PID control algorithm ,Alarm

7.2.6 HMI software development

The operator station is a Windows-based PC. The LNS (LonWorks Network Services) architecture is used. It provides the foundation for interoperable LonWorks installation, maintenance, monitoring, and control tools. Based on the LCA (LonWorks Component Architecture) data sever (Echelon Co., 1997), a data communication software Lon_Data_Com and an HMI software are developed using Visual Basic 6.0. Figure 7.5 is the block diagram of the HMI software.

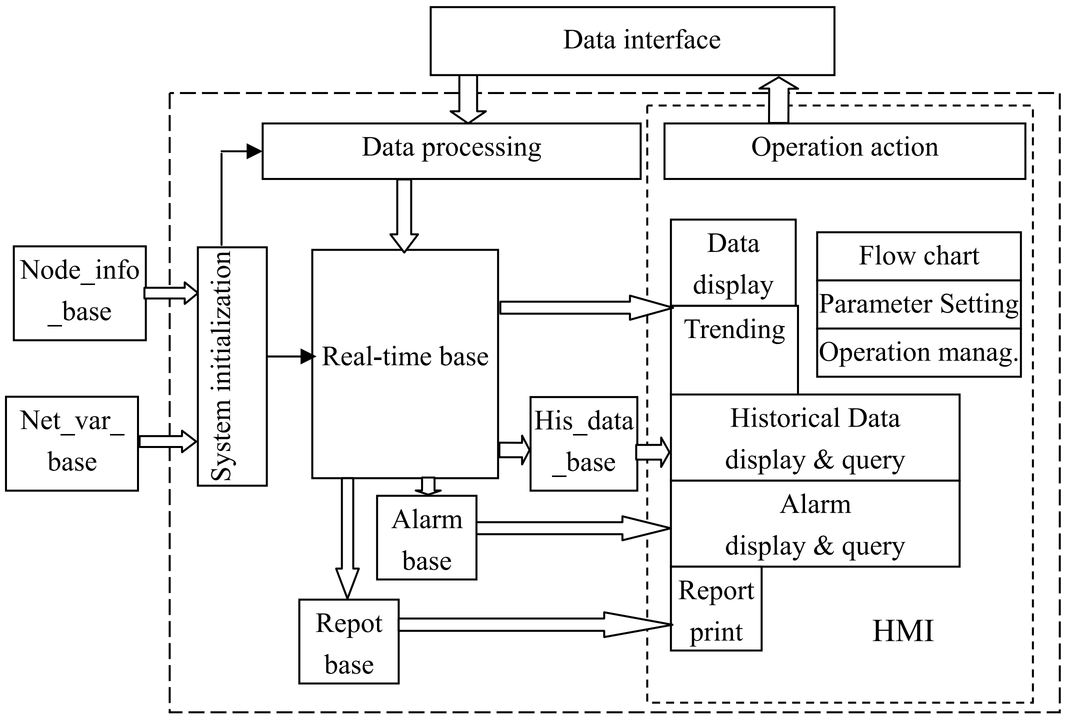


Figure 7.5: Block diagram of HMI software

7.2.7 System application

The systems we developed based on the LonWorks technology have been successfully applied in several industrial process control projects such as the waste water treatment facility, the natural gas separation facility, and the gas and liquid hydrocarbon treatment facility in China. As an example, the application requirements and the system configuration for the gas and liquid hydrocarbon treatment facility are introduced as follows.

The gas and liquid hydrocarbon treatment facility is one of the units in a refinery. At the field device level, traditional analog instruments and control valves are still used. However, the control and data acquisition system needs to be implemented in a distributed form. The requirements of the I/O points for this unit include:

- 4 – 20 mA analog input signal 36 points
- Pt 100 RTD signals 10 points
- E type thermal couple signals 14 points
- Dry contact signals 6 points
- Relay output signals 2 points
- Closed control loops 8

According to the above requirements of the I/O points, the system is designed to have 16 smart nodes including: five AI, five PID, two RTD, two TC, one TCC, and one DIO. Two of operator stations with Windows NT and the PCLTA-10 adapter cards are used in the system, so they are hot stand-by with each other. The system configuration is shown in Figure 7.6. It is worthy to notice that all the control loops and logic control are implemented locally within one node, not via the fieldbus, to guarantee the control performance.

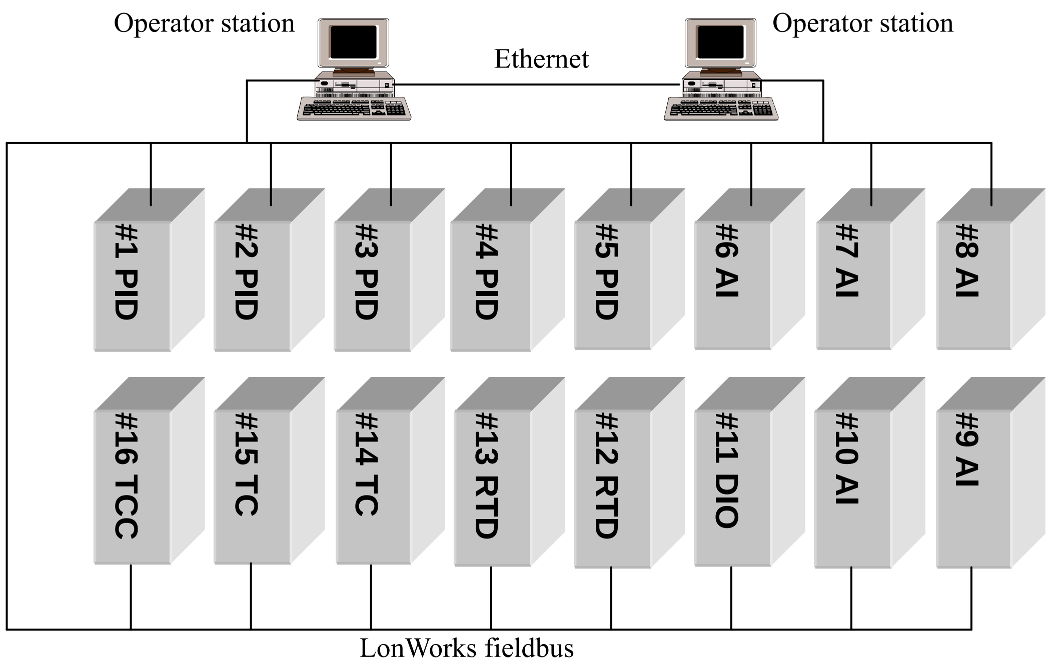


Figure 7.6: System hardware configuration

Figure 7-7 shows one snapshot of the HMI (Human Machine Interface) process flow graphics for the system application in the gas and liquid hydrocarbon treatment facility.

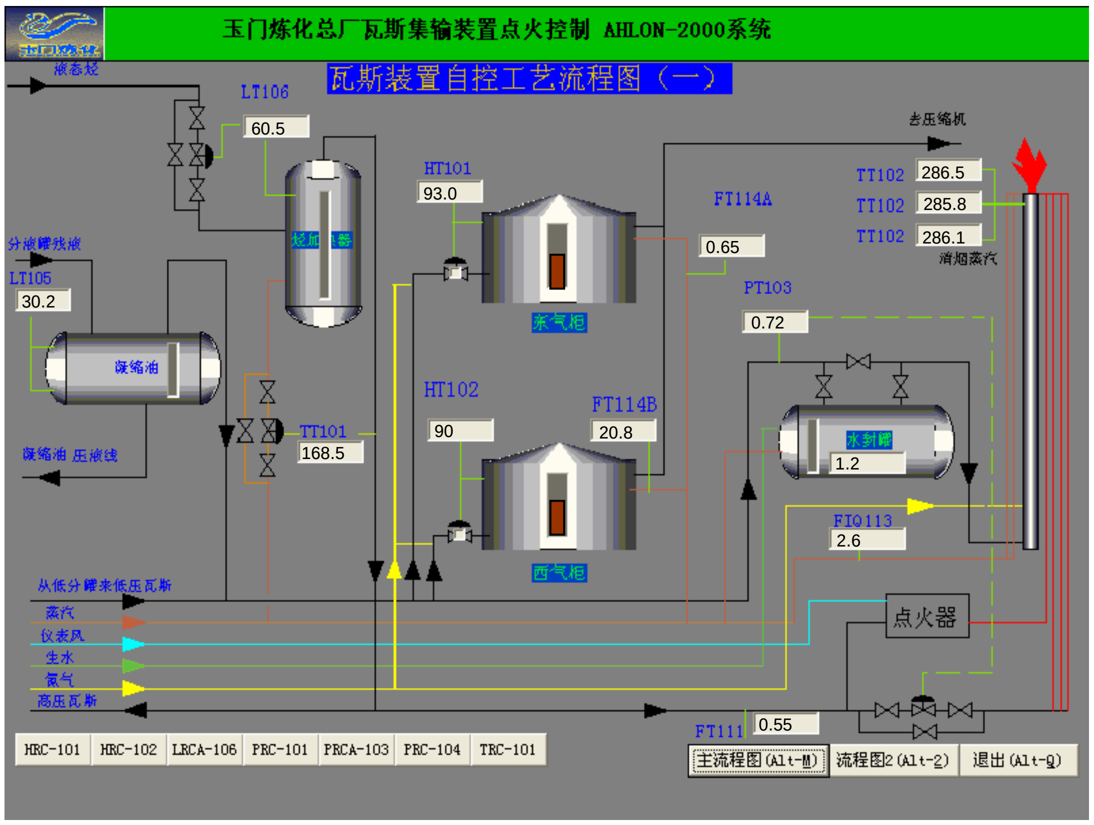


Figure 7.7: Snapshot of HMI process flow graphics for the system application in the gas and liquid hydrocarbon treatment facility

7.3 Development and application of a fieldbus sensor system

7.3.1 Background

In order to forecast an earthquake, it is necessary to monitor the information of the movement of earth in a certain area. One of the methods to do this is to measure continuously the changes of temperatures and levels of several water wells in this area. In the past, the way to implement this method in most areas of China is to install the measuring instruments with paper card recorder at the water well site. And the responsible persons must go to every site to collect the paper cards recorded every day. Then these cards are transferred in a certain period to the monitoring center of this area. In the center, all the data on the cards from all water wells are stored and processed by inputting the data into a computer in which the special processing software is installed. After that, the processing results are presented to the upper organization. This way clearly has the following shortcomings: heavy tasks, needing more persons, the delay of data transfer and processing, low efficiency and accuracy. To solve these problems, a new kind of solution to monitor such information needs to be designed and developed.

7.3.2 Analysis of application requirements

For the application above, two kinds of sub-application were developed. One is the measurement application that is executed locally at the sites of water-wells scattered in a large area. Its function is to get the temperature and level of the water in the well accurately and sensitively. The other is the supervisory application that is executed at the place called monitoring center which is far from the sites of water-well. This application functions as processing all data from each of water well and letting other clients share the result and data. Hence, the measurement application is distributed and the supervisory application is centralized, whereas the clients are distributed. The information exchange must be realized between the two applications, i.e. the two places of their executing. So a communication network connecting these places is required. From the viewpoint of data flow, the processing procedure can be described in Figure 7.8.

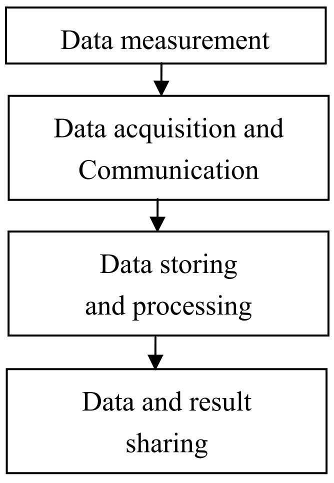

Data measurement
Data acquisition and Communication
Data storing and processing
Data and result sharing

Figure 7.8: Processing procedure of application

Data measurement is to measure the temperature and level of the water in the well by using the related sensors and to send out the results of measurement. These activities are carried out in the site of water well where there is not any person on duty and the public power and telephone network are unavailable in some sites. The temperature sensor is required to be more than 100 meters below the water surface of well whereas the level sensor cannot be at that depth because of the requirement of high sensitivity. So the two sensors cannot be integrated together and must be separated. In our solution, the two sensors are designed to be fieldbus type in order to avoid the signal transmission error and the congestion of cables in the well cylinder, and to reduce the cost of cable. The sensors also must be low power because of the use of battery power where no public power is available.

Data acquisition is to collect all data from all water-wells in one area and then store all data into a database of one computer that is in the monitoring center of the area. Because each well is far from another, generally about 100 kilometers, the way of data transmission is a problem to be solved. The answer is to use the public telephone communication network. There are two selections for it. One is to use the cable telephone network. Another is to use the wireless mobile phone network. Both will be used in our design. The special communication interface devices are developed to connect the public telephone communication network from two sides of sensors at the well sites and computer in the monitoring center. The communication interface device is different depending on the communication side. At the site of water-well, the sensor side, the interface device named as the data concentrator is designed to be low-power consumption as the sensors, and to have the abilities of data store of one week, of data conservation when power down. The interface device is a bridge over the sensors and the public telephone network. It first acquires the data from sensors below the well, then stores and forwards them in a local database and into the monitoring center through the public telephone network.

Data storing and processing are carried out in the computer at the monitoring center. The computer can communicate with all data concentrators at the sites of water-wells by using the communication interface device developed through the public telephone network. The application software on this computer includes the following four components: data communication, database, data presentation and data processing.

In order to realize the sharing of data and results processed by many users who may be in different places, the LAN and a Web server are required. Under such conditions, the users authorized can explore the data and results through the LAN and Internet.

7.3.3 System architecture

Based on the analysis of the system application requirement above, a sensor network should be designed. The sensor network requires the following properties:

- Scalability. This makes the sensor network scalable for its components. When a new component becomes available for the network, it should be joined to the existing sensor network without affecting its operation.
- Easy installation and wiring cost savings.
- Easy maintenance and more information. This means that the sensor network not only provides the measured data, but also the diagnostic information and operational information

which benefits the maintenance of the system.

- Easy to use and cost-effective.

By considering the properties mentioned above, a three-level sensor network system is proposed as illustrated in Figure 7.9.

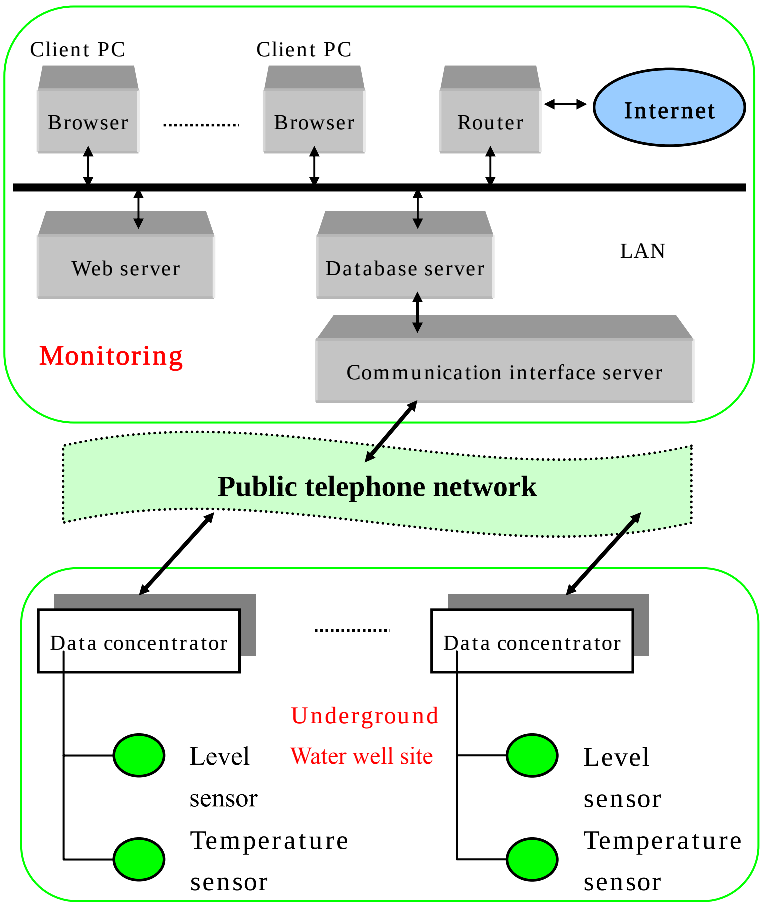


Figure 7.9: System architecture of earthquake monitoring system

At the lowest level of the system, a fieldbus connects all the sensors together for measuring the levels and temperatures of the water-wells that are not far from each other, within 1 kilometer. The fieldbus is a digital, multi-drop, bi-directional and serial communication channel that can meet the requirements of easy installation and maintenance as well as wiring cost savings. In one segment of fieldbus, there must be a data concentrator to work as a master that manages all the sensors connected as slaves in a small area. The hardware and software of the sensor will be discussed in detail below. At the middle level, all the data concentrators in the system are connected through modems to the top-level computer via the PSTN (Public Switched Telephone Network). The connection between the data concentrators and the computer is established separately by dialing automatically or manually depending on the communication requirement. The details of the data concentrator are described below. The top level of the network system is personal computer based platform and a personal computer is required to be the center of the system that completes the tasks including the data acquisition from the sensors in the water-well sites, the management of the data concentrators, the analysis of the data and the presentation of the results. The computer could be a member of a LAN at the top level based on Ethernet. So it can share the data and other resources with other computers in the LAN. The Web based services can also be achieved through Internet at the top level.

7.3.4 System software

As shown in Figure 7.9, from bottom to top, sensor nodes, data concentrator, database and web servers and clients all have their own software. The first two are based on non-PC embedded system. The latter two are based on PC platform with Windows OS. The software and sensor fieldbus protocol (SFP) of sensor node is described in the following section. For the data concentrator, it works as two roles because it is a master in the sensor fieldbus and a slave when communicating with database server in the center via the public telephone network as indicated in Figure 7.10. The software components of data concentrator can be outlined in Figure 7.11.

The component for telephone network functions as the slave of the database server to receive or send message from or to the PC as a database server. The component for sensor fieldbus is to manage the sensor nodes and schedule the sensor communication over the fieldbus as a master. The component for local database is to store and manage the data from sensor nodes. The last component of software for data concentrator is to test and diagnose itself and all sensor nodes on the fieldbus.

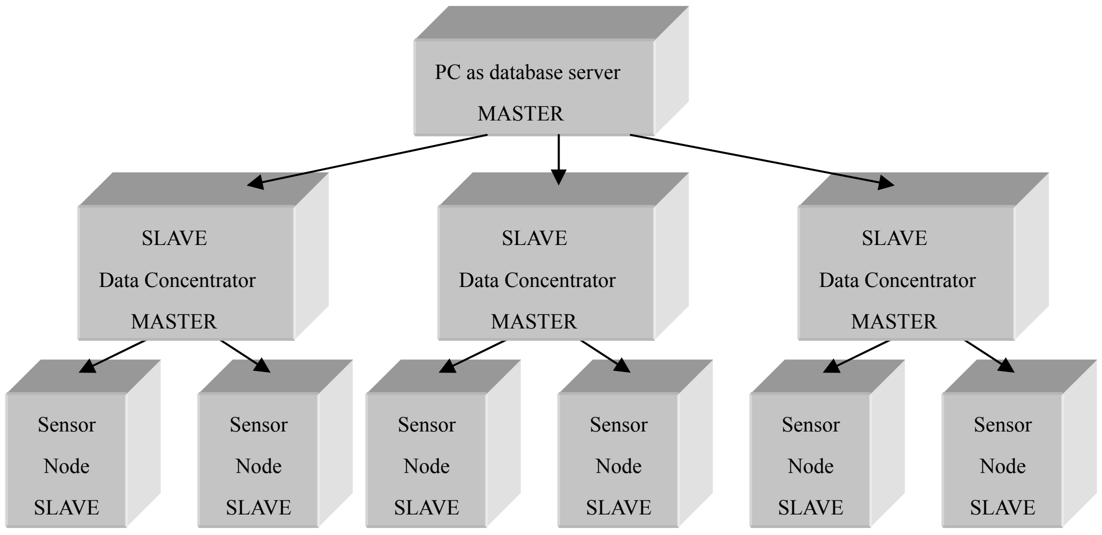


Figure 7.10: Data access structure in software

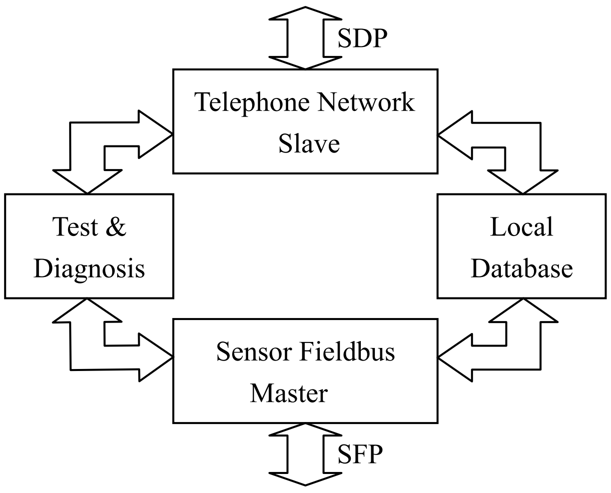


Figure 7.11: Software components of data concentrator

The self-defined protocol (SDP) has been developed for the data transmission via the public telephone network. Figure 7.12 shows the frame formats used by SDP. All frames transmitted by the data concentrator or PC as a database server are preceded by a specified number of hexadecimal "FF" characters which are called the preamble to the frame. They are required in the physical layer protocol to synchronize the receiver. The protocol only uses destination address in each frame, which identifies different data concentrator at different sites of water-well. Command field is one byte long. The commands for communication and management have been defined to meet application requirements. The time stamp field is defined to have two purposes, indicating the data time and solving the problem of communication being broken during data transmission. The data are transmitted as a packet consisting of varying number of frames. If data communication is broken at some middle frame of data packet for some reason, the broken point can be positioned according to the time stamp contained in the last data frame received. Then the data transmission can be redone from that broken point. Therefore, a reliable data transmission can be achieved over the unreliable public telephone network by the time stamp technique. Data field contains the data or parameters having their internal structure. Check field are CRC (cyclic redundancy check) codes used for error detection.

Preamble	Delimiter	Address	Command	Time Stamp	Data	Check

Figure 7.12: SDP frame format

At the layer of monitoring center in Figure 7.9, B/S (Browser/Server) structure is used. So, the client part is totally independent on the application and standard, and it does not need a special software other than the Internet Explorer and OS. The data processing software and the software for communication, database and Web are required on servers. All these should be developed specially for the application. Figure 7.13 depicts the interaction of the software mentioned above. The Web server is system center. It has following functions:

- Accept the HTTP requests from different users by browser.
- Process user requests and return responding static and dynamic web pages.
- Interconnect with database.
- Generate the display curve and/or image.

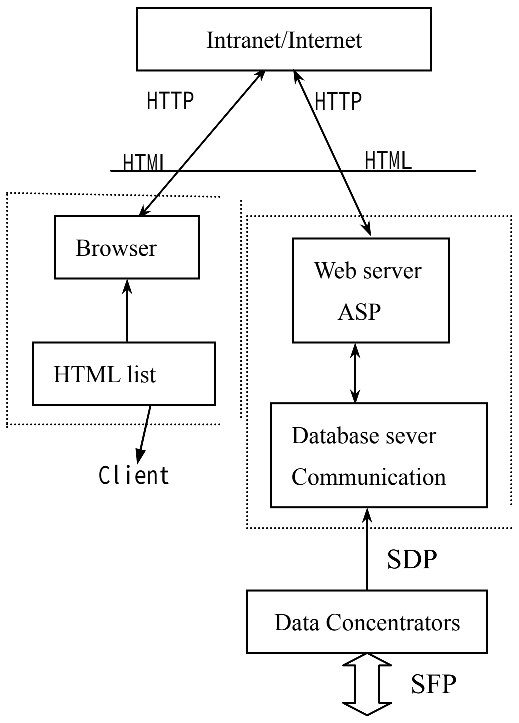


Figure 7.13: Interaction of software components

After the data concentrators collect all data from sensor nodes and pass them to database, the Web server completes almost all of the other system functions while client PC at user side is thin.

7.3.5 Sensor hardware development

The smart sensor contains some or all of the following functions in addition to basic transduction functions: sensor excitation, analog gain, data conversion, control processor, digital signal processing, filtering, digital data bus interface, monitoring and diagnostic functions (Bowen and Smith, 1995). For our application, the sensor architecture is designed. It includes two components of hardware and software as shown in Figure 7.14. The hardware component of the sensor contains a sensor, signal conditioning circuits, microcontroller, and the fieldbus interface. The component of the software mainly encompasses data sampling, data processing, communication protocol, and self-diagnosing.

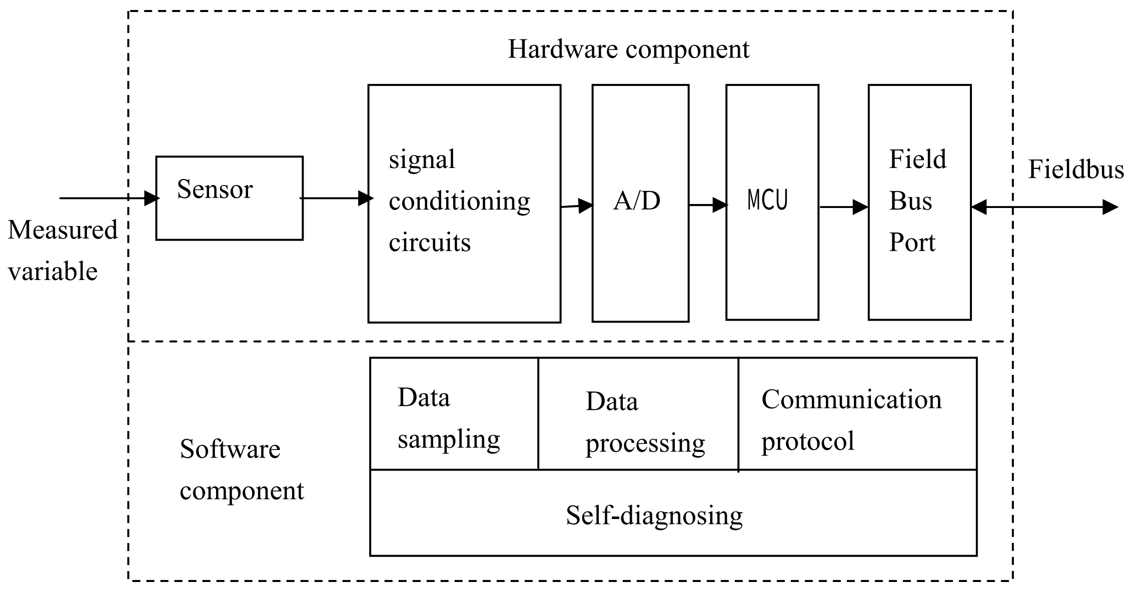
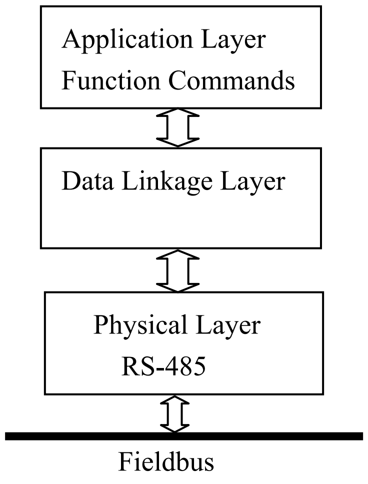


Figure 7.14: Sensor architecture

In the Figure 7.14, the sensor may be a silicon pressure sensor or RTD sensor. The pressure sensor that is used for measuring the water level works on piezoresistance effect. It has a back pressure inlet to avoid the influence of atmosphere variation when measuring the level of water-well. The RTD sensor detects the temperature of deep water-well by using Pt1000 to get a high sensitivity and accuracy. The signal conditioning circuit converts the signal from the sensor to voltage into the A/D converter. The microcontroller as the kernel of the sensor hardware completes all the computation functions. The output of the sensor takes the form of fieldbus signal which is serial, bidirectional and multi-drop communication to decrease wiring harness weight and complexity and to increase more information.

The sensor fieldbus protocol is designed based on the OSI (Open System Interconnection) 7-layer model. It utilizes 3-layer model which includes physical layer, data linkage layer and application layer as shown in Figure 7.15. It is a simplified fieldbus protocol. In the physical layer, RS-485 interface standard is used. In the data linkage layer, master-slave management mode, access method of polling and CRC (Circular Redundancy Check) check are utilized, and the data frame format is defined as shown in Figure 7.16. The application layer is made of the function commands, which are designed according to the requirements of functions on the sensor fieldbus communication.



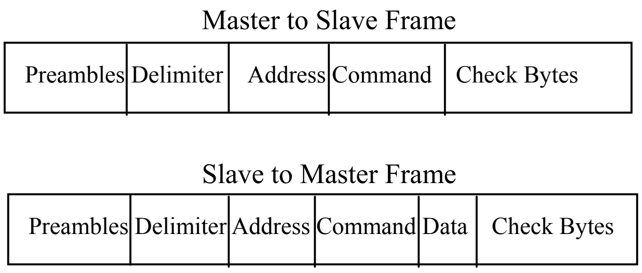


Figure 7.15: Model of sensor fieldbus

Figure 7.16: Data frame format

7.3.6 Concentrator hardware implementation

The data concentrator as the master of the sensor fieldbus is also designed based on the MCU. Figure 7.17 shows its hardware scheme.

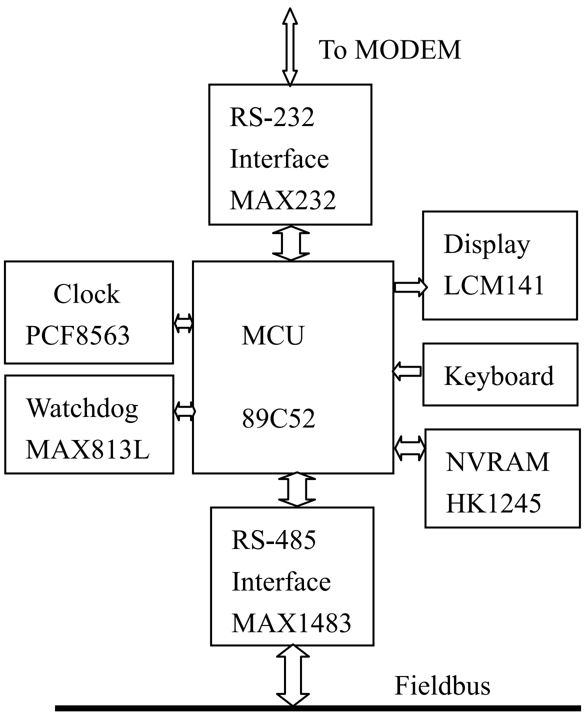
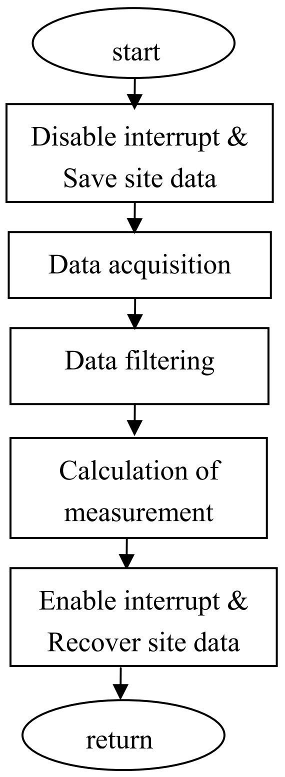


Figure 7.17: Hardware implementation of data concentrator

The data concentrator works as a gateway to achieve the connection between the top PSTN and the lower sensor fieldbus. It also has functions of data storage, display and keyboard.

In the sensor, its software is required to complete two big tasks. One is the internal processing task that includes data acquisition, filtering, calculation of measurement and self-diagnosing. The other is the communication response task to the master, which includes the acceptance of command from the master, command analysis and corresponding response. The two tasks are designed to do in two service programs of interrupt, time interval interrupt for internal processing task and serial communication interrupt for communication response task. The general algorithms of the two programs are shown respectively in Figure 7.18 and Figure 7.19. The main program only does some initiating tasks. When the MCU gets free during completing different tasks, it is forced to sleep to reduce its power consumption.

By analyzing the software requirements of the data concentrator, there are four types of tasks to be done by the software. They are the sensor fieldbus communication task, the PSTN communication task through MODEM, display task and keyboard task. These tasks are driven respectively by the following events: time interval interrupt event for polling, serial communication interrupt event, display refresh time interval event and keyboard scanning event.


start
Disable interrupt & Save site data
Data acquisition
Data filtering
Calculation of measurement
Enable interrupt & Recover site data
return

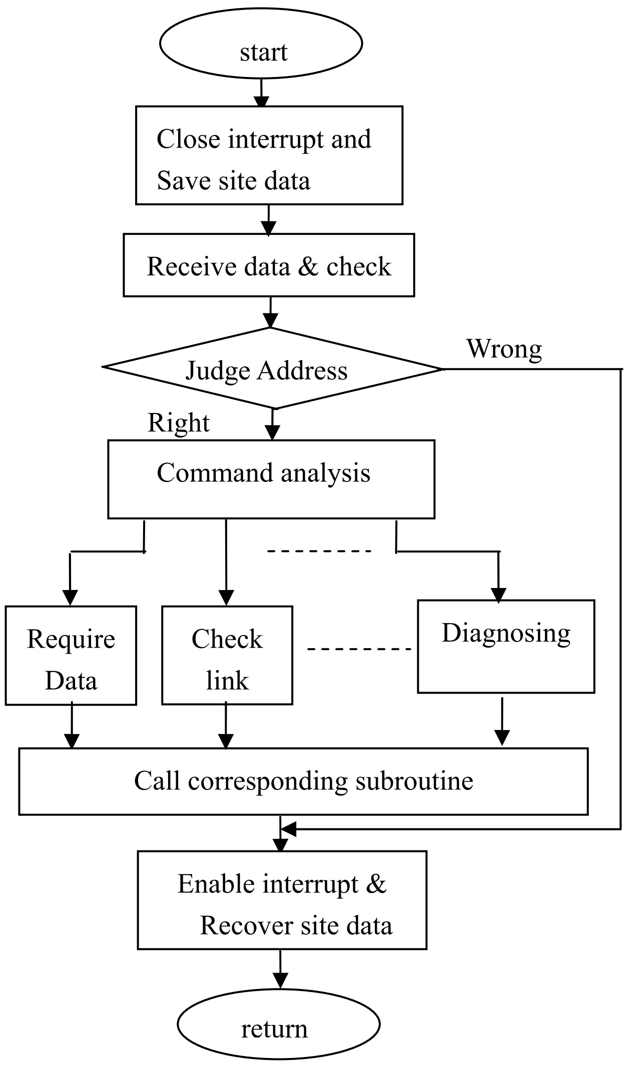


Figure 7.18: Algorithm of internal processing task Figure 7.19: Algorithm of communication task

7.3.7 Application of system

The system designed was put into practical application in the earthquake monitoring center of a city area where there were three water-wells located in three different counties near the city. Figure 7.20 shows the geographical locations of this application.

In this application, two fieldbus sensors for temperature and water level, and one data concentrator were installed at each site of the water-well. The photographs of data concentrator, temperature sensor and level sensor are shown in Figure 7.21. Two computers were used in the monitoring center as a server of database and web, as well as a client. The public telephone network is used as the communication line between the data concentrators in the well sites and the server in the center. Modem devices are needed at both sides. The time stamp technique mentioned above is proven to be very successful by practical application. It achieved the reliable data transmission through the unreliable network, which is casually broken during transmission.

The Windows 2000 Server+ IIS (Internet Information Server) and SQL Server 2000 were used as the system software of the server computer. Software for communication master and data presentation was developed by VB6.0 language. The software for data processing was supplied by a third party. Figure 7.22 shows the measurement results of level and temperature of the water-well, which were presented on the server.

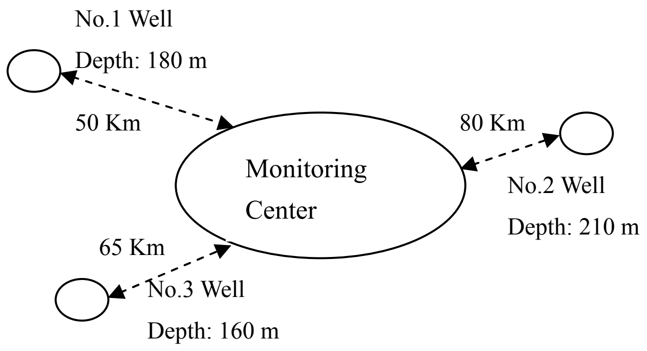


Figure 7.20: Geographical location of application

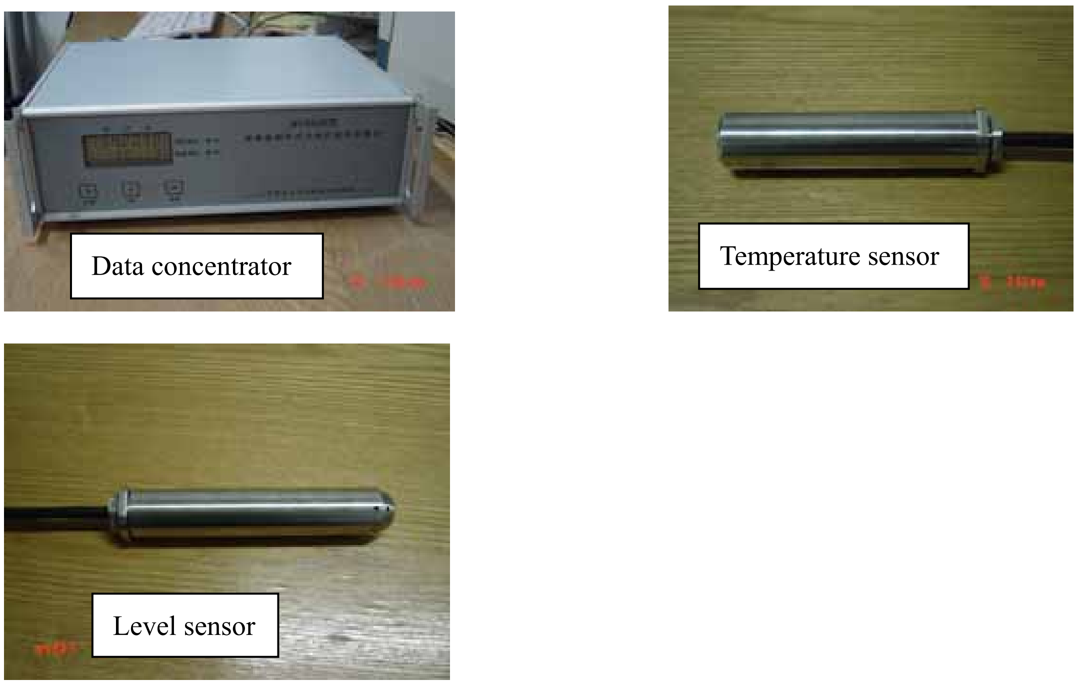
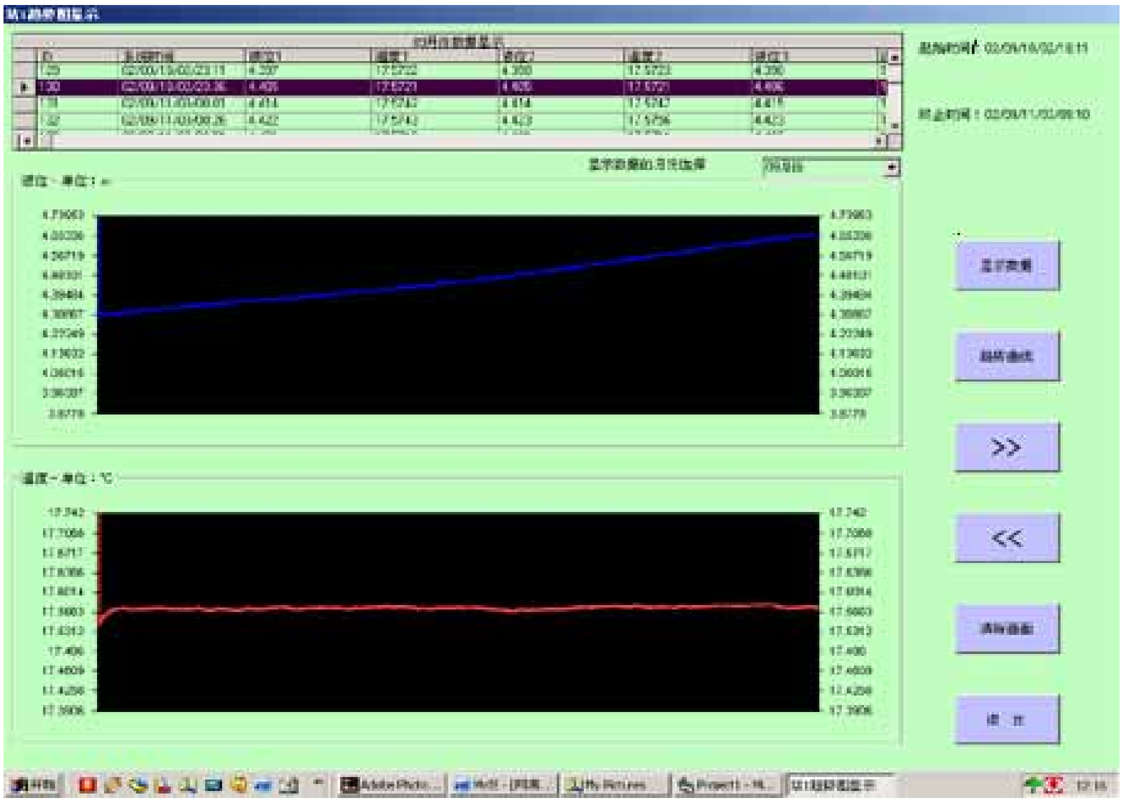


Figure 7.21: Photographs of concentrator and two sensors



ID	[illegible]	液位1	温度1	液位2	温度2	液位3
129	02/09/10/02/23:11	4.397	17.5722	4.398	17.5723	4.398
130	02/09/10/02/23:36	4.405	17.5721	4.405	17.5721	4.406
131	02/09/11/03/00:01	4.414	17.5742	4.414	17.5742	4.415
132	02/09/11/03/00:26	4.422	17.5742	4.423	17.5756	4.423

起始时间：02/09/10/02/18:11

终止时间：02/09/11/03/09:10

Figure 7.22: Historical data of measurement

7.4 Summary

Two fieldbus systems were successfully developed and put into practical applications. The two fieldbus systems were characterized to be scalable, flexible, open, inexpensive and easy to use.

First, a low-cost fieldbus control system was designed by considering general requirements of process control system based on the LonWorks technology. Its development was illustrated by describing the hardware and the software of the node and the HMI software on the operator station. An example of the application was described.

Secondly, a design of distributed system for monitoring the temperature and level underground water-well was proposed and implemented as a total solution. In this system distributed measurement and centralized supervisory were achieved simultaneously. A lot of available technologies in fieldbus sensor, public telephone network, database, and web, were integrated to design and implement the architecture of the system and the structure of software. To overcome the problem of communication being broken during data transmission, the time stamp technique for data message was proposed. The results of application proved that the system design is practical and useful.

Chapter 8

Conclusions and Future Research

In this chapter, we summarize the contributions of this thesis and suggest the directions for future research.

8.1 Summary

A Fieldbus Control System (FCS) is a distributed system composed of field devices and control and monitoring equipments and it is integrated into the physical environment of a plant or factory. The FCS is increasingly being used in the automation arena because of the advantages of the fieldbus. Since 1980s, more than 50 different names of available fieldbuses have emerged. However, researches for the FCS lag far behind their current practice. In this thesis, we proposed a systematic methodology to analyze, design, and develop the FCS by considering several issues in the FCS. Main contributions are summarized as follows:

- **Timing analysis method for FCS**

First, we can characterize and classify different fieldbuses from both the fieldbus protocol models and the fieldbus medium access control mechanisms. Based on them, the timing characteristics of the FCS are analyzed in detail. The control period in the FCS for a control loop is formulated and analyzed. It is equal to the sum of the execution times required by the function blocks distributed in fieldbus devices, the communication times between these function blocks, and the margin time reserved to guarantee all of periodic tasks to be completed within a control period. We proposed methods of determining these times. We also showed the stability condition for normal operation of the fieldbus control loop. This condition gives the lowest bound for the control period of the FCS. The bound is obtained by considering the communication and the control computation at the same time. The stability condition also represents that the communication tasks and the computation tasks are not independent, but have a time precedence relationship with each other. The analysis and experimental results show that the execution times and the margin time are dominant in a control period, whereas the communication time is secondary, and also the execution time of the PID function block might be different if located at different fieldbus devices. Therefore, in order to shorten the control period, the PID function block must be located at a fieldbus device where the execution time is minimal. Also, an appropriate configuration must be chosen to reduce the number of the communication links.

The effects of communication times, computation times, and the jitter of control period on the performance of control are discussed by evaluating the performance criteria such as IAE, ITAE, the overshoot, and the settling time based on the simulation results. In general, increases of communication times, computation times, and the jitter of control period worsen the control performance. The larger these times and jitter are, the worse the control performance is. For a given control period, the distribution changes of computation or communication times in the control period have no effect on the control performance. This result tells us the fact that the different delay components are commutative for a given control period from the control point of view. There are upper bounds for the delays of communication time, computation time, and the jitter of control period, respectively, for a specified performance criteria. In an extreme case, the fieldbus control system becomes unstable if the delays of these times and the jitter exceed the upper bounds.

- **Effective control algorithms to improve FCS's performance**

To overcome the bad effects caused by the delays of communication time, computation time, and the jitter of control period, and to guarantee the control performance of the FCS, control algorithms for the FCS were addressed. First, the delay problem in the FCS is analyzed and categorized from the control theory point of view. There are two models: the pure delay model and the control period model to handle the delays in the FCS. The differences between the two models lead to different designing problems for the control algorithms of the FCS. For the control period model used in our work, the problem is how to select an appropriate control period and how to compensate the period jitter. Therefore, a modified PID control algorithm and a model predictive control algorithm were proposed to overcome the effects of the delays and the jitter on the control performance. The effectiveness of the two control algorithms were analyzed and verified by simulation runs. The simulation results showed that the modified PID control algorithm works well for the common process having the first-order model with dead time. This implies that the stability of the fieldbus control loop can be assured under the unpredictable and varying time delays caused by the fieldbus. The simulation results also tell us the fact that the delay of the plant and the delay of the communication and computation are not commutative. It was shown that the single predictive control algorithm as one of optimal control algorithms has a far better control performance than the modified PID algorithm. If the plant model is not known, the modified PID control algorithm can be used. It can give an acceptable control performance even in the case of a large jitter range of control period. The predictive control algorithm can give better control performance if the plant model is obtainable and the jitter range of the control period is small. Furthermore, the parallel computation way for the modified PID control algorithm was proposed.

- **Modeling and simulating approach for FCS**

By analyzing the FCS in detail, we proposed an object-oriented, hierarchical, and hybrid modeling approach for the FCS. Based on this approach, the simulation models for the FCS including the fieldbus devices, fieldbus segment and the plant were successfully developed using the Matlab environment. Both the application computation and the communication are considered in the models at the same time. Also, both the control and the communication performance can be calculated in the models. It provides us a good platform to study the FCS, for example, to design, validate, analyze or evaluate a control algorithm and the fieldbus performance.

- **Evaluation techniques of fieldbuses**

By analyzing the requirements of data communication and installation environment for fieldbuses from the user standpoint, we proposed a hierarchy performance criteria based on the hierarchy model of the fieldbus protocol. A complete set of evaluation metrics, both quantitative and qualitative, were given while considering the special requirements of fieldbus. The calculation formula of the efficiency of message and the response time for the commonly used CAN fieldbus are given as an example. The static indices of 6 types of fieldbuses were also proposed. As a case study, the experimental results for the FOUNDATION Fieldbus were presented, including the indices of efficiency of message, response time, and efficiency of fieldbus. The simulation results show that the priority mechanism for the acyclic message transfer on the FOUNDATION Fieldbus works effectively. Both experimental and simulation conclusions conform to those obtained by analyzing the mechanism of the FOUNDATION Fieldbus protocol. A general procedure for selecting a fieldbus system was shown. A complete set of detailed indices for the technical data and the strategic criteria were proposed in order to support system designers and users to select an appropriate FCS.

- **Development and applications of FCSs**

By mastering the fieldbus technology and analyzing the application requirements, we developed two fieldbus systems, and put them successfully into practical applications. The two fieldbus systems are characterized to be scalable, flexible, open, low-cost, and easy to use. A low-cost fieldbus control system was designed based on the LonWorks technology. Its development was illustrated by describing the hardware and the software of the node and the HMI software on the operator station. A design method of distributed system for monitoring the temperature and level underground water-well was proposed and implemented as a total solution. In this system distributed measurement and centralized supervisory were realized simultaneously. A lot of available technologies in fieldbus sensor, public telephone network, database, and web, were integrated to design and implement the architecture of the system and the structure of software. The results of application proved the system design to be practical and useful.

8.2 Future research

There are many unanswered questions in the relatively new FCS field. The research presented in this thesis provides a foundation for future research efforts for the FCS. In the following, we explore a few directions for future research.

New fieldbus medium access control methods should be researched. A dynamic method can be proposed based on the closed-loop control concept with the objective of satisfying both the communication and application performance. The property of application data can be used to control the data transmission. For example, unchangeable or little changeable data can be deleted from transmission queue to save the fieldbus bandwidth in order to enhance other data response time. The idea is to couple the design of the communication and the application instead of the current separate design of the communication and the application.

Models and simulation software for more complete fieldbus protocols should be further considered. More functions such as monitoring more communication performance criteria should be enriched. New fieldbus protocols and control algorithms can be designed and validated using such simulation tool. The simulation speed in the Matlab environment becomes slow when the number of fieldbus nodes increases. How to accelerate the simulation is another challenge.

More effective or optimal control algorithms for the FCS should be proposed in order to achieve better or optimal control performance under the time-varying delays. The digital fieldbus devices enable various flexible control algorithms such as the switched control algorithms. Parallel computation mechanism at the fieldbus device level should be also investigated to implement complex control algorithms in the fieldbus devices with low computing capability.

The interoperability of heterogenous fieldbuses at application level needs research under the situation where many fieldbuses coexist.

BIBLIOGRAPHY

Agostino, Cesar, Ahualli Federico, Belisario, Nahimir, and Bustamante Eduardo; “Automation of a Direct Reduction Plant Using PLC/PC,” IFAC Intelligent Components and Instruments for Control Applications, 341-346, Buenos Aires, Argentina (2000)

Alberto, L.-Garcia and Indra Widjaja; Communication Networks: Fundamental Concepts and Key Architectures, p.102, McGraw-Hill, New York, U.S.A. (1999)

Almeida, L., R. Pasadas and J.A., Fonseca; “Using a Planning Scheduler to Improve the Flexibility of Real-Time Fieldbus Networks,” Control Engineering Practice, 7, 101–108 (1999)

Angel, M. Leon Chavez; “Quality of Service and Fieldbuses,” 4th IFAC International Conference on Fieldbus Systems and Their Applications, 185-189, Nancy, France (2001)

Andrew, C.; “Fieldbus: The Foundation for Field Control Systems,” Control Engineering, 41, 77-80 (1994)

Andrew, S. Tanenbaum; Computer Networks, 3rd ed., Tsinghua Press & Prentice Hall, Inc., Beijing, China (1997)

Audsley, N. C., A. Grigg, “Timing analysis of the ARINC 629 databus for real-time applications,” Microprocessors and Microsystems, 21, 55-61 (1997)

Barretto, M.L.; Courtiat, J.-P.; De Saqui-Sannes, P.; “Experience in Using Estelle for the Specification and Verification of a Fieldbus Protocol. FIP,” Computer Networking - COMNET 90, 295, (1990)

Beldiman, Octavian and Gregory C. Walsh; “Predictors for Networked Control Systems,” Proceedings of the American Control Conference, 2347 – 2351, Chicago, U.S.A. (2000)

Blevins, T. , Duffy, J.; Willems, R.; “DCS Integration of Fieldbus,” ISA TECH/EXPO Technology Update Conference Proceedings, 51, 753-762, Chicago, USA (1996)

Blum, I. and G. Juanole; “Comparing the Networks CAN and ARINC 629CP with Respect to the Quality of the Service Provided to an Automatic Control Application,” Proceedings of the Fieldbus Conference, 128-135, Magdeburg, Germany (1999)

Boettcher, J., Traenkler, H.R.; “Trends in Intelligent Instrumentation,” IFAC Proceedings Series, 15, 241-248, (1990) Bowen, Mark, Gerry Smith; "Considerations for the Design of Smart Sensors," Sensors and Actuators, A 46-47, 516-520 (1995)

CAN in Automation, International Users and Manufacturers Group; CANopen Application Layer and Communication Profile, CiA Draft Standard 301, Revision 4.01 (2000)

Cavalieri, S., S. Monforte; "Schedulability Analysis of Periodic and Asynchronous Information Flow in IEC 61158 Type 1 Fieldbus," 4th IFAC International Conference on Fieldbus Systems and their Applications, 127- 134, Nancy, France (2001)

Cavalieri, S.; "Exploiting Data Link Layer Features to Realize Communication Models in Fieldbus System," Computer Standards and Interfaces, 18, 139–158 (1996)

Cena, G., L. Durante, A. Valenzano; "Standard Fieldbus Networks for Industrial Applications," Computer Standards & Interfaces, 17, 155-167 (1995)

Cena, G., A. Valenzano; "New Efficient Communication Services for ISO 11898 Networks," Computer Standards & Interfaces, 22, 61-74 (2000a)

Cena, G., A. Valenzano; "Delay Analysis of Priority Promotion Systems," Computer Communications, 23, 1252-1262 (2000b)

Cena, G., L. Durante, A. Valenzano; "A New CAN-like Field Network Based on a Star Topology," Computer Standards & Interfaces, 23, 209-222 (2001)

Chidambaram, M.; Computer Control of Processes, Alpha Science International Ltd., Pangbourne, UK (2002)

Crowder, Robert S.; "Fieldbus Performance," SHIP STAR Associates, www.shipstar.com (1996)

Cutler, C.R., Ramaker, B.C.; "Dynamic Matrix Control - A Computer Control Algorithm," Automatic Control Conference, San Francisco, U.S.A (1980)

Decotignie, J.-D.; "A Perspective on Ethernet – TCP/IP as a Fieldbus," 4th IFAC International Conference on Fieldbus Systems and their Applications, 185-189, Nancy, France (2001)

DeltaV^{TM} website, www.easyDeltaV.com (accessed in 2003)

Dietmar, Loy, D. Dietrich, H.-J. Schweinzer; Open Control Networks, LonWorks/EIA 709 technology, Kluwer Academic Publishers, London, UK (2001)

Durante, Luca, Adriano Valenzano; "On the Performance of the IEC 61158 Fieldbus," Computer Standards & Interfaces, 21, 241- 250 (1999)

Echelon Co.; LCA Object and Data Server Programmer's Guide, Palo Alto, U.S.A (1997)

Egan-krieger, G. V., T. Stein, J. Rahn; “Object Oriented Device Control Using the CAN Bus,” Nuclear Instruments and Methods in Physics Research A, 352, 204-206 (1994)

Eidson, J.C., Steven A. Siano, Stan P. Woods; “A Research Prototype of a Control System Based on Smart Transducers,” ISA Transactions, 35, 17-24 (1996)

European Committee for Electrotechnical Standardization; “General Purpose Field Communicaton System,” vol.2, (PROFIBUS), EN50170/2 (1996)

European Committee for Electrotechnical Standardization; “General Purpose Field Communicaton System,” vol.3, (FIP), EN50170/3 (1996)

Farsi, Mohammad and Manuel Bernardo Martins Barbosa; CANopen Implementation: applications to industrial networks, pp. 36-39, Research Studies Press Ltd. Baldock, Hertfordshire, England (2000)

Fieldbus Foundation; “Foundation ^{TM} Specification,” Austin, U.S.A. (1996)

Fieldbus Foundation; “Technical Overview, Foundation ^{TM} Fieldbus,” Austin, U.S.A. (1998)

Franco, L. R.H.R. and Henriques, A. M.; “Using AI Algorithm to Improve Consistencies in Real-Time Fieldbus,” 4th IFAC Symposium on Intelligent Components and Instruments for Control Applications, 227–232, Buenos Aires, Argentina (2000)

Franco, L.R.H.R.; “Transmission Scheduling for Fieldbus: A Strategy to Schedule Data and Messages on the Bus with End-to-End Constraints,” Proceeding of International IEEE Joint Symposia on Intelligence and Systems – Intelligence in Automation and Robotics, 148-155, (1996)

Gelenbe, Erol et. al.; Network System Design, Gordon and Breach Science Publishers, New York, U.S.A. (1999)

Glanzer, David A., Cianfrani Charles A.; “Interoperable Fieldbus Devices: a Technical Overview,” Proceedings of the 1996 Anniversary Symposium on Instrumentation in the Chemical and Petroleum Industries, 33-40, Cleveland, U.S.A. (1996)

Henderson, W., D. Kendall A. Robson; “Improving the Accuracy of Scheduling Analysis Applied to Distributed Systems,” International Journal of Time Critical Computing Systems, 20, 5-25 (2001)

Hofstee, J.W., D. Goense; “Simulation of a CAN-based Tractor-Implement Fieldbus according to DIN 9684,” Journal of Agriculture Engineering Research, 68, 89-100 (1997)

Hofstee, J.W., D. Goense, “Simulation of a Controller Area Network-Based Tractor-Implement Data Bus according to ISO 11783,” Journal of Agriculture Engineering Research, 73, 383-394 (1999)

Hohwiller, Luc, Wendling Serge; “Fieldbus Network Simulation Using a Time Extended Estelle Formalism,” IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems - Proceedings, 92-97 (2000)

Hong, S. H., Lee Seong Geun; “Performance Analysis of the Data Link Layer in the IEC/ISA Fieldbus by Simulation Model,” IEEE Symposium on Emerging Technologies & Factory Automation, 2, 593-598, Kauai, U.S.A. (1996)

Hong, S. H., Seong Jun Ko; “Analysis of Real-Time Data Transmission in the DLL of IEC/ISA Fieldbus,” Proceeding of ISIE'98, 694-699 (1998)

Hong, S. H.; “Scheduling Algorithm of Data Sampling Times in the Integrated Communication and Control Systems,” IEEE Transactions on Control System Technology, 3(2), 225–230 (1995)

International Standard Organization; “Road Vehicles-Interchange of Digital Information -Controller Area Network for High-Speed Communication,” ISO 11898 (1993)

Jimenez, E., J.M. Miruri, F.J. Martinez de pison, and M.Gil; “Supervised Real-Time Control with PLCs and SCADA in Ceramic Plant,” 6th IFAC workshop on Algorithms and Architectures for Real-Time Control, 223-228, Palma de Mallorca, Spain (2000)

Leen, G., D. Heffernan; “TTCAN: a New Time-Triggered Controller Area Network,” Microprocessors and Microsystems, 26, 77-94 (2002)

Lenhart, Gerald W.; “Fieldbus-Based Local Control Networks,” InTech, 41(8), 31-34 (1994)

Lewis, R.; “Modeling Control Systems Using IEC 61499,” 22, the Institution of Electrical Engineers, U.S.A (2001)

Lian, F.L; “Analysis, Design, Modeling, and Control of Networked Control System,” PhD Thesis, University of Michigan, 40- 45 (2001)

Lindner, Klaus-Peter; “Fieldbus: a Milestone in Field Instrumentation Technology,” Measurement and Control, 23 (9), 272-277 (1990)

Livani, M.A., J. Kaiser, W. Jia; “Scheduling Hard and Soft Real-Time Communication in a Controller Area Network,” Control Engineering Practice, 7, 1515-1523 (1999)

Leon-Garcia, A. and I. Widjaja; Communication Networks: Fundamental Concepts and Key Architectures, p102, McGraw-Hill, New York, U.S.A. (1999).

Li, Lingqi, Hanasaki Koichi, Xiangyu Wei, Yanbin Pang, Zheng Liu, Youhua Wu; "Integration of fieldbus into DCS," Proceedings of the SICE Annual Conference, 1043-1046, Morioka, Japan (1999)

Liu, J.W.S; Real-Time Systems, pp. 43- 44, Higher Education Press & Pearson Education, Beijing, China (2002)

Mahalik, N.P.; Moore P.R.; "Fieldbus Technology Based, Distributed Control in Process Industries: A Case Study with LonWorks Technology," Integrated Manufacturing Systems, 8(4), 231-243 (1997)

Mahalik, N.G.P.C., S.K. Lee; "A Study on Production Line Automation with LonWorks Control Networks," Computer Standards & Interfaces, 24, 21-27 (2002)

Martins, E. and J. A. Fonseca; "Improving Flexibility and Responsiveness in FTT-CAN with a Scheduling Coprocessor," Proceedings of the 4th IFAC Conference on Fieldbus Systems and their Applications, 91-97, Nancy, France (2001)

Motorola; LonWorks Technology Device Data, Phoenix, U.S.A. (1996)

MTL Instruments Group; "Intrinsic Safety Principles and Practice," Hazardous Area Products Data Resource (2001)

Navet, N., Y.Q. Song; "Reliability Improvement of the Dual-proority Protocol under Unreliable Transmission," Control Engineering Practice, 7, 975-981 (1999)

Navet, N., Y.-Q. Song, F. Simonot; "Worst-case Deadline Failure Probability in Real-time Applications Distributed over Controller Area Network," Journal of Systems Architecture, 46, 607-617 (2000)

Navet, N., Y.-Q. Song; "Validation of In-vehicle Real-time Applications," Computers in Industry, 46, 107-122 (2001)

Open DeviceNet Vendor Association; DeviceNet Specification, vols.1 and 2，Boca Raton, FL (1994)

Pang, Yanbin and Hirokazu Nishitani; "Evaluation Techniques of Fieldbuses", SICE Annual Conference, 1679-1684, Fukui, Japan (2003)

Pang, Yanbin，Xiangyu Wei, Youhua, Wu; "Sensor Network Based on LonWorks Technology," Proceedings of the SICE Annual Conference, 897-900, Morioka, Japan (1999)

Park, Eik-Dong, Uk-Youl Huh; "Stochastic Control of Communication Network with Time Varying Delay," 6th IFAC Workshop on. Distributed Computer Control Systems, Sydney, Australia (2000)

Patzke, Robert; "Fieldbus Basics," Computer Standards & Interfaces, 19, 275-293 (1998)

Raja, P., J. Hernandez, L. Ruiz, F. Guidec and J. D. Decotignie; "Simulating Fieldbus Applications with DRUGH Simulator," Computers in Industry, 27, 43-51 (1995)

Rawlings, J.B., Meadows, E.S. and Muske, K.R. Nonlinear model predictive control: a tutorial and survey. In Proc. ADCHEM'94, Kyoto, Japan (1994)

Reza, S. Raji; "Smart networks for control," IEEE Spectrum, 31(6), 49-55 (1994)

Richalet, J., Rault, A., Testud, J.L., Papon, J.; "Model Predictive Heuristic Control: Application to Industrial Processes," Automatica, 14, 413-428 (1978)

Robert Bosch GmbH; CAN Specification 2.0, Stuttgart, Germany (1991)

Rodd, M.G., K. Dimyati, L. Motus; "The Design and Analysis of Low-cost Real-time Fieldbus Systems," Control Engineering Practice, 6, 83– 91 (1998)

Savkin, A. V. and Evans R. J.; Hybrid Dynamical Systems: Controller and Sensor Switching Problems, Birkhauser, Boston, U.S.A (2001)

Sawamura, Yoshiki, Yuh Yamashita, Hirokazu Nishitani; "Compensation for Transmission Delay in Remote-Control System under Measurement Noise," SICE Annual Conference, 558-561, Osaka, Japan (2002)

Sempere, Paya Victor M., Jorge Mataix Oltra, Estanislao Utrilla Gines; "Modeling and Evaluation of Systems for the Interconnection of Industrial Communications Networks," Proceedings of the Fieldbus Conference, 136-145, Magdeburg, Germany (1999)

Sevillano, Jose Luis, Arturo Pascual, Gabriel Jimenez, Anton Civit-Balcells; "Analysis of Channel Utilization for Controller Area Networks," Computer Communications, 21, 1446-1451 (1998)

Thiele, Dirk, Terry Blevins, and Willy Wojsznis; "Device Based Process Control in Foundation Fieldbus," Proceedings of the Fieldbus Conference, 112-117, Magdeburg, Germany (1999)

Thomesse, J. P.; "A Review of the Fieldbuses," Annual Reviews in Control, 22, 35-45 (1998)

Thomesse, J. P.; "Fieldbuses and Interoperability," Control Engineering Practice, 7, 81-94 (1999)

Thomesse, J. P., M. L. Chavez; "Main Paradigms as a Basis for Current Fieldbus Concepts," Proceedings of the Fieldbus Conference, 2-15, Magdeburg, Germany (1999)

Thomas, A. B.; "Robust Stability Conditions for SISO Model Predictive Control Algorithms", Automatica, 33 (7), 1357-1361 (1997)

Tian, G.Y., Z.X. Zhao, R.W. Baines; "A Fieldbus-based Intelligent Sensor," Mechatronics, 10, 835-849 (2000)

Tindell, K., A. Burns and A.J. Wellings; "Calculating Controller Area Network (CAN) Message Response Times," Control Engineering Practice, 5 (8), 1163-1169 (1995)

Tipsuwan, Y., M.-Y. Chow; "Control Methodologies in Networked Control Systems," Control Engineering Practice, 11, 1099–1111 (2003)

Tovar, E. and F. Vasques; "Guaranteeing Real-Time Message Deadlines in Profibus Networks," Proceedings of the 10th Euromicro Workshop on Real-Time Systems, 81-87, Berlin, Germany (1998)

Tovar, E., F. Vasques, A. Burns; "Supporting Real-Time Distributed Computer-Controlled Systems with Multi-hop P-NET Networks," Control Engineering Practice, 7, 1015-1025, (1999)

Tovar, E., F. Vasques; "Distributed Computing for the Factory-Floor: a Real-Time Approach Using WorldFIP Networks," Computers in Industry, 44, 11-31 (2001)

Tovar, E., F. Vasques, A. Burns; "Communication Response Time in P-Net Networks: Worst-Case Analysis Considering the Actual Token Utilization," Real-Time Systems, 22, 229-249 (2002)

Vitturi, S.; "Some Features of Two Fieldbuses of the IEC61158 Standard," Computer Standards & Interfaces, 22, 203-215 (2000)

Walsh, Gregory C., Hong Ye, and Linda G. Bushnell; "Stability Analysis of Networked Control Systems," IEEE Transactions on Control System Technology, 10(3), (2002)

Walrand, J., Kallol Bagchi, and George W. Zobrist; Network Performance Modeling and Simulation, 197-218, Gordon and Breach Science Publishers, Amsterdam, Netherlands, (1998)

Xie, C., J.-S. Pu, P.R. Moore; "A Case Study on the Development of Intelligent Actuator Components for Distributed Control Systems Using LONWORK Neuron Chips", Mechatronics, 8, 103-119 (1998)

Zech, Kurt; “Understanding Fieldbus,” Chemical Processing, 58 (3), 6 (1995)

Zhivoglyadiv, P. V., and Middleton, R. H.; “Networked Control Design for Linear Systems”, Automatica, 39, 743–750 (2003)

List of Publications

Journals:

(1) Yanbin PANG and Hirokazu NISHITANI, "Timing Analysis and a Modified PID Control Algorithm of Fieldbus Control System", Journal of Chemical Engineering of Japan, Sep. 2004 (To be published)

(2) Yanbin PANG, Hirokazu NISHITANI, and S.H. YANG, "Analysis of Control Period in a Fieldbus Control System", Measurement and Control, 2004 (Submitted)

Proceedings of International Conferences:

(1) Yanbin PANG, Xiangyu WEI, and Youhua WU, "Distributed Sensor System and its Application", 37th SICE Annual Conference, July 29-31, Chiba, Japan, pp. 907-910, 1998

(2) Yanbin PANG, Xiangyu WEI, and Youhua WU, "The Sensor Network Based on Lonworks Technology", 38th SICE Annual Conference, July 28-30, Morioka, Japan, pp. 897-900, 1999

(3) Yanbin PANG and Xiwei LIU, "The Design of the Fieldbus Sensor for Measuring the Level and Temperature of the Water Well Monitoring Earthquake", 41st SICE Annual Conference, Aug. 5-7, Osaka, Japan, pp. 569-574, 2002

(4) Yanbin PANG, Hirokazu NISHITANI, and Lingqi LI, "Design of Distributed System for Monitoring the Temperature and Level of Underground Water Well", 1st Kyoto International Symposium on Underground Environment, May 17-18, Kyoto, Japan, pp. 393-397, 2003

(5) Yanbin PANG and Hirokazu NISHITANI, "Evaluation Techniques of Fieldbuses", 42nd SICE Annual Conference, Aug. 4-6, Fukui, Japan, pp.1679-1684, 2003

(6) Yanbin PANG and Hirokazu NISHITANI, "Stability of Fieldbus Control System", 10th APCChE (Asian Pacific Confederation of Chemical Engineering) Congress, Oct. 17-21, Kitakyushu, Japan, 2004 (Accepted)