Research on Energy Saving and Emission Reduction Technology Based on Field Communication and Embedded Control Computer Data Exchange Algorithm

Pingping Qian*
Zhongnan University of Economics and Law, Wuhan, China
*Corresponding author email: pingpingqian@zuel.edu.cn

Abstract. In addition to meeting the basic characteristics of traditional transactions, the concurrent control of embedded mobile real-time transactions should also focus on priority inversion, unnecessary transaction restart and global data consistency, performance of hybrid transaction systems, energy saving and emission reduction. When using traditional control algorithms, scheduling control is mainly based on data structure characteristics, and it is easy to cause data stack congestion. In order to solve the above problems, a stack scheduling control method of embedded data structure using multi-parameter neural network is proposed. Combined with the widely used switched Ethernet technology in defences, communications, aerospace, industrial control and other fields, a prototype of a star topology embedded software test system is introduced.

1. Introduction
Embedded control systems require high frequency, low delay, and low jitter to exchange control and feedback data. The high bandwidth of Ethernet is sufficient to support high-frequency data exchange, and it has become an ideal solution for embedded control networks. However, the difficulty of Ethernet as a communication medium is mainly manifested in the negative impact of non-deterministic network delay on high-speed high-precision multi-axis synchronous embedded. Switched real-time Ethernet is equivalent to increasing the network bandwidth, but the delay caused by its queuing is still uncertain. Time synchronization is the basis for implementing multi-axis synchronous embedded on an asynchronous control network. IEEE 1588 has a theoretical mechanism that is superior to NTP, but its performance is affected by time stamp accuracy and line asymmetry, especially in harsh industrial field environments. The Kalman filter algorithm can effectively eliminate the interference of noise on the measured value. At present, the control method based on network delay prediction has become a mainstream research direction.

Web technology is used to design the network networking of the underlying data system of the Internet of Things to realize the sharing of the underlying data information of the Internet of Things. Based on the Internet platform, the server/client (C/S) architecture is used to design the IoT network, and the bus transmission channel model is constructed for data transmission and scheduling. An embedded system based on multi-parameter neural network is proposed. The data structure stack scheduling control method. The simulation proves that the data structure stack scheduling control method of the embedded system based on the multi-parameter neural network has good dynamic control and scheduling balance, and effectively guarantees the stability of the embedded system. At the same time, this paper propo
ses a design method based on the embedded Internet of Things underlying data exchange system. Firstly, the overall design framework analysis of the underlying data exchange system of the Internet of Things is carried out, and then the functional module design is carried out, and the software development and design of the underlying data exchange system of the Internet of Things is carried out under the embedded Linux platform, and the intelligent information of the underlying data of the Internet of Things is combined with Visual DSP++ technology Processing, completing the system optimization design, and finally carrying out system testing, showing the superiority of the system designed in this article [1].

2. Data exchange algorithm design

2.1. Principle of data structure stack scheduling control

In the process of stack scheduling control of the data structure of the embedded system, x represents the data scheduling set to build the data structure stack storage model of the embedded system, h(A, B) represents the topology of the embedded system, and RY represents the For the time scale feature of the load data information flow, the scale feature value is cyclically and iteratively controlled using equation (1), and the control process conforms to the probability of normal distribution [2].

\[ RY = \frac{x}{h(A, B)} \times p \]  

The cyclic iterative control of the scale eigenvalues realizes the balanced scheduling of the data structure stack of the embedded system.

2.2. Data structure stack scheduling control method of embedded system under neural network

Assuming that vi represents the resource demand parameter of the embedded system, use this parameter and the characterization parameter of the system task urgency represented by \( \gamma_i \) as the input variable of the neural network, and \( p_i \) represents the priority of the control data stack task as the output variable, using the formula (2) Describe the real-time task set of n tasks that control the stack of data structure

\[ T(\tau_1, \tau_2, ..., \tau_n) = \frac{v_i \times \gamma_i}{p_i \times n} \oplus T_i \]  

Among them, \( T_i \) represents each data stack task cycle. In the optimization control process of the data structure stack scheduling of the embedded system, it is assumed that \( C_i \) represents the maximum task calculation time, which is defined as the maximum execution time required to execute a data structure stack task operation without interference. Condition of \( T_i = D \), bring T calculated by equation (2), and use equation (3) to calculate the actual idle time of the stack scheduling of the actual data structure of the embedded system

\[ \gamma_i = T - C_i \times \gamma_{max} \]  

Among them, \( \gamma_{max} \) represents the minimum idle time of the actual data structure stack scheduling of the embedded system, \( R \) represents the urgency of the task of the embedded system, \( V_i \) represents the demand of system resources, the larger the \( V_i \), the more important the task, and \( p_i \) represents the prior
ity of the task scheduling, the value is defined as the priority index of the embedded system data structure stack task is scheduled. In the optimization control process of the data structure stack scheduling of the embedded system, \( w_i(k) \) represents the weight of the data structure stack task. The weight is divided into fixed weight and compensation weight, which are represented by \( w_{0i}(k) \) and \( \Delta w_i(k) \) respectively. Calculate any data structure stack task using formula (3) Weights:

\[
 w_i(k) = w_{0i}(k) + \Delta w_i(k) + R_i
\]

\[
 w_{0i}(k) = \frac{\{w_{01}(k), w_{02}(k),..., w_{0n}(k)\}}{T(\tau_1, \tau_2, ..., \tau_n)}
\]

\[
 \Delta w_i(k) = \frac{w_i(k)}{w_{0i}(k)}
\]

Among them, \( \Delta w_i(k) \) represents the maximum weight coefficient of the data structure stack task \( \{w_{01}(k), w_{02}(k),..., w_{0n}(k)\} \) represents the compensation weight coefficient of the data structure stack task.

### 2.3. Realization of Data Structure Stack Scheduling Optimization Control

During the optimization control of the data structure stack scheduling of the embedded system, the priority decision of the acquired data structure stack task is used as the basis to form a data structure stack scheduling control model, which effectively implements the data structure stack scheduling control of the embedded system. In the optimization control process of the data structure stack scheduling of the embedded system, suppose that the burst data stored by the data structure stack represented by A during \( [t_0, t_1] \) does not cause data source congestion, which is denoted as \( T_{BSP} \), but the event will occur within the time period represented by \( [t_1, t_2] \), then the data structure stack represented by A will generate data overflow phenomenon, if the communication efficiency between the data structure stack represented by A and other candidate data structure stacks is 100%, use formula (5) to calculate the probability of data overflow caused by the data structure stack represented by A:

\[
 A'' = 1 - \frac{P_{DC,m}}{P_{DC,m}'} \times T_{BSP} / T_{Cycle} \times 0.3
\]

In the process of optimizing the stack scheduling of the data structure of the embedded system, comprehensively considering many factors such as the storage capacity of the data structure stack, the duty ratio of the work, the data generation rate, the size of the memory, etc., model

\[
P_{x-m} = \prod_{i=1}^{x} \left[ 1 - \left( R_{Oh} + R_f \right) \left( A'' \right) \right]
\]

Among them, \( x \) represents the data transmission link, \( R_{Oh} \) represents the data transmission rate, \( R_f \) represents the data transmission process.
3. Design of embedded field communication computer control system

3.1. How the embedded Web server works
The embedded Web server adopts the B/S structure. The client accesses the embedded Web server through the browser. The embedded Web server responds to the client request and fuses the sensor measurement data back to the client browser.

3.2. System hardware platform design
After fully considering the cost and application requirements, the processor of this system uses Philips' LPC2220 chip, which is a 32-bit processor based on ARM7TDMI, which can achieve a 75MHz CPU operating frequency, can support real-time simulation and embedded tracking, and is particularly suitable for industrial control. On this basis, the system expanded the 2M NOR Flash chip 36VF1601C as a program memory, 2M SDRAM as a memory. The network control chip adopts 10M Ethernet control chip CS8900A-IQ. In addition, the system also expanded the RS232 interface. The system hardware block diagram is shown in Figure 1.

3.3. PCI bus interface chip PCI9054
As a local bus that does not depend on a specific processor, the PCI bus supports a 32-bit or 64-bit bus width. The bus frequency is usually 33 MHz or 66 MHz (PCI 2.0). When working at 33MHz and 32 bits, the theoretical maximum data transfer rate reaches 132MB/s. The PCI bridge chip is equivalent to the intermediary between the PCI bus and the local logic. Intuitively, all the host's operation of the adapter's local logic is implemented by operating the PCI bridge chip. The function of the PCI bridge chip can be divided into two parts: the set of functions that must be implemented to meet the requirements of the PCI specification. The set of functions necessary for data transmission and control between the loc

![System hardware block diagram](image-url)
al logic of the adapter. PCI9054 is a 32-bit, 33MHz PCI bus master I/O accelerator launched by PLX, which can provide strong support for PCIV2.2, and burst transmission can reach 132MB/s. PLX integrated data pipeline structure technology, including DAM engine, programmable PCI initial pointer, slave data transmission mode and PCI message mechanism [3].

3.4. Software design

3.4.1. Intelligent serial port control firmware design. The intelligent serial port control firmware mainly processes and receives serial data. At the beginning of the program, the serial port parameters, send/receive buffer and related settings are initially set; when DSP detects that the PC needs to send data, it will take over the data and forward it to the external interface; DSP detects the external interface when data comes in, the data on the interface is actively received and forwarded to the PC. Since the DSP takes over the data transmission and reception of the external interface at the bottom layer, the burden on the PC is reduced. The sending of data is carried out through a combination of query and interruption, and the reception of data is carried out through an interruption method combined with timeout processing. The main program flow of the intelligent serial port control firmware is shown in Figure 2.

![Figure 2](image.png)

Figure 2. Block diagram of the main program of the serial port intelligent control software.

3.4.2. Intelligent parallel port control firmware design. Different from the standard protocol of serial interface, the application of parallel interface is often targeted. The following only uses a specific data input and data output method as an example.

The DSP receives the frame start pulse sent by the external device and enters the interrupt processing program. After recognizing the frame header flag, it extracts the frame length information A from the subsequent data, and then reads A-2 valid data and stores it between the DSP and the PC. As the FIFO of the data buffer (it can also be a dual-port RAM), after receiving a frame of data, it sends an interrup
t request to the PC and notifies the PC to take the data. The timing diagram of communication with a parallel input device is shown in Figure 3.

![Data input timing diagram](image1)

**Figure 3.** Data input timing diagram.

The PC sends data to the DSP and notifies the DSP in the way of the flag (or interrupt). The DSP first detects the presence of the frame header identifiers 0x55AA and 0xAA55. Once determined, it sends a start pulse to the connected device, followed by 83 fixed command words, and then sends an end pulse to the end. Externally connected equipment. The timing diagram of communication with a parallel output device is shown in Figure 4.

![Data output timing diagram](image2)

**Figure 4.** Data output timing diagram.

3.4.3. **Software design on the host side.** Driver: Because this design selects the PCI bus interface chip provided by PLX, you can use the library functions in the PLXSDK package provided by PLX to complete the corresponding functional operations. Serial port test program: The serial port test program consists of three parts: serial port parameter setting, data sending and data receiving. Serial port parameter
r setting: The serial port parameter setting mainly includes the configuration of the parameters of the four serial ports such as baud rate, data bit, stop bit, parity bit, flow control and so on. The serial port parameter setting interface in the test program is shown in Figure 5.

![Figure 5. Serial port parameter setting interface in the test program.](image)

Figure 5 Serial port parameter setting interface in the test program. Serial port sends and receive data: The serial port sends and receive data function is realized by multithreading. Each serial port has two threads responsible for receiving and sending. For each serial port receiving or sending, it is implemented by sampling and querying, that is, each thread is cyclically querying the status of receiving or sending, and if the status meets the conditions, the receiving or sending operation is performed; if the conditions are not met, then wait for a while and then enter the next cycle, continue to query [4].

4. Experimental testing
In order to verify the application performance of the method in this paper in the realization of the underlying data exchange control of the Internet of Things, an experimental test was conducted. The experiment was built on the Visual DSP++ 4.5 platform, and the underlying data processing module of the data exchange system was configured through BASE0 and BASE1. Build a human-computer interaction platform in the LOCAL configuration register, use the local bus transmission control protocol to implement host computer communication and dual-port RAM transmission scheduling, and perform the underlying data exchange of the Internet of Things in the embedded platform according to the above test environment description to obtain the bit error rate. The comparison results are shown in Table 1. The tr
additional exchange system and the exchange system studied in this article are compared and analysed for output stability. The results are shown in Figure 6.

Table 1. Comparative analysis of bit error rates of different switching systems.

| Serial number | Traditional switching system (%) | This article exchange system (%) |
|---------------|----------------------------------|----------------------------------|
| 1             | 95                               | 46                               |
| 2             | 89                               | 38                               |
| 3             | 90                               | 32                               |
| 4             | 92                               | 40                               |
| 5             | 96                               | 41                               |

The experimental results show that the comparative analysis of the two exchange systems shows that the bit error rate of the exchange system in this paper is much lower than the bit error rate of the underlying data exchange of the Internet of Things in the traditional exchange system, indicating that the embedded data exchange system of the embedded Internet of Things proposed in this paper has the feasibility of realization is greater.

![Figure 6](image_url)

Figure 6. Comparison of output stability of different switching systems.

It can be seen from Figure 6 that the traditional switching system has a large fluctuation frequency during data exchange, resulting in low stability, while the system designed in this paper has low fluctuation frequency and high stability when performing data exchange. In summary, using the system designed in this paper can realize the underlying data exchange of the Internet of Things, and the output bit error rate is lower and the stability is better [5].
5. Conclusion

This article designs a low-cost, high-performance embedded real-time Ethernet embedded controller, based on standard Ethernet technology, without changing the Ethernet hardware. The PID algorithm of the RBF fuzzy neural network designed in this paper can adjust the control parameters adaptively according to the system delay. Compared with the traditional PID algorithm, it has better steady-state performance and dynamic following performance.

References

[1] Mokihi, Z. Vitale, G. Aletti, G., Sacchi, V., Mauri, T., & Colombo, V., et al. Pain control with ultrasound-guided inguinal field block compared with spinal anesthesia after hernia surgery: a randomized trial. Surgery, 157(2) (2015) 304-311.

[2] Chen, A., Zhao, Y., Li, P., Zhang, X., Peng, R., & Huang, H., et al. Angular dependence of exchange bias and magnetization reversal controlled by electric-field-induced competing anisotropies. Advanced Materials, 28(2) (2016) 363-369.

[3] Cho, S. J. Yun, S. H. & Jeon, J. W. A powersaving dvfs algorithm based on operational intensity for embedded systems. Ice Electron Express, 12(3) (2015) 20141128-20141128.

[4] Prentice, J. C. A. Charlton, R. J., Mostofi, A. A., & Haynes, P. D. Combining embedded mean-field theory with linear-scaling density-functional theory. Journal of Chemical Theory and Computation, 16(1) (2020) 354-365.

[5] Rautaray, S. S. & Agrawal, A. Vision based hand gesture recognition for human computer interaction: a survey. Artificial Intelligence Review, 43(1) (2015) 1-54.