The Research of an On-demand Threaded Interrupt Technology of Serial Port Based on the Power Edge Smart Device

Wenkui She*, Fan Yang*, Shuang Wang and Yuanyu Li

1 Aostar Information Technologies CO., Ltd, Chengdu, Sichuan 610000, China
2 State Grid Information & Telecommunication Group CO., Ltd., China

*Email: shewenkui@126.com

Abstract. At present, the common problems facing the low-voltage power distribution system lie in plenty of equipment and wide range of power distribution. In order to address these problems, the overall architecture of the cloud-pipe-side-end is introduced through the IoT system in electric power distribution, where the end device is connected to the low-voltage power distribution system via the serial bus of the edge device and carries on data collection and real-time edge computing. As the core node, the edge device is required to have excellent real-time capability of timely detection and real-time response to the fault of various low-voltage power distribution facilities. In consideration of this, this paper proposes an on-demand serial port interrupt threading design to ensure the real-time performance of an electrical edge smart system. According to the experimental results, as the number and load of serial port devices increase, the real-time task dispatching rate continues to decline until the transformation, and the impact on the real-time task dispatching rate is limited after the threaded interrupt. That is to say, the method is effective in ensuring the dispatching rate of real-time tasks for the edge smart terminals.

1. Introduction

Due to the deep integration between traditional industry technology and IoT technology, the Internet of Things (IoT) in electric power distribution has emerged as a novel type of power network. In general, it refers to the distribution network applied for the new-generation electric power systems to meet the demand for lean management of the power distribution network and support the rapid development of the energy Internet. In addition, it also plays a role in achieving the comprehensive perception, data integration, and intelligence of the power distribution network through the full interconnection, intercommunication, and interoperability between the power distribution network facilities. Within the power distribution network, the 0.4KV distribution network is regarded as the most complex part, and encounters such problems as plenty of equipment and wide range of power distribution, which makes it difficult to carry out supervision on the maintenance of power operation, thus leading to a blind spot. The network data acquisition of existing low-voltage distribution and automation systems should be developed into the architecture of a cloud-edge node-edge device. Besides, as the core of edge node, the distribution transformer supervisory terminal unit (TTU) is required to be capable of fast iteration for achieving the edge integration of both new and existing multiple services and cloud collaboration. To solve such problems as long fault identification time and slow response at the time of processing switchgear failure, power line short circuit, and leapfrog trips that could occur to the low-voltage power distribution system, Ref. [1] proposed a fault handling
system of distributed generation which was based on edge computing. The system is capable to perform timely detection and make real-time response to the fault of low-voltage power distribution facilities. What is significant to improving the capability of real-time edge computing lies in the capabilities of the edge smart terminals of core nodes to conduct data collection and analysis. At this stage, about 80% of the low-voltage distribution network devices still use the serial port to access, for example, 485, 232, etc. Despite the introduction of wireless technology by some sites and the connection of sensor network to the smart terminal, the convergence side remains dependent on the serial ports to access the edge smart terminal. There are also some circumstances where wireless sensor networks can be used straightaway to communicate with the master station. For example, a communication management mechanism was designed in Ref. [2] to monitor load and leakage of the low-voltage distribution network. Apart from that, wireless sensor networks were applied to exchange data with smart circuit breakers for monitoring the progress in data collection, command forwarding, data and status storage between the master station and the smart circuit breaker. Currently, however, this method is not necessarily suitable for all scenarios due to such factors as geographic location and equipment transformation. Therefore, the low-voltage distribution network terminal device is most commonly used to perform edge calculation by accessing the edge smart terminal serial port. As the number of devices that access through smart terminal serial port increases, it is inevitable to affect the real-time task dispatching rate of the smart terminal system and lead to a failure in the assignment of key applications. In order to ensure the real-time processing capability of system edge computing, this paper proposes an on-demand threaded interrupt technology based on the Linux operating system of smart terminals.

2. The Edge Computing Operating System for the Internet of Things in Power Distribution

2.1. The Overall Architecture Internet of Things in Power Distribution
Adopting the overall architecture of the cloud-pipe-side-end, the Internet of Things in power distribution involves a large number of sensing nodes at the end layer which complete cloud-side interconnection through pipelines could achieve such as the perception of realizing state quantity, monitoring of electrical quantity, and support the Internet of Things in power distribution field through cloud computing, big data, artificial intelligence and other related technologies in the cloud platform. Through the access to the end-layer equipment, the edge is on the edge of the network close to the source of things and data which achieve data collection, storage, and edge computing applications by using edge smart terminals as the carrier. Edge computing apps are capable to achieve such purposes as prediction load, fault research, judgment, the analysis of line loss, the early warning of risk, the application of charging pile, the analysis of user the data, compensation of reactive power, commutation switch, and so on [3-4].

The edge smart terminal represents a critical link in the Internet of Things in power distribution and plays a crucial role in achieving massive device access, real-time data collection, and edge computing. All those actions require support from the operating systems on which the edge smart terminal relies.

2.2. Intelligent Operating System Based on Edge Computing
To enhance the capability of edge computing, a containerization technology was proposed in Ref. [5] to better support edge smart services and achieve fast connection and smart application in the digitalization of distribution networks based on the Linux system. Figure 1 shows the overall architecture of the edge computing operating system, where the bottom layer is the edge smart terminal hardware that runs the Linux-based edge computing operating system and carries out the operation required by the container engine which generally uses an industrial-grade CPU.
There are two types of container selection technology. One is the Docker container led by the smart chip company State Grid, and the other is the lxct led by Huawei. Although both technologies have demonstrated their respective advantages, Docker is not considered an alternative solution for lxct. Referred to as a feature of the Linux kernel (especially namespace and Cgroup), "lxct" allows some sandboxed processes to be conducted in a relatively independent space and makes it easy to control how their resources are allocated. Based on the characteristics of the underlying Linux kernel, Docker develops a higher-level toolset with multiple powerful functions on the upper layer, for example, portable cross-machine deployment, application-centric, automatic construction, and sharing. Besides, Docker features a more powerful tool ecosystem, which makes both technologies capable to achieve resource isolation for containers, despite the Docker ecosystem being more powerful. Notably, edge computing can also be performed by applying non-containerized technology. In that situation, however, the requirements for edge smart terminal operating systems will be extremely demanding. Therefore, with edge computing applied inside smart terminals, the Linux system is bound to be accepted as the mainstream intelligent operating system intended for edge computing.

With the extensive application of Linux systems on edge smart terminals, edge computing apps tend to place higher requirements on the real-time performance of the system. Unfortunately, Linux is a timesharing system with poor capabilities of real-time computing. In Ref. [6], the use of interrupt threading and SMP-based task priority balancing methods was proposed to improve the real-time performance of the kernel. In Ref. [7], real-time improvement technologies were introduced for Linux, including interrupt threading, Mutex instead of spinlock, priority inheritance and deadlock detection, the prioritization of the wait queue, and the preemption methods intended for large kernel lock. In Ref. [8], an analysis was conducted regarding the negative impact of process dispatching, interrupt processing, kernel lock mechanism, and virtual memory on real-time performance, based on which a threaded interrupt method was proposed on the basis of system calls. A new real-time model of the Linux operating system was described in Ref. [9], including the design of interrupt managers and resource managers. The application of these two designs is effective in improving the real-time performance of the operating system. In Ref. [10], a study was conducted on the design of a real-time data transmission system based on ARM9 and GPRS. By transplating the linux2.6 operating system with the real-time response in the 32-bit arm microprocessor S3C2440A, the system makes a direct modification to the Linux kernel for interrupt threading, thus further improving the performance of the system in real-time responsiveness, data processing and transmission rate. It can be seen that threaded interrupt is a key technology to improve Linux real-time performance. Besides, since a large number of end devices are connected to edge smart terminals, the interrupt frequency generated by the hardware interface increases, as does the probability of edge computing apps being preempted by interrupts. Moreover, the serial port remains the mainstream collection method now for edge smart terminals to get connected to the end device. Thus, this paper focuses on combining interrupt threading and the serial port.

3. Analysis of Related Technology of Serial Port Collection Based on Linux
3.1. Analysis of Serial Communication Technology Based on Linux

The Linux serial communication architecture is shown in Figure 2.

Linux serial port communication consists of two major parts. One is the access to the serial port in user mode, involving common interface operations such as open, read, write, and ioctl. The other is the tty architecture of the serial port terminal driver in the kernel mode. The latter is regarded as the core of the entire serial port driver, involving the tty core, tty line regulations, and tty drivers. The tty core is used to define the file_operations structure common to tty devices through tty_io.c. While the line discipline layer is responsible for transforming the input data, so as to achieve the members in the tty_disc structure through n_tty.c. The tty driver corresponds to the realization of the hardware serial port driver.

Figure 3 shows a typical process of sending and receiving serial data. During the course of sending data, the tty core layer obtains the data to be sent to a tty device from a user before transmitting the data to the tty line for discipline analysis. The parsed data is transmitted to the tty driver layer, while the driver layer converts the data to the format that can be sent to the tty device. In the process of receiving data, the data received from the terminal tty device is passed on to the tty driver layer for line discipline analysis. Then, it enters the tty core layer, and ends up being obtained by the user.

Therefore, in the context of Linux serial communication architecture, the tty core and tty line discipline represent the standard framework layer, and the optimization of serial port is mainly concentrated in the user space and tty driver layer. In Ref. [11-14], the multi-threaded serial port communication method is applied in the user space layer to carry out the reception, extraction, transmission, and storage of data, thus
achieving the purpose of real-time serial port communication. In this paper, the nuc970 chip will be combined to analyze the serial port driver in the tty driver layer. In nuc970_serail.c, nuc970serial_start_tx is applied to send data to the interface of the hardware layer, before the data is transmitted to the serial control register through transmit_chars. Receive_chars provide an interface purposed to receive data. Once receiving external data, the serial port will initiate the interrupt callback function nuc970serial_interrupt, activate receive_chars in nuc970serial_interrupt to read the data from the serial port control register and pass it on to the tty core layer for later processing. Figure 4 shows the flowchart followed by the nuc970 chip serial port interrupt handler.

![Flowchart of Serial Port Interrupt Processing](image)

**Figure 4.** Interrupt processing of serial ports.

It can be seen that the process of sending and receiving data is carried out by the serial port interrupt handler. Thus, the more frequently the serial port is used, the longer the CPU will be trapped in interrupt processing. Consequently, the un-dispatched edge computing real-time tasks would last a long time.

### 3.2. Technology Analysis Based on Linux Interrupt

The interrupt is a mechanism behind the interaction between hardware and software. As for the generation of an interrupt, it undergoes three processes: equipment, interrupt control, and processor. The interrupt controller acts as a bridge between the connected device and the processor. After a device generates an interrupt, it needs to be forwarded by the interrupt controller to reach the processor. As time goes forward, the interrupt controller has undergone two stages: PIC (Programmable Interrupt Controller) and APIC (Advanced Programmable Interrupt Controller). As for the realization of the Linux system interrupts, it includes two critical stages. In the first one, peripherals are associated with system interrupts. Take the nuc970 chip as an example, nuc970_init_irq will be deployed to associate the peripheral with the interrupt ID when the Linux system is initialized. The peripheral driver binds the interrupt ID with the interrupt handler through the request_irq interrupt registration function.

The other one relates to interrupt response and processing. When an interrupt is triggered by the peripheral, the PIC transmits the signal to the CPU. To begin with, the CPU will perform on-site protection, such as user stack, kernel stack. Then, it starts to call the interrupt response phase related to the CPU architecture, identify the Linux system corresponding to the hardware interrupt ID and call the interrupt handler associated with the interrupt ID. After the return of interrupt handler, the Linux operating system conducts on-site recovery operations. Inside the standard Linux kernel, interrupt is treated as the highest priority execution unit. Regardless of whatever the kernel is processing at the time, the system will respond...
to that event immediately and execute the corresponding interrupt processing code as long as there is an interrupt event. Figure 5 shows the assignment of real-time tasks, interrupt responses, and interrupt handlers. It can be seen from the figure that when real-time tasks are assigned, the real-time task assignment will be interrupted in case of an interrupt. At this time, the CPU will switch to interrupt response and interrupt processing. Until the interrupt process gets complete, the CPU switches back to the real-time task assignment. Therefore, if the system has edge computing real-time tasks in progress and the interruption occurs frequently, it is barely possible for the real-time tasks to proceed.

The threaded interrupt is purposed to carry out interrupt processing, which is because a kernel thread allows the interrupt response to return immediately after the interrupt processing thread is wakened, thus making the priority of the interrupt processing thread lower compared with the real-time task and causing delay to the assignment, as shown in Figure 6.

After the threaded interrupt is adopted, the interrupt will function as a kernel thread and be given different real-time priorities, as a result of which the real-time task can be prioritized over the interrupt thread. In doing so, the real-time task can be performed as the highest priority execution unit, despite a real-time guarantee that remains under severe load. In Ref. [15], it was suggested that solve the problem that real-time tasks cannot be scheduled for a long time through interrupt threading in the bayonet monitoring system of the vehicle, simply apply threaded interrupt technology will cause the equipment data to be unable to be collected in time and makes the corresponding real-time the task hungry. In consideration of this, the next part of this paper will propose an on-demand serial port interrupt threading method based on the low-voltage distribution network on the electric power edge smart device.

4. Research on On-demand Serial Port Threaded Interrupt Technology Based on Electric Power Edge Smart Devices
4.1. Analysis and Design of the Serial Port Interrupt Threading Based on the Edge Smart Device on the Low-voltage Power Distribution Network

The edge smart terminal equipment used in this paper is the NUC970 series chip developed by Nuvoton Technology, which relies on ARM926EJ-S single-core architecture with a maximum frequency of 300MHz. Due to its single-core mode, every hardware interruption will interrupt the currently running process, as mentioned above, thus hindering the real-time task from being assigned. Take the 0.4KV low-voltage power distribution network smart terminal as an example, there are eight distributed channels suffering serial interruptions. The serial ports ttyS0, ttyS1, ttyS2, ttyS3, ttyS5, ttyS6, ttyS7, and ttyS8 correspond to interrupt ID 36, 37, 38, 43, 44, 40, 45, 41, respectively. Figure 7 details the serial port interrupt printing information about the Linux operating system.

```
nuc970-uart.0: ttyS0 at I/O 0x0 (irq = 36) is a NUC970
nuc970-uart.1: ttyS1 at I/O 0x0 (irq = 37) is a NUC970
nuc970-uart.2: ttyS2 at I/O 0x0 (irq = 38) is a NUC970
nuc970-uart.3: ttyS3 at I/O 0x0 (irq = 43) is a NUC970
nuc970-uart.5: ttyS5 at I/O 0x0 (irq = 44) is a NUC970
nuc970-uart.6: ttyS6 at I/O 0x0 (irq = 40) is a NUC970
nuc970-uart.7: ttyS7 at I/O 0x0 (irq = 45) is a NUC970
nuc970-uart.8: ttyS8 at I/O 0x0 (irq = 41) is a NUC970
```

Figure 7. Serial Interrupt Printing Information of Linux OS.

Among them, ttyS0 represents a smart terminal debugging serial port intended for debugging and maintenance of the system. As the communication interface between the smart terminal and the Bluetooth module, ttyS1 is used for message interaction with external Bluetooth devices. As the 485 ports of the smart terminal to the external, ttyS2, ttyS6, and ttyS8 are applied for data collection and control of external devices. As the 232 port of the smart terminal to external, ttyS3 is used for data collection and control of external devices. As the communication interface between the smart terminal and the 4G module, ttyS5 is applied to control the normal online and offline operations of the 4G module. As the communication interface between the smart terminal and the broadband carrier wave module, ttyS7 is purposed to communicate with external devices through the carrier wave module.

The functional interfaces are detailed above for each serial port. As for the smart terminal, there will be a large number of edge devices getting connected to the southbound interface. If the number of external devices connected through the serial port is denoted as \( n \), according to the aforementioned analysis of the interruption principle, the response time of interruption of each serial port is indicated by \( t_i \), and the time consumed by the interrupt handler is denoted as \( t_j \). Taking \( T \) as a dispatching period of the Linux operating system, the dispatching rate \( P \) of a real-time task per unit time can be expressed as formula (1):

\[
P = \frac{T \cdot p - \sum_{i=1}^{n} t_i - \sum_{j=1}^{m} t_j}{T}
\]

In formula (1), \( P \) represents the current dispatching rate of real-time tasks in a unit time without interruption. It can be seen from this formula that as the number of access devices \( n \) increases, the dispatching rate of real-time tasks per unit time \( P \) is in decline. Therefore, if traditional Linux interrupt technology was applied, there would be no way to ensure the real-time performance of the system. Through the above-mentioned analysis, the threaded interrupt technology can be applied to run the interrupt handler as a kernel thread, thus reducing competition with real-time tasks and improving the dispatching rate of real-time tasks. However, the application of threaded interrupt technology alone to reduce the running time of interrupts will hinder the timely collection of device data, and the hunger of corresponding real-time tasks. In order to ensure the real-time performance and throughput capacity of the system, an on-demand serial port interrupt threading technology is developed in this paper.

4.2. The Design of On-demand Serial Port Threaded Interrupt Technology
In this study, the NUC970 smart terminal device developed by Nuvoton is exemplified, and on-demand serial port interrupt threading technology is developed through the two dimensions of serial port function and access device.

The first one is divided by the function of the serial port. Rarely used as a debugging serial port, ttyS0 is intended mainly for on-site operation and outputting critical information on the system. Besides, the program printing information can be outputted to the file system for reducing the frequency of the access to the serial port. Therefore, to ensure the timeliness of operation, ttyS0 as a debugging serial port is exempt from the threaded interrupt transformation. In comparison, the other serial ports ttyS1-8 are classed into two categories by function. One is the command-type serial port, such as ttyS5 for controlling 4G modules, and the other is the collection-type serial port, such as ttyS2, ttyS6, and ttyS8 for the purpose of data collection through the 485-serial port. The command type serial port comes with high real-time requirements, but the flow of interrupt processing time is not long-lasting. To ensure that the command type serial port processing program monopolizes the CPU, the threaded interrupt transformation is skipped.

The second dimension is divided according to the access equipment. Take the access scenario of the 0.4KV low-voltage station area as an example, a typical access scenario of distribution and transformation terminal in the 0.4KV low-voltage station area is shown in Figure 8.

Figure 8. Access Scenario of 0.4KV Low-voltage Station Area.

Among them, main fuse, circuit breakers, branch switches are categorized into remote control equipment, which requires relatively high real-time performance. Since serial port interrupt processing is conducted on a temporary basis, there is no need to apply threaded interrupt technology. The environmental monitoring devices, electricity meter, and smart capacitors in the station area are classed into telemetry.
equipment. As the real-time requirements are not particularly demanding, threaded interrupt transformation could be conducted. However, it remains necessary to set different processing priorities for different devices, for example, the environmental monitoring device is employed to collect real-time temperature, humidity, and other information in the station area, while an auxiliary decision-making device is applied for operation and maintenance. In this case, the priority after threaded interrupt will be higher, despite being relatively lower than the real-time task of the system. Regarding the electricity meter and smart capacitor, the priority of applying threaded interrupt is lower since the smart terminal collects only power information and capacitor compensation information. Besides, the real-time requirements are not too demanding.

As for threaded interrupt, high-priority tasks and low-priority tasks are categorized as real-time tasks, but the priority of both is lower compared with the edge computing real-time tasks in the system. According to formula (1), the dispatching rate of edge computing real-time tasks after threaded interrupt can be expressed as formula (2):

$$P = \frac{T^* p - \sum_{j=1}^{n} t_j - \sum_{j=1}^{r} t_{ij}}{T}$$

In the formula (2), \(x\) represents the number of devices connected to the serial port after threaded interrupt transformation. Based on the theoretical analysis, it can be known that after the serial port is threaded, as the number of connecting devices rises, although the overall interrupt response time \(\sum_{i=1}^{n} t_i\) remains unchanged, the time processing the competition between the interrupt handler and the real-time task is reduced, which ends up improving the real-time the dispatching rate of the task.

Based on the above theoretical analysis, it is considered necessary to divide the serial function for the existing edge smart devices for better adapting to the threaded interrupt transformation. The results obtained after transformation are listed in Table 1.

| name of serial ports | function of serial ports                                      | apply threaded interrupt transformation or not |
|----------------------|----------------------------------------------------------------|-----------------------------------------------|
| ttyS0、ttyS1、ttyS5   | debugging and controlling built-in serial port                | no                                            |
| ttyS2               | access remote control equipment                               | no                                            |
| ttyS6               | access high real-time telemetry equipment                      | yes high priority                              |
| ttyS3、ttyS7、ttyS8   | access low real-time telemetry equipment                       | yes low priority                               |

Additionally, in practice, the variations in the number, types, and business requirements of the equipment in the station area cause the requirements on real-time performance to be different. Therefore, this paper proposes a method that allows users to make dynamic adjustment to the priority of interrupt threads, so as to meet different demands, which will be detailed in the following part.

4.3. Implementation of On-demand Serial Port Interrupt Threading Method Based on Edge Smart Devices on the Low-voltage Power Distribution Network

In order to demonstrate the improvement achieved by the system to real-time edge computing app dispatching rate after the threaded interrupt transformation, an on-demand serial port interrupt threading technology will be implemented in this study under Linux 3.10.x version, from two perspectives. One is the threaded interrupt transformation based on the serial port driver and the other implements dynamic adjustment for the priority of threaded interrupt.
4.3.1. The Threaded Interrupt Transformation based on Serial Port Driver

In the serial port driver processing function nuc970_serial.c, the interface nuc970serial_startup is applied to complete the hardware parameter configuration related to the serial port driver and to initialize the interrupt processing function. In Linux 3.10.x, the interrupt handler could be threaded by using request_threaded_irq. Combined with the above-mentioned design of serial port driver threaded interrupt transformation, the process of transformation is developed, as shown in Figure 9.

- Call the nuc970serial_startup serial port hardware to initialize interface and initialize the serial port related registers;
- If the serial port is ttyS3, ttyS6, ttyS7, or ttyS8, the request_threaded_irq interface is used to initialize the serial port interrupt thread;
- Otherwise, the traditional interrupt processing interface request_irq is used to register;
- Return after serial port interrupt registration is completed.

![Figure 9. Transformation Processing of Threaded Interrupt transformation.](image)

Figure 10 shows the pseudocode of the interrupt processing flow after threaded interrupt transformation of the serial port. It can be seen from the figure that the interrupt processing thread is triggered and returned to the interrupt response program. The interrupt thread runs asynchronously.

```plaintext
on each IRQ reached
do
    interrupt response program;
    find interrupt handler;
    awake interrupt thread;
done

Interrupt thread:
    interrupt handler;
```

![Figure 10. Pseudocode of Serial Port Driver After the Threaded Interrupt.](image)

After the interrupt handler is threaded, the priority of the serial port interrupt processing thread is 50, and it is set to the SCHED_FIFO dispatching mode. Thus, throughout the operation of the interrupt thread, the process of the thread interrupt will be suspended in case of a real-time edge computing process with a
higher priority than 50. In this circumstance, the system will shift to performing the real-time task, thus ensuring the prompt operation of real-time tasks and enhancing the real-time nature of the whole system.

4.3.2. Achieve the Dynamic Adjustment of the Priority of Interrupt Thread

Due to the differences in on-site access equipment, it is necessary for the priority of thread interrupt to be dynamically modified. The priority of the threaded interrupt of the system is 50 in default. In order to make the serial port interrupt threading technology more consistent with the scene, the dynamic adjustment is made in this paper to the priority of interrupt threading method in the kernel mode. By registering the character device as the access interface of the user-mode program, the user is permitted to modify the priority of serial port thread interrupt in real time through commands. Figure 11 illustrates the process of implementing dynamic adjustment to the priority of thread interrupt.

![Diagram](image)

**Figure 11.** Achieving Dynamic Adjustment of the Priority of Threaded Interrupt.

- ioctl calls the user interface and passes the thread interrupt id and the priority size needing to be adjusted to the kernel-mode;
- The kernel processing program is carried out to obtain the corresponding pid structure through find_get_pid;
- All threads of the kernel are polled through do_each_pid_task to find out the task_struct structure corresponding to pid;
- Find sched_setscheduler, set thread priority, and return success.
- Otherwise, it returns as a failure.

In this section, an on-demand serial port interrupt threading method is designed and implemented on the basis of the real-world low-voltage distribution network. The next section will be dedicated to verifying the improvement to the real-time performance of edge smart terminals using this method through experiments.

5. Experiment and Analysis

The last section finished the analysis, design, and implementation of the serial port thread interrupt based on the edge smart device on the low-voltage distribution network. In this section, experiments are conducted to compare the real-time performance between old (before applying) and new (after applying) serial port interrupt threading technology.
5.1. Experimental Scenarios

This experiment set is based on the edge smart terminal of nuc970 chip, the configuration of which is shown in Table 2.

| Configuration                  | Value                  |
|-------------------------------|------------------------|
| CPU cores number (core(s))    | 1                      |
| CPU frequency (MHz)           | 300                    |
| CPU manufacturer              | Xintang                |
| CPU model                     | ARM926EJ-S             |
| total physical memory (M)     | 64                     |
| nanflash size(M)              | 512                    |
| operating system              | Linux3.10.x            |

The serial port accessed device information is indicated in Table 3.

| Serial Port | Function          |
|-------------|-------------------|
| serial port 0 | debug port      |
| serial port 1 | bluetooth       |
| serial port 2 | air circuit breaker |
| serial port 3 | sensing device  |
| serial port 5 | 4G               |
| serial port 6 | sensing device  |
| serial port 7 | carrier device   |
| serial port 8 | Sensing device  |

In this experiment, a real-time task is performed in the smart terminal system, and the analog sensing equipment is accessed on serial port 3, serial port 6, and serial port 8. The number of sensors is 10, 20, 30, 40, and 50 respectively. Each sensing device sends message of different lengths to the serial port on continued basis. At this time, the changes in dispatching rate of real-time tasks within 1 minute are observed before and after threaded interrupt technology is applied under the program.

5.2. Experimental Results

Scenario 1: When each sensor device sends a 10-byte message continuously to the serial port, the change to dispatching rate of real-time tasks before and after applying threaded interrupt is obtained, as shown in Figure 12. The horizontal axis represents the number of the connected sensor devices, while the vertical axis indicates the real-time task dispatching rate:

![Figure 12](image-url)
Scenario 2: When each sensor device sends a 100-byte message continuously to the serial port, the changes in dispatching rate of real-time tasks before and after applying threaded interrupt are observed, as shown in Figure 13. The horizontal axis represents the number of the connected sensor devices, and the vertical axis indicates the real-time task dispatching rate:

![Figure 13. Real-time Task Dispatching Rate before and after Threaded Interrupt at 100 bytes.](image)

As revealed by the test results, with the increase in the number and load of serial port devices, the real-time task dispatching rate shows a decreasing trend on a continued basis until the transformation, and the impact on the real-time task dispatching rate is limited after the threaded interrupt. Besides, it makes little difference to the real-time task dispatching rate after the transformation as the length of messages sent by the serial port increases. Therefore, the on-demand serial port interrupt threading method is effective in ensuring the dispatching rate of real-time tasks of edge smart terminals as the number and load of serial port devices are on the increase.

6. Conclusion
In order for adaptation to the development of the edge computing architecture required by the Internet of Things in power distribution, the Linux system is set to become the mainstream edge computing type intelligent operating system. However, the Linux operating system is not a real-time operating system. In the 0.4KV low-voltage power distribution network, a large number of devices are widely distributed, most of which are interconnected through serial ports. When the load of serial devices increases, it will inevitably affect the assignment of edge computing real-time type tasks in the Linux operating system of the edge smart terminal. In consideration of this, the actual scenarios of low-voltage distribution networks are combined in this paper to design and implement an on-demand threaded interrupt method based on edge smart devices for ensuring the dispatching ability of edge computing real-time tasks, and it is demonstrated through simulation experiments that this method is effective in ensuring the dispatching capabilities of edge computing real-time tasks in edge smart terminals. Furthermore, the impact of serial port threaded interrupt on real-time tasks is verified in a single-core scenario. In the future, the scenarios of multi-core, APIC and CPU affinity will be integrated to further study the impact of threaded interrupt technology on the capabilities of high-performance edge smart terminals in terms of real-time task assignment.

References
[1] Weihong Zhang, Xiaolong Chen, Shun Wan, et al, 2019 A distributed distribution network fault processing system based on edge computing technology Distribution & Utilization 58 28-33
[2] Xiaojiang Li, Moufa Guo, Wei Gao 2014 Development of communication manager for supervisory control of load and leakage current in low-voltage distribution network Electrician 5 18-22
[3] Jiqian Li 2019 The company fully deploys the construction of ubiquitous power Internet of Things State Grid Mag, 4 1-105
[4] Yang Li, Jie Chen, Qiang Gao 2020 Research on future development model of power grid company under ubiquitous electrical internet of things Information and Dynamic 5 426-33
[5] Zheng Nie, Jianmin Zhang, et al, 2020 Technologies and Application Scenario Design for Making Distribution Transformer Terminal Unit Being a Containerized Edge Node Automation of Electric Power 44(3) 154-61
[6] Chuan dong Zhang 2015 The Improvement Scheme of Real-time Scheduling Strategy Based on Embedded Linux Kernel Journal of JiNing University 3 87-90
[7] Shouchang Sun, Hongfang Han, Yu Meng 2007 Improvement and realization of the Real-time technology of Embedded Linux Microcomputer Information 23(35) 67-9
[8] Jun Hou, Yanfeng Zhang 2007 Embedded real-time kernel for determining interrupt and preemption delay Microcomputer Information 23(4) 49-50
[9] Baoyu Zhou, Degang Kong, Hongwei Zhao, et al, 2011 Research on Enhancing Embedded Linux's Real-Time Performance Journal of Jilin University 29(4) 58-62
[10] Xiaoyun Wu 2011 Research of Real-time Data Transmission System Journal of shangluo University 25(6) 56-60
[11] Yifan Zhou, Guoyu Yan, Ming Lu 2017 Multithreading Serial Communication Technology and Application in Fire Monitoring New Product & Tech 17(9) 55-8
[12] Rangchen Chen 2017 Analysis on the use of multi-threaded serial communication technology in Beidou navigation Wireless Internet Technology 15 8-9
[13] Gang Liu 2020 Design and Implementation of High Performance Dual-Serial Communication Model Based on EM9000 Moderncomputer 3 1048
[14] Penghui Wang, Guang Chen, Pingping Bao 2017 The development of GPS logistics vehicles data acquisition and storage system Electronic Design Engineering 25(11) 9-13
[15] Hongyang Xu 2016 Detecting system of intersection vehicles based on embedded Linux,” Microcomputer and application 35(12) 87-8