Design of Real Time Communication Software Based on ReWorks Operating System

Jianwei Zhang¹, Xingke Tian¹ and Yi Man¹
¹North China Vehicle Research Institute, Beijing, 100072, China
*Corresponding author’s e-mail: sxdtzys254@163.com

Abstract. This paper briefly introduces the functions and characteristics of the embedded operating system ReWorks. Aiming at the design requirement of real-time communication software, the task mechanism of ReWorks operating system is analyzed, the design process of real-time communication software is given, and the software development and debugging process using ReDe integrated environment under ReWorks environment is introduced. The data transmission and communication of the vehicle information system are realized.

1. Introduction

The embedded operating system is oriented to specific application requirements and can be designed and optimized to improve efficiency and reduce costs. At present, the mainstream general embedded operating systems mainly include Linux, VxWorks, Windows CE.net and so on. The real-time embedded operating systems for control, communication and other fields include WindRiver's VxWorks and so on. The main embedded operating systems in China include Kylin series Linux, ReWorks and so on.

The wide application of embedded operating system has brought great convenience to our life and work. At the same time, the reliability and controllability of the embedded operating system have become increasingly demanding, especially in the application related to national security and social stability. In the construction of information system in the fields of national defense, aerospace, electric power and transportation, it is necessary to use domestic hardware and software as much as possible.

ReWorks/ReDe is an embedded real-time operating system and integrated development environment independently developed by the 32nd Institute of Computing Technology. ReWorks/ReDe provides a good software development and operation platform for embedded systems. Especially, ReWorks application programming interface is compatible with VxWorks, driver code is compatible with VxWorks binary, and supports software engineering tool integration. Its technology has reached the level of the mainstream embedded operating system abroad. ReWorks operating system is a scalable embedded real-time operating system constructed by micro-kernel and a group of core components that can be tailored. It has functions of interrupt management, memory management, clock management and so on. It can not only satisfy the general embedded system, but also meet the needs of the large embedded operating system.

This paper is based on the domestic hardware environment represented by Loongson 2K1000 and the domestic software development environment represented by ReWorks/ReDe. The design of multi-channel real-time communication system software is realized. The information transmission is efficient, real-time, stable and reliable.
2. ReWorks introduction

2.1 ReWorks characteristics

ReWorks has the following characteristics:

- **Strong real-time**: Kernel can preempt, interrupt can be nested, interrupt management mechanism with fast response, efficient task scheduling, context switching algorithm, providing priority inheritance, priority ceiling protocol, system scheduling can be predicted.
- **Tailorability**: Using the architecture of the micro-kernel and object-oriented design method, it can provide component-based kernel services and maximize the tailorability of the kernel.
- **Applicability**: It supports a variety of mainstream CPU/DSP chips and hardware architectures, and provides typical device drivers. Support dynamic loading/unloading of applications and provide rich field-oriented peripheral components, such as communication, CLDC/MIDP, graphics components to meet the needs of military and civilian applications.
- **Standardization**: Support C/C++, Java programming language, provide system call interface conforming to POSIX 1003.13-2003/POSIX 1003.1-2001 specification, ensure the portability of application.
- **Reliability**: Provide MMU-based spatial domain protection mechanism, support device-driven management in user mode, implement fault monitoring, isolation and recovery framework, prevent user application failures leading to system crash, and enhance system reliability.
- **Scalability**: Provide core extension interface, support component dynamic interpolation including driver, support scheduling algorithm injection and dynamic configuration, improve system maintainability.
- **Accessibility**: The supporting ReDe development environment integrates design, development, configuration, debugging, simulation, testing, operation and deployment, which is convenient for users and improves the efficiency of application development.
- **Sustainability**: Independent intellectual property core, with excellent maintenance and upgrading capabilities, part of the integrated components and subsystems are derived from open source, and develop synchronously with relevant organizations.

![Figure 1. Reworks Architecture Diagram](image)

2.2 ReWorks function

(1) Microkernel and Component Architecture

ReWorks is a scalable embedded real-time operating system built by a microkernel and a set of tailorable core components. Microkernels provide strong real-time scheduling, basic memory management, clock management, interrupt management and other functions. Core components
provide tailorble functions such as timers, regions, partitions, signals, events, semaphores, messages and so on to meet the needs of embedded systems from deeply embedded systems to high-end complex embedded systems. ReWorks's micro-kernel and hierarchical component structure determine its inherent modularization characteristics, and provide a basic guarantee for improving the system's upgrade and maintenance ability. The adaptability of the target processing environment and the system's expansion ability.

(2) Task Management

ReWorks provides efficient real-time multi-task management, supports 256 task scheduling priorities and has a flexible and configurable task scheduling algorithm. The preemptive scheduling, time slice rotation scheduling and RMA (rate monotonic algorithm) are implemented. The task context can be switched quickly and task scheduling can be predicted.

ReWorks adopts a preemptive scheduling strategy based on priority. Meanwhile, for mission critical tasks, the exclusive operation of tasks is allowed. Once they enter operation state, other tasks with higher priority cannot preempt scheduling until the task is blocked or completed.

For periodic tasks in real-time systems, reworks implements a monotone rate scheduling algorithm. It has the schedulability analysis ability of periodic tasks, which ensures that the system will not exceed the deadline of each real-time task even in the case of short-term overload and ensures the predictability of system behavior.

(3) Storage Management

In order to ensure real-time, ReWorks is dedicated to memory. For this reason, ReWorks provides a variable size and fixed size area memory management mechanism, which is responsible for memory allocation and release.

Regional memory management uses efficient buddy algorithm to dynamically allocate and release memory. For various core objects, ReWorks allocates and releases memory through pre-allocated object management method to improve operational efficiency.

(4) Interrupt Management

Interrupt response time is the key index of the real-time system. ReWorks adopts a core preemptive design, and provides a mechanism to associate interrupt handling with tasks. Enables interrupt processing tasks of the main process to run in user space. It improves the reliability of the system, reduces the nesting level of interrupt and the critical area of the system and shortens the delay time of interrupt.

ReWorks supports 256 levels of interrupts and nesting of interrupts.

(5) Device Management

In real-time systems, most of the time critical operations are often related to I/O. ReWorks customized the device driver framework to optimize I/O throughput and response speed. At the same time, ReWorks implements pluggable management of device drivers.

I/O devices are managed through device driver tables. The operation of the device includes initialization, opening, closing, reading, writing, control, polling, etc.

ReWorks supports character device, network device and block device management. Including multi serial port card, audio and video card, infrared interface, I2S, I2C, SPI, digital acquisition card, network interface, LCD, touch screen, USB, 1553, can, NVRAM, flash, DOC, dual port RAM and other equipment.

ReWorks provides binary compatible device driver management with VxWorks, which provides a fast migration method for VxWorks applications.

(6) Clock / Timer Management

ReWorks clock management provides time slices for rotation scheduling. It is used for timeout management of semaphore, event, message, signal and other operations.

Based on the watchdog mechanism, ReWorks provides timer tasks to realize the unified timing triggering of specific functions in the system. Meanwhile, it implements the timer interface in line with POSIX specification and completes timing triggering through an asynchronous signal mechanism.

ReWorks implements a high-precision clock. It provides nanosecond timing.
(7) Task Communication

In order to meet the needs of task communication, synchronization and mutual exclusion, Reworks designs and implements four communication mechanisms: semaphore, message queue, event and asynchronous signal.

Semaphore is an effective means of sharing resources and synchronous management provided by Reworks. Semaphores include mutually exclusive semaphores and count semaphores. Mutual semaphore is used for mutual operation in a critical area and supports priority inheritance and priority ceiling to solve the problem of priority inversion. Count semaphores are used to manage a specific number of resource entities.

Message queuing provides formatted data communication and synchronization mechanisms. It supports message priority queuing and data exchange among tasks, tasks and ISR. When receiving a message, if the message queue is empty, the task can choose to wait, timeout wait or return unconditionally.

Reworks provides 32 event flags for each task. Each event flag has two states: set and clear. The number of events does not accumulate. When in the set state, indicates that the related event has occurred. Tasks and ISR can use event flags for synchronization and communication.

Asynchronous signals provide a UNIX like "soft interrupt" mechanism. There are 64 signals per task. Reworks supports real-time queuing signals and allows users to customize asynchronous signal processing functions to ensure that every real-time signal is processed.

(8) File System

Reworks realizes the virtual file system layer by abstracting each module of the file system. Thus, it can support a variety of file systems, including fat, JFFS, NFS and other file systems.

(9) Network Protocol

Reworks provides TCP, UDP, IP, ICMP, ARP and other network protocols. It provides a standard socket programming interface and implements typical Internet application protocols such as HTTP, FTP, TFTP, Telnet, sun RPC, and related services.

In addition, Reworks realizes the network management information base based on MIB-II standard and enhances the management and maintenance function of the system.

(10) VxWorks Compatibility Layer

Through the encapsulation of VxWorks System call interface and related macro definitions, Reworks realizes VxWorks source compatibility on the application layer. At the same time, the binary compatibility of VxWorks driver is realized by referring to VxWorks driver framework.

3. Design of real time communication software

3.1 Hardware environment

The hardware environment of the real-time communication software is realized by the way of Loongson 2K1000 main processor + FPGA + interface. Loongson 2K1000 is a new medium-sized CPU with low power consumption and advanced performance launched by Longxin Zhongke Technology Co., Ltd. in 2017. 2K1000 processor integrates two GS264 processor cores. The peripheral interface of the chip includes two PCIE2.0, one SATA2.0, four USB2.0, two DVOs, 64 bit DDR2/3 and many other interfaces. With Loongson processor as the core, CPU and other core processing units are integrated on the single board. Through FPGA to expand the rich external interface, it can meet the requirements of 1553B, GPIO and other interfaces. Firmware BIOS implements hardware initialization, detection and boot the operating system kernel. In terms of the operating system software, Reworks operating system is the basic software platform. Provide support in device driver layer, develop and support environment layer to integrate common compilation tools, various supporting environments and basic software layer and provide support for subsequent application deployment.
3.2 Software Function Design

The real-time communication software undertakes the task of data forwarding and protocol conversion between various software. Generally, it has the characteristics of high real-time requirements, large amount of data, numerous external interfaces and so on. As the main part of software system data exchange, real-time communication software plays an essential role in the normal and effective operation of the system. In the software system, the main data interface modes are network, serial port, bus and so on. Different interfaces need reasonable interconnection to realize information exchange.

3.2.1 Signal Acquisition

The real-time communication software needs to collect the start and brake signals of the vehicle and send the collected data through the bus. At the same time, it is necessary to send the external input control signal to the controller. Therefore, the real-time communication software needs to read and write the input and output modules of digital and analog quantities.

3.2.2 Data Communication

In the real-time communication software, the main interface mode includes network and bus. Software modules are interconnected through the network. After the network connection is established, TCP or UDP is used for data communication. Most of the real-time communication software is a console application program, which does not need to design the interface only prints the necessary information in the console. In view of the characteristics of high real-time requirements and large amounts of data in real-time communication software, information interaction software mostly adopts multi task structure: data receiving task, data processing task and data sending task. The real-time communication software can be divided into five parts: initialization, interrupt processing, data receiving, data processing and data sending. Software initialization completes network interface, disk partition, interrupt setting, task startup and other functions. Because the data transmission of other software units is unpredictable, the interrupt setting is used for processing. When an external interrupt is generated, the interrupt service program is called to inform the data processing task to read the data. The data receiving task receives the notice from the interrupt service program, receives the data from other software units from the network port and stores the data in the corresponding data buffer. The data processing task takes the data to be processed from the receiving data buffer, analyzes, processes and reorganizes the data according to the data type. Then the processed data is put into the corresponding sending data buffer. The data transmission task is to take out the data to be sent from the data transmission buffer, add the frame header, frame tail and check information and then send it to the corresponding destination address.

The real-time communication software processing process is as follows:
1. Initialize the network and memory of the board. Set network card IP and available memory size and so on. At the same time, set the necessary parameters. Start three tasks: data receiving, data processing, and data sending.

2. The data receiving task puts the received data into the data receiving buffer. Send signals to data processing tasks at the same time.

3. The data processing task takes data from the data receiving buffer after receiving the signal. It parses, reassembles the frames according to the protocol and puts them into the data transmission buffer.

4. The data sending task sends the data of the data sending buffer to the corresponding destination address.

![Task Diagram](image)

**Figure 3. Task Diagram**

3.3 Software Development and Debugging

Before running the executable file on the target machine, it is necessary to configure the parameters of the target machine, including the IP address of the debugging network port of the target machine and the starting mode of the software on the target machine. The target computer for information interaction has multiple network interfaces, one of which is used as the debugging network interface to connect with the host computer and the remaining network interfaces communicate with other software units. The debug network port is set by the command `esetifconfig`, which is located in the same network segment as the host computer. The remaining network ports can be set directly by the command or during software initialization. Reworks has two startup modes: network startup and hard disk startup which are set by instructions.

The software development process in ReDe is as follows:

1. In ReDe, the compiled project source code is built and generated executable.

2. Send the executable to the Reworks target board. There are two ways to do it. One is to set the folder where the executable file is located to the TFTP file path in the ReDe file view. After starting the target board, use the command "load Sal; g" to make Reworks start the application in a network way. The other is to copy the .bin executable file to the C-disk directory of Reworks system by using FTP tool on the host computer. After restarting the target computer, Reworks starts the application program from the hard disk.
When ReDe is required to debug the developed program, it is necessary to open "allow program debugging connection" in the project configuration file. The refactoring project is debugged on the host computer after the target machine starts the executable after refactoring.

4. Conclusion
In this paper, the design and development of real-time communication software is completed based on the Reworks operating system with the main processor of Loongson 2K1000. This paper describes the selection of hardware and software, task division, development and debugging. The final real-time communication software has high data transmission efficiency, fast response speed, stable operation and good performance in real-time and big data fast transmission.

References
[1] Liu W, Zhang W, Xing T, Wang Y. (2001)VxWorks Operating System and Real Time Multitask Programming. SCM and Embedded System Application,5:12-21.
[2] Wang X, Zheng C, Cai H. (2011)Design of SD Card Driver Based on Reworks Platform. Modern Electronic Technology, 12:86-88.
[3] Huang X. (2011)Implementation of Context Switch of Reworks on TRICORE. Computer Engineering, S1:368-370.
[4] Zhou X. (2013)Research on the Automatic AGM Based on Reworks. Computer Application and Software, 5:211-215.