Optimization and Realization Technology of Embedded Electromechanical Control and Start

Guo Zhitian; Liu Xiuyun; He Chunhua

Qingdao University, Qingdao, Shandong, 266071, China

Key Words: embedded; electromechanical control; start; optimization

Abstract. Through analysis on start plan of electrometrical and boot process of core based on NOR flash for embedded electromechanical control system, this paper designs the plan with the shortest power hour in boot loader stage, and it simplifies and optimizes core on the basis of modification, configuration of embedded Linux and modularization technology, so it can shorten start time of core as well as optimize and realize quick start plan of embedded electromechanical control system. The experience demonstrates that embedded electromechanical control system after optimization can effectively shorten start time of system on the basis of meeting reliability and stability.

1. Introduction

With the development of computer technology and integrated circuit, high-end electromechanical control system becomes increasingly complicated and it presents characteristics [1, 2, 3] of high system integration and processing large-scale data etc. Faced with quick development in electromechanical control system, the traditional PLC can not meet its requirement, while the embedded electromechanical system with characteristics of micro-core, simple system, high real-time, strong exclusivity etc has indicated high advantages and it has completely replaced traditional PLC in some fields. As one kind of special computer system, application of embedded electromechanical control system determines that it must have the above characteristics, so tits start and quick start system have become to be the important topic on study of electromechanical control system.

On the basis of analyzing and optimizing start process of embedded electromechanical control system, for the plenty of sections consuming large time on its start, it optimizes start plan and makes necessary modification, configuration for core, as well as makes realization on mixed file system by combing with detailed application environment of embedded electromechanical control system. This paper makes study on effect of network reconfiguration on black-start process, design, safety operation, circuit flow etc, it establishes black-start model power system that completely considers start, network reconfiguration, load restoration, systematically solves black-start problem of power system.

2. Analysis on Start Process Time of Embedded Electromechanical Control System

2.1Start process of embedded electromechanical control system

In the start process of embedded electromechanical control system, there are 4 major components participate in it: Bootloader, core, root file system, application program, and start process is indicated by diagram 1. Bootloader is the first software implemented in the process of system start; it has high reliance relations to hardware of target board, undertaking initialization of microprocessor, memory and serial port etc and load task of core. After guiding load procedure to complete initialization of underlying hardware and load task of core image, it will jump to start procedure and code enforcement of core. The other parts of start process is processed by init procedure of root file system in user space, these content relies on detailed application demand.

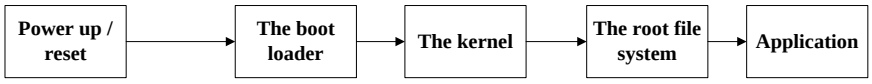

Diagram 1 Start process of embedded electromechnical control system

2.2Bootloader analysis on embedded electromechnical control system

In the embedded system, nonvolatile flash memorizer (NOR and NAND type) have replaced EPROM and EEPROM, and become to be the mainstream of memory equipment. nonvolatile flash memorizer is divided into 2 types by structure, NOR flash occupies most shares in flash market with memory of 1 to 16MB, while NAND flash is mostly used in products with plenty of memorizer.

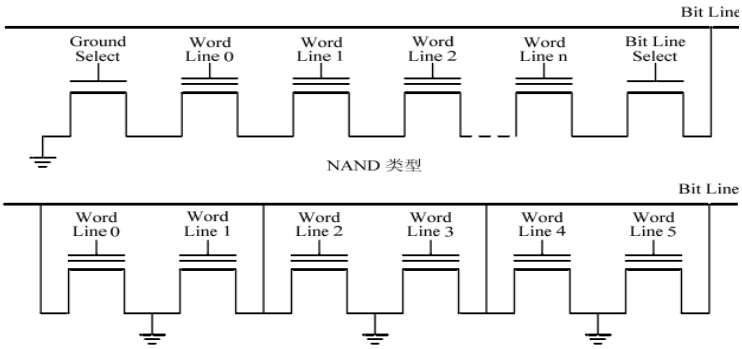

Diagram 2 Internal structure of NOR and NAND

The guidance process based on NOR flash is generally divided into 2 stages, which is indicated by diagram 3. State I usually has high reliance relations to system structure of CPU, initialization of core hardware such as CPU and memory etc are completed in this stage, its code is realized by assemble language and has higher enforcement efficiency. State II completes initialization of non-core functional equipment, such as network equipment and flash memorizer equipment etc, and it completes guidance work of core, its code is programmed by C language, it has low correlation with system structure of CPU and stronger portability. The start processes of 2 stages neither based on NOR flash and time analysis are as follows:

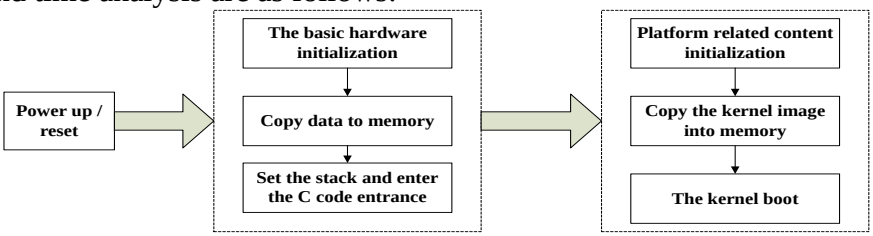

Diagram 3 Guidance process based on NOR flash

Usually, stage I consumes little time, it is less than 1s to complete this stage, stage II consumes the most time in the bootloader guidance process. In the stage II, it is key point to accelerate bootloader state speed on how to reduce time of copying core image file from NOR flash to memory.

2.3 Analysis on file system of embedded electromechnical control system

Below the virtual file system is file system based on flash memorizer and memory, such as CRAMFS file system, JFFS/JFFS2 file system, YAFFS/YAFFS2 file system and RAMFS file system etc, characteristics of each kind of file system will be developed in the subsequent paper. Below real file system is Linux memory technology device (MTD), which covers the entire memory devices, such as usual ROM, RAM, flash and DOC etc. Because of characteristics and functions of these devices are different, in order to make these devices with different technology can use the same tool and port, Linux core designs MTD subsystem. It is indicated by diagram 4, after memory device of designed file system structure setting up configuration of code drive procedure, memory space of each MTD is managed by one MTD user module. User module needs

extra flash memorizer information so as to realize reflection and reading operation of physical block of image file for the managed MTD, the date structure of bad block management is defined as follows:

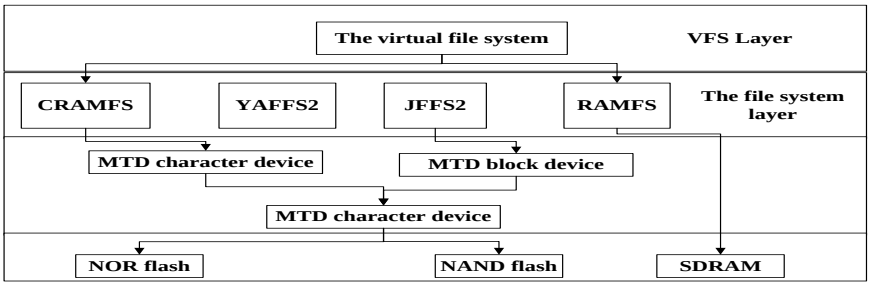

Diagram 4 Structure of embedded Linux file system

```
ypedef struct{
   struct mtd_info *mtd;      //    Obtain equipment information of MTD
   unsigned char nand_flag;      // bit zone of NAND equipment
   unsigned int blks;      // NAND flash block number of MTD equipment division
   unsigned int * vblk_map;      //    the first address of effective block table
}FSBBM;
```

When effective block table has physical block address in this block, file system will obtain corresponding physical block address, read data into system cache. If it can not find corresponding physical block address in the effective block, which indicates that this block has not been reflected, then file system firstly reflects this block into the effective block table and then obtains physical block address from effective block table, finally reads data into system cache. The read operation procedure of CRAMFS file system is indicated by diagram 5.

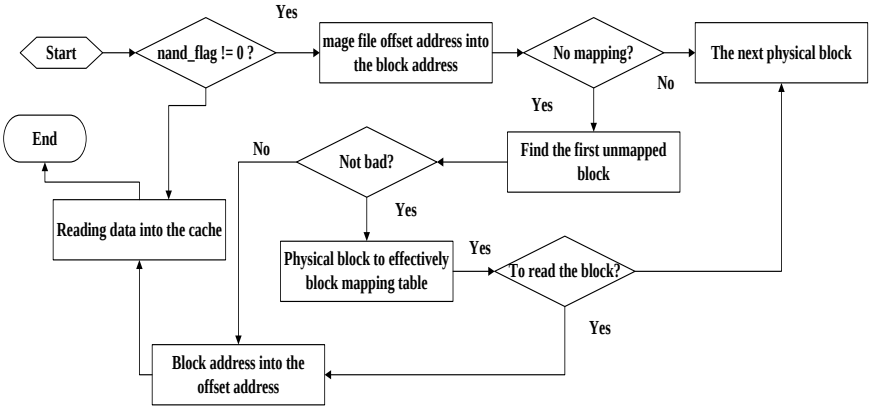

Diagram 5 Flow process of reading operation

3 Design Optimization on Embedded Electrometrical Control System and Operation System

3.1 Quick start plan of bootloader based on NOR flash

The design and realization in bootloader stage are closely related to hardware platform of embedded electromechanical control system. It should guarantee correct guidance after system is power on, it must master memory space and address division of microprocessor adopted by hardware platform, and prevents address conflict of external equipment and memorizer etc. Table 1 describes address reflection of memory space of Au1 250 SOC.

Table 1 Space reflection of physical memory for Au1 250SOC

Initial physical address	End physical address	amount（MB）	Function
0X0000 0000	0X0FFF FFFF	256	（KSEG 0/1）
0X1000 0000	0X11FF FFFF	32	I/O device of external device bus
0X1200 0000	0X13FF FFFF	32	Reserve
0X1400 0000	0X17FF FFFF	64	I/O device of system bus
0X1800 0000	0X1FFF FFFF	128	reflection of memory space
0X2000 0000	0X7FFF FFFF	1536	reserve
0X8000 0000	0XEFFF FFFF	1792	Without using
0XF000 0000	0XFFFF FFFF	256	Detection debugging

From physical address 0X18000000 to 0X1FFFFFF, totally 128MB address space, from this we can use it to reflect memorizer such as NOR flash, NAND flash, ROM etc and network equipment etc. It needs to be noted that in the start plan based on NOR flash, it should guarantee normal start of system, abnormal reset vector 0X1FC0　0000 must be included in the reflection range of NOR flash, and the initial terminal address of guiding this code must be in the abnormal reset vector.

3.2 Design optimization on time consumption of electrometrical control and start

The clear operation of memory is relatively time-consumption, BSP in PowerPC8548 E is uses function bootClear 0 to make clear operation on memory, its bootclearQ function is indicated as follows, function of this function is to make clear operation on other memory address except for interrupt vector of 4k, while memory of PowerPC8548E is 64M, so it needs to make read 0 operation with memory of 64M-K, this operation needs about 3.1 seconds, time consumption is longer, it is one large item that can make time optimization, writer uses function bootClear (void) to realize clear operation of memory.

```
LOCAL void bootClear (void)
{
/* fill from the bottom of memory to the load image */
fillLongs ((UINT "SYS—MEM—BOTTOM,
((UINT) RAM_DATA_ADRS - STACK_SAVE -
(UINT)SYS_MEM_BOTTOM) I
sizeof (long), 0);
I* fill from the load image to the top of memory */
fillLongs ((UINT *)end, ((UINT)SYS_MEM_TOP - (UINT)end) I
sizeof(long), 0);
}
```

3.3 Optimization and realization of embedded Linux file system

For system initialization procedure of root file system, because embedded system is usually worked under single user model, developer can use init procedure provided by BusyBox suite which is specially developed for embedded system to complete start work of system.

Init procedure of BusyBox is the first application program implemented in the target board; it is also the symbol link of /bin/busybox, init process of BusyBox is indicated by diagram 6, this inittab file will implement the following actions:

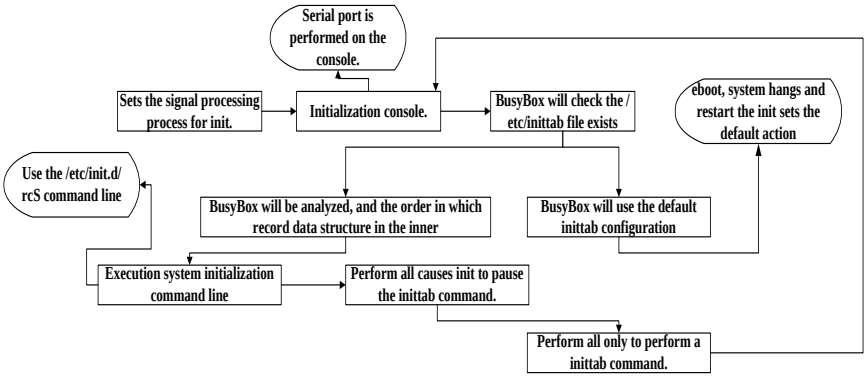

Diagram 6 Optimization process of Embedded Linux file system

① Set etc/init.d/rcS as initialization file of system.

②It will start /bin/sh procedure when corresponding process is ended.

③ If init is restarted, set /sbin/init as the procedure it can implement.

④ It will implement corresponding procedure when push combination key.

⑤ Tell init, it can implement uamount command to unload all the file system, use read-only model when unload is failed so as to protect file system. Copy these chained library parts and relevant symbol to the /lib content of root file system. After Glic parts are installed in the root file system, we can use them when application program is running.

4. Optimization Demonstration of Electrometrical Control

In the test, we respectively use vmlinux core image without compression and zImage file through compression by gzip algorithm, these 2 core images are generated under the same core configuration with different form. It chooses JFFS2 of usual embedded system to compare with mixed file system adopted by this plan. As for design problem, it adopts method of reading CPU real-time clock. In the bootloader stage, read one real-time clock time and printed to terminal through serial port after initialization of basic hardware is completed. When core starts to construct virtual file system and then print time stamp again. Tests for quick start of embedded electromechanical control system are implemented 20 times, it makes tests for 20 times before and after optimization, the average value of test result is indicated by table 2. It needs to be described that size of core image in the first stage will directly affect result after optimization, the larger of image file, and the more quick start time reduced after optimization. MTD division in the second stage will also directly affect result after optimization, the larger of file system region, the more time of start time after optimization.

Table 2 Time consumption comparison of system before and after optimization (unit: second)

Item name	Time consumption of stage I	Time consumption of stage II	Total time consumption of system start
Before optimization	8.30	7.13	15.43
After optimization	6.24	1.06	7.30
Time reduction	2.06	6.07	8.13

As it is indicated by diagram 7, the time change curve correspond to 2 kinds of different strategies in T = 3 hours. Although start optimization strategy of using T = 12 as target can make system obtain much more control effect in the longer time period, in the previous 3 hours, its optimization effect is obviously inferior to optimization strategy of unit start of using T = 3 as target.

Diagram 7 Comparison of different optimization time periods

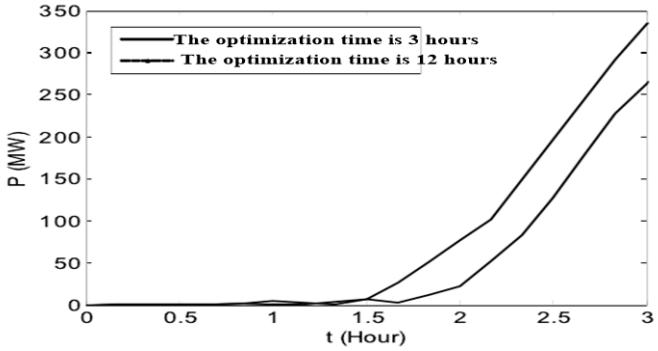

Diagram 7 Comparison of different optimization time period

5 Conclusions

In the aspect of embedded file system, it describes micromation method and basic usage of MTD, using CRAMFS file system as root file system, combing with realization method of mixed file in YAFFS/YAFFS2 file system to make description. This mixed file system meets the high reliability requirement of embedded electromechanical control system, and realizes quick mounting of root file system. After optimization and realization of embedded electromechanical control system is completed, we use the refrigeration unit control in the refrigeration and cooling system, make demonstration on optimization result of electromechanical control system and get corresponding test data.

References

[1] Ge Rui，Dong Yu，Lv Yuechun. Balckout Accident Analysis and Illumination on Chinese Grud Operaton. Grid Technology，2007，31(3)：1-6

[2] Bian Hongyuan. Study and Realization on Industrial Robot Based on PLCControl [D]. Nanjing：Southeast China University，2005.

[3] Deng Gaoshou，Pan Hongxia. Application and Development of PLC in Industrial Automation [J]. Mechanic Management and Development，2006，（3）：99-100

[4] Yin Hongyi. PLC Theory and Practice M]. Beijing：Tsinghua University Press，2008.

[5] Huang Yourui，Ling Liuyi，Chen Zhencui. Design and Development of Embedded System [M]. Beijing：National Defence Industrial Press，2009.

[6] Yan Feng，Yan Ping，Hu Xinyuan，Yi Runzhong. Study and Realization of Start Technology Based on Au1200 NAND Flash [J]. Mechanotronics，2008，（11）：37-40. .

[7] Deng Bao, Zhao Xiaodong. Study on Embedded Interconnection Based on Serial RapidIO[J]. Aeronautical Computing Technique, 2008, 33(5):29-32.

[8] Chul Lee，Sung Hoon Baek，Kyu Ho Park. A Hybrid Flash File System Based on NOR and NAND Flash Memories for Embedded Devices[J] . IEEE TRANSACTIONS ON COMPUTERS，2008，57（7）：1002-1008.

[9] Huang Liang, Liu Fuyan. High-speed Interconnection Networl Based on RapidIO and Mempry Reflection [J].Computer Engineering, 2008,12(7): 44-45.

[10] Yang Bin. Theory and Application of Embedded System Principle [M]. Chengdu: SouthWest JiaoTong University Press, 2006.