ORIGINAL RESEARCH ARTICLE

Design and Development Based on Embedded System

Huiguo Peng,Weiyong Yao,Weiyi Lin

Information Engineering College, Panzhihua University of Technology, Sichuan, China

ABSTRACT

The demand for equipment has been unable to meet the current and future high performance application and development needs with the rapid development of electronic technology and upgrading of traditional industries in China. At the same time, intense market competition and technical competition require that the development cycle of the product to be as short as possible. Obviously, the software and hardware of embedded system are becoming more important foundation of technological innovation in various fields. The embedded system is a combination of advanced computer technology, semiconductor technology, electronic technology and the specific application of various industries. This determines that it must be a technology-intensive, capital-intensive, highly fragmented and innovative knowledge integration system. The embedded type is based on the application of computer technology-based hardware and software can be tailored to adapt to the application system on the function, reliability, cost, size, power and other strict requirements of the special computer system. The emergence of consumer appliances and cheap microprocessors embedded systems will form larger application areas in daily life to as the embedded system usually has low power consumption, small size and high integration. Embedded employment is widely embedded in the social needs of large talent. The content of the paper is clear in the embedded control system overview - development - application writing ideas in three chapters gradually. Chapter 1 introduces the definition, frame, characteristic, development history, current situation and prospect of the embedded control system in detail. The development of the embedded control system mainly discusses the development steps and methods. The third chapter embedded control system. This paper discusses the practical application of embedded control system in public life. As the application of these research results in the final summary of the various aspects of the system capacity and analysis of the existing problems for further research to provide a direction and valuable experience.

KEYWORDS: Electronic technology embedded system control computer integrated system

Citation: Huiguo Peng, *et al.* Design and Development Based on Embedded System. 1(1): 1–9.

Correspondence to: Weiyi Lin, Information Engineering College, Panzhihua University of Technology, Sichuan, China. weiyionly@126.com

1. Introduction

1.Embedded systems

The computer system can handle and manage a variety of data including the text, numbers, pictures and various instructions. People want to create a variety of intelligent machines and these machines need a brain system. Some of these small machines need to give them a small set of can be embedded in the 'brain' system. How smart the 'brain' depends on its software. This type of software is hidden in some of the larger systems to manage and control these systems with a microprocessor-specific software hardware system called embedded computer systems, usually is embedded systems. As the embedded system itself is a very broad extension of the term, the entire product with the embedded characteristics of the control system can be called embedded systems. Therefore, it is difficult to give it an accurate definition. In general, the embedded system can be divided into hardware and software parts, the hardware generally consists of high-performance microprocessors and peripheral interface circuit while the software consists of application platform and program made up of hardware abstraction layer, operating system and board support package.

2. The purpose and meaning of the question choosing

Embedded system technology has been widely used in industrial control systems, information appliances, communications equipment, medical equipment, intelligent instruments and other fields such as mobile phones, ADA, MP3, handheld devices, smart phones, set-top boxes and others. It can be said that embedded systems are everywhere.

The market demand for embedded systems is rapidly growing. Enterprises are putting their efforts to use the development of embedded systems to keep pace with the market demand and the need of competition. However, the development of embedded systems based on chaos and methods of research is seriously lagging behind at the same time which is related to its characteristics into the city system.

3. The embedded system of advanced programming language

The Ada language was a powerful generic system development language developed and used by the US Department of Defense in the 1970s, initially for Ada83. It supports modular, independent compilation, co-processing and other functions. Its reliability, maintainability, readability are quite good. Later, it was improved in order to support the OOP (Object-Oriented Programming) better and formed the current widely used Ada95. Ada language can greatly improve the system's clarity, reliability, maintainability and other performance indicators [2,3]. It is the only language designated by the US Department of Defense which can be used in the development of military systems.

The C language was a system programming language studied and put into use by Dennis Richie in 1972 at AT& Bell Labs. Its design goal is to make C has efficiency of assembly language and high-level language programmability. Its most representative application is the UNIX operating system. From the mid-20th century, C language involved in real-time system has been generally welcomed and now it is the most widely used embedded system programming language. C ++ was developed and put into use by Bjarne Stroustrup in Bell Labs in 1995. C + + in support of modern software engineering, OOP, structured and other aspects of the C has been fruitful improvements, but slightly weak in the program code capacity, execution speed, complexity of the program than the C language program performance.

Modula-2 was developed by Nicklans Wirth in the late 1970s under the system design language developed by Pascal and Modula. Its main goal is to improve Pascal in terms of modularity, system programming and co-processing. Modula-2 has a strong type of inspection capabilities and a wealth of low-level support. Therefore, it can be used to design a complete real-time program without the support of assembly language. Modula-3 was developed in 1988 by DEC (Digital Equipment Company) and ORC (Olivetti Research Center) according to Modula-2 developed and put into use in the system development language. The goal is to design a powerful but easy-to-use, general-purpose programming language. It has improved Modula-2 in terms of co-processing, OOP, automatic garbage collection and support for C and UNIX.

Java is the network language and embedded systems are in the function, price, size, power consumption, time to market have special requirements. So, the Java language is not suitable for embedded systems applications by the speed and code capacity constraint. However, Sun Company is not willing to give up this development potential of the huge application market. He improved and modified Java then released J2ME (Java2 Micro Edition). It is a subset of the Java API that contains only the key features of Java and is designed specifically for embedded systems with harsh requirements for memory. J2ME roughly divided the object applied into two categories which are the equipment with internal memory in between 128KB ~ 512KB and equipment with internal memory is greater than 512KB. Different user interfaces and software packages will be provided according to different categories.

4. To solve the key issues

The key issues to be addressed are:

1. The various modules of the test of hardware.

2. The design and debugging of software.

2. ARM Processor Architecture and ARM Instruction Set

1.The register and processor mode

1.ARM has 7 basic working modes:

User: Non-privileged mode, most of the tasks performed in this mode. Limit your memory access and you cannot read hardware directly.

Normal program execution mode

FIQ: This mode will be entered when a high-priority interrupt is generated.

High-speed data transmission and channel processing

IRQ: This mode will be entered when a low priority interrupt is generated.

The usual interrupt handling

Supervisor: This mode will be entered when a reset or soft interrupt instruction is executed.

A protection mode used by the operating system

Abort: This mode will be entered when an exception is accessed.

Virtual storage and storage protection

Undef: This mode is entered when an undefined instruction is executed.

Software simulation hardware coprocessor

System: Uses the privileged mode of the same set of registers as the User mode.

Privileged operating system tasks

2. Register group

In the 26-bit system, ARM processor has twenty-seven registers. Some of them are only used under certain conditions, so usually only sixteen registers can be used at a time.

• Register 0 to Register 7 is a general purpose register and can be used for any purpose. Unlike the 80 x 86 processor, a particular register is required to be used as a stack access or the result of a mathematical calculation is placed in an accumulator like the 6502 and the ARM processor is highly flexible in register usage.

• Registers 8 through 12 are general-purpose registers but use their shadow registers when switching to FIQ mode.

• Register 13 is typically used as an OS stack pointer but can be used as a general purpose register. This is an operating system problem, not a processor problem, so if you do not use the stack, you can corrupt your code freely as long as you later restore it. Each processor mode has a shadow register for this register.

• Register 14 holds the address of the return point in full to facilitate the writing of subroutines. When you execute the branch with the connection, the return address is stored in R14. Besides, the exit address is stored in R14 when the program is run for the first time. All instances of R14 must be saved to other registers (not actually valid) or on a stack. This register has shadow registers in each processor mode. Once the connection address has been saved, this register can be used as a general-purpose register.

• Register 15 is the program counter. It holds the status of the processor in addition to holding the twenty-six digits of the address currently used by the program.

For clarity, provide the following chart:

User mode SVC mode IRQ mode FIQ mode APCS

R0 ------- R0 ------- R0 ------- R0 a1

R1 ------- R1 ------- R1 ------- R1 a2

R2 ------- R2 ------- R2 ------- R2 a3

R3 ------- R3 ------- R3 ------- R3 a4

R4 ------- R4 ------- R4 ------- R4 v1

R5 ------- R5 ------- R5 ------- R5 v2

R6 ------- R6 ------- R6 ------- R6 v3

R7 ------- R7 ------- R7 ------- R7 v4

R8 ------- R8 ------- R8 R8_fiq v5

R9 ------- R9 ------- R9 R9_fiq v6

R10 ------ R10 ------ R10 R10_fiq sl

R11 ------ R11 ------ R11 R11_fiq fp

R12 ------ R12 ------ R12 R12_fiq ip

R13 R13_svc R13_irq R13_fiq sp

R14 R14_svc R14_irq R14_fiq lr

---------- R15 / PC ---------- pc

The rightmost column is the name used by the APCS code. APCS, ARM Procedure Call Standard, provides a compact mechanism for writing routines that can be intertwined with other routines. The most notable point is that there

are no clear restrictions on where these routines come from. They can be compiled from C, Pascal or can be written in assembly language.

APCS defines:

• Restrictions on the use of registers.

• Use the stack convention.

• Pass / return parameters between function calls.

• A stack-based structure that can be 'backtracked' to provide a list of functions (and given parameters) from the point of failure to the program entry.

Condition:

• N = 1 - result is negative, 0 - result is positive or zero

• Z = 1 - result is 0,0 - the result is not 0

• C = 1 - carry, 0 - borrow

• V = 1 - result overflow, 0 result does not overflow

Q:

• Only ARM 5TE / J architecture support

• Indicates whether the enhanced DSP instruction overflows

J bit:

• Only ARM 5TE / J architecture support

• J = 1: The processor is in Jazelle state

Interrupt interrupt bit:

• I = 1: Disable IRQ.

• F = 1: Disable FIQ.

T Bit:

• Only ARM xT architecture support

• T = 0: The processor is in ARM state

• T = 1: The processor is in Thumb state

Mode bit (processor mode bit):

• 0b10000 User

• 0b10001 FIQ

• 0b10010 IRQ

• 0b10011 Supervisor

• 0b10111 Abort

• 0b11011 Undefined

• 0b11111 System

When the processor executes in ARM state:

• All instructions are 32 bits wide.

• All instructions must be word aligned.

• So the value of pc is determined by bits [31: 2], bits [1: 0] are undefined (so the instruction cannot be halfword / byte aligned).

When the processor executes in Thumb state:

• All instructions are 16 bits wide.

- All instructions must be halfword aligned.
- So the pc value is determined by bits [31: 1], bits [0] are undefined (so the instruction can not be byte aligned).

When the processor executes in Jazelle state:

- All instructions 8 bits wide
- Processor implementation of word access to take four instructions at a time

3. Embedded Linux Programming Environment

1.The use of Linux compiler vi

1.vi mode

Vi has three modes which are the command line mode, insert mode and the bottom line mode. The following specific describes the function of each mode.

(1) Command line mode.

Users in the use of vi editing files, the initial entry for the general model. In this mode, you can move the cursor up and down to 'delete characters' or 'line delete' and other operations, you can also 'copy', 'paste' and other operations, but you cannot edit the text.

(2) Insert mode.

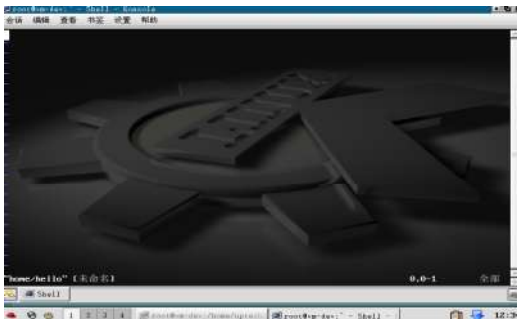
Only in this mode, the user can edit the text input. The user can press the Exs key to return to the command line mode.

(3) Bottom line mode.

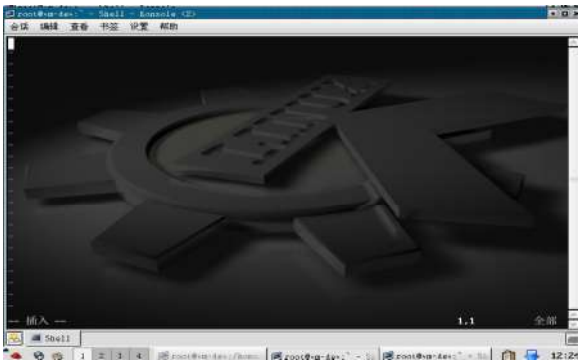
In this mode, the cursor is on the bottom line of the screen. The user can save or exit the file. You can also set the editing environment such as looking for a string, column number and others.

2.vi the basic process

(1) Enter vi, that is entering vi hello in the command line,. At this point into the command line mode, the cursor is located at the top of the screen.



(2) In the command line mode, enter i into the insert mode, as shown in the following figure, you can see, in the bottom of the screen shows 'insert' that insert mode, in this mode can enter text information.



(3) Finally, in the insert mode, enter Esc which changes the current mode into the command mode and in the bottom line, enter ': wq' (save out) to the bottom line mode, as shown below.

2. gcc compiler

(1) Pretreatment

At this stage, the compiler compiles the stdio.h in the above code, and the user can view it using the gcc option '-E', which causes gcc to stop the compilation process after the preprocessing ends.

(2) Compilation

Gcc compiles the code into assembly language.

(3) Compilation

The compilation phase is to convert the '.s' file into the target file. The reader can use the option '-c' to see the binary code of the assembly code that has been converted to '.o'. The statement is as follows.

[Root @ localhost Gcc] # gcc -c hello.s -o hello.o

(4) Link

After successful compilation, it entered the link phase. Here is an important concept related to the function library.

3. Make project manager

1. Makefile introduction

When the make command is executed, a Makefile is required to tell the make command what to do to compile and link the program.

First, we use an example to illustrate the Makefile's writing rules. This example comes from the GNU make use manual, in this example, our project has 8 C files, and 3 header files, we have to write a Makefile to tell the make command how to compile and link these files. Our rule is:

(1). If this project is not compiled, then all of our C files should be compiled and linked.

(2). If a few C files of this project are modified, then we only compile the modified C file and link the target program.

(3). If the project's header file is changed, then we need to compile the C file that references these header files and link the target program.

2. Makefile basic structure

(1) Objects that need to be created by the make tool, usually the target file or executable file.

(2) The object to be created depends on the amount of documents.

(3) Create each target need to run the command.

3.Makefile variable

Makefile is often contains a lot of files and commands, which is reason of existence of Makefile. Here you can give a little more complex Makefile to explain.

In the above example, let's look at the rules of edit:

Edit: main.o kbd.o command.o display.o \\

Insert.o search.o files.o utils.o

Cc -o edit main.o kbd.o command.o display.o \\

Insert.o search.o files.o utils.o

We can see that the [.o] file string is repeated twice and if our project needs to add a new [.o] file, then we need to add two places (should be three places, there are A place in clean). Our makefile is not complicated, so add it into two places will not be very tired. However, we may forget a need to join the place when the makefile is complicated which led to the compiler failed. So, for easy maintenance of makefile, we can use variables in makefile. The variable for makefile is also a string that understands that the macro in the C language may be better.

4. Conclusions

With the rapid development and widespread use of Internet / Intranet technology, many companies have built their own local area network. They not only can quickly publish and communicate information through the network, but also search and access to information quickly through the network. The network has changed our daily life and brought profound changes in enterprise management. The establishment of Web-based recruitment management system is to meet the needs of the development of the times.

Embedded system has great potential in future development with embedded systems as a symbol of the post PC era has arrived. However, the embedded system is a very close combination of hardware and software topics especially related to bios development and boot loader development. You are required to have a deeper understanding of the hardware. Learning embedded systems required us to master a lot of knowledge. So, the order of learning must not be chaos. We should start and carry on step by step start in understanding the embedded system development architecture. Build the development environment needs: hardware platform, compiler, debugger, RTOS, C / C ++ library, protocol stack and others. It is best for embedded scholar to first analyze the source code of a RTOS - UCOS which is the easiest then re-analysis of a communication protocol stack implementation - TCP / IP which is the most practical. Proficient in a single-chip development integration environment - keil C is the most classic while proficient in a MCU development integrated environment - ADS 1.2 is the most popular.

Through this graduation design, I have had more specific understanding on my computer hardware and the entire architecture of the computer from the bottom to the top of the application layer and the middle of those agreements. I also have a detailed understanding on the infiltration into the lives of people in various fields embedded system design and development process. The ability to solve problems for me to understand the problem is an excellent calcination and my computer level is also a great improvement.

References

1. Kang Yimei, Zhang Yongge, Li Zhijun, Hu Jiang, Wu Wei. 'Embedded Software Test'
2. Liu Yongtao, editor. ARM embedded architecture and interface technology'
3. Ge Yuhui, Tian Jingbing, Tang Lenggang. Comparative study of human resource management model [J]. Commercial Research, 2002,252 (8): 76
4. Tian Hongmin, Lu Weifeng. Design and Implementation of Human Resource Management System. Journal of Jiangxi Education Institute (General Sciences) .2003 (6).
5. Wang Feng, Zhang Jing, He Wenjuan, Internet-based human resources management system, microcomputer development, 2003 (9) P (95 ~ 97).
6. Yu Hong Chuan, Zhang Zhisheng, Shi Jinfei. ERP Human Resource Subsystem Solution Research and Implementation Modern Manufacturing Engineering, 2003 (1) P (12 ~ 14).
7. Timothy Budd. Object-oriented Java programming thinking. Beijing: Tsinghua University Press, 2002 (8)
8. Chen Jinhui, Wang Jinghao, ed. (XML and Java programming Daquan) Beijing: China Railway Publishing House 2002 (2), 36.
9. ZSC / Pacific Internet Institute compiled, JAVA programmers must read: basic articles (2) object-oriented programming concept. Pacific Internet.
10. US Way S. Horstmann, et al. The latest Java2 core technology volume workers: Principles. Beijing: Mechanical Industry Press, 2002 (2).
11. 'JDBC and Java database programming' Beijing: China Electric Power Press, 2002 (3), 34.
12. Hua Qingyuan, et al., editor. Embedded Linux operating system