Research on Application of Computer Recognition Technology in C Language Programming Modeling System

Yikun Zhao
Hebei University of Technology, China
zyk_1214@163.com

Abstract. C language programming is more and more favoured by the majority of technical personnel in embedded systems. The application of C language technology in computer software programming can effectively avoid unnecessary language logic problems, ensure the smooth progress of programming work and effectively improve the quality and efficiency of programming. For the development of C language embedded system, the programming ideas of system software are explained, the functional module division based on hierarchical design is given, and the realization methods of project organization, program framework design, module reuse design, etc. in the software development process are clarified. To solve the contradiction between C language flexibility and application development engineering. Although it is introduced for the ARM platform, the basic experience and algorithms are also suitable for software design on other embedded platforms.

Key words: C language, computer software programming, embedded system, software.

1. Introduction
The wide application of C language is mainly due to its excellent portability, which can run on a variety of software or hardware platforms with different architectures; the grammatical mechanism is compact, flexible in use and can directly access the hardware; and has high operating efficiency. Compared with the general form of software programming, the embedded system has its particularity. It is oriented to a special-purpose computer system, which has both the versatility at the application level and the complexity of hardware operation. Therefore, using an efficient and reasonable programming framework and design process to standardize the management of the programming process is very beneficial to improve the safety and reliability of the program, and at the same time can improve the efficiency of software development, execution, and maintenance [1]. On the basis of conventional software programming, apply hierarchical thinking, modular thinking, and object-oriented technology to the embedded C language programming system, and divide the modules, hierarchical architecture, interrupt service programming, and system software design in embedded programming. Based on the current popular software engineering ideas, this article applies object-oriented design technology and layered technology to C language embedded system programming, and discusses the system design ideas, layered practice, and program architecture in the development of C language embedded systems.
2. System design ideas
Whether it is process-oriented thinking or object-oriented thinking, it is to better transform development requirements into software module division, and then into program functions that can be implemented with code. In actual system development, it does not mean that object-oriented programming languages such as C++ or Java must be used for object-oriented programming [2]. The encapsulation and inheritance of program modules can also be realized with C language. The key is how to reflect the high level of module division. "Cohesion, low coupling" characteristics, improve the reusability and scalability of the code. With the increasing scale and complexity of embedded software systems, how to better divide modules and develop complex software that can work correctly has become a major problem facing system design.

3. Module division
Module division refers to dividing a complex and comprehensive program into small sections according to different functions that can be realized, and each small section can realize a specific function. The software should minimize the connection with external modules, improve the portability of the program, and clarify its achievable functions. In general, embedded systems include hardware drive modules and software function modules [3]. A hardware drive module corresponds to a specific hardware, and the basic requirements of low coupling and high cohesion should be met as much as possible when dividing software functional modules. When designing modules with different functions, there are two important principles: (1) Module independence, there will be no information cross-confusion between each functional module, no mutual influence and interdependence, function instructions in a module do not require other modules The connection between modules is realized through function interface. (2) Orthogonality, each functional module is uniquely determined, and the variables in it are unambiguous with the functions it implements. There are the following precautions when dividing modules:

(1) The data and external functions that a certain module provides to other functional modules need to be declared with the extern keyword in the header file (.h). (2) The global variables and function information in a function module must be declared with the static keyword at the beginning of the .c file. (3) A module is composed of a .h file and a .c file. The interface of this module is declared in the header file (.h). (4) No variables can be defined in the header file (.h file). Defining variables is a concept in the assembly stage, and memory will be allocated according to requirements.

4. Layered architecture of embedded system development
Layering technology is an important means to cope with the increasing complexity of software and the continuous expansion of functions. Through the use of layered technology, many complex issues can be divided and simplified, transformed into specific application functions, and multi-layered structures and middleware technologies have been derived, which play an increasingly prominent role in software development activities [4]. As the application complexity of embedded systems continues to increase, the use of layered technology to rationally design embedded systems has become the key to improving software development efficiency, execution efficiency, and maintenance efficiency.

4.1. Layered architecture design method
Layering is to better meet the needs of various functions in programming, to decompose an overall function into small functions, implement them in layers, and then integrate them. In this way, it is necessary to decompose the code into modules of different conceptual levels, determine the connection of each module, make reasonable connections, and finally realize the complex overall function (Figure 1). There are fixed principles that need to be followed when layering: (1) When layering, try to reduce the correlation between layers, do not affect each other and restrict each other, and reduce the portability of modules. If there is an error or problem in a certain layer, it does not affect the entire program, but only affects the realization of some functions. The technicians can modify and improve this layer in a targeted manner, without affecting other layers. (2) Each layer needs to provide support for the upper layer of the layer, and each layer can independently solve a specific problem, such as frequency sampling.
function, which can be divided into data collection layer, data processing layer, data display layer, etc. The layers are connected to each other to realize functions. (3) Try to achieve a high degree of cohesion between layers and low coupling between each other. The modules are divided within each layer to achieve the best modularity and the strongest portability. The simpler and clearer the function is, the easier it will be to realize it. However, if the function goal is too detailed, it will cause more calls and increased complexity. Therefore, a reasonable division of levels and modules requires continuous experimentation.

4.2. Application of layering technology

Through the hierarchical design of the embedded system, it is helpful to clarify the hierarchical structure and optimize the organization of functional modules, making the system design process agile and flexible, and the product function extensible [5]. The common functional module division is to design the system logic architecture around the central processor/controller, and adopt a process-oriented design idea, which is divided into functional modules such as input/output, application scheduling, device driver, and network communication. Such a division method can make full use of the processing power of the system and perform refined storage space management, but it also brings the shortcomings of cross-repetition of application logic, strong dependence on hardware, etc., it is difficult to expand the function, and the code reusability is also poor. Using the design ideas and hierarchical design methods described in this article to implement object-oriented and decentralized design of embedded systems, the system logic architecture can be divided into the following four levels (Figure 2):

4.2.1. Application management layer. Mainly realize interface interaction, business logic scheduling and other functions.

4.2.2. Algorithm protocol layer. Mainly realize the functions of model algorithm, protocol analysis, file management, database management, etc., such as position conversion calculation, compass pointer calculation, etc.

4.2.3. Function expansion layer. It mainly realizes the independence of devices, and provides functions such as universal processing and interface access of various devices, such as LCD line, circle, and rectangle processing, sensor data conversion and other functions.

4.2.4. Hardware driver layer. It mainly realizes the independence of hardware, and provides hardware operation primitive functions, such as LCD positioning, writing points, writing bytes, sensor data acquisition and other functions.

Figure 1. Layered architecture of embedded system development
5. Program design based on ARM embedded

The selection of the basic APM related program design is the most important link. In the design plan, there are certain requirements for the computer's hardware, which must be carried out under the mode-driven program, and the flow arrangement of the program in the design must meet the requirements of the program code. At the lowest level of the central processing of the drive mode in the computer hardware, the driver is established and the programming structure is constructed, and the program mode of interconnecting with the external network is realized by the way of constructing the information platform. However, the difficulty of such a design can appear in the specific operation, which is mainly manifested in the complexity of the structure in the coding and the difficulty of the later connection operation, and the most suitable connection mode cannot be formed in the docking. Here you can solve some of the difficulties encountered in the design through the C language related program editing method. When the embedded program is disconnected from the external device, the system will analyse the data at the disconnected point, and divide the accident cause system into internal causes and external causes. When the data connection happens externally, you need to pay attention to the management and repair of data variables [6]. The debug interrupt for variables is a kind of external unsynchronized event. When dealing with interrupt-related variables, choose the most appropriate parameter value, pay special attention to the change trend of the variable in the next program run, to prevent the disconnection from happening again. In general, after the editor's optimization scheme option is turned on by default, the operation of variables will be arranged in the memory as much as possible. Interrupt service routines often notify the application that an external event has occurred by changing some global variables, and these global variables should not be optimized. The writing style of c language should also be considered for efficiency.

For example, in the operation of array elements, "Amry [idx, 4] = &~1;" is better to be changed to "Amry [Yij>>2] = &~1;." The symbol ">>" mentioned here is one This kind of shift operation has corresponding machine instructions, and the "/" symbol is a division operation, and the algorithm is much more complicated. Of course, advanced compilers can generally optimize such statements, but there is no guarantee that all compilers have this feature.
5.1. LCD terminal (system I/O)

LCD terminal software is an important content of the system I/O category, mainly including LCD character display (English 8×16 dot matrix, Chinese character 16×16 dot matrix), LCD drawing (dot, line, circle, surface, bitmap, graphic rotation) etc. The 320×240-pixel LCD display can display 15 rows×40 columns of English characters, or 15 rows×20 columns of Chinese characters, and basically realize the display of graphics/images with better resolution. The most basic program for LCD display is the drawing program, and its prototype is as follows:

```
Void LCD Pixel (into, inty, charcolor)
```

Among them, x and y are the coordinates of the point, the origin of the coordinates is in the upper left corner, and color is the gray level of the point. The display of characters and bitmaps uses a dot matrix method. Lines, circles and surfaces are implemented using corresponding algorithms. Graphic rotation requires the use of coordinate transformation functions. The basic function for outputting a string to the screen is Print f, and its prototype is as follows:

```
Void Print f (Print,)
```

This is a variable parameter function, which is completely similar in function to the print f standard library function. In order to realize the processing of variable parameters, use stdarg. Define some macros in h. Print f analyses each format character, and processes each escape character (such as \n, \t, \b, \r, \v, etc.) accordingly. Print the formatted string at the appropriate position on the screen. Print f also calls a scrolling function Screen Scroll, which scrolls the screen up several lines when the cursor is at the bottom line. The Print f function not only provides a good method for LCD as a character terminal, but also provides convenience for program debugging. We can use the Print f function to print some information where the program may go wrong, which provides a considerable convenience for us to track the program. The use of Print f function in embedded system programming is very obvious.

5.2. Algorithm optimization

After selecting the data type, the algorithm of the program needs to be optimized. Optimization is mainly embodied in improving the running speed of the program and reducing the amount of code of the program, that is, optimizing the time complexity and space complexity of the program. The optimization of time complexity and space complexity are often opposite to each other. Reducing time complexity will increase space complexity. Conversely, reducing space complexity will increase time complexity. This needs to be used in specific situations. A compromise optimization that is more in line with system requirements. Most of the single-chip system simulation development environment does not provide performance evaluation of time complexity and space complexity, and developers need to perform performance evaluation to optimize the algorithm. The length of the generated code after compilation is shown in Table 1.

| Type of data       | Unsigned char | Char | Unsigned int | Int | Unsigned long | Long | Float |
|--------------------|---------------|------|--------------|-----|---------------|------|-------|
| Number of bytes    | 1             | 1    | 2            | 2   | 4             | 4    | 4     |
| Code length        | 12            | 19   | 43           | 43  | 134           | 134  | 508   |

The optimization of time complexity is mainly reflected in the execution time of the code. The execution time of a piece of code from calling to returning determines the efficiency of the code. The execution time can be calculated in the following ways. (1) Use timer: The time complexity can be estimated with the help of the microcontroller's own timer. Set the timer to automatic reload mode, start the timer before a certain piece of code is called, stop the timer after the call is completed, and read the timer to estimate the time complexity of the code. (2) Using an oscilloscope: You can use an oscilloscope to estimate the time complexity. During the execution of the code, a pulse is generated from a TCH0 port, and the running time of the code is estimated by the pulse width time displayed by the oscilloscope.
6. Conclusion
This article discusses the computer programs that are currently being studied extensively, and starts with computer assembly language and C language, and continuously introduces the inevitable connection between the two, and the operation of the programs in the process of connecting and embedding the two languages. It also introduces the computer program environment that uses both. In the process of some function transformations, the particularity of the language is revealed. At the same time, in the process of introducing the ARM embedded system, the computer program management is better improved, and the computer is in the program self-checking and checking calculations. Played a huge role in the process. In the future research, we will continue to study the programming language in the computer, continue to deepen and improve the program, and better make the computer program serve the public.

References
[1] Baftiu, N. ANALYSIS AND USE OF CRYPTOGRAPHY TECHNIQUES IN PROGRAMMING LANGUAGE C. Prizren Social Science Journal, 4(3) (2020) 1-10.
[2] Abe, K., Fukawa, Y., & Tanaka, T. Dealing with Stumbling in C Language Programming Using Visual Programming Environment. International Journal of Learning Technologies and Learning Environments, 3(1) (2020) 35-50.
[3] Li-na, L. I. U. Reform and Practice of 1+ e Fusion Teaching in the Course of C Language Programming. Computer & Telecommunication, 1(6) (2020) 49-51.
[4] Wei-wei, Y. U. N. Thinking on the Teaching Scheme of C Language Programming Based on Critical Thinking. Computer & Telecommunication, 1(3) (2020) 66-68.
[5] Feng, X. The Case Design Method in C Language Programming Teaching. Creative Education Studies, 7(4) (2019) 498-503.
[6] Xu, D., Yi, F., & Li, M. Exploration and Practice of C Programming Course Based on the Teaching Concept of Learning Community. Open Journal of Social Sciences, 8(11) (2020) 141-155.