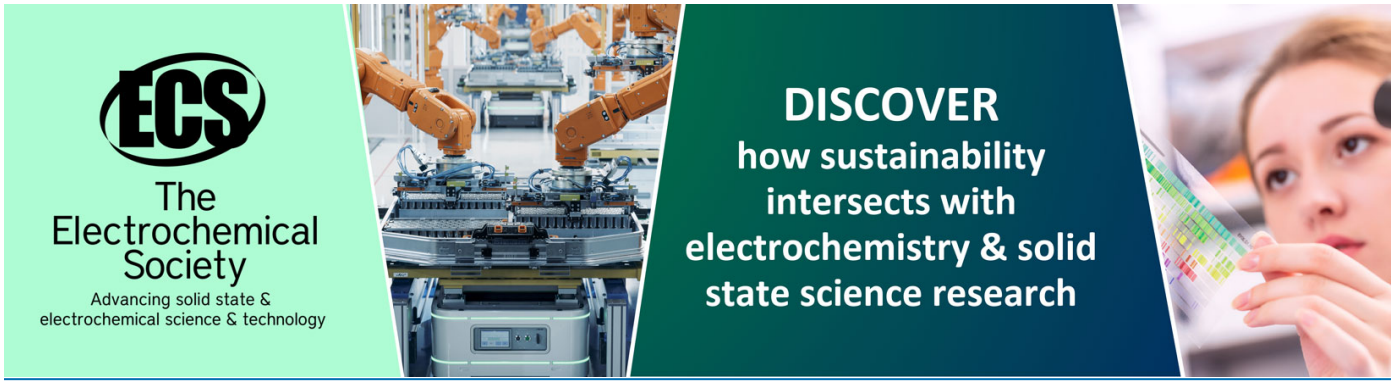
Research on embedded file management system based on Forth virtual machine



Research on embedded file management system based on Forth virtual machine

Chunmei Wan[1] and Hongbing Dai[2]

[1]School of Information science & engineering, Yunnan University, Kunming City, Yunnan Province, 650504, China

[2]School of Information science & engineering, Yunnan University, Kunming City, Yunnan Province, 650504, China

*Corresponding author's e-mail: hbdai_it@126.com

Abstract. In view of the lack of file management mechanism in the current embedded Forth operating system, a file management system is proposed in this paper. Embedded Forth system in astronomy, image processing, artificial intelligence has a wide range of applications, such as is the current popular embedded Forth system cannot be updated online, on-line modification and online programming problem, this subject adopts Forth, delayed word virtual machine technology, build operating system based on embedded Forth file management system of the SD card, to write a Forth system with SD card drive function, Forth buffer management, dynamic load, the editor's build, complete Forth communication between the system and the SD card, load the related data from SD card, Make changes in the Forth system and save them to an SD card. To build a small file management system in the embedded Forth system and realize online editing, loading and updating in the embedded system.

1. Introduction

Forth is a language invented by Chunk Moor in the United States in 1969 [1]. Forth language has interactive, portability, strong self-expansion ability, high speed code generation ability, and can quickly construct an operating system [2]. Using data stack and return stack is one of its major features [3]. Forth development is first used in the astronomical aspects [4], because of the excellent characteristics of Forth language, Forth got rapid popularization and use of language, is wider and the artificial intelligence, instrument processing, data acquisition, process control, image processing, such as computing technology direction has involved [5-6], in a number of cutting-edge products or key components of the system can be found in the sign of the Forth. Since the first Forth version appeared in 1969, there have been several Forth versions [7], such as pdp-11, Forth88, AmForth, SwiftX, PunyForth, etc. The system is based on the design idea of Forth virtual machine (FVM). The embedded operating system based on The Forth virtual machine (FVMOS)[8] was formed, and standards such as FIG Forth, forth-79, forth-83, ANSI x3.215-1994 [9], ISO/ iec15145:1997, forth-2012 [7] were successively formed. Currently, the Forth language is more widely used in embedded software and hardware [10-11].

In the past use of embedded system, the maintenance personnel constantly encountered the trouble of continuous operation and maintenance on site [12]. When many embedded firmware running online has bugs or needs to be upgraded, it often needs to be updated and maintained on site after downtime, or even taken away from the site to the laboratory to solve the problem and re-burn the

updated and maintained firmware into the system. In some special applications, a kind of online loading, online editing, online update can achieve online interaction of the operating system. For example, in the continuous observation of the remote space will expect to have this kind of operating system support: you can create the observation, control, collection, processing, communication, monitoring and a series of concurrent tasks, when a task application fails, the system outage and will not affect the other cases, the program execution by remote terminal load corresponding code files, modify online, online update, the task to run again. file management system is a bridge between the embedded system and the external memory. The embedded Forth system can realize online editing, online loading and online updating. Forth systems such as PcForth, Forth 88, fig-forth and MMSForth[13] are non-embedded operating systems, whose operation requires the cooperation of the Windows operating system. The core code files are stored in the file system recognized by Windows, such as Fat16, Fat32 and NTFS. It can change the code file in non-embedded Forth system. Since Forth is mostly applied in embedded environment, it is an urgent problem to update, edit and load programs online.

In this paper, the architecture and related operating systems of the embedded Forth system were studied, and the file management system of the embedded system based on the Forth virtual machine was proposed. Due to the portability, standardized interface, large capacity, small volume, fast transmission rate and moderate price of the SD card [14], the SD card was proposed as an external disk. Therefore, the driver between the embedded Forth system and the SD card is written, and the communication between the embedded Forth system and the SD card is realized. In this paper, AmForth language is used as the experimental language to read the blocks in the SD card, and the data of the buffer in the Forth system is written back to the SD card to realize the online loading, so there is no need to modify and burn down the system. The research on Forth at home and abroad shows that dynamic loading based on virtual machines is still lacking. Forth system is mainly applied in the embedded field. For an operating system applied in the embedded field, the significance of timely handling various emergencies is self-evident, and timely handling of corresponding events is the key [15]. Therefore, the dynamic loading of program in the embedded system based on Forth virtual machine is studied. In this paper, an implementation method of the file management system based on Forth virtual machine is presented.

2. Overall architecture of embedded file managed system based on Forth virtual machine

The current embedded Forth operating system is an operating system with real-time and multi-task execution. In the case of a bug in the real-time multi-task operating system that needs to be fixed, the file management system can realize online modify and online repair of the bug without downtime. Is realized in a project on the time slice rotation method, memory management, for the emergence of loopholes can complete the corresponding task by disruptions in a timely manner, through the synchronous mutex corresponding mechanism, but not modify online, on-line load, online update these gaps, this paper Forth in real-time multitasking operating system under the premise of loopholes in the implementation of real-time multitasking online modification, the online updates. The embedded file management system of the Forth virtual machine is an integral part of the Forth real-time multitasking operating system and ACTS as a file system. The Forth system is written from the source code of the entire Forth system through the assembly language and the colon definition and core word of Forth. In an embedded environment, the existing Forth system needs to be cross-compiled into an embedded development platform. The corresponding event processing is carried out on the target platform, and the result is obtained. During this process, if something goes wrong with the program, you have to go back to the original step, go down, modify it again, compile it, and check the result again. The development of Forth for an embedded system is shown in the figure 1.

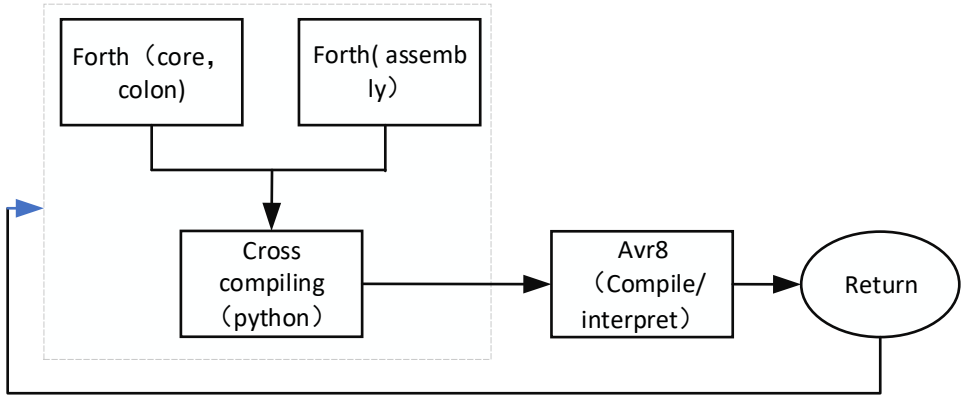


Figure 1. Traditional embedded forth development process

Due to traditional embedded development not implemented in the process of loading, online update online, on-line modification and therefore in the event when something goes wrong, can only downtime repeat this process, but down in complex environment is not chance, this paper introduced the file management system can well solve the encounter bugs to outage situation, because the SD card portability, large capacity, this paper chooses SD card as a disk, the Forth to build embedded system developing process. The source code loaded with Forth system was compiled to the target platform, linked to the SD card, and the programs were stored in the SD card. If any errors were made, the blocks where the screen editor loader was loaded were directly called on the target platform to change the blocks and run.The SD card development process for using the SD card in an embedded system is shown in the figure 2.

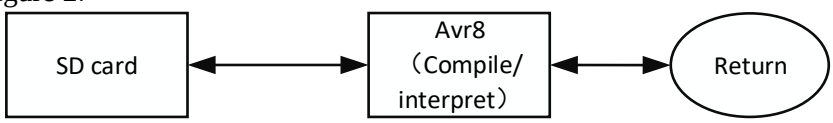


Figure 2. SD card in an embedded system

The SD card is used as the disk in this paper, and the disk is used as an extension of the main memory of Forth system. The SD card is divided into continuous blocks, each of 512 bytes in size. In a Forth system into a special area in main memory to store data from the SD card, set Forth in the area of computer memory in the address of high end, is composed of two disks buffer, each disk buffer to 512 bytes, when you need to block of data on the SD card, just block in the SD card put reads an input stream of the block coexist in a single disk buffer. In this paper, SD card is introduced as a hardware support of file management system. The first task to establish a file management system in the Forth virtual machine embedded system is to enable communication between the two. After the communication between them is realized, storage units are allocated for the Forth system in main memory to provide a place for the data exchange of the file management system, which is the basis of the file management system.

3. Design and implementation of file management system

3.1 communication between the embedded Forth system and the SD card

The SD card provides two communication modes: SD mode and SPI mode. You can select any of the communication modes and select it in the first reset command after the power is turned on. This paper selects the SPI mode SD card. SPI (Serial Peripheral Interface) is a full-duplex, byte-based synchronous communication protocol. The SPI bus operates in master-slave mode. The device driver framework should include the master-slave structure, including all SPI device registration mechanisms and device management mechanisms, as well as the SPI device data transfer interface. The function pointer callback is used to perform specific hardware-related processing. SPI mode SD

card communication principle, in SPI mode, SD card is connected to SPI controller as SPI device, SD card has four signal lines: cs (host to card chip selection signal), CLK (host to card clock signal), MISO, MOSI. As the host of the SPI system, the SPI controller connects the CS chip selection signals of the SD card through GPIO and selects the SD card chip. The clock of the SD card is provided by the SPI controller. Data exchange between the host and the SD card through the MISO and MOSI data lines. When writing an SD card driver, you need to define the hardware interface and functionality of the device. During the initialization of the program, the initialization function of the hardware device needs to be initialized. Its definition code is as follows:

```
PORTB 4 portpin: /ss \ optional port interface
PORTB 5 portpin: _mosi \ input port interface
PORTB 6 portpin: the interface of the _miso \ output port
PORTB 7 portpin: interface to the _clk \ clock signal port
```

In SPI mode, the SD card works by using commands, and for each command, the SD card has a corresponding return value. When SD card is pre-set, it works in SD mode. To enter SPI mode, a CMD 0 needs to be sent, and the SD card will enter the idle state after responding correctly to the CMD 0 command. When the host computer transmits the bus, when the CS chip select signal is selected, the low level is enabled. Following the forth-2012 standard, the definition is as follows:

```
: - MMC
Spi. ss pin_output \ select CS signal line and set it as firing exit
Spi.ss high \ set it to low level
;
```

SD card adopts command driven mode, and communicates with SPI host by sending commands and waiting for reply. Before starting data transmission, SD card should first send data transmission commands and wait for the correct reply before starting transmission. Each SD card command is made up of six bytes. In SPI mode, the SD card does not use CRC validation to protect data, but the CRC bits of command, reply, and data still need to be retained and ignored during transmission. However, the command CMD 0 that operates from SD mode to SPI mode requires CRC validation. In the underlying read-write operation, the SD card interacts with the SPI host as an SPI device, and realizes the data transmission from host to card and from card to host through the SPI bus.

3.1.1 SD card initialization

The SD card driver in this article supports the SD specification standard 1.0. In Specification 1.0, the maximum block length of the SD card is 512 bytes. In the initialization process of the SD card, it is necessary to identify the specification version of the card and the type of the card. After the SD card is powered on, it is in SD mode by default. Therefore, before sending CMD 0 to reset, it must be ensured that the CS signal is active low. This is guaranteed by the hardware circuit design. The 1.x version of the SD card only needs to check the voltage because it does not support large capacity cards. Considering that in the initial stage of power-on of the SD card, the voltage rise process takes about 64 clock cycles to reach the normal operating voltage of the SD card. In addition, it takes about 10 clock cycles to synchronize with the SD card. Therefore, after the SD card is powered on, it needs to delay at least 74 clock cycles. Since the design of this paper uses the hardware-controlled SPI mode, its transmit clock is automatically sent by its SPI module. Waiting for the corresponding time, the hardware SPI will automatically send at least 74 pulses before starting to send CMD 0 for reset operation and SPI mode selection. Following the Forth-2012 standard, the algorithm is described as follows:

```
: waitsd (---)
10 0
The do
C @ $FF = spi
If unloop exit then
Loop. "error";
```

```
: mmc_init
0 # mmc_ buf!
Mmc_ + spi
Mmc_reset
Dup 1 -
If-mmc 100 xor exit then
Drop
Mmc_sd?
Dup 0 <
If -mmc 200 xor exit then today
Mmc_waitsd pet-name ruby
200 mmc_length attending
Then or - MMC;
```

The mmc_+spi word defines the host as the initialization of the SPI module, enters the SPI initialization, defines its spi enable bit, the master/slave device, the rate is 128, the mode is 0, and the working mode is set for its SPI communication; mmc_reset is Entering the idle state, the word does not have a corresponding return flag. It sends at least 74 pulses before entering the command CMD 0, and sends 74 pulses to enter the IDLE state. It waits for the value of R1 to be returned. If the value of R1 returned is 1, the entry is entered. IDLE idle state; dup 1 - Check if the SD card enters the idle IDLE state. If it does not directly exit the entire program; mmc_sd? After entering the idle state, send command 47, command 51, detect the power-on state, and 200 mmc_length is the size of the sector. Defined, set to 512 words capacity, -mmc cancels chip selection, ends initialization.

3.1.2 read and write

The main task of the SD card is to achieve data read and write operations. During the read and write operation, pay attention to the attributes of the card, the size of the block, the length and number of bytes that can be stored, the number of blocks that can be erased at one time, and the timeout period of the card operation. The initialization module of the SD card has specified that the length of the read/write data of the SD card is 512 bytes, so it must be 512 bytes when reading and writing the module. Since the SD card communicates with the SPI master in the form of commands and responses, data transmission must be performed before each data transmission, before sending the corresponding read and write commands and waiting for a correct response. The SD card data transmission is carried out in the data token mode, and all bytes are transmitted in the high-order (MSB) transmission mode. Following the Forth-2012 standard, the algorithm for reading data on the SD card is described below.

```
: mmc_read (n xl xh -- c|-1) \ length, sector addr
    +mmc
    Mmc_blk2addr \ addr*512, block->byte
    11 mmc_cmd \ addrL addrH 11 --, CMD17
    Mmc_(read) \ n c -- c, 0=ok, -1=timeout
    -mmc;
```

Following the Forth-2012 standard, the algorithm for writing data on an SD card is described as follows:

```
: mmc_write (n xl xh -- c|-1) \ length, sector addr
    +mmc
 Mmc_blk2addr \ addr*512, block->byte
    18 mmc_cmd \ addrL addrH 18 --, CMD24
    Mmc_(write) \ n c -- c, 0=ok, -1=timeout
    -mmc;
```

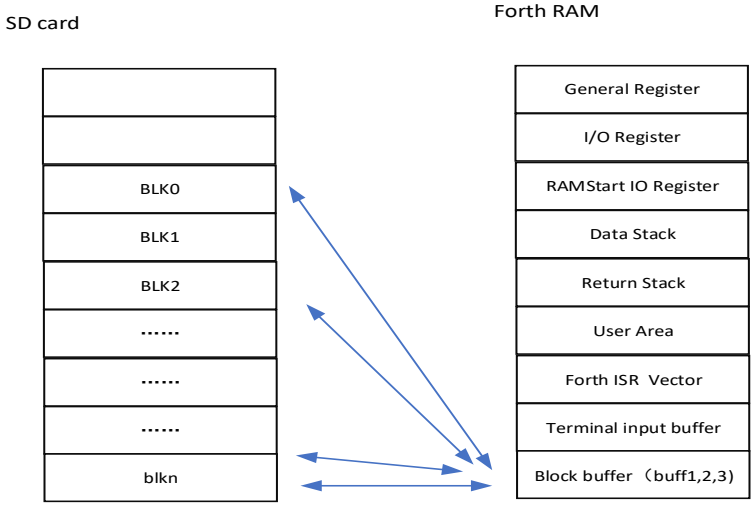
Through the management of SPI master device, a piece of data is read out from SD card into the buffer. The read and write operation of SD card is the basis of building file management system, and

the two are inseparable. In the read-write module that introduces SD card, a variable is used to replace the buffer address. After buffer management is added, the buffer address is allocated by the buffer management word.

3.2 buffer pool management

Buffer is an important part of the system. Due to the limited capacity, it is very important for the efficiency of the buffer replacement algorithm. Most systems only consider the buffer hit ratio and forget about the cost of write operations. The high address end of Forth's main memory is set up with three buffers for data exchange between the SD card and main memory. Here, an array of buffer Pointers is set to store information about each buffer and its data, such as the address of the buffer, the data in the buffer is from the SD card, the update flag of whether the buffer has been updated, and the address pointer of the next buffer. The buffer pointer array is next to the unit where the buffer data is stored.

The buffer holds data that is read from and ready to be written back to the SD card. The management of the buffer is actually the management of the buffer pointer array to achieve. The SD card buffer management, in fact, the SD card buffer pointer array management to achieve. The strategy for managing the SD card buffer pointer in this article is to use the pointer prev to point to the current buffer being used so that the pointer use points to the next available buffer. When the required block, the system will search the buffer pointer prev, if the required disk block has been in a certain disk buffer, then the buffer pointer will be moved to the first item of the buffer pointer array; If you want the disk does not exist, in the buffer system by the end of the buffer pointer array referred to in a pointer to the buffer allocated to a new SD card block, block in the SD card read the data in the buffer when testing the current use of the data in the buffer are updated, if the contents of the buffer is modified, the pointer points out that need to be updated, the content of this buffer will be the first to be written back to disk file, then the new SD card block read into the buffer. In this way, the blocks on the SD card can always be updated at the appropriate time to reflect changes or additions made by the user to the blocks, while the system minimizes the number of reads and writes to the SD card. The mapping relationship between SD card and Forth system memory is shown in figure 3.



Figer3. The mapping relationship between SD card and Forth system memory

In this article, three 524-byte buffers are set, where the buffer size for data is 512 bytes. Blk means block number and 4 units are used for storage, because the capacity of SD card is large and two units cannot represent the block number it represents, while lun means logical unit number. Since this paper is based on no file system, the value of logical unit number is constant as 0.Flag is used to indicate the buffer is updated, the original is 0, the updated is 8000, next is the link to the next buffer pointer, takes up two storage unit, buffer using the next pointer links form a buffer pool, data storage buffer

size is 512 bytes, at the end of each buffer empty two storage unit to prevent buffer overflow, its design process of buffer as shown.

3.2.1 Buffer pool

Manage to buffer is its basic framework. The buffer uses the next pointer link to form a buffer pool. The currently available buffer is prev and the next available buffer is use, to determine the buffer structure. The framework for buffer initialization is shown in the figure 4.

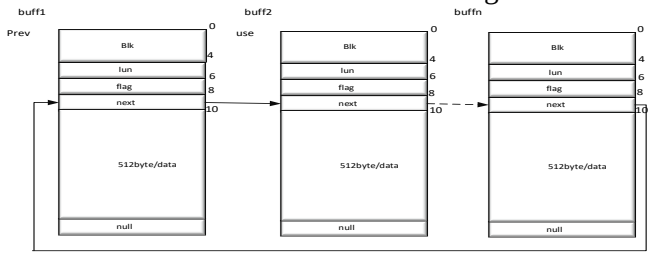


Figure 4. Initialization result

In the whole program design, the hexadecimal counting method is adopted. Firstly, the variable word and allot word of Forth language are used to allocate the corresponding address unit for the buffer in memory.

3.2.2 Buffer pool allocated memory address

Ring buffers are used in this article, and the Forth system has three SD card buffers that form a ring. If buffer 1 of SD card is used first, then buffer 2, and finally buffer 3, and the system needs to accept another piece of data, the system chooses buffer 1 to store the new data. This is because buffer 1 is the least active buffer at that time, which is called the most recently unused algorithm. The SD card buffer is managed using the most recent and longest unused algorithm. The most recently unused algorithm is reflected in the word (block). When the buffer space is insufficient, the most recently unused buffer is replaced first.The most recently used buffer is a modified buffer.

```
: update (---)
Prev @ @ 2 dup
> d 0.
If prev @6 + dup @8000 or swap!Then \ flag = 8000
0. D =
If prev @6 + dup @8000 or swap!Then \ flag = 8000
;
```

Buffer pointer to an array of update unit (flag) has a very crucial role in the buffer management, when the flag is set to true, write the data in the buffer back to SD card this incident will not happen, however, when the buffer is allocated to update the unit is true when a new SD card block, new blocks on the SD card before the data read from the SD card in the contents of the buffer will be written back to SD card in the piece where it was before. On the other hand, if the content of the update unit is 0, the system assumes that the contents of the buffer are consistent with the contents of the block in the SD card, and therefore does not write back to the SD card.

The next buffer is set to the currently used buffer defined in the word (buffer); Empty the data of the current buffer emptybuff; Write the updated buffer back to the SD card, using the word flush; The block in the SD card is allocated a buffer, and the buffer address is returned. The data in the SD card is not read into the buffer, but is simply allocated an available buffer, using the word buffer.

```
: buffer (series - buffer *)
Lun @use @dup 6 + @0 <
If dup savebuf then
>, r, r.\ lun
R @ 2!\ series
```

```
R > dup (buffer)
;
```

When the block number of the SD card data block is given, the word buffer checks whether the buffer to be allocated to the block number in the buffer pool is updated. If so, the data in the buffer is written back to the original block in the SD card.

3.2.3 Read/write SD card data to buffer pool

When a block in an SD card is needed, assign an available buffer to the block and read it, reading the contents of the specified block into the allocated buffer. When a block of data from an SD card is needed, check if the block is already in the buffer, and if so, make the buffer pointer prev point to the address of the buffer. If not, find the currently available buffer and check whether the block in the buffer has been updated. If the data in the buffer has been updated, first write the data in the buffer back to the corresponding block in the SD card, and then read the data in the block in the required SD card into the current buffer. If the data in the buffer has not been updated, read the block data in the required SD card directly into the current buffer. The Buffer simply allocates a Buffer to an SD card block, and the block reads the contents of the specified SD card into the allocated SD card Buffer.

If the block number is less than or equal to zero, it is an invalid block number; if the block number is greater than zero, it is a valid block number; if the block number is greater than zero, it is a valid block number; if the block number is greater than zero, it is assigned to the LBLK unit where the block number is stored; if the logic number is -1, it is detected; if it is -1, an error message is displayed. Block is the most important definition for communication with SD card. If any piece of data on the SD card is needed, just provide the Block number on the SD card, and the Block will read the Block data into an SD card buffer and return the buffer address. The user can access the data on the SD card block with the normal command to access main memory. If the command update is executed after changing the data on the buffer, the block will definitely write the modified data back to the SD card.

The difference between Buffer and block is that Buffer only allocates the Buffer and does not read the contents of the specified SD card into the Buffer. When you write new data to a blank SD card block, you should use the command Buffer.

3.3 file management system

Driven by the embedded real-time multi-task application scenario, the file management system appeared. In order to realize the online loading, online editing and online updating of the existing vulnerabilities in the embedded environment, the application could complete the repair of vulnerabilities without downtime. file management system USES SD card as the carrier of its source code. In order to perform good online loading, online editing and online modification, the stability of file management system has great requirements.

3.3.1 screen editor

Since there is no screen editor in the embedded Forth system, it is necessary to use the screen editor when using the file management system to provide a visual operation interface for online editing and online updating. A screen editor should be built in the embedded Forth system to facilitate subsequent use.

To realize screen editor, the key value of the virtual terminal should be mapped first, so that the editor can be used in the virtual terminal. The cursor is set to set the action for each key value. Since the size of a buffer storage data unit in this article is set to 512 bytes, the size of the screen editor is 8*64=512 bytes, corresponding to the buffer data unit. Following Forth-2012 standard, the definition is as follows:

```
Constant # 8 lines
64 # constant cols
```

Build a good screen editing program, use list to display the contents of blocks when using it, use

edit to call screen editing program, and use the corresponding keys on the keyboard to operate the blocks to be changed. Define corresponding key values for keyboard key definitions, specify corresponding actions for each function key to be used, and complete online editing, online updating and online modification of file blocks.

3.3.2 file management system

In the screen editor of the need to modify, update the program after the completion of the modification to start the file management system to complete the program online operation. Forth system source code consists of advanced colon and low-level assembly the definitions of the core word, have a special one word in a Forth system - delay, at compile time, not its function, its function after delayed to word, use the delay of the Forth system in file management system word function of dynamic load loading system of the core word definition.

A file management system is established using the delay word definition of the Forth system. When the data to be loaded is in the SD card, start up the Forth system terminal, use the load word to load, build a file management system for the corresponding embedded Forth system with SD card as the hardware, and submit the blocks in the SD card to the text interpreter. The text interpreter accepts the command line from the terminal, separates the commands from it, and in the process searches for the commands currently encountered. If it is a word already in the dictionary, it performs the actions specified in the dictionary to complete the specific functions. After executing the command line, the text interpreter returns to the terminal to wait for the next line of command to be entered. A text interpreter usually means a definition of INTERPRET, which is a key definition in the Forth loop and plays a key role in the text interpreter. The idea is to isolate the words in the input stream and, if it is defined in the dictionary, execute it.

The core word of the file management system is load, which relates the dynamic communication between the SD card and Forth memory. Before the implementation of file management system, the construction of screen editing files should be implemented first. As an auxiliary tool, it is convenient to use the visualization of file management system again.

When the load word is run, the block number you want to load is supplied to the load word. ['] is the word used to define delay words in the forth system and can only be used in colon definitions. At runtime, is sends the compile address of the followed word source into the pfa unit of the deferred word source-block, and execute. The program exception handling is introduced into the file management system to make the program have good fault tolerance. When the running error occurs, the load word can catch the exception in time and even handle it, so that the load word can recover from the running error and continue to execute.

4. Experimental verification and analysis

In the experimental testing process, the embedded system AmForth system based on open source was adopted. AmForth is a 16-bit Forth system that runs on the ATmega328p chip. ATmega328p is a high-performance, low-power AVR 8-bit microcontroller that supports SPI mode, has a clock frequency of 16MHz, 32KB FLASH, 2KB RAM and 1KB EEPROM storage space.ATmega328p can provide pin change and transform corresponding potential, complete the sending of corresponding read and write commands, and provide a good platform for file management system

On this hardware platform, the above algorithm is implemented. During this experiment, the buffer is first initialized using the buff built:init to assign addresses in main memory. Second, the SD card is initialized so that it is in SPI slave mode. Set up the word "test" to test its reading and writing accuracy:

```
: test1 Buff1 0 1234. Bread drop
Buff2 0 1235. Bread drop
Buff3 0 1236. Bread drop;
```

Now modify the block number not 1234. Using the specification 1234.edit.Exit after editing, and use 1234.load to load the edited block content to realize online update.

As for the embedded file management system based on Forth virtual machine, it works normally and quickly when switching to different types of SD khaki, which indicates that the stability of the embedded file management system based on Forth virtual machine is excellent.

5. Conclusion

In this paper, the embedded Forth virtual system can't on-line edit, online update and other issues, to achieve its online update, online modification, online loading under Forth embedded system. For embedded applications, it is beneficial to the embedded system, which can bring good development to embedded development. The embedded operating system based on Forth virtual machine is to operate on the stack. Only one data structure makes the structure of the system more Concise and convenient for many application scenarios and the file management system proposed in this paper can bring convenience to embedded applications. When a program needs to be modified, it does not need to be executed in an outage. As long as a virtual terminal is linked, the corresponding bug can be modified. The design of SD card makes it more flexible. The corresponding bug, using the design of the SD card makes it more flexible. The file management system realizes online update, online modification and online loading of the embedded Forth system on the basis of such hardware, making it more flexible in embedded applications.

References

[1] Wikipedia. Charles H. Moore. (2019). http://enwikipedia. org/wild/Charles H._Moore.
[2] Dai H.B, Yang W.M, Wang L.Q, Zhou Y.L. (2014). Research and Implementation of Multi-Objective Forth Self-Generator. Computer Application Research, 2014,31(4).
[3] Dai H.B. (1993). the development of a new type of high-efficiency microcomputer FORTH language. Journal of the Graduate School of the Chinese Academy of Sciences, 1993,10(1).
[4] Liu D.L, Li X.H, Zhang H. (1988). The Fourth Generation Computer High-Level Language--FORTH. People's Posts and Telecommunications Press, 1988
[5] forth lnc. featured forth applications. (2009).http: //www. forth. com/resources /app Notes.
[6] JAMLES R. Space-related applications of forth. (2009). http: //forth. gsfc. nasa.gov/.
[7] Forth200x Committee. Forth 2012 standard. (2019). http://forth-standard.org/
[08] Dai H.B, Zhou Y.L, An H.P, Huang Z.J. (2018). Design and implementation of multi-task scheduling algorithm for embedded Forth virtual machine architecture. Application Research of Computers,2019,36(02):472-475+485.
[9] ANSI. ANSI X3. 215-1994, American national standards for information systems programming languages Forth. New York: Amarican National Standards lnstitute,1994
[10] Lu M.G, Zhuang J.F. (2013). Trial group 3D printer.2013 -06. http: //www.figtaiwan.org/figtaiwan.org/2013FigTopics.htm.
[11] Xu X.Z. (2013). Cortex M4 swiftx forth Interrupt. http: //www. figtaiwan.org/figtaiwan.org/figtaiwan.org/2013FigTopics.htm.
[12] Dai H.B, Zhou Y.L, An H.P, Mei H. (2019). Research on Embedded Multitasking Operating System Architecture Based on Forth Virtual Machine. Application Research of Computers, 2019, 36(02):476-480.
[13] forth lnc. featured forth applications. (2019). https://www.forth.com/resources/forth-programming-language/#2436_Multiprogramming
[14]SD Group. (2005). In: SD Specifications-Part 1: s Physical Layer Specification Ver-sion 2.00 Draft.
[15] Huang Z.J, Dai H.B, Wang L. (2019), Research on real-time scheduling algorithm of embedded Forth operating system. Application Research of Computers ,36(09):2700-2703+2721.