Pakistan Journal of Information and Technology 2(1): 52-57, 2003

A File System Implementation for Microcontroller using Ferroelectric Random Access Memory

[1]W. L. Tse and [2]W. L. Chan
[1]Department of Building and Construction, City University of Hong Kong
Tat Chee Avenue, Kowloon Tong, Hong Kong
[2]Department of Electrical Engineering, Hong Kong Polytechnic University, Hung Hom
Kowloon, Hong Kong

Abstract: In this paper, a low-cost and effective nonvolatile memory system is developed and applied in microcontroller applications. A Ferroelectric Random Access Memory (FRAM) is employed as the core component in the memory system due to its faster operation times, larger number of write operations, lower power consumption and cheaper selling price. All these advantages are due to the physical structure of ferroelectric materials. In order to utilize the FRAM flexibly and effectively, a practical file system is created. This file system is able to support real-time database implementation for complicated situations. With the newly developed nonvolatile memory system, the development time of microcontroller applications can be greatly shortened so as to keep up with the fast pace of commercial market. A prototype of 8051-based electronic board has been built to demonstrate the performance of the new memory system.

Key Words: Microcontroller, Ferroelectric Random Access Memory, File System, Real-Time Database

Introduction

Nowadays, data collection is very common in most microcontroller applications. The major reason is that these applications are customer-oriented. More and more user-friendly features are added to satisfy customers' needs and historic data is necessary for these implementation. For example, a user hitting a hot key on his mobile phone can recall the previously-dialed phone number which is obviously stored in the mobile phone. Data collection consists of acquisition and storage of data, which must be retained in the absence of power. This type of data is stored in off-chip memory instead of cache and scratchpad memory (Panda, 1999). As a result, a low-cost and effective nonvolatile memory system is essential in these applications, which consists of two important things, namely nonvolatile memory and a file system.

In this paper, Ferroelectric Random Access Memory (FRAM) technology is employed in this memory system. It is a new technology to enable memory to retain data without power. An electronic circuit interface with this kind of memory is very compact and consumes small amount of power. In addition, a very practical file system is developed on the FRAM to utilize it in flexible and effective manner. Several files for program settings and user preferences can be stored using this file system. User-friendly features can be added easily by simply modifying the contents of these files. With the file system, real-time database implementation is made possible to cater for complicated situations. In fact, some researchers recognize that microcontroller applications requires database implementation in the very near future (Williams, 2000). In this way, the product can catch up with the fast pace of commercial market (Lewis, 2002). A prototype of the 8051-based electronic board is built to demonstrate the performance of the new file system.

Materials and Methods

The Ferroelectric Random Access Memory Technology: Memory technologies can be divided into two categories, volatile and nonvolatile. Volatile memories lose their contents when power is removed, nonvolatile memories do not. Traditional volatile technologies include Static RAM (SRAM) and dynamic RAM. The common factor is that they are RAM-based technology, Random Access Memory. The main advantage of RAMs is that they are easy to use, read and write operations are similar. The main disadvantage of traditional RAMs is that they can only be used for temporary storage. Traditional nonvolatile technologies are based on ROM-based technology, Read Only Memory. Various improvements to the underlying technology have created Flash and Electrically-Erasable-and-Programmable ROM (EEPROM), these variations of ROM can be written.

Ramtron's FRAM is a new generation of nonvolatile memory that combines high-performance and low-power operation with the ability to retain data without power. FRAM has the fast read/write speed and low power of battery-backed SRAM and eliminates the need for a battery. EEPROM and Flash require long write times, wear out after being written a small number of times and use a large amount of power to write data. FRAM writes instantly, has virtually unlimited endurance and requires very little write power. FRAM performance is superior to EEPROM or Flash in three primary areas. Firstly, FRAM is much faster. The advantage is dramatic for write operations. It is faster for reads as well but the difference is not so dramatic. Secondly, FRAM offers virtually an unlimited number of writes compared with one million for EEPROM. FRAM uses much less power than other nonvolatile memories. When comparing with battery-backed SRAM, FRAM is much cheaper. According to Barr (Barr, 2001), SRAM with battery back-up is very expensive and its application is limited to the storage of a few hundred bytes of system-critical information. This does not happen to FRAM. As you see later, we employ a FRAM of size 32 kbytes in our tested circuit.

The advantages of using FRAM are because of its physical structure. At the core of Ramtron's FRAM memory technology are tiny ferroelectric crystals integrated into the memory cell that allows FRAM products to operate like fast nonvolatile RAMs. When an electric field is applied to a ferroelectric crystal, the central atom moves in the direction of the field.

Internal circuits sense the charge required to move the atom. If the electric field is removed from the crystal, the atom stays in position, preserving the state of the memory. FRAM memory needs no periodic refresh. When power fails, FRAM memory instantly retains its data. In this case, Ramtron FRAM FM1808 is used and the file system is developed in this memory chipset. To control the file system, a 8051 MCU is employed, which is very commonly adopted in industry.

Architecture of the Proposed File System in FRAM: Under the proposed file system, the file contents are stored in several blocks. A block is the unit of data that actually gets transferred (i.e. read or write) to or from a storage device (Grosshans, 1986). In this way, each block contains parts of the file contents. A link-listed technique is adopted to link all these associated blocks in an ordered list such that a file can be easily manipulated. In the file system, there are a total of five regions defined in the memory. They are the file system signature, file system configuration, File Allocation Table (FAT), root directory (ROOT) and file data content (FILECONT). The five regions serve for different purposes and they are described in detail in the following sections.

File system signature is located in the first 16 bytes in the FRAM. This 16-bytes location is used to store a null-terminated string which would be assigned to a certain value when the file system is first initialized. In this case, it is always set to "FILESYS 1" which stands for the signature of the file system. This string can be used to check for the validity of the file system. MCU is only required to read its content to compare it with the signature. If they are not the same, the file system is regarded as invalid or being corrupted. Repair or re-initialization is required on the file system to fix defects. This description can be used to determine the version number of the file system. Thus, compatibility can exist in different versions.

File system configuration occupies the next 16 bytes. As earned from its name, it stores the configuration information of the FRAM including the total memory size, file system size, block size, the total number of blocks, the total number of file entries in ROOT, FAT address, ROOT address and FILECONT address. MCU should read this information before using the file system. As each entry in this region has two bytes in size, the file system can manage a memory up to the size of 64 kbytes. As mentioned before, a block is the smallest unit manipulated by the file system for each operation. It is usually set to a value greater than one so as to speed up data transfer. The total number of blocks is determined by the memory size. It is varied under different size of the FRAM and so does the location of the next three regions.

The size of File Allocation Table (FAT) region is equal to the total number of blocks times two. It stores the status of all blocks in FILECONT. Two bytes of information are required to represent the status of each blocks. The position of this two-byte entry represents the location of the block in FILECONT, that is, the first entry stands for the first block in FILECONT region. In each entry, the first byte ($F_7F_6F_5F_4F_3F_2F_1F_0$) is for control purposes with its most significant bit F_7 indicating the availability of the current block and other bits are reserved for future use. If F_7 is set to zero, the block is available for allocation. Else, the block has been occupied. The second byte represents the next block followed by the current one. IF it is set to 0xFF, the current block is the last one for that file. Else, there is another block required for processing. In this way, the whole file is stored in a list of blocks and all these blocks can be searched by linked-list technique.

Root Directory (ROOT) consists of several entries. Each entry has sixteen bytes in length and stores the information of a file such as file size and file name. Firstly, there is a control byte ($R_7R_6R_5R_4R_3R_2R_1R_0$) in each entry to indicate its availability. If R_7 is set to zero, the entry is available for allocation. Else, the entry has been occupied and it is already belonging to a certain file. The next byte represents the location of the first block storing the file contents. It should not be equal to 0xFF. Then, the size of the file is stored in the next two bytes as an unsigned integer format. The maximum allowable file size is thus 65536. File name is the last item, which is in the form of null-terminated string. 12 bytes are allocated for a file name and are sufficient for a typical application. The total number of entries in this region can be configured, which in turn affects its size.

The last region, file data content (FILECONT), consists of several blocks that the actual file contents are stored. As mentioned before, each block has a fixed length. For a non-fully occupied block, the remaining memory space cannot be used. If the size of a block is too large, the memory will not be fully utilized leaving many unused locations in occupied blocks. If it is too small, the list of blocks for a typical file will be too long causing poor efficiency (Harbron, 1986). An optimal size of a block is required to be determined to make balance between the utilization rate and the efficiency.

Results and Discussion

The nonvolatile memory system is implemented in a 8051-based tested circuit which consists of 89C55 MCU, FM1808 FRAM, Static RAM (SRAM), RS232 serial interface, LCD display module and PS/2 keyboard. The FRAM is used as the nonvolatile memory in which the newly proposed file system is developed for file storage purposes. A file saved in the FRAM under the file system comes from a personal computer connected to the circuit via RS232 serial link. Fig. 1 shows the prototype of the tested circuit.

Hardware Design: The FM1808 is a 32 kbytes FRAM employing an advanced ferroelectric process (Ramtron, 2000). The interface of the FRAM with 8051 MCU is similar to that of a typical SRAM except the chip enable signal. First, bytewide FRAM memories latch each address on the falling edge of chip enable. This allows the address bus to change after starting the memory access. Since every access latches the memory address on the falling edge of chip enable, users should not ground it as they might with SRAM. A decoder circuit is required to qualify addresses with an address valid signal if they do not already. In this case, two NAND gates are used in the decoder circuit. Fig. 2 shows the circuit connection between the FRAM and a 8051 MCU.

Atmel 89C55 MCU is a 8051-based MCU with 256 bytes internal RAM and 20 kbytes Flash (Atmel, 2000). With RS232 serial interface, LCD display module and PS/2 keyboard, the MCU is programmed to accept user commands from a keyboard. Then, the circuit waits for serial data sent from the remote PC and saves it in the FRAM as a file under the newly proposed file system.

Settings of the New File System in the FRAM: The considerations for determination of the settings of the

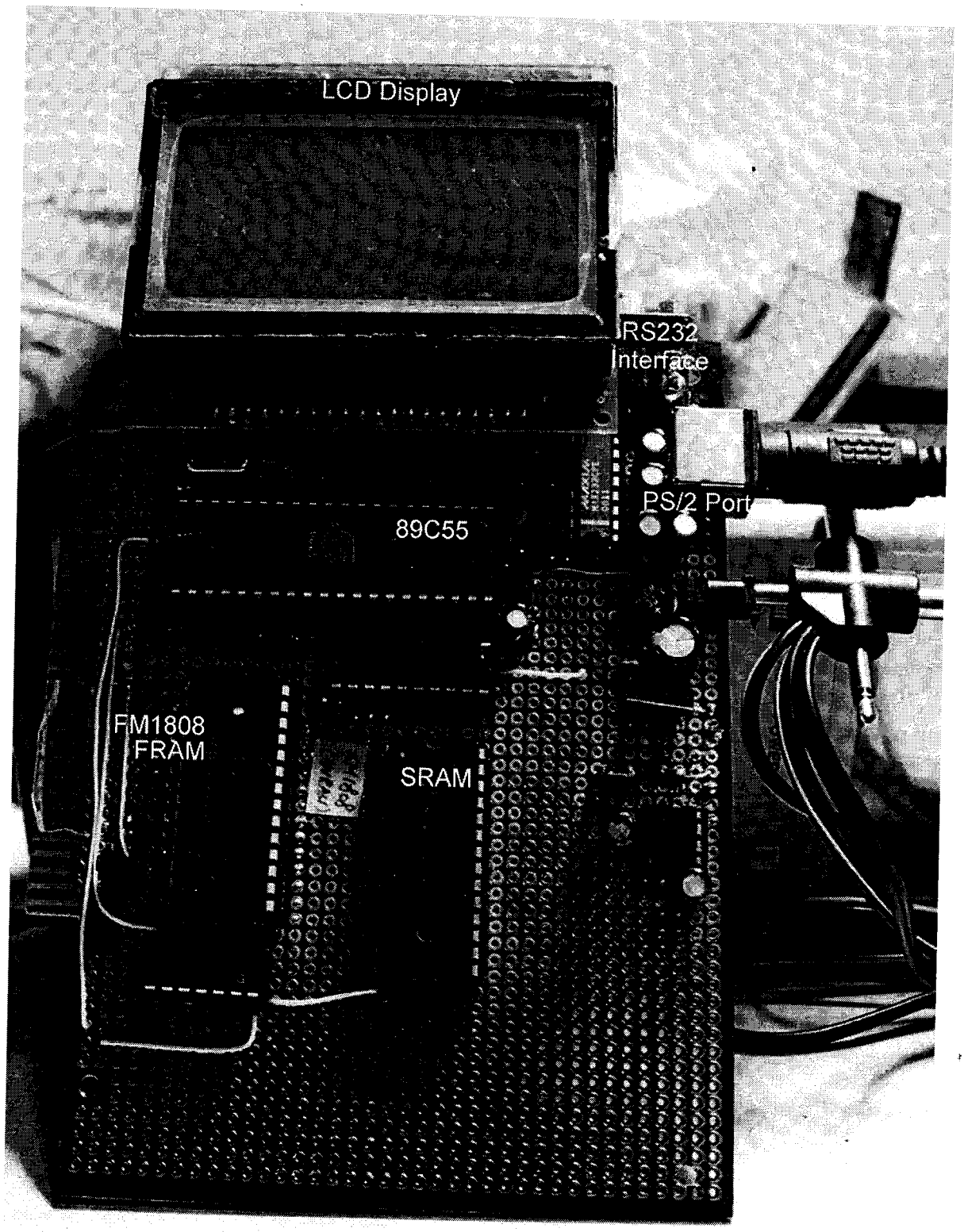

Fig. 1: The Prototype of the Tested Circuit

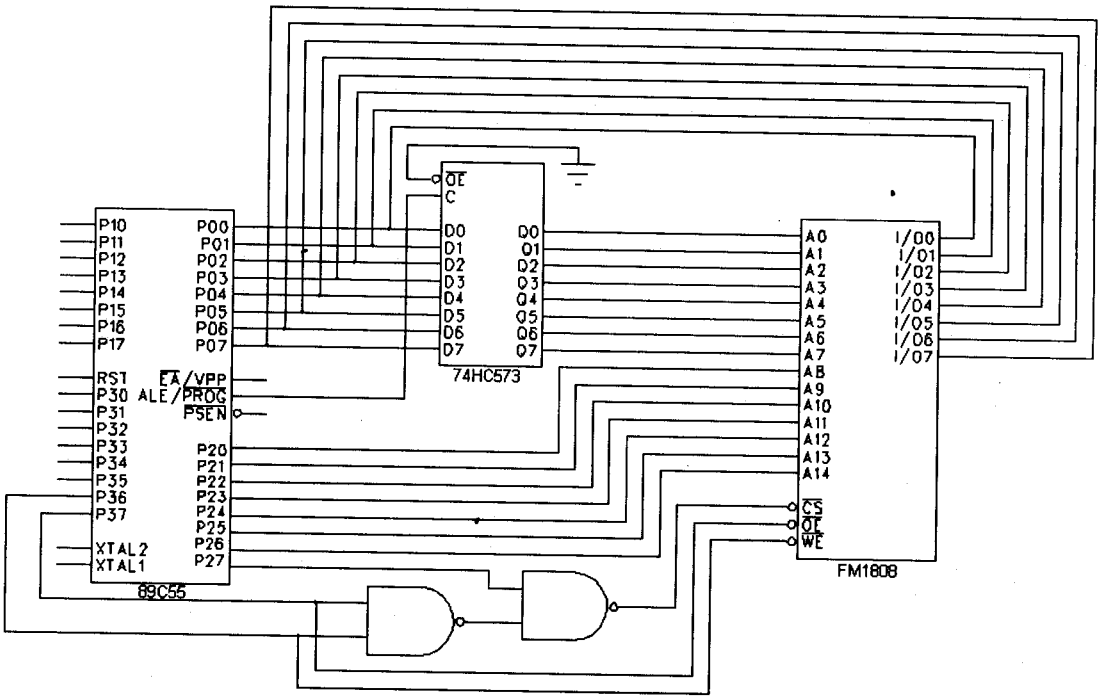

Fig. 2: The Circuit Connection between the FRAM FM1808 and the MCU 89C55

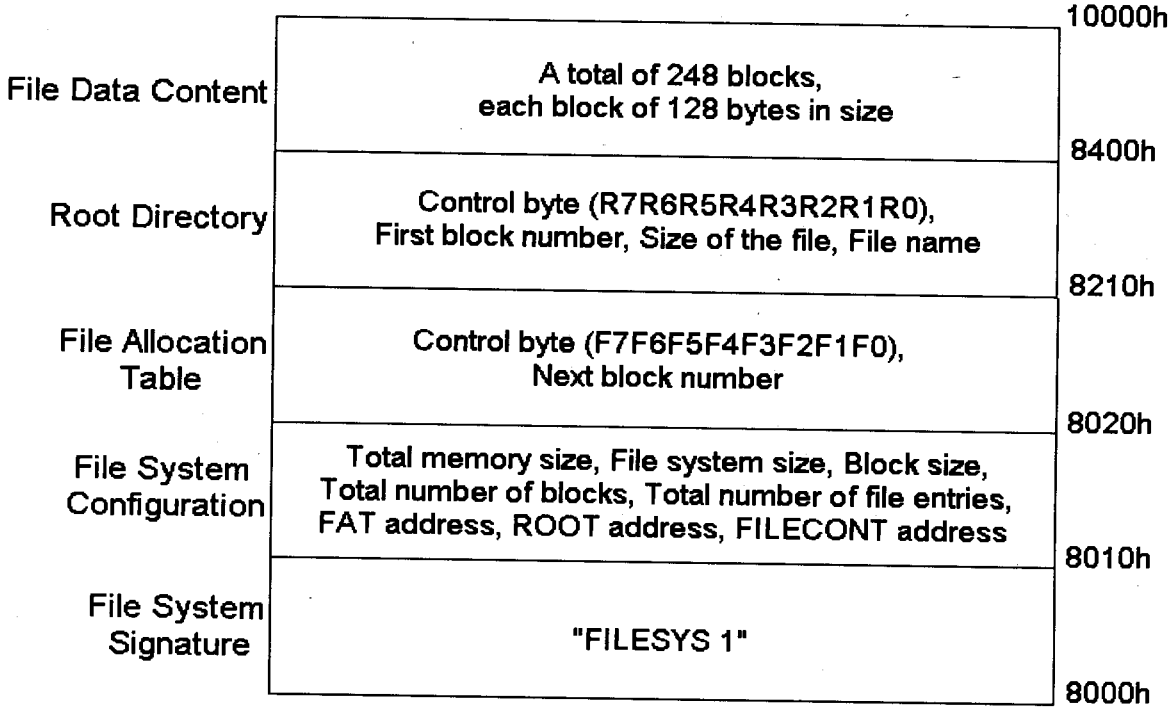

Fig. 3: The Structure and Memory Locations of the New File System

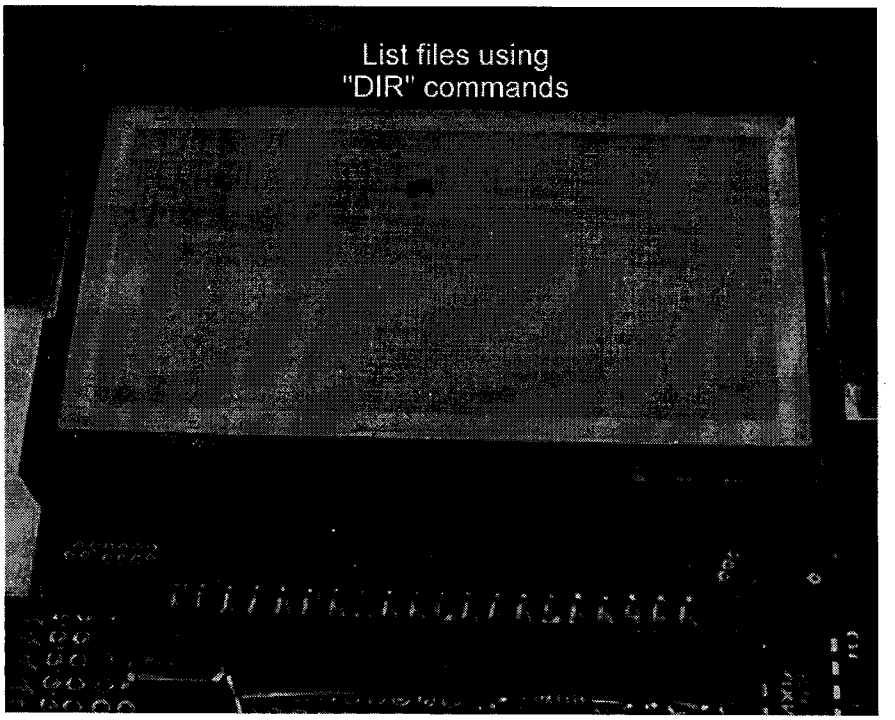

Fig. 4: The File "img1.gp1" is Listed out in the LCD Display of the Tested Circuit

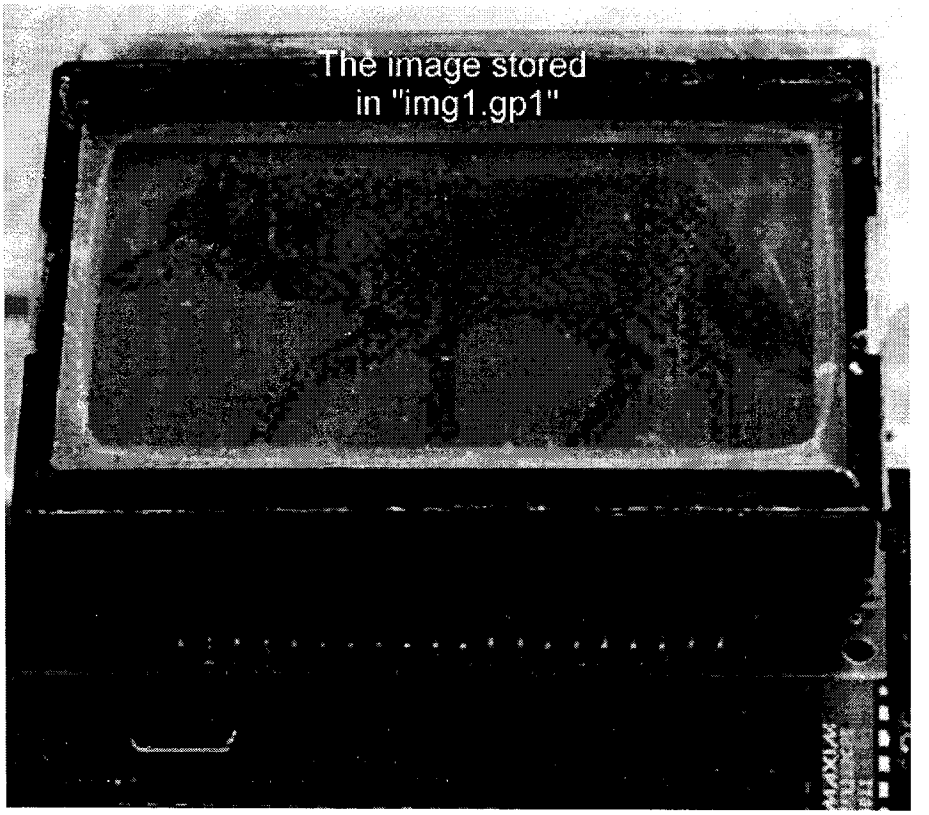

Fig. 5: The Image Stored in the File "img1.gp1" is Shown in the LCD Display

file system are the utilization rate of the FRAM and the speed of the system. Two main parameters are required to be determined first. They are the size of memory for storing file information and the block size. The first item include the memory space for the first four regions (i.e. file system signature, file system configuration, FAT and ROOT) of the new file system. The size of this item should be as small as possible so as to reserve more space for the actual file usage. In this case, we set it to 1 kbytes leaving 31 kbytes for actual use. Over 96 % of memory space is used for the actual file content, which is a rather high utilization rate. Next, the block size of the FM1808 FRAM is set to be 128 bytes. This value is a typical file size for storing program parameters and user preference settings in microcontroller applications. The amount of unoccupied memory space in each sector is minimized and thus the FRAM can be fully utilized.

In this way, there are a total of 248 blocks in FILECONT regions. The size of FAT is equal to 496 bytes (= 2 x 248). ROOT occupies another 496 bytes (1024 - 496 - 16 - 16). The total number of files that can be stored is 31 which is enough for most applications. As the FRAM is addressed at 8000h, the location of all five regions are offset by this value, which are shown in Fig. 3.

The Application Interface of the New File System: To facilitate the interface with the new file system, a set of procedure calls are developed using C language, which are listed as follows:

- *void format()* – assign the file system signature "FILESYS 1", assign the correct file system configuration for the FRAM and fill 0x00 in all locations in FAT, ROOT and FILECONT.
- *void info(char xdata *s)* – retrieve the signature and store it in a variable *s*.
- *unsigned int diskfree()* – return the total number of unoccupied bytes in the FRAM.
- *bit savefile(char xdata *fn, unsigned char xdata *p, unsigned int num)* – save the data pointed by *p* as a file with the file name *fn* and *num* is the expected number of bytes to be saved.
- *bit delfile(char xdata *fn)* – delete the file with its name *fn* in the file system.
- *bit readfile(char xdata *fn, unsigned char xdata *p, unsigned int *num)* – read the file *fn* and save its contents to the buffer pointed by *p* with *num* equal to the total number of bytes saved.

  For the last three procedures, they return one if there is no error. Otherwise, zero is returned.

Demonstration: The tested circuit is connected to a Personal Computer (PC) via RS232 serial communication. A user keys in the command "GET" and then inputs the requested file name. The file in the PC is then sent to the tested circuit and is saved using the same file name. Fig. 4 shows the file "img1.gp1" in the file system by using the command "DIR". This file is sent from the PC and is read in the tested circuit. It is a bitmap file. By using the another command "GPH", the image stored in this file can be displayed. Fig. 5 shows the image displayed on the LCD. The idea of the newly proposed file system is proved to be a success. Also, the tested circuit is a useful teaching aid to introduce the concept of file system and electronic circuit design to students in lecture.

Conclusion

Our aim to develop an effective nonvolatile memory system for microcontroller applications is achieved. A totally new file system has been successfully implemented in a Ferroelectric Random Access Memory. Unlike the battery-backed SRAM, no extra circuit for battery is required to retain data. The system is very competitive and will be widely adopted as it provide a compact and simple hardware design. With the newly developed nonvolatile memory system, the development time of microcontroller applications can be greatly shortened so as to keep up with the fast pace of commercial market.

In this paper, both hardware and software aspects of the system are fully explored. Readers are welcome to employ the technical contents of this paper for their future uses. The software copy of the new file system can be obtained by sending a request to our email addresses bcwltse@cityu.edu.hk or eewlchan@polyu.edu.hk.

References

Atmel Corporation, 2000. Data sheet – 8-bit Microcontroller with 20K bytes Quick Flash AT89C55WD, Atmel Corporation, USA.

Barr, M., 2001. Memory Types, Embedded Sys. Programming. 14:5: 103-104.

Grosshans, D., 1986. File Sys. Design and Implementation. Prentice Hall, NY, USA.

Harbron, T.R., 1986. File Sys. Structures and Algorithms. Prentice Hall, NY, USA.

Lewis, D.W., 2002. Fundamentals of Embedded Software: Where C and Assembly Meet. Prentice Hall, NY, USA.

Panda, P.R., N. Dutt and A. Nicolau, 2000. Memory Issues in Embedded Systems-On-Chip Optimizations and Exploration. Kluwer Academic Publishers, NY, USA.

Ramtron International Corporation, 2000. Data sheet – FM1808 256kb Bytewide FRAM Memory, Ramtron, USA.

Willimas, T., 2000. Embedded Devices Find Links to Large Databases, Embedded Sys. Dev. 3:2: 22-25.