Study of High-Capacity Data Processing Method in Testing System

Zhi-Gang Ma[1,2], Wen-Yi Liu[1,*] and Wen-Dong Zhang[1]

[1]*Key Laboratory of Instrumentation Science & Dynamic Measurement, North University of China, Ministry of Education, Science and Technology on Electronic Test & Measurement Laboratory, Taiyuan, Shanxi, 030051, P.R. China;* [2]*College of Information Science and Engineering, Shanxi Agricultural University, Taigu, Shanxi, 030801, P.R. China*

Abstract: In various kinds of testing systems, large-capacity data file is common. File processing method based on C, C++ and MFC was introduced in this paper. In the light of functions in Win32 API, reading, data decomposing and analysis method of large-capacity data file were presented. According to practical requirement, the data processing software is designed. By experimental verification and practical data testing, the processing method posed in this paper can analyze and process large-capacity data file effectively. The processing method presented in this paper has been employed in several space measurement projects successfully and obtained satisfactory analysis result, which provided the reliable basis for relational tests and experiments.

Keywords: Algorithm, application program interface (API), data processing, high-capacity data, testing system.

1. INTRODUCTION

For the past few years, with the rapid development of information technology, several test and measurement fields are needed for transmission, storing large amounts of data. For example, several kinds of vibration, shock, and high-speed image data often need to be collected and stored in testing system, and some usually requires a high sampling rate and a longer acquisition time. Therefore, the amount of data processed by a computer is often very large, usually up to a few dozen GB, even up to several tens of GB [1, 2]. In connection with so huge amount of data, it is necessary to adopt efficient programming language and the rational design of algorithms. Data processing algorithms discussed in this paper is based on the Visual C++ 6.0.

2. READ AND WRITE OF BINARY FILES

"File" means an information collection stored in a computer external media (such as hard discs, CD-ROM, etc.). Files are generally divided into text files and binary files. This paper discusses the data files belonging to a binary file. Typically, the data in the program will be cleared from memory after the end of the program runs. It is inevitably encountering the problem of permanently storing the data, thus, we can still use the data when the program is ending. So file operations is indispensable.

2.1. File Operations Based on C and C++

In C language, file operations are performed by the structure named as FILE. When which is used, "fopen" function returns a pointer to FILE structure firstly, and functions "fread" and "fwrite" are used for reading and writing the files. In C++, ofstream class (file stream) can be adopted for the file reading and writing operations, the relevant functions are "read","write", etc.

2.2. File Operations in MFC

CFile is used for file operations in MFC, which provides a number of functions, such as, "Write" and "Read" are adapted to writing and reading data from files respectively.

In addition, "GetLength" function can get the file size, and this function returns a value of type DWORD (32-bit unsigned integer). It means that "GetLength" function can get the correct number of bytes in size can not exceed 4GB files. If the file exceeds 4GB, it is necessary to consider other ways to get its size. Although the data processing does not need to know the size of the file, this parameter can be estimated the progress of data or other necessary information, so it is important.

3. FILE OPERATIONS WITH WIN32 API FUNCTIONS

Win32 API provides a number of functions related to the operation and function, including: "CreateFile", "ReadFile", "WriteFile" and "FindFirstFile", etc. This paper is discussed mainly based on Win32 API functions.

3.1. "CreateFile" Function

"CreateFile" function is used to open and create files, pipes, communication services, devices, and consoles. This paper is only concerned with how to use this function to create or open a file. Its syntax is:

```
HANDLE CreateFile
(
LPCTSTR lpFileName,
DWORD dwDesiredAccess,
DWORD dwShareMode,
LPSECURITY_ATTRIBUTES lpSecurityAttributes,
DWORD dwCreationDisposition,
DWORD dwFlagsAndAttributes,
HANDLE hTemplateFile
);
```

3.2. "ReadFile" Function

"ReadFile" function is used for reading data from the specified file. Its syntax is:

```
BOOL ReadFile
(
Handle HANDLE hFile,
LPVOID lpBuffer,
DWORD nNumberOfBytesToRead,
LPDWORD lpNumberOfBytesRead,
LPOVERLAPPED lpOverlapped
) ;
```

3.3. "WriteFile" Function

"WriteFile" function is used for writing data to the specified file, the syntax is:

```
BOOL WriteFile
(
HANDLE hFile,
LPCVOID lpBuffer,
DWORD nNumberOfBytesToWrite,
LPDWORD lpNumberOfBytesWritten,
LPOVERLAPPED lpOverlapped
);
```

3.4. "FindFirstFile" Function

To get detailed information files, "FindFirstFile" function can be adopted. Its syntax is:

```
HANDLE FindFirstFile
(
LPCTSTR lpFileName,
LPWIN32_FIND_DATA lpFindFileData
);
```

After this function is executed successfully, the variable "lpFindFileData" will get a WIN32_FIND_DATA structure, which contains several file information, this paper is more concerned about the parameters nFileSizeHigh and nFileSizeLow, which represent high 32-bit and low 32-bit of file size respectively. Therefore, it can obtain the file size does not exceed 16 MTB, which can meet the actual needs at this stage.

4. ANALYSIS OF LARGE-CAPACITY DATA

In testing system, test data obtained from each channel needs to be transmitted to the analyzing computer in data file format, and analyzed. Data analysis includes: channel data separation, channel data integrity analysis, etc. When necessary, the continuous changing analog signal (voltage, temperature, level, pressure and other parameters) should be shown in curve or exported quantized value. In addition, the analysis course should generate a log file, which is used to give the results of data analysis, including testing duration, the effective number of frames, whether dropped frames and other information. General data analysis process is shown in Fig. (**1**).

4.1. Data Separation

Since various kind of test data have different sampling rate, sampling time, etc., the raw data obtained is often a mixture of various types of data. Therefore, before analyzing, the raw data must be separated and generated separate files. Each file is corresponding to a kind (or channel) of data.

Fig. (**2**) is a case of a package structure with a mixed series of data frames. The sample contains two types of data, namely digital data (packet tail is 0X "AA + AA") and analog data (packet tail is 0X "E7 + E7"). Packet count (4 bytes) is designed for classifying packets in order to judge the integrity of the package, and two types of data packets are counted separately.

"Data separation", also called "Data unpack", which is used for separating various types of data and generating new data file respectively. In the separation process, it is necessary judge the continuity of packet count. If not continuous, then an error message should be generated and written to the log file.

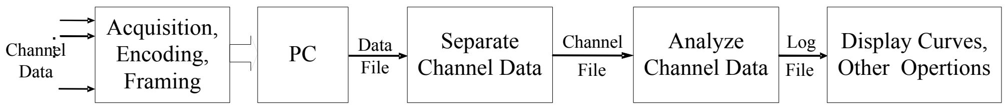


Fig. (1). Universal process of data analysis.

Packet Head (2Bytes) 0X"55+55"	Valid Data (92Bytes)	Packet Count (4Bytes)	Packet Tail(2Bytes) 0X"AA+AA" or " E7+E7"
100 Bytes			

Fig. (2). Package structure of mixture framed data.

Before data separation, the data must to be read from the data file. If the data file is small (for example, no more than 100MB), which can be directly read into memory completely. While, for a large amount of data (for example, more than 1GB), common analysis method is "block reading" → "join data " → "block analysis", and finally achieve "full analysis". That is, firstly read out (*e.g.* 128KB) block of data for analysis; Put the remaining unused portion of the block into a temporary buffer; Read the next block, then join the previous remaining data and current block of data and perform analysis the whole data, thus so on, until the analysis is complete. The process is shown in Fig. (**3**), "analyze data" refers to the judgment of all data packet structure in the current block to determine if the packet data belongs to the "digital data" or "analog data", and then the effective data is written into the corresponding file.

4.2. Channel Data Analysis

Although the frame structure is different of each type of channel data, he frame structure is designed to have a pattern, so the flexibility and universality of analysis algorithm should be considered adequately. A typical data frame structure shown in Fig. (**4**).

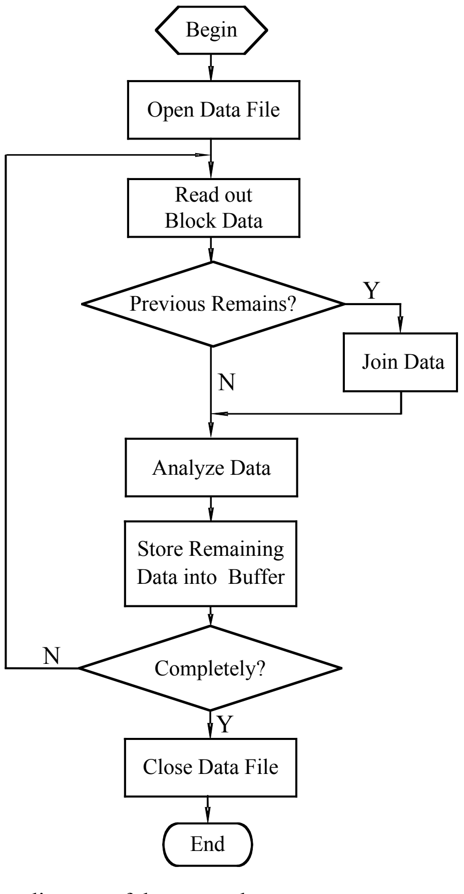


Fig. (3). Flow diagram of data unpack.

```
TmpDat = new UCHAR[BlkLen];
fp.Read(TmpDat, BlkLen);
FCnt1 = 0; FCnt2 = 0; //previous and current frame count
for(i=0; i< BlkLen; i++)
{
  if((TmpDat[i]==FHead[0])&&(TmpDat[i+1]== FHead[1]))
  {
    if((TmpDat[FLen-2]==FTail[0])&&(TmpDat[FLen-1]==
FTail[1]))
    {
      //Find a valid data frame;
      //Judge the continuity between the adjacent frame (2-byte
count for example)
      FCnt1 = FCnt2;
      FCnt2 = (TmpDat[FLen-4]<<8) + TmpDat[FLen-3];
      if ((FCnt1+1)!= FCnt2)
        {
          //Write an error message to the log file
        }
    }
  }
}
```

Fig. (5). Typical code of data analysis.

As before mentioned, different channels often contain different data frame structures. In order to adapt to different data analysis, relational analysis parameters (frame head, data length, frame count number and frame tail) should be set. Data analysis procedures should be adapted to analyze different data frame structure, typical code as follows in Fig. (**5**).

4.3. Design of Data Analysis Software

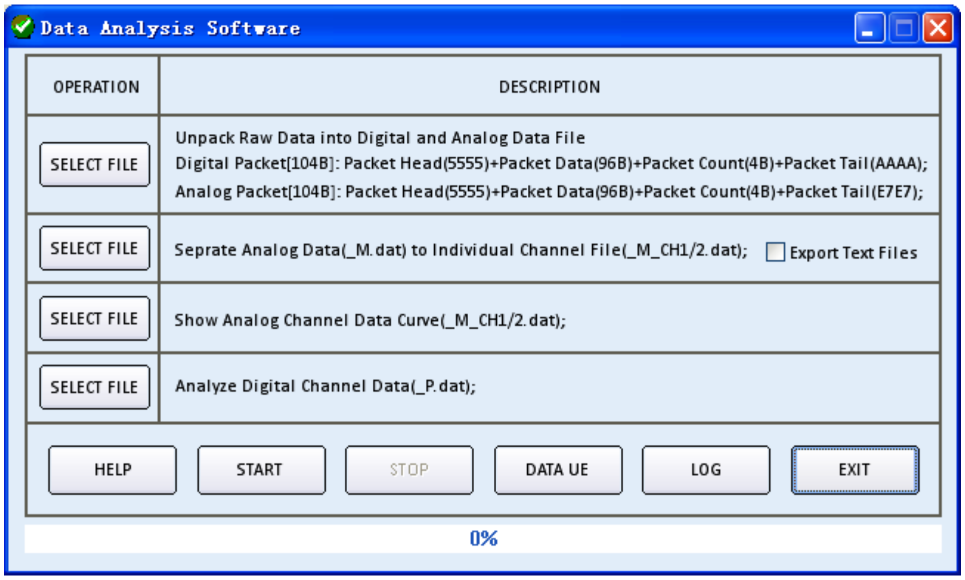
In this paper, Visual C++ 6.0 is adopted for data analysis software design, which is shown in Fig. (**6**). According to the general process of data analysis, the software is divided into "data unpack", "channel data analysis", "channel data export or curve shows" and other modules.

CONCLUSION

Data is an important indicator and foundation of system performance and working conditions. In this paper, large-capacity data in testing system processing method issued, combined with C++, Win32 API functions, such as support

Frame Head (2 Bytes) 0X"EB+90"	Frame Data (DLen Bytes)	Frame Count (CntLen Bytes)	Frame Tail (2 Bytes) 0X"14+6F"
FLen Bytes			

Fig. (4). A typical data frame.



OPERATION	DESCRIPTION

SELECT FILE	Unpack Raw Data into Digital and Analog Data File
	Digital Packet[104B]: Packet Head(5555)+Packet Data(96B)+Packet Count(4B)+Packet Tail(AAAA);
	Analog Packet[104B]: Packet Head(5555)+Packet Data(96B)+Packet Count(4B)+Packet Tail(E7E7);

SELECT FILE	Seprate Analog Data(_M.dat) to Individual Channel File(_M_CH1/2.dat); ☐ Export Text Files
SELECT FILE	Show Analog Channel Data Curve(_M_CH1/2.dat);
SELECT FILE	Analyze Digital Channel Data(_P.dat);

Fig. (6). Data analysis software.

for file operations, discussed the analysis methods of large-capacity binary data files, and designed the data processing software in Visual C++ 6.0 environment. Through the self-test data and the actual test data analysis and processing, the analysis method has high efficiency and stability, better able to deal with large-capacity data. In addition, how to analyze large capacity and even massive amounts of data need to employ more advanced technologies, such as: multi-threaded programming [3, 4], memory mapping, parallel technology [5], multi-core technology [6], "Big Data" operation [7-10], cloud computing [11, 12], these are the follow-up studies of this paper's direction.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

The authors wish to thank for the support in part by National Natural Science Foundation of China (No. 51275491).

REFERENCES

[1] K. Chen, and W. Zheng, "Cloud computing: system instances and current research", *Journal of Software*, vol. 20, pp. 1337-1348, May. 2009.

[2] D. Zhu, and Y. Song, "Accomplishment of algorithm of massive data analysis and processing", *Journal of Changchun University*, vol. 21, pp. 42-45, Aug. 2011.

[3] Y. Li, Y. Zhao, M. Li, and Y. Du, "Tread partitioning algorithm for speculative multithreading based on path optimization", *Journal of Software*, vol. 23, pp. 1950-1964, Aug. 2012.

[4] X. Hu, S. Pan, Y. Hu, and X. Li, "Mitigating voltage emergency in simultaneous multithreading processor by memory level parallelism aware thread scheduling", *Chinese Journal of Computes*, vol. 36, pp. 1065-1075, May. 2013.

[5] H. Wei, J. Yu, H. Yu, and M. Qin, "A method on software pipelined parallelism for data flow programs", *Chinese Journal of Computes*, vol. 34, pp. 889-898, May. 2011.

[6] H. Liu, and P. Zhao, "A multi-core fair memory scheduling model", *Chinese Journal of Computes*, vol. 36, pp. 2191-2199, Nov. 2013.

[7] X. Meng, and X. Ci, "Big data management: concept, techniques and challenges", *Journal of Computer Research and Development*, vol. 50, pp. 146-169, Jan. 2013.

[8] J. Li, and X. Liu, "An important aspect of big data: data usability", *Journal of Computer Research and Development*, vol. 50, pp. 1147-1162, Jun. 2013.

[9] S. Wang, H. Wang, X. Qin, and X. Zhou, "Architecting big data: challenges, studies and forecasts", *Chinese Journal of Computes*, vol. 34, pp. 1742-1752, Oct. 2011.

[10] X. Meng, Y. Li, and J. Zhu, "Social computing in the era of big data: Opportunities and challenges", *Journal of Computer Research and Development*, vol. 50, pp. 2483-2491, Dec. 2013.

[11] P. Wang, L. Zhang, C. Ren, and Y. Guo, "The analytical model and simulation research in phase space of cloud computing", *Chinese Journal of Computes*, vol. 36, pp. 286-296, Feb. 2013.

[12] Z. Liu, Z. Wen, and H. Zhang, "Cloud computing and cloud data management technology", *Journal of Computer Research and Development*, vol. 49, pp. 26-31, Jun. 2012.

Received: September 22, 2014 Revised: November 30, 2014 Accepted: December 02, 2014