Reverse Engineering Megadata MD3000 Database File Format
Swi Hong Tjan

1Electrical Engineering, Politeknik Negeri Bandung, Bandung 40012, Indonesia
*Corresponding author. Email: tjan.sh@polban.ac.id

ABSTRACT
Megadata MD3000 is a SCADA master used to monitor and control an Indonesian railway company's microwave communication. The master stores all of the system configurations in a proprietary database file format. This paper introduces a SCADA file format from Megadata. The files consist of digital, analog, RTU name, sequence control, hybrid diesel, RTU status calc, RTU scan, and module list database. The file format needs to know to extract the system's data. The method to find the file format is reverse engineering. Its steps are to collect and classify files based on file types, collect the stored data, compare manually file's raw with actual data, and summarize its format. Past research has obtained the files. The real data gathering uses system tools by following the user manual. Analyze the file data by manual comparison and display using xxd tools. The result, file data elements are 1-bit logic, 1-byte or 2-byte integer, IEEE754 single-precision float, ASCII character encoding, and fixed-length null-terminated string. The aname.dat, dname.dat, and rname.dat file data format are 32 bytes null-terminated string. File format for digpt.dat, anapt.dat, seqc.dat, hybrid.dat, rtucalc.dat are an array of records with 14, 46, 8, 10, 8 byte/record. File scanlist.dat contains the RTU scan list at Offset 0 and module list at offset 48.

Keywords: Reverse Engineering, Megadata, File Format, MD3000, SCADA

1. INTRODUCTION
Megadata MD3000 is a SCADA master used to monitor and control an Indonesian railway company's microwave communication [1]. It started in operation in 1990. Because of age and no longer supported by the vendor, the master SCADA is not functioning while its RTUs that scattered around Java still work well. The master used 68000 microprocessor proprietary hardware, so its software cannot run outside the system [2]. One way to continue its functions is to do master SCADA reengineering using the available hardware and software. The first step in this process is retrieving and evaluating data stored in the master database. This data contains the system configurations. The database uses a proprietary format, so its file format must discover before retrieving its content.

1.1. Related Work
A similar file format works ranges from general to specific file. This paper uses a similar method with files from previous work using standard data types.

1.1.1. File Format Works
Other works include the approach to analyze a general binary file, a log file, a specific database file, an encrypted database, and the SQLite database. Pehnack describes how to analyze a binary file for reverse engineering [3], while Zelenyuk explains a detailed step to reverse engineer a database file [4]. Schuster discovers the Microsoft Vista event log file format by comparing Vista log files in binary form with textual form [5]. Choi et al. analyze the encrypted database file format using the reverse-engineering method [6]. They found the unencrypted database use SQLite and OLE compound files. A standard to examine the SQLite files for forensic analysis set by Nemetz et al. [7].

1.1.2. Previous work
MD3000 SCADA master uses a proprietary operating system called Megadata Operating System [8]. Its file system does not recognize by DOS, Windows, or Linux. The previous work is reverse engineering the Megadata file system, so its files can be retrieved [9].
1.1.3. Data type standards

Character and numeric are typical data stored in a file. A string consists of an array of characters. The C-style is null-terminated while the Pascal-style is length prefixed [10] [3]. Standard character encoding includes ASCII, EBCDIC, and UTF-8 [10]. Numeric consists of an integer and float. Depends on the value range, an integer store is 1, 2, 4, 8, or 16 bytes [3]. The float stores as IEEE 754, mostly with sizes 4 or 8 bytes (single-precision/double-precision).

1.2. Our Contribution

This paper introduces a SCADA file format from Megadata. The files consist of digital, analog, RTU name, sequence control, hybrid diesel, RTU status calc, RTU scan, and module list database.

1.3. Paper Structure

The rest of the paper is organized as follows. Section 2 describes the background. Section 3 explains the methodology, follow by the result in Section 4, and the conclusion in Section 5.

2. BACKGROUND

Reengineering is one of the methods used to transform a legacy/proprietary system into a new system while maintaining compatibility with the existing hardware, software, or operation [11]. It usually consists of reverse engineering and forward engineering [12]. Reverse engineering is a process of analyzing the legacy system to identify its component and relationship or create an abstraction of process flow. Forward engineering is a process of the physical implementation of a system from abstraction/design [13].

Reverse Engineering Megadata MD3000 Database File Format is one of the steps in reengineering master SCADA. Figure 1 is the fishbone process from the proprietary Master SCADA to a new one. Its light blue color is the database file format, and the green one is the previous work [9].

3. METHODOLOGY

The method to discover the database file format is reverse engineering. Its steps are to collect and classify files base on file types, collect the actual data, compare manually file’s raw with real data, and summarize its format.

4. RESULTS

Past research has obtained data files [9]. Collect the file’s real data using system tools by following the user manual [1]. Analyze the file data by manual comparison and display using xxd tools.

4.1. Collect and Classify Files

Figure 2 is part of the data files list obtained from past research [9]. According to the training manual, the database file has a .DAT extension [1]. This research will focus on the .DAT file type.

```
drwxrwxrwx   1344 ..
drwxrwxrwx     80 ..
-rw-rw-rw- 1048576 PICTURE.CRT
-rw-rw-rw-  49920 PICINDEX.CRT
-rw-rw-rw-   78080 DYNADATA.IMG
-rw-r-r-r   127232 DIGPT.DAT
-rw-r-r-r    98560 ANAPT.DAT
-rw-r-r-r     4352 RNAME.DAT
-rw-r-r-r     69888 ANAME.DAT
-rw-r-r-r   291072 DNAME.DAT
-rw-r-r-r      292 RTU_NAME.HLP
```

Figure 2 MD3000 Data File List

4.2. Collect the File’s Real Data

The actual data collection uses database tools by following the training manual [1]. Chapter 15 Database Compiler in the training document explains about database structure and software operation. It includes a select, display, and edit data. Figure 3 through Figure 9 are examples of the database display output that contains the file’s real data.

```
Digital database (mmem DPT) 144

dig pt name (mmem DName) = MICROWAVE 1 - URGENT

dig FU CS etc (mmem DBITs)

On = AS AUD

Off = FU CS NAL NAR NLG NLR IV DP SG CN

Digital Device Type (mmem DDEVice) = 1 (MICROWAVE)

dig time delay (mmem DTDLY) = 0

dig link PID (mmem DLPID) = 0

dig pt name (mmem DName) = MICROWAVE 1 - NON URGENT

dig FU CS etc (mmem DBITs)

On = AS AUD

Off = FU CS NAL NAR NLG NLR IV DP SG CN

Digital Device Type (mmem DDEVice) = 1 (MICROWAVE)

dig time delay (mmem DTDLY) = 0

dig link PID (mmem DLPID) = 0
```

Figure 3 Digital Database Sample Output
Figure 4 Analog Database Sample Output

| Parameter                        | Value       |
|----------------------------------|-------------|
| ana name (mnem ANName )          | 48 VOLT BUS VOLTAGE |
| ana FU NAL etc(mnem ABITs)On    | 0.00        |
| ana name (mnem ANName )          | 48 VOLT BUS VOLTAGE |
| ana FU NAL etc(mnem ABITs)On    | 0.00        |
| ana pt RTU (mnem ARTU )          | 11 (SASAKSAAT) |
| Analogue Point Type (mnem ADevice) | 10 (POWER) |
| ana eng unit idx (mnem AEUN )    | 1 (VOLTS)   |
| ana xxxxxx fmt (mnem AFMT )      | 6 (XXX)     |
| ana deadband % (mnem ADEB )      | 6           |
| ana raw max (mnem ARMAX )        | 2100        |
| ana raw min (mnem ARMIN )        | 0           |
| ana hi alarm limit (mnem AHigh )| 260         |
| ana lo alarm limit (mnem ALow )  | 165         |
| eng value max (mnem AMAX )       | 237.00      |
| eng value min (mnem AMIN )       | 0.00        |

Figure 5 Sequence Control Sample Output

| Parameter                        | Value       |
|----------------------------------|-------------|
| sequence control (mnem SEQ_CON)  | 0           |
| day of week (mnem SCDAY)         | 2 (TUESDAY) |
| hour of the day (mnem SCHOUR)    | 10          |
| minute of the hour (mnem SCMIN)  | 0           |
| type of point (mnem SCTYPE )     | 12 (DIESEL, CON) |
| action to be done (mnem SCACT)   | 2 (START)  |

Figure 6 RTU Scan List Sample Output

| Parameter                        | Value       |
|----------------------------------|-------------|
| RTU scanlist (mnem RTU_TB)       |             |
| RTU telm str addr (mnem RTAD )   | 1           |
| pretrans time msec (mnem RPRET  )| 0           |
| site name (mnem RNNAME )         | SCC BANDUNG |
| Dig on-line input pi (mnem DIGIP)| 1331        |

Figure 7 Hybrid Diesel Sample Output

| Parameter                        | Value       |
|----------------------------------|-------------|
| RTU name (mnem HRTU )            | 23 (PLABUAN) |
| Analogue result pid (mnem ANARES)| 1375        |
| Analogue input #1 pi (mnem ANAIP1)| 259         |
| Analogue input #2 pi (mnem ANAIP2)| 260         |
| Dig on-line input pi (mnem DIGIP)| 1331        |

Figure 8 RTU Status Calc Sample Output

| Parameter                        | Value       |
|----------------------------------|-------------|
| Module list (mnem MD_LIST)       | 0           |
| first PID for module (mnem MFPID)| 1008        |
| module str addr (mnem MSTRAD )   | 1 (SCC SEMARANG) |
| module address (mnem MADD )      | 1           |

Figure 9 Module List Sample Output

4.3. Compare File's Raw with Real Data

The following compares the digital, analog, RTU name, Sequence Control, RTU status calc, RTU scan, and module list database output files with their raw data files.

4.3.1. Digital Database

The sample digital database output shows in Figure 3. Digital database number 144 is "MICROWAVE 1 – URGENT", and number 145 is "MICROWAVE 1 - NON-URGENT". By searching those text, it found in the dname.dat file. Using the xxd tool, Figure 10 displays its content in the form of hexadecimal and ASCII. Figure 11 uses EBCDIC in place of ASCII.
The ASCII encoding shows the correct text. The end of the string is 00. It is an ASCII encoded null-terminated string.

The digital number 144 starts at offset 1200H or 4608 decimals, while number 145 is 1220H or 4640 decimals. The difference is 4640-4608 = 32 byte. The offset divide by the difference is 4608/32 = 144 and 4640/32 = 145. So, it is a fixed maximum length string.

The analog number 64 starts at offset 800H or 2048 decimals, while number 65 is 820H or 2080 decimals. The difference is 2080-2048 = 32 byte. The offset divide by the difference is 2048/32 = 64 and 2080/32 = 65. So, it is a fixed maximum length string.

The binary data of number 144 is 00001000 00000000 00000000. Then the analog database number 64 “220 VAC BUS VOLTAGE” and number 65 is “48 VOLT BUS VOLTAGE”. Searching those text, it found in the aname.dat file. Using the xxd tool, Figure 13 displays its content in the form of hexadecimal and ASCII.

**4.3.2. Analog Database**

The sample analog database output shows in Figure 4. Analog database number 64 “220 VAC BUS VOLTAGE” and number 65 is "48 VOLT BUS VOLTAGE". Searching those text, it found in the aname.dat file. Using the xxd tool, Figure 13 displays its content in the form of hexadecimal and ASCII.
Searching them in anapt.dat, it found in the rname.dat file. The using the xxd tool, Figure 15 displays its content in the form of hexadecimal and ASCII.

The RTU number 11 at offset 160H or 352 decimals, while number 12 is 180H or 384 decimals. The difference is 384 - 352 = 32 byte. The offset divide by the difference is 352/32 = 11 and 384/32 = 12. So, it is a fixed maximum length string. The same as the digital database, it is an ASCII encoded null-terminated string with 32 bytes maximum length.

4.3.4. Sequence Control database

The sample sequence control output shows in Figure 5. The sequence control number 0 contains 2, 10, 0, 0, 12, 2, while number 1 contains 2, 11, 0, 0, 12, 1. Because all of them are less than 256, assuming all the data stores as 1 byte. Number 0 becomes 02, 0A, 00, 00, 0C and 02. Number 1 becomes 02, 0B, 00, 00, 0C and 01. Figure 16 shows the contents of the seqc.dat file. The corresponding data for number 0 is at offset 0(yellow) and number 1 at offset 8(grey). It is a fixed-length record of 8 bytes.

4.3.5. Hybrid Diesel Database

The sample of hybrid diesel output shows in Figure 7. The sequence control number 0 contains 16, 1356, 147, 148, 659, while number 1 contains 23, 1357, 259, 260, 1331. Because most of them more than 256, assume all the data store as 2 bytes. Number 0 becomes 00 10, 05 4C, 00 93, 00 94, 02 93. Number 1 becomes 00 17, 05 4D, 01 03, 01 04, 05 33. Figure 17 shows the contents of the hybrid.dat file. The corresponding data for number 0 is at offset 0(yellow) and number 1 at offset 10(grey). It is a fixed-length record of 10 bytes.

4.3.6. RTU Status Calc Database

The sample of RTU status output shows in Figure 8. The RTU status calc number 0 contains 11, 7516, 64, 65, while number 1 contains 12, 7517, 80, 81. Because some of them are more than 256, assume all data store as 2 bytes. Number 0 becomes 00 0B, 1D 5C, 00 40, 00 41. Number 1 becomes 00 0C, 1D 5D, 00 50, 00 51. Figure 18 shows the contents of the rtucalc.dat file. The corresponding data for number 0 is at offset 0(yellow) and number 1 at offset 8(grey). It is a fixed-length record of 8 bytes.

4.3.7. RTU Scan, and Module List Database

The sample of the RTU scan list and module list output. The scanlist.dat file stores both data. The RTU scan list data is 1, 0, and "SCC BANDUNG." The data module list number 0 contains 1008, 1, 1, 1, while number 1 contains 1024, 1, 1, 2. Assume the integer data less than 256 stores as 1 byte, the integer more than 256 stores as 2 bytes.
Number 0 becomes 03 F0, 01, 01, 01. Number 1 becomes 04 00, 01, 01, 02. Figure 19 shows the contents of the scanlist.dat file. The corresponding data for the RTU scan list is at offset 0 (yellow). The corresponding data for the module list number 0 is at offset 48 (grey) and number 1 at offset 56 (green). It is a fixed-length record of 8 bytes.

C: >xxd -g1 -l100 -u -c 8 -S0 scanlist.dat
0000000: 01 00 00 06 04 B0 00 00 .......... 0000008: 53 43 43 20 42 41 4E 44 SCC BAND
0000010: 55 4E 47 00 00 00 00 00 UNG..... 0000018: 00 00 00 00 00 00 00 00 .......... 0000020: 00 00 00 00 00 00 00 00 .......... 0000028: 03 F0 01 01 01 00 00 00 .......... 0000030: 04 00 01 01 02 00 00 00 .......... 0000038: 04 10 01 01 03 00 00 00 ..........

Figure 19 File scanlist.dat ASCII XXD

5. CONCLUSION
The MD3000 database data elements are 1-bit logic, 1-byte or 2-byte integer, IEEE754 single-precision float, ASCII character encoding, and fixed-length null-terminated string.

The aname.dat, dname.dat, and rname.dat file data format are 32 bytes null-terminated string. File format for digpt.dat, anapt.dat, seqc.dat, hybrid.dat, rtucalc.dat are an array of records with 14, 46, 8, 10, 8 byte/record. File scanlist.dat contains the RTU scan list at Offset 0 and module list at offset 48.

ACKNOWLEDGMENT
This work was supported by UPPM Politeknik Negeri Bandung.

REFERENCES
[1] R. Bradhurst, Training Course for the Microwave Radio Supervisory System for PJKA Indonesia, vol. Volume 2 Software Course, French Forest, NWS: Megadata Pty Ltd, 1990.

[2] A. McRae, "The Architecture of a Distributed Substation Management System (A description of the Megadata Substation Management System)," in Distribution Automation/Demand Side Management conference, Palm Springs, California, 1993.

[3] A. Pehnack, "How to Approach Binary File Format Analysis Essential knowledge for reverse engineering," Chicago, 2015.

[4] S. Zelenyuuk, "Database Reverse Engineering, Part 2: Main Approaches," 6 January 2018. [Online]. Available: https://medium.com/@MorteNoir/database-reverse-engineering-part-2-main-approaches-ae9355b2d429.

[5] A. Schuster, "Introducing the Microsoft Vista event log file format," Digital Investigation, Vols. 4, Supplement, p. S65 – S72, 2007.

[6] J. Choi, J. Yu, S. Hyun and H. Kim, "Digital forensic analysis of encrypted database files in instant messaging applications on Windows operating systems: Case study with KakaoTalk, NateOn and QQ messenger," Digital Investigation, Vols. 28, supplement, pp. S50-S59, 2019.

[7] S. Nemetz, S. Schmitt and F. Freiling, "A standardized corpus for SQLite database forensics," Digital Investigation, Vols. 24, Supplement, pp. S121-S130, 2018.

[8] Megadata, Supervisory System Equipment, Megadata Programmers Manual, Frenchs Forest: Megadata Pty Ltd, 1988.

[9] S. H. Tjan, "Reverse Engineering File System Of Megadata Operating System," ICIEVE, Bandung, 2019. Abstract ABS-12.

[10] S. D. Burd, System Architecture sixth edition, Boston: Course Technology Cengage Learning, 2011.

[11] F. Weil, "Legacy Software Reengineering," UniqueSoft, LLC, Palatine, IL, USA, 2015.

[12] E. J. a. C. J. H. Chikofsky, "Reverse Engineering and Design Recovery: A Taxonomy," IEEE Software, vol. 7, no. 1, pp. 13-17, 1990.

[13] R. a. G. I. G.-R. d. a. P. M. a. E. C. Perez-Castillo, "Reengineering Technologies," IEEE Software, pp. 13–17, 2011.