Real-time Error Measurement System for MVB Protocol

Su Goog Shon[1] and Soo Mi Yang[2]

[1] Department of Information and Telecommunication Engineering, The University of Suwon, Hwasung-city, Gyunggi-do, 445-743, Korea

[2] Department of Internet Information Engineering, The University of Suwon, Hwasung-city, Gyunggi-do 445-743, Korea

Abstract

Recently, there are lots of control equipments in a train such as traction control, air conditioners and even internet access. For this reason, vehicle network must allow for the big amount of transmission data and must ensure the high reliability. After investigating about characteristics of multifunction vehicle bus, an error detection and analysis system is proposed. The proposed error analysis system can be used to verify high reliability of data transmission over multifunction vehicle bus. We explain how to implement the embedded error analysis system based on an ARM processor. The proposed system can detect all kinds of errors that IEC 61375 standard refers to.

Keywords: *Reliability, Measurement, MVB, TCN, IEC, Linux, TCP/IP*

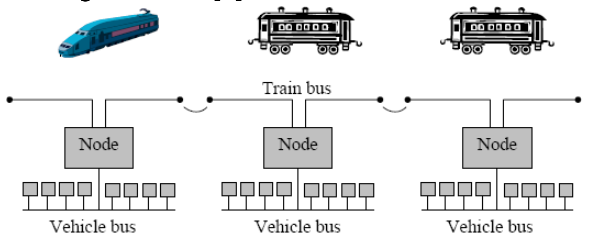
1. Introduction

The recent challenge for railway industry is to make automatic electronic coupling of the vehicles through a data communication network. Over the last several years, a modern, versatile communication system on board trains, both to interconnect equipment located inside a railway vehicle and to allow communication between different vehicles, has been studied.

At the train level, the data communication system should configure itself when vehicles are coupled and interconnected on the track. At the vehicle level, manufacturers should assemble pre-tested units, such as doors or air conditioners manufactured by subcontractors. Manufacturers could reduce development costs by utilizing one standard. Train operators could reduce spare parts and simplify maintenance and part replacement. Automatic electronic coupling of the vehicles with standardization could improve train safety and maintenance.

An international standardization of data communication has been studied at both the train and vehicle levels. Especially in Europe, it is important to ensure cross-border traffic by standardizing track profiles, pneumatic hoses, traction voltages, operating procedures, and so on. Trains need a standard form of data communication for train control, diagnostics, and passenger information. Such a data communication network was specified by International Electro-technical Committee (IEC) as the Train Communication Network (TCN). The IEC groups worldwide deputies from over 20 countries worked several years within the IEC's Working Group 22 on the definition of the Train Communication Network [1][2]. The TCN was adopted as the international standard IEC 61375 in 1999 [3].

Train Communication Network is a real-time data network proposed for use on trains, consists of two different networks with somewhat different protocols [4], and is called the distributed control system. The TCN architecture also addresses all relevant configurations found in rail vehicles. It comprises the train bus connecting the vehicles and the vehicle bus connecting the equipment aboard a vehicle. The train bus architecture is divided into a Wire Train Bus (WTB) interconnecting all vehicles, and in each vehicle a MVB (Multifunction Vehicle Bus), as according to the TCN standard. The general architecture of a train communication network is shown in Figure 1 from [4].



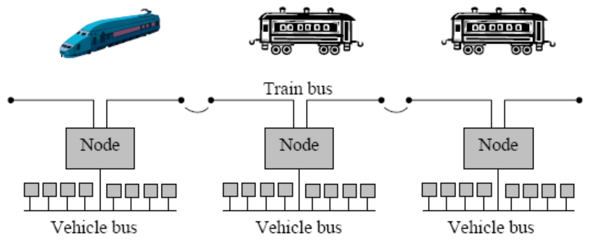


Fig. 1 Train Communication System

Error detection is a crucial part of the train safety and maintenance. The train network must satisfy minimum levels of requirements for message frame transmission integrity. The integrity simply means that a sufficient number of uncorrupted frames are delivered to perform target functions. The TCN is relatively known as having excellent error detection properties [5][6], even though there is no error detection scheme that can detect all possible errors. It is important that train operators or maintenance staff should note whether any error has occurred.
Currently, error analyzers for MVB are based on signal or frame levels. This paper is interested in frame-based error analyzers that basically can capture and analyze frames over the MVB bus and can tell whether errors have occurred. The staff of a maintenance department is of particular interest about how many errors do occur over the MVB. Even though there are some commercial error analyzers available from Siemens, Duagon, etc, they still have some problems, such as missing error status information due to slow communication speed of RS-232C, or inconvenience due to very short communication length of PCI or PCMCIA.
In this paper, we propose a fast and user-friendly error analyzer that can detect transmission errors of the data communication system on a train. So, the staff can easily obtain the results of error analysis via Ethernet, which is fast and lengthy relatively. Also, characteristics of frames are reviewed on the MVB in a train and the error coding scheme is discussed. The error measurement system corresponding with IEC standard is designed and implemented based on an ARM processor and Linux operating system. It can capture all kinds of corrupted frames on the MVB and monitor the line status. We explain the internal architecture of our approach. Finally, we draw conclusions from the actual implementation.

2. MVB Controller

The MVB protocol is used for connecting equipment within a single vehicle (e.g., a rail car) or within different vehicles in closed train sets. Each vehicle with MVB has its own vehicle bus connecting on board equipment, such as sensors and actuators for doors, brakes and air conditioning system. The MVB enables considerable reduction in the amount of cabling and increased reliability with respect to conventional wiring.

The MVB is a standard communication medium to transport and exchange data among attached nodes. These nodes, which are physically connected to the bus, may vary in function, size, and performance at a physical layer level.
Figure 2 shows a diagram of MVBC (Multifunction Vehicle Bus Controller) made by ADtranz [7], which is possible to be an MVB class 2 or a higher device that can have a processor and may exchange data. For the MVB data transport, it requires the MVBC that handles the bus traffic and arbitration without participation of external application CPU. The MVBC and CPU can exchange data through a common memory called the traffic store. The MCU in the MVBC of Figure 2 coordinates all data transfers between the MVB and the traffic store.

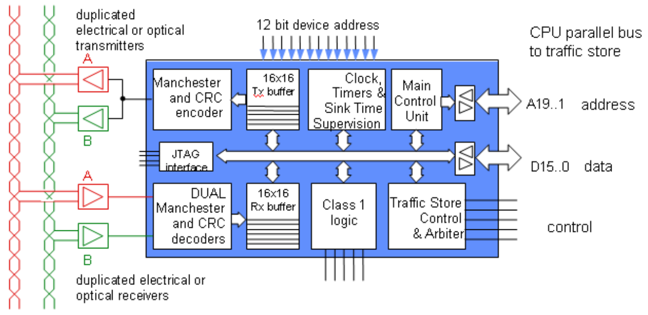

duplicated electrical or optical transmitters
12 bit device address
CPU parallel bus to traffic store
A
B
Manchester and CRC encoder
16x16 Tx buffer
Clock, Timers & Sink Time Supervision
Main Control Unit
A19..1 address
JTAG interface
D15..0 data
A
B
DUAL Manchester and CRC decoders
16x16 Rx buffer
Class 1 logic
Traffic Store Control & Arbiter
control
duplicated electrical or optical receivers

Fig. 2 MVBC ASIC by ADtranz

All information and data pertaining to the MVBC are found in the traffic store (memory). The traffic memory stores two types of data: process variables and messages that TCN buses transport. Process variables reflect the train's state, such as speed, motor current, and operator's commands. Message data carry infrequent but possibly lengthy information, for instance, diagnostics or passenger information. Message length varies between a few bytes to several kilobytes. The traffic memory is a shared memory to interface the MVB with the application CPU. This store is visible to both application CPU and MVBC.
Devices with MVBC are operated on maser or slave mode. A port is used to provide a means to control access, data storage and interrupts such that data is read consistently when it is accessed simultaneously by the bus and by the application CPU. This requires at least two memory pages: one is accessed for the bus access, the other for application CPU. Each device owns a certain number of ports configured. Two kinds of ports exist: physical and logical ports.
The traffic memory is divided into 4 partition areas such as logical address space, device address, service area, etc. In this paper, memory configuration mode 3 is used, where the port index table maps the logical port addresses to the device address of the traffic memory which are

specified in the master frames. The Port Control and Status Register (PCS) contain all the relevant information pertaining to one port. This information is used by MVBC to determine how it should handle the related port. The PCS includes the following information: port related information (function code, port description, event, etc.), data consistency check, telegram report, transfer acknowledge bits, and check sequences. It is important for the MVB error analyzer to enable the loading data from the PCS. In addition, the two interrupt request signals must be connected directly or via an interrupt controller to the CPU.

3. MVB Characteristics

Transmission errors can occur due to corruption from network transmission noise. Detecting every possible corrupted frame is inherently impossible because any detection technique cannot find out all sets of bit errors.

MVB adopts the Manchester encoding technology according to the IEC 61375. The fixed frequency is 1.5Mbps, and there is an 8-bit cyclic redundancy code (CRC) following each 64-bit frame data. The frame of MVB can be divided into two parts: master and slave frame, which consist of START, DATA, CRC and END.

The general formats of frames on the MVB are shown in Figure 3 and discussed first. Frames start with a start delimiter preamble of 9 bits. Frame data includes from one to four data payload sections, with each payload being 16, 32, or 64 bits in size. Frames with more than 64 bits of data are broken into multiple 64-bit data payloads as shown.

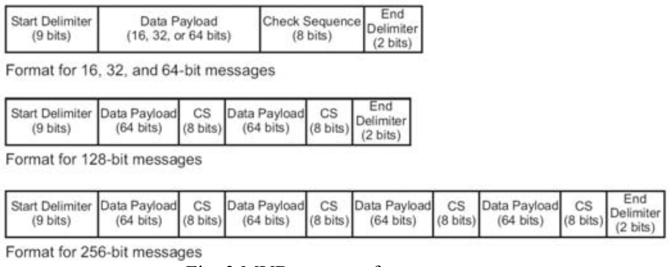


Fig. 3 MVB message formats.

Each data payload section is protected by an 8-bit Check Sequence (CS). The end of each frame is denoted by a 2-bit End Delimiter sequence. Frame length is inferred from the detection of an End Delimiter. The MVB uses a Check Sequence protecting every data payload segment of 16, 32, or 64 bits. The Check Sequence use the CRC polynomial for MVB, which is given by (1)

$$G(x) = x^7 + x^6 + x^5 + x^2 + 1. \tag{1}$$

This polynomial guarantees a Hamming distance of 4. The CS encoding used by the MVB is known to detect successfully 1-bit and 2-bit errors [7]. The overall Hamming distance of 8 is obtained from the combined operation of CRC and Manchester coding.

The MVB uses well designed error coding schemes. The MVB has the ability to detect two types of errors caused by noise during transmission: invalid delimiter encoding and check sequence values. The mechanism to detect errors in MVB is based on observing Manchester bit encoding errors and detecting mismatches of CRC between sent and received.

The MVB uses two kinds of telegrams: the master and the slave frame. The master frame is a 16-bit word which consists of an F-Code (4bits) and an address (12 bits).

All buses pertaining to the TCN provide two basic medium accesses: periodic (for data like process variables) and sporadic (for on-demand data traffic, such as messages). Periodic and sporadic data traffic share the same bus, but devices treat each separately. Figure 4 shows the TCN basic period.

One device acting as master controls periodic and sporadic data transmission, which guarantees deterministic medium access. To accomplish this, the master alternates periodic and sporadic phases. Traffic is divided into basic periods of fixed duration—either 1 or 2 ms on the MVB. At the start of a period, the master polls the process variables in sequence during a certain time period—the periodic phase. To reduce traffic, urgent data are transmitted every period and less urgent variables are transmitted with an individual period every second, forth, eight, and so on basic period, with the longest period being 1,024 ms.

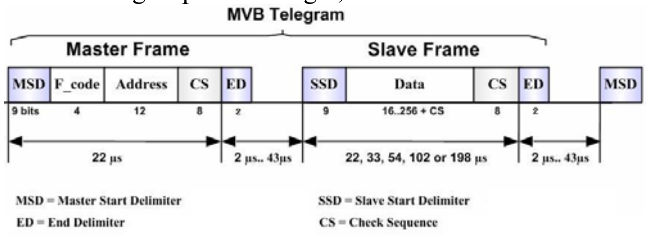


Fig. 4 MVB telegram

Decoder accepts Manchester signals, strips delimiter and stores received data in receive buffer. Error information is transferred to the telegram analysis unit, where the information is processed. Some are passed to the Main Control Unit (MCU) for further actions.

Telegram analysis unit of the MVBC plays key role to detect sorts of errors. Telegram analysis unit which consists of four registers: Frame Counter, Error Counter, Master Frame Register, and Master Frame Reg. Duplicate Exception handles the following tasks: reporting incoming slave frames, timeout mechanisms, telegram error handling, master frame registers, and telegram error recordkeeping. For example, the interrupt "Slave Frame Checked" is asserted by the announcement of the decoder that a valid or erroneous Slave Frame has arrived. The numbers of frames sent and received are recorded in the Frame Counter Register (FC) for obtaining quantitative

results regarding the quality of the bus. The number of erroneous frames sent or received is recorded in the Error Counter Register (EC). When 65,535 frames have been received, the interrupt "Frames Evaluated Interrupt" will be asserted [7].
Table 1 shows types of error that telegram analysis unit can detect and its register name used as in IEC 61375-1. The column of implementation in Table 1 means that each type of errors can be detected by using the proposed error detector.

Table 1. Error types and registers

Types of error	Types of register	Implementation
Total Frame counter	FC	0
Master Frame check	MFC	0
Slave Frame check	SFC	0
Erroneous Master Frame	EMF	0
Erroneous Slave Frame	ESF	0
Duplicate master frame	DMF	0
Duplicate slave frame	DSF	0
Bus timeout interrupt	BTI	0
Reply timeout interrupt	RTI	0
Frame evaluated interrupt	FEV	0
Data transfer interrupt	MCU	0
Tx queue exception	TQE	0
Rx queue exception	RQE	0
Transmit queue complete	TQC	0
Receive queue complete	RQC	0
Externel Interrupt 0-3	EI	0
All master frame transmitted	AMFX	0

4. Implementation of MVB Error Measurement System

Recently, there have been researches to develop embedded network equipment [8]. For the error analysis for the MVB frame transmissions, a MVB error analysis system is designed. The analysis system consists of both a MVB error detector and an error analyzer as shown in Figure 5. The MVB error detector can capture the MVB frames on the line and can tell what kinds of error occur or not. The MVB error analyzer can tell the results of error analysis for frames captured.

MVB nodes are connected to the bus in series. Figure 5 shows an example system to gather MVB data for this paper. There is a maser MVB node and a slave MVB node. The MVB error detector is connected between the master and slave MVB node. Both the MVB master and the slave send and receive frames over the line. Then, frames captured from the MVB error detector pass through a decoder, a receive buffer, and a telegram analysis unit of the MVB. Whenever there is an error referred in Table 1, an interrupt event is asserted. Error analyzer on the Windows PC gets the results of error analysis system via TCP/IP communication channel and displays that on the screen.

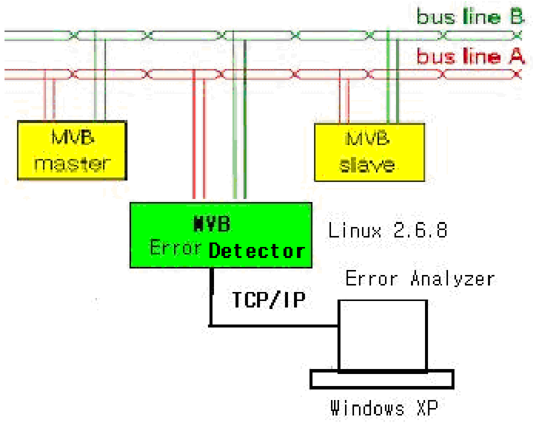


Fig. 5 MVB error analysis system

Figure 6 shows the embedded MVB error detector implemented. The error detector mainly consists of MVB controller and CPU. The MVB controller is physically connected to MVB bus and is operated as one of MVB slave nodes. This node can only get command and data as a sink on the bus. The data, including all the process variables and messages, are stored into the traffic memory of 256KB interfaced to the MVB controller and are shared between the MVB controller and CPU (ARM). The ARM can access the traffic memory and its MVB data, and can transfer to the MVB error analyzer program on the PC via TCP/IP interface. One of the jobs that the ARM offers is to convert frame data of the MVBC into Ethernet format.

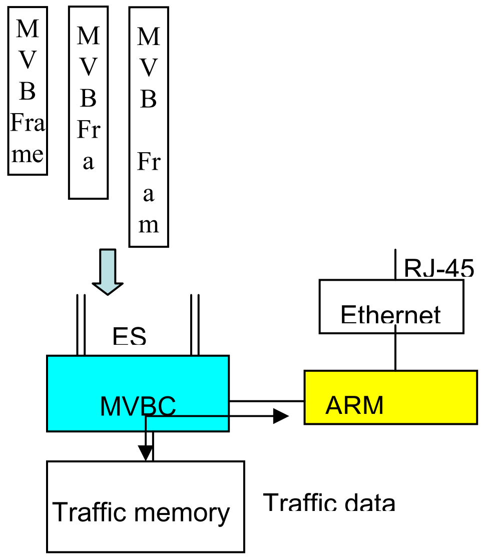


Fig. 6 Embedded MVB error detector

Figure 7 describes the details for the ARM board. For this research, the Samsung 2410 of ARM 920T is used. The ARM 2410 is directly connected to MVB controller through ARM bus and gathers and process the MVB data. The MVBC handles the ARM access made to the traffic memory or internal registers in the MVBC.

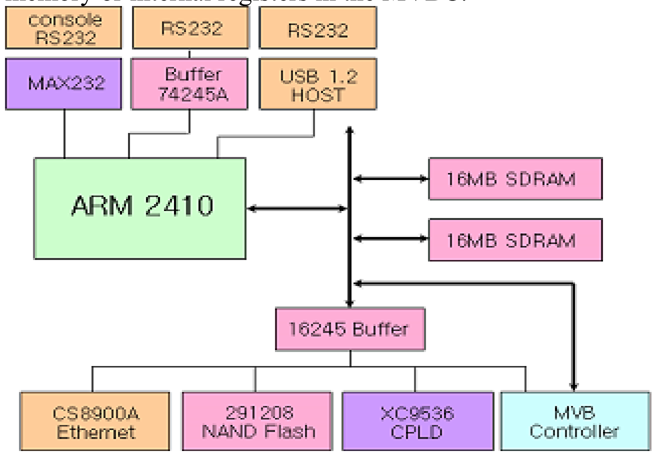


Fig. 7 Details of MVB ARM board

Figure 8 shows the software platform for embedded MVB error detector. The system based on Linux kernel 2.6.8 consists of MVBC driver, Ethernet driver, TCP/IP protocol stack, and application program that fetch, store, and process the traffic memory data. It uses the programming languages ANSI C for programming their devices.

For this embedded system, a boot loader (uboot) is used to initialize the system and load the Linux kernel and so on. All the Linux kernel, Linux file system, drivers, and application programs are stored on NAND flash memory, and loaded into the SDRAM after booting. After booting, RS-232C or Ethernet interface is used for debugging and loading the application programs.

The MVB board driver is to process interrupt routine. It can initialize the traffic memory, store PCS information to the portlist, and register the PCS on the MVBC. Also, it can deal with the data communication between the traffic memory and DPRAM.

Ethernet driver routine can offer the socket communication to the PC-based MVB error analysis program, where the socket program is operated on the non-blocking mode, and may connect to the PC program whenever it requires to connect.

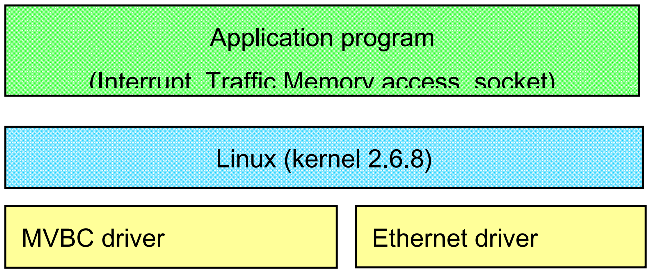


Fig. 8 Software platform for MVB error detector

5. Results

The proposed MVB error detector is implemented as shown in Figure 9. The MVB controller board is shown on the top and the ARM board is placed underneath it. The MVB error detector is connected between a master and a slave MVB node as shown in Figure 5 and can capture the MVB frames that move from the master to the slave and from the slave to the master.

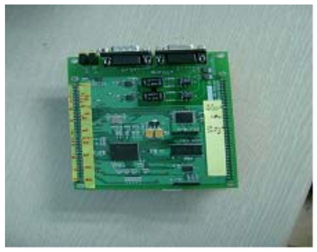
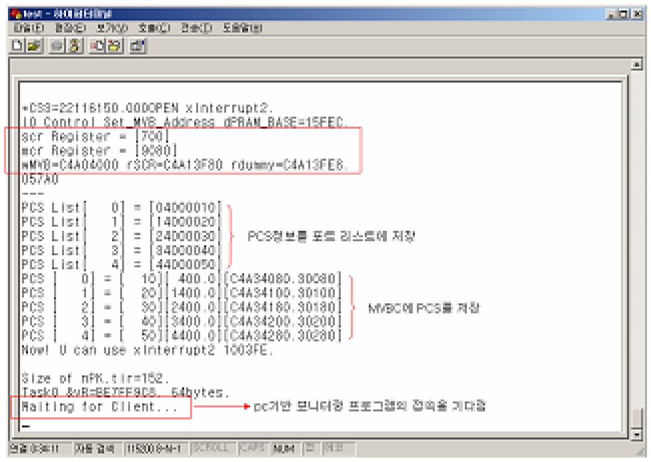


Fig. 9 Implemented MVB error detector

After booting the detector, the initialization process is shown in Figure 10. The MVB error detector starts to listen to both the bus and the Ethernet port. The initialization process is also shown by using the PCS registers' information.



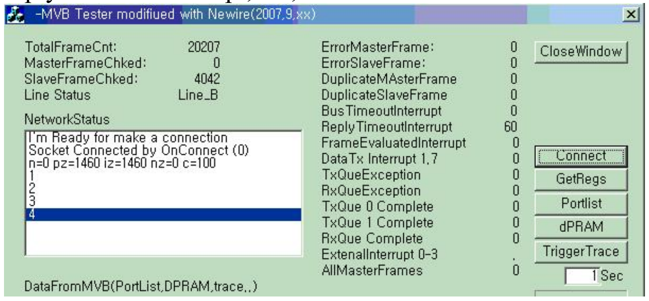
```
*CS3=22116150.000OPEN xInterrupt2.
IO Control Set MVB Address dPRAM_BASE=15FEC.
scr Register = [700]
mcr Register = [9080]
wMVB=C4A04000 rSCR=C4A13F80 rdummy=C4A13FE8.
057A0
---
PCS List[   0] = [04000010]
PCS List[   1] = [14000020]
PCS List[   2] = [24000030]
PCS List[   3] = [34000040]
PCS List[   4] = [44000050]
PCS [    0] = [   10][ 400.0][C4A34080.30080]
PCS [    1] = [   20][1400.0][C4A34100.30100]
PCS [    2] = [   30][2400.0][C4A34180.30180]
PCS [    3] = [   40][3400.0][C4A34200.30200]
PCS [    4] = [   50][4400.0][C4A34280.30280]
Now! U can use xInterrupt2 1003FE.

Size of nPK.tir=152.
Task0 &vR=BE7FE9C8. 64bytes.
Waiting for Client...
_
```

PCS정보를 포트 리스트에 저장
MVBC에 PCS를 저장
pc기반 모니터링 프로그램의 접속을 기다림

Fig. 10 View of MVB error detector initialization

Figure 11 shows the MVB error analysis program working on Windows with approximately 140KB binary application code. It is a simply GUI based socket program as shown in Figure 11. It can decide whether the frames are master or slave frames from the information for the F_Code. The error analysis program can tell the error analysis results, such as total frame, count, frame checked, reply timeout interrupt, etc, as listed in Table 1.



```
-MVB Tester modifiued with Newire(2007,9,xx)

TotalFrameCnt:          20207     ErrorMasterFrame:          0
MasterFrameChked:           0     ErrorSlaveFrame:           0
SlaveFrameChked:         4042     DuplicateMAsterFrame       0
Line Status            Line_B     DuplicateSlaveFrame        0
                                  BusTimeoutInterrupt        0
NetworkStatus                     ReplyTimeoutInterrupt     60
I'm Ready for make a connection   FrameEvaluatedInterrupt    0
Socket Connected by OnConnect (0) DataTx Interrupt 1,7       0
n=0 pz=1460 iz=1460 nz=0 c=100    TxQueException             0
1                                 RxQueException             0
2                                 TxQue 0 Complete           0
3                                 TxQue 1 Complete           0
4                                 RxQue Complete             0
                                  ExtenalInterrupt 0-3       .
                                  AllMasterFrames            0
DataFromMVB(PortList,DPRAM,trace,,)
```

Fig. 11 MVB error analysis program

6. Conclusion

After investigating about MVB characteristics, an error detection and analysis system is designed and implemented. The error analysis system can show all the error information referred by IEC 61375 about PCS internal register value, error status, frame status, and link status after capturing MVB frames.

This implementation contributes to the management and maintenance of the MVB node. It can offer to the maintenance people a user-friendly way to monitor all the equipment on a vehicle.

Before this research, the error analysis was based on the RS-232C serial interface, which limits the speed of analysis. With the Ethernet interface, the error analysis can capture all the frames on the line now.

For future work, we plan to develop MVB protocol analysis system with the automatic configuration because maintenance staff does not want to spend time installing or configuring anything in order to monitor their systems. They just want to perform maintenance.

Acknowledgments

This work was supported by the GRRC SUWON 2010-B5 program of Gyeonggi province. The advice received from K. H. Shon of Intercon system ltd is highly appreciated.

References

[1] G.Fadin and F.Cabaliere, "IEC TC9 WG22 train communication network dependability and safety concepts, "World Congress on Railway Research 97, 1997.
[2] H.Kirrmann and P.A. zuber, "IEC/IEEE train communication network." 1996.
[3] IEC 61375-1 Standard Train Communication Network: Part (1) General Architecture (2) Real-time Protocol (3) Multifunction Vehicle Bus (4) Wire Train Bus (5) Train Network Management (6) Train Communication Conformance Testing, 1999.
[4] UIC 556 Standard, Information Transportation on the Train Bus, 1999.
[5] H. Kirrmann and P. A. Zuber. The IEC/IEEE Train Communication Network, IEEE Micro, 21(2):81–92, March/April 2001.
[6] Philip Koopman, " Analysis of the Train Communication Network Protocol Error Detection Capabilities", Technical Report, Carnegie Mellon University, Pttsburgh, PA, USA, Feb. 2001.
[7] ABB Daimler-Bensz Transportation (Switzerland) Ltd, "Multifunction vehicle Bus Controller," Adteanz, 1997.
[8] Thomas Nolte, "Share-Driven Scheduling of Embedded Networks," Malardalen University Press Dissertations No. 26, May, 2006.

Su Goog Shon (corresponding author) received his B.S. (1982) degree in Electrical Engineering from Seoul National University, his M.S. (1984) degree in Electrical Engineering from Seoul National University, and his Ph.D. (1996) degree in Electrical and Computer Engineering from the University of Texas at Austin. He is an assistant professor in the Department of Information and Telecommunication at the University of Suwon in Korea. His research interests include computer and embedded system, network protocol, network simulation, and network programming.

Soomi Yang received the B.S., M.S. and Ph.D. degrees in computer engineering from Seoul National University of Seoul, Korea, in 1985, 1987 and 1997 respectively. From 1988 to 2000, she was a researcher at Korea Telecom Research Center where she worked on telecommunication network, internet and information security. From 2000 to 2001, she was a visiting scholar at UCLA, USA. From 2002 to 2004, she was a faculty of the Suwon Science College. Since 2004, she has been on the Faculty of the University of Suwon, Korea, where she is a professor of computer sciences. Her research interests in information security include access control, network security, and secure system software.