

CANTALKER™

Multi-CAN Analyzer & Converter

User's Manual

(CAN-USB / CAN-Ethernet Analyzer included)



D&K Information Communication Technology

<http://www.dnkict.com>

Revision History

Revision	Comment	Date	Author	Approver
1.00	Initial release	2011/1/28		
1.01	Bus Statistics Added	2011/1/30		
1.02	CAN Send Message Tool Modified	2011/2/1		
1.03	Monitor Graph Added	2011/2/6		
1.04	Firmware Upgrade Function Added	2011/2/10		
1.05	Toolbar added for CSM and Monitor	2011/2/21		
1.06	CAN-USB Analyzer Added	2011/3/14		
1.07	Appendix added for application note	2011/5/2		
1.1	Monitoring Modified and ECU Simulator Added	2012/8/15		
1.12	Updated ECU Simulator	2012/10/5		
1.13	Updated Monitor	2013/3/7		
1.15	J1939 Protocol Supported	2013/3/24		
1.16	Filter Modified	2013/10/20		
1.17	OBD-II Protocol Dialog Upgraded	2013/11/3		
1.18	Feature updated	2014/1/12		
1.2	CAN2Ethernet Analyzer Added	2014/2/25		
1.3	High-Level USB Win32 API Added	2014/3/23		
1.4	C Interpreter Added	2015/6/27		
1.42	Monitor Data Format and USB Port Selection added	2015/7/25		
1.43	Unlock all extended protocol analysis features.	2020/1/8		

Table of Contents

1 Introduction	9
1.1 Purpose.....	9
1.2 Scope.....	9
1.3 Definition, acronyms, and abbreviations.....	9
1.4 References.....	10
2 Overview	11
2.1 Features.....	11
2.2 Performance.....	12
2.3 Hardware.....	13
2.3.1 Multi-CAN Analyzer & Converter Board.....	13
2.3.2 Interfaces.....	16
3 Software & SDK	19
3.1 Multi-CAN Analyzer & Converter Windows Application, CANTALKER.....	19
3.2 USB Device Driver.....	19
3.3 CAN Tester Application (RS232/USB용, Ethernet용).....	21
3.4 Converter Active-X Control.....	22
3.5 Microchip CAN Bit time calculator Windows Application.....	22
3.6 Low-Level USB Interface API.....	23
3.6.1 BOOL LoadMpUsbAPIDll (HWND hwnd).....	23
3.6.2 BOOL OpenPipe (void).....	24
3.6.3 void ClosePipe (void).....	24
3.6.4 BOOL SendPacket (BYTE *data, UINT32 *pSize, UINT32 timeOut).....	24
3.7 High-Level USB & RS232 Interface API.....	24
3.7.1 int DNK_InitDLL (void).....	25
3.7.2 int DNK_GetDeviceCount (void).....	25
3.7.3 int DNK_InitUSBDriver (int DeviceHandle).....	25
3.7.4 int DNK_InitRS232Driver (int port, int baudrate).....	25
3.7.5 int DNK_CloseDriver (void).....	26
3.7.6 int DNK_CloseDLL (void).....	26
3.7.7 int DNK_SetCanInit (void).....	26
3.7.8 int DNK_SendCanData (CAN_FRAME *canFrame).....	26
3.7.9 int DNK_GetTotalRcvCanDataCount (void).....	26
3.7.10 int DNK_ClearRcvCanaData (void).....	26
3.7.11 int DNK_GetRcvCanData (CAN_FRAME *can)	26
3.7.12 int DNK_SetClearCanError (void).....	27
3.7.13 int DNK_ReqCanStatus (CAN_STATUS *status).....	27
3.7.14 int DNK_ReqErrorCount (UINT8 *txErrCnt, UINT8 *rxErrCnt).....	27
3.7.15 int DNK_SetCanID (BYTE extMode, UINT32 id).....	27
3.7.16 int DNK_SetCanBaudRate (UINT32 brate).....	27
3.7.17 int DNK_ReqDeviceInfo (DEVICE_INFO *devInfo).....	27
3.7.18 int DNK_SetRcvMode (UINT32 mode).....	28
3.7.19 int DNK_ReqRcvMode (UINT32 *rcvMode).....	28
3.7.20 int DNK_ReqCanFilters (FILTER_INFO *filterInfo).....	28
3.7.21 int DNK_ReqVersion (UINT8 *major, UINT8 *minor).....	28
3.7.22 int DNK_ReqSpeedInfo (UINT32 *canSpeed, UINT8 *rs232SpeedIdx).....	28
3.7.23 int DNK_ReqTcplpInfo (TCPIP_INFO *plInfo).....	28
3.7.24 int DNK_ReqCurTimeStamp (UINT32 *timeStamp).....	29
3.7.25 int DNK_SetCanBitTiming (BRGCON *brgcon).....	29

3.7.26 int DNK_SetFilters (FILTER_INFO *filterInfo).....	29
3.7.27 int DNK_SetTcplpInfo (TCPIP_INFO *pInfo).....	29
3.7.28 int DNK_SeRs232Baudrate (UINT8 baudrate).....	29
3.7.29 int DNK_CloseC2EConnection (void).....	29
4 Packet Mode.....	30
4.1 Overview.....	30
4.2 Physical Setup.....	30
4.3 Handshake.....	31
4.4 General Packet Format.....	31
4.5 Packet Formats.....	34
4.5.1 Set CAN Initialize (0x80).....	34
4.5.2 Set CAN ID (0x81).....	34
4.5.3 Set CAN Bitrate (0x82).....	34
4.5.4 Set RS232 Baudrate (0x83).....	35
4.5.5 Send CAN Message (0x84).....	36
4.5.6 Receive CAN Message (0x85).....	36
4.5.7 Request CAN Status (0x86).....	37
4.5.8 Response CAN Status (0x87).....	37
4.5.9 Indicate CAN Error (0x88).....	38
4.5.10 Set CAN Bitrate Parameters (0x89).....	38
4.5.11 Clear CAN Errors (0x8A).....	39
4.5.12 Set CAN Filters (0x8B).....	39
4.5.13 Request Version (0x8C).....	40
4.5.14 Response Version (0x8D).....	40
4.5.15 Close C2E Connection (0x90).....	40
4.5.16 Request CAN Filters (0x91).....	41
4.5.17 Response CAN Filters (0x92).....	41
4.5.18 Request Device Information (0x93).....	41
4.5.19 Response Device Information (0x94).....	41
4.5.20 Request Speed Information (0x95).....	42
4.5.21 Response Speed Information (0x96).....	42
4.5.22 Set TCP/IP Information (0x97).....	42
4.5.23 Request TCP/IP Information (0x98).....	43
4.5.24 Response TCP/IP Information (0x99).....	43
4.5.25 Request Time Stamp (0x9A).....	43
4.5.26 Response Time Stamp (0x9B).....	43
4.5.27 Receive CAN Message with time stamp (0xA0).....	43
4.5.28 Set CAN Receive Mode (0xA1).....	44
4.5.29 Request CAN Receive Mode (0xA2).....	44
4.5.30 Response CAN Receive Mode (0xA3).....	44
4.5.31 Request CAN Error Count (0xA4).....	44
4.5.32 Response CAN Error Count (0xA5).....	45
4.5.33 Acknowledge (0xC0).....	45
4.5.34 No Acknowledge (0xD0).....	45
5 Terminal Mode.....	46
5.1 Overview.....	46
5.2 How to enter terminal Mode.....	46
5.3 Terminal Commands.....	46
5.3.1 INI (CAN Initialize).....	46
5.3.2 EID (Extend ID Mode, 2.0B).....	46
5.3.3 SID (Set CAN ID).....	47
5.3.4 CBR (CAN Bit Rate).....	48
5.3.5 SCP (Set CAN Parameter).....	48
5.3.6 RBR (RS232 Baud Rate).....	48

5.3.7 RCV (CAN Message Receive).....	49
5.3.8 SND (CAN Message Send).....	49
5.3.9 STS (CAN Status).....	50
5.3.10 CLR (Clear CAN Error).....	50
5.3.11 SIP (Set IP Address).....	51
5.3.12 SSM (Set Subnet Mask).....	51
5.3.13 SGW (Set Gateway Address).....	51
5.3.14 SMA (Set MAC Address).....	51
5.3.15 SIM (Set IP Mode).....	52
5.3.16 SPT (Set C2E TCP Port No).....	52
5.3.17 CPW (Change Telnet Password).....	52
5.3.18 DTS (Display Time Stamp).....	53
5.3.19 VID (Change USB Vendor ID).....	53
5.3.20 PID (Change USB Product ID).....	53
5.3.21 EXT (Exit Terminal).....	53
6 Protocol Analyzer Application (CANTALKER).....	54
6.1 Overview.....	54
6.2 CAN Message List.....	54
6.3 Menu.....	55
6.3.1 File.....	55
6.3.2 Edit.....	56
6.3.3 Protocol.....	57
6.3.4 Interface.....	66
6.3.5 Trigger.....	68
6.3.6 Option.....	69
6.3.7 Help.....	69
6.4 Tool Bar.....	70
6.5 Status Bar.....	70
7 C Interpreter.....	71
7.1 Overview.....	71
7.2 Limitations.....	71
7.3 Console Keys and Commands.....	72
7.3.1 Keys.....	72
7.3.2 Commands.....	72
7.4 Integrated Function List.....	73
7.4.1 Standard Input / Output Functions.....	73
7.4.2 Standard Library Functions.....	75
7.4.3 Standard String Functions.....	76
7.4.4 Standard Time Functions.....	77
7.4.5 Standard Mathematics Functions.....	78
7.4.6 CANTalker High Level Functions.....	79
7.4.7 TCP/UDP Network Functions.....	80
7.4.8 Serial Communication Functions.....	81
8 Firmware Upgrade.....	83
8.1 Overview.....	83
8.2 Preparations.....	83
8.3 Procedure.....	83
9 Appendix1.	
Multi-CAN Analyzer를 이용한 차량 ECU 모니터링.....	85
9.1 들어가며.....	85
9.2 Multi-CAN Analyzer 소개.....	85
9.3 ECU 모니터링.....	87
9.3.1 OBD-II PID란?.....	87

9.3.2 ECU 모니터링을 위한 전송 메시지 설정..... 89
9.3.3 ECU 모니터링을 위한 수신 메시지 설정..... 90
9.3.4 실차량에서 모니터링 수행..... 91
9.3.5 맺음말..... 92
9.3.6 참고자료..... 93

Table of Tables

Table 1: Factory Default Value.....	16
Table 2: CAN Connector Pin Map.....	17
Table 3: RS232C Connector Pin Map.....	17
Table 4: PID List.....	33
Table 5: Terminal Mode Help.....	47
표 6. PID 목록 및 설명.....	87
표 7. Mode 별 설명.....	88

Table of Figures

Figure 1: Multi-CAN Analyzer Board Layout.....	13
Figure 2: CAN-USB Analyzer PCB Layout.....	14
Figure 3: CAN-Ethernet Analyzer PCB Layout.....	15
Figure 4: Multi-CAN Analyzer Front Panel.....	16
Figure 5: Multi-CAN Analyzer Rear Panel.....	17
Figure 6: CAN Error State Machine.....	38
Figure 7: Microchip CAN Bit Timing Calculator.....	39
Figure 8: Protocol Analyzer Application.....	54
Figure 9: Monitor CAN ID & Data.....	63
Figure 10: C Interpreter Console Window.....	71

1 Introduction

1.1 Purpose

본 문서는 Multi-CAN Analyzer (CANTALKER™) 제품의 개요, 하드웨어 및 소프트웨어에 대한 전반적인 사용 설명을 목적으로 한다

1.2 Scope

본 문서에서 다루는 부분은 다음과 같다.

1. 하드웨어 사양 및 설명
2. 소프트웨어 종류 및 설명
3. 패킷모드를 위한 통신 프로토콜 설명
4. 터미널 모드 사용법
5. Analyzer 응용프로그램 CANTALKER의 설명
6. Firmware Upgrade 설명

1.3 Definition, acronyms, and abbreviations

ACK	Acknowledge
ARP	Address Resolution Protocol
BRP	Baud Rate Prescaler
C2UA	CAN2USB Analyzer
C2EA	CAN2Ethernet Analyzer
CAN	Controller Area Network
DHCP	Dynamic Host Configuration Protocol
DMIPS	Drystone Million Instruction Per Second
DLL	Dynamic Linked Library
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
NACK	No Acknowledge

PID	Product Identification
PRSEG	Propagation Time Select
RTR	Remote Transmission Request
SAM	Sample of the CAN Bus Line
SEG1PH	Phase Buffer Segment 1
SEG2PH	Phase Buffer Segment 2
SEGPHTS	Phase Segment 2 Time Select
SJW	Synchronization Jump Width
TCP	Transmission Control Protocol
UART	Universal Synchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
VID	Vendor Identification
WAKFIL	CAN Bus Line Filter Enable

1.4 References

1. CAN to RS232C Converter Manual, Maeul Software
2. CAN to Ethernet Gateway Manual Maeul Software

2 Overview

기존 CAN Converter 제품은 전송속도가 매우 느린 RS232C 통신을 사용하거나 성능이 낮은 8비트 MCU를 사용하여 고속의 Ethernet 통신을 하더라도 CAN 버스의 모든 메시지를 처리 할 수 없었던 근본적인 문제를 안고 있어 Analyzer로 사용하기에는 다소 부족한 면이 있었고 특정 메시지들만 수신하고 처리하기 위한 제어기의 용도로 적합했다.

본 Analyzer는 이의 단점을 보완하기 위해 고성능 MCU를 사용하여 CAN 버스의 모든 메시지를 실시간으로 처리할 수 있게 하였으며, 휴대성과 활용성을 높이기 위해 기존 스위치로 모드 변경하는 제품과는 달리 언제든지 RS232C, USB 그리고 Ethernet 인터페이스를 동시에 사용 할 수 있게 했다. USB를 이용하는 경우는 외부 전원이 추가로 필요로 하지 않는다. 또한 CAN Bus의 절연(Isolation)으로 외부 전기적 충격으로부터 내부 회로를 보호할 뿐만 아니라 기존 안정성을 위해 제어기들의 전원 GND를 모두 연결해야 했던 번거로운 문제를 없앴다.

본 문서는 Multi-CAN Analyzer(MCA), CAN-USB Analyzer (이하 C2UA) 그리고 CAN-Ethernet Analyzer (이하 C2EA)에 대해 모두 설명하고 있으나, C2UA는 Ethernet 관련 부분이, C2EA는 USB 부분의 기능이 제외된다.

제품 모델로 C2UA 제품에는 MSC2U100, C2EA 제품에는 MSC2E100, 그리고 MCA 제품에는 MSMCA100 이 있다.

2.1 Features

본 Analyzer는 다음과 같은 특성 및 기능을 갖고 있다.

1. 시중에 나와 있는 대부분 CAN Converter 나 Analyzer는 고속의 USB를 사용함에도 성능이 달려 CAN 데이터가 고속으로 물리는 경우 데이터가 빠지거나 분석 프로그램등이 먹통되는 문제가 있어서 범용적으로 안정성 있게 사용 불가하나, 본 제품은 120 DMIPS로 동작하는 고성능 32 bits MCU내장하여 1 Mbps CAN 버스의 모든 메시지의 실시간 처리가 가능함.
2. RS232C, USB 그리고 Ethernet 인터페이스를 내장하여 CAN 프로토콜을 동시에 다양한 인터페이스로 변환하여 실시간 데이터 송수신 (Converter로서 사용가능, C2UA는 Ethernet없음, C2EA는 USB없음)
3. RS232C의 최대 통신속도는 460,800 bps 지원
4. 10/100 Mbps Ethernet 통신속도 지원하며 LAN cable 형식 자동인식
5. 1 Mbps High Speed CAN을 지원하며, CAN 2.0A와 2.0B 동시 지원함 (CAN 2.0 Active Mode)
6. CAN BUS가 Isolation 되어 외부 전기적 충격으로부터 보호됨
7. CAN 종단저항(Termination) 사용을 외부 스위치로 결정할 수 있음
8. MCP2551 / 82C250 호환 CAN Transceiver 사용
9. USB 2.0 Full Speed 지원 (C2EA 제외)
10. IP / ARP / ICMP / UDP / TCP / DHCPc / DHCPs / TELNET / C2E 프로토콜 지원 (C2UA 제외)

11. USB전원(C2EA제외) 또는 DC5V 외부 전원 사용 가능함
12. 공장초기화 점퍼 내장하여 모든 설정값을 출하시의 상태로 변경 할 수 있음
13. Power, Status, Error, CAN TX, CAN RX에 대한 LED지원 (C2UA는 Status없음, C2EA는 Power 없음)
14. Watchdog timer동작으로 비정상 동작시 자동 리셋됨
15. 설정값은 Nonvolatile 메모리에 저장되어 전원 off시에서 값 유지됨
16. 동작 전압 및 최대 전류는 DC5V / 400mA 이상
17. 제품크기 140 x 35 x 110 mm (가로x높이x깊이, C2UA / C2EA는 61 x 28 x 90)
18. 전용 Analyzer 응용프로그램 CANTALKER 제공
19. ECU Simulator / CANopen / OBD-II PID / J1939 / DeviceNet(일부지원)
20. 10us 해상도 Time Stamp지원
21. Filter / Trigger / 특정 CAN ID 모니터링 및 산술계산 지원 / 전송할 CAN 메시지 프로그래밍지원
22. 특정 메시지에 대한 실시간 Graph 그래픽 지원
23. 송수신 메시지 저장 및 텍스트 형식 Export 지원
24. Firmware Upgrade 지원
25. SDK 제공으로 VB/VC/C# 등에서 편리한 프로그래밍 환경 제공
26. C Interpreter 제공으로 CAN메시지의 입출력을 실시간 프로그래밍 할 수 있음

2.2 Performance

본 Analyzer는 USB / RS232C / Ethernet 인터페이스를 지원하고 Terminal 과 Packet모드를 지원한다. 단 한 가지 인터페이스에 대해서는 동시에 Terminal모드와 Packet모드를 지원하지 않고 둘중 하나만 사용할 수 있다. 모든 인터페이스는 동시에 모두 사용할 수 있으며 모든 인터페이스가 연결된 경우 수신된 CAN메시지는 동시에 모든 인터페이스로 전달될 수 있다. 또한 동시에 모든 인터페이스에서 CAN 메시지를 송신 할 수 있다.

RS232C의 경우 최대 460800bps의 속도를 갖지만 1Mbps의 CAN속도보다 낮고 Protocol Overhead로 인해 모든 데이터를 수신하지 못할 수 있다. USB와 Ethernet의 경우는 전송속도가 충분히 빠르므로 수신된 모든 CAN 메시지를 전송 받을 수 있다. 따라서 모든 메시지를 수신하고자 한다면 반드시 USB나 Ethernet을 사용해야 한다.

주의사항으로 RS232C의 baudrate를 115200bps 초과하여 사용하는 경우는 반드시 고속을 지원하는 USB2Serial Converter등을 사용해야 하며, 고속이 되는 경우는 전송 데이터에 오류가 발생할 확률이 높아질 수 있다는 것을 사전 인식하고 사용해야 한다.

2.3 Hardware

Analyzer를 사용하기 위해 다음과 같은 하드웨어 구성품을 필요로 한다.

(*주의. 본체 이외에는 상품에 포함되지 않으므로 개별 구매해야 함)

1. Multi-CAN Analyzer본체
2. USB mini cable
3. 1:1 RS232 cable
4. Ethernet cable
5. SMPS DC5V 1A 이상 Power Supply

2.3.1 Multi-CAN Analyzer & Converter Board

Figure 1 는 Multi-CAN Analyzer 내부에 있는 PCB 의 Layout을 보여 준다. 각 기구 부품별 설명은 아래와 같다.

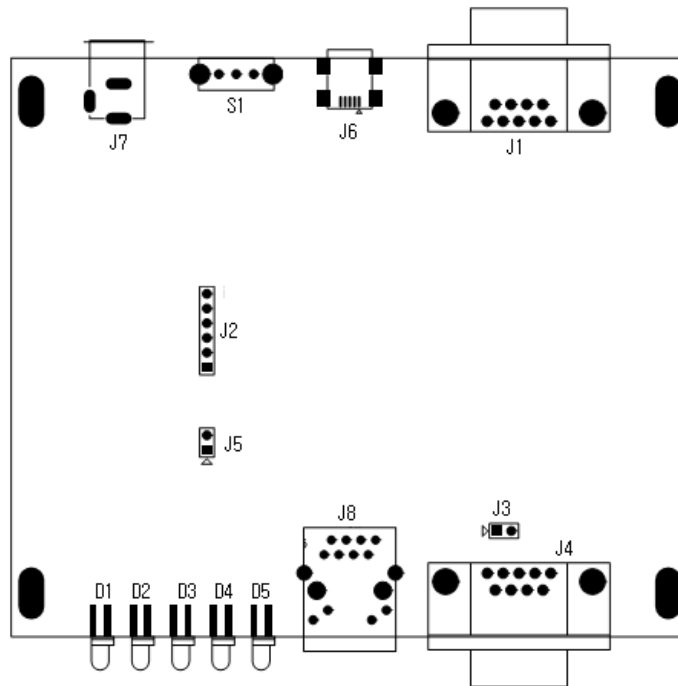


Figure 1: Multi-CAN Analyzer Board Layout

1. D1 : Power LED로 전원이 인가되면 점등됨
2. D2 : Error LED로 CAN통신 Bus Off인 경우 점등됨
3. D3 : Status LED로 IP를 할당중이면 점멸되고 할당 완료되면 점등됨
4. D4 : CAN TX LED

- 5. D5 : CAN RX LED
- 6. J1 : RS232C DB9 Female Connector
- 7. J2 : ICSP Connector로 펌웨어 업그레이드 또는 Debugging에 사용됨
- 8. J3 : CAN 종단저항 점퍼이나 사용되지 않음
- 9. J4 : CAN DB9 Male Connector
- 10. J5 : 공장초기화 점퍼로 공장출하시 설정값으로 초기화 함
- 11. J6 : USB mini Connector
- 12. J7 : DC5V Jack
- 13. J8 : RJ45 LAN Connector
- 14. S1 : CAN 종단저항 스위치

Figure 2 은 CAN-USB Analyzer 내부 PCB의 Layout을 보여준다. 각 기구별 설명은 아래와 같다.

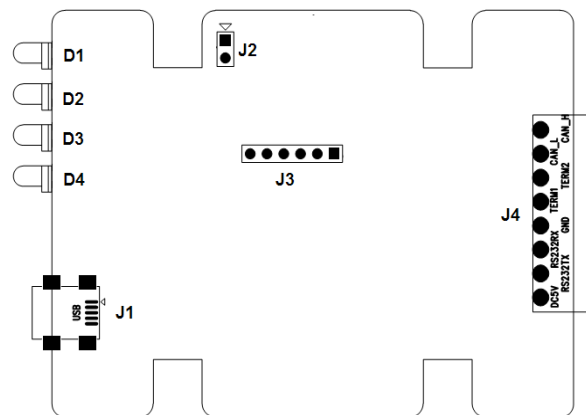


Figure 2: CAN-USB Analyzer PCB Layout

- 1. D1 : Power LED 로 전원이 인가되면 점등됨
- 2. D2 : Error LED로 CAN통신 Bus Off인 경우 점등됨
- 3. D3 : CAN TX LED
- 4. D4 : CAN RX LED
- 5. J1 : USB mini Connector
- 6. J2 : 공장초기화 점퍼로 공장출하시 설정값으로 초기화 함
- 7. J3 : ICSP Connector로 펌웨어 업그레이드 또는 Debugging에 사용됨
- 8. J4 : CAN / RS232C / DC5V 및 CAN 종단 저항 연결 터미널
CAN 종단저항을 사용하기 위해서는 TERM1과 TERM2를 연결함
RS232_RX 터미널은 PC의 9pin RS232C컨넥터의 2번, RS232C_TX는 3번, GND는 5번 pin

과 연결함

Figure 3 은 CAN-Ethernet Analyzer 내부 PCB의 Layout을 보여준다. 각 기구별 설명은 아래와 같다.

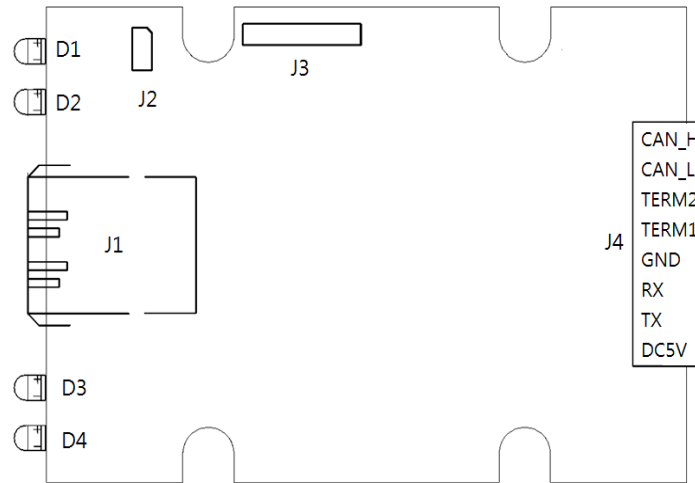


Figure 3: CAN-Ethernet Analyzer PCB Layout

1. D1 : Status LED로 IP를 할당중이면 점멸되고 할당 완료되면 점등됨
2. D2 : Error LED로 CAN통신 Bus Off인 경우 점등됨
3. D3 : CAN TX LED
4. D4 : CAN RX LED
5. J1 : RJ-45 LAN Connector
6. J2 : 공장 초기화 점퍼로 공장출하시 설정값으로 초기화 함
7. J3 : ICSP Connector로 펌웨어 업그레이드 또는 Debugging에 사용됨
8. J4 : CAN / RS232C / DC5V 및 CAN 종단 저항 연결 터미널

CAN 종단저항을 사용하기 위해서는 TERM1과 TERM2를 연결함

RS232_RX 터미널은 PC의 9pin RS232C컨넥터의 2번, RS232C_TX는 3번, GND는 5번 pin

과 연결함

MCA의 J5 나, C2UA / C2EA의 J2 를 연결하여 출하시 값으로 초기화 하는 경우 설정값은 다음과 같다. 아래 내용중 Ethernet과 관련된 부분은 C2UA와 상관 없고, USB와 관련된 부분은 C2EA와 상관없다.

Model	MSMCA100
Serial No	10129001
Version	1.0
USB VID/PID	04D8/FFEE
CAN_ID	FFFFFFFF (All Accepted)
CAN BaudRate	1000 kbps
CAN Param	00/00/00
RS232 BaudRate	115200 bps
Error Warning	0
Rx/Tx Warning	0/0
Rx/Tx Bus passive	0/0
Bus off	0
Overrun	0 (HW) :0 (RX FIFO) :0 (TX FIFO)
Error Count	0 (RX) /0 (TX)
Extend ID mode	0
Fifo Overrun	USB (RX:0/TX:0) , UART (RX:0/TX:0)
Mask1,2	00000000,00000000
Filter1-2	00000000,00000000
Filter3-6	00000000,00000000,00000000,00000000
IP Address	192.168.10.100
Subnet Mask	255.255.255.0
Gateway	192.168.10.1
MAC Address	00-04-A3-14-6E-1C
Port Number	D431
IP mode	Static
CAN Time Stamp	Enable

Table 1: Factory Default Value

2.3.2 Interfaces

Figure 4와 Figure 5는 각각 Analyzer의 앞면과 뒷면의 판넬 이미지를 표시한다. LED의 색깔은 Power, CAN Tx/ CAN Rx는 녹색, Error는 빨간색, Status는 주황색이나 색은 변경 될 수 있다.

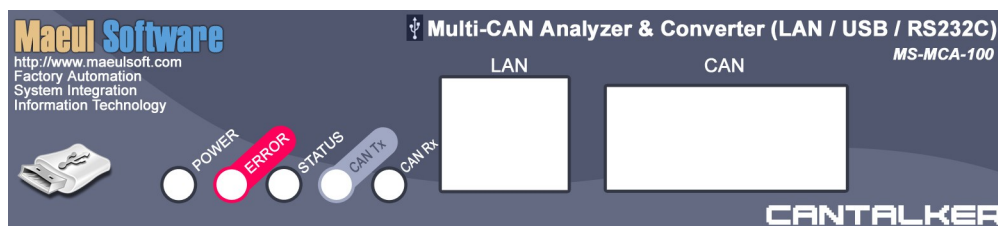


Figure 4: Multi-CAN Analyzer Front Panel

DB9 Male 형식 CAN 컨넥터의 핀배열은 다음과 같다. GND는 CAN Transceiver에 공급되는 전원의 GND로

특별한 경우가 아니라면 연결할 필요가 없다.

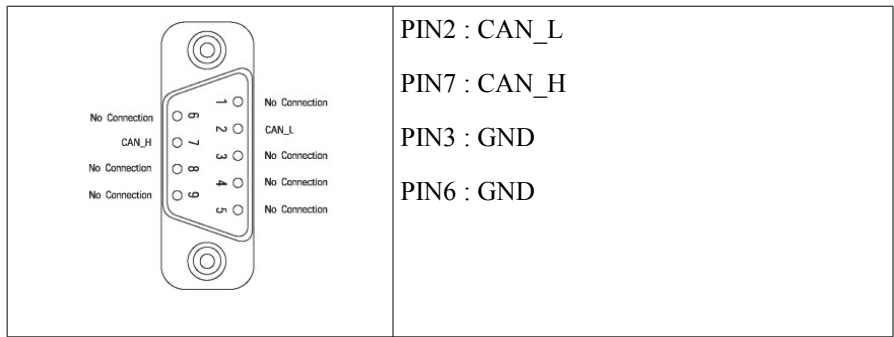


Table 2: CAN Connector Pin Map

Figure 5의 TERM은 CAN의 120 Ohm 종단저항을 연결하기 위한 스위치로 [ON] 방향으로 이동시 종단저항이 연결된다. 일반적으로 종단저항은 CAN 버스의 양끝단에 위치하는 노드만 연결하므로 상황에 따라 ON/OFF 해야 한다.

RS232C 컨넥터에 연결하는 케이블은 RX와 TX라인이 Cross가 아닌 1:1 형식을 반드시 사용해야 한다. 보통 1:1 케이블은 한쪽은 Male, 다른쪽은 Female 로 되어 있다.



Figure 5: Multi-CAN Analyzer Rear Panel

DB9 Female형식 RS232C컨넥터의 핀배열은 다음과 같다.

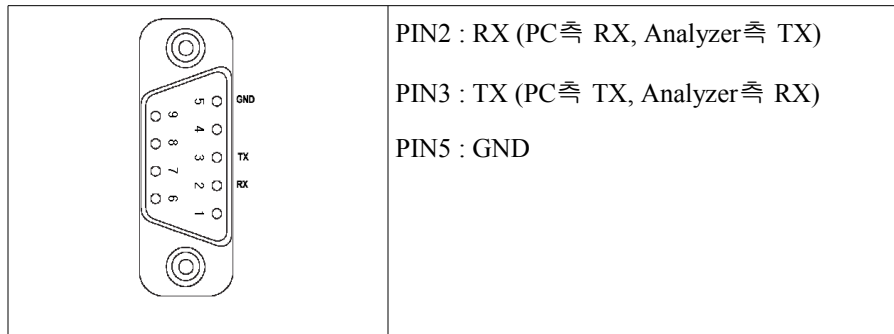


Table 3: RS232C Connector Pin Map

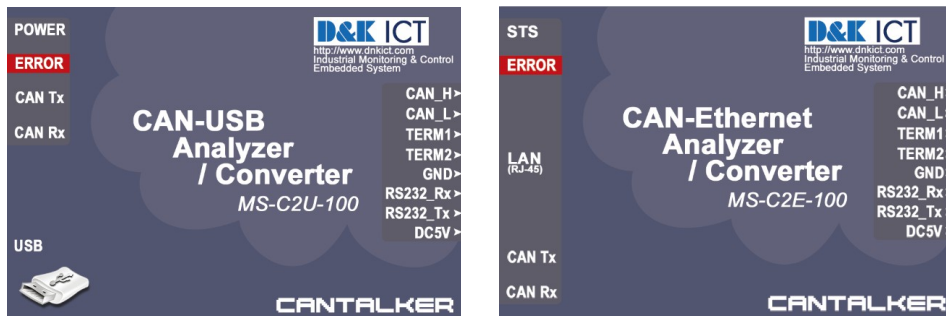
RS232C 인터페이스를 터미널 모드로 사용하는 경우 하이퍼 터미널과 같은 시리얼 터미널 응용프로그램을 사용할 수 있는데 가급적 하이퍼터미널은 사용하지 말며, Teraterm, 이야기, Putty, 새롬데이터맨 같은 프로그램을

을 이용한다. 이유는 하이퍼터미널은 가끔 정상적인 통신이 되지 않는 문제가 발생하기 때문이다.

DC5V의 경우는 극성과 전압을 반드시 사용전에 확인해야 한다. 만약 비정상적인 입력이 인가되는 경우 보드의 부품에 손상이 발생할 수 있으며 이런 경우는 당사의 책임 범위를 벗어난다. 전원은 항상 동일 전압을 유지 하는 정전압 어댑터를 사용해야 하며 최소 지원되는 전류는 500mA 이상 되어야 한다. 경우에 따라 오래된 PC 나 USB허브를 사용하는 경우 USB에서 공급되는 전류가 모자라는 경우가 있을수 있으니, 이런 경우는 외부전원을 사용하는 USB허브를 사용하는게 좋다.

지원되는 USB의 규격은 2.0이며 속도는 Full Speed로 최대 12Mbps 이다. 통신속도가 최대 1Mbps인 CAN버스의 모든 메시지는 RS232C를 제외하고 USB와 Ethernet 인터페이스를 사용하는 경우에 처리될 수 있으므로 Analyzer로서 사용되는 경우 가급적 USB나 Ethernet을 사용해야 한다.

C2UA의 경우 케이스 윗면에 보여지는 이미지는 아래 좌측 그림과 같고 C2EA는 오른쪽 그림과 다음과 같다. LED색깔은 Power/CAN RX/CAN TX는 녹색, ERROR는 빨간색 그리고 STS(Status)는 노란색 이며 변경될 수도 있다.



3 Software & SDK

Analyzer를 사용하는데 지원되는 소프트웨어에 대해 본절에서 설명된다. 지원되는 모든 응용프로그램들은 Windows OS (2000/XP/Vista/Win7 32/64bit)만 지원하며, RS232C 터미널 모드나, Telnet은 관련 기능이 지원하는 모든 OS에서 사용할 수 있다. USB인터페이스를 사용하려면 반드시 드라이버를 설치해야 하는데 자세한 설명은 3.2 절에서 설명된다.

3.1 Multi-CAN Analyzer & Converter Windows Application, CANTALKER

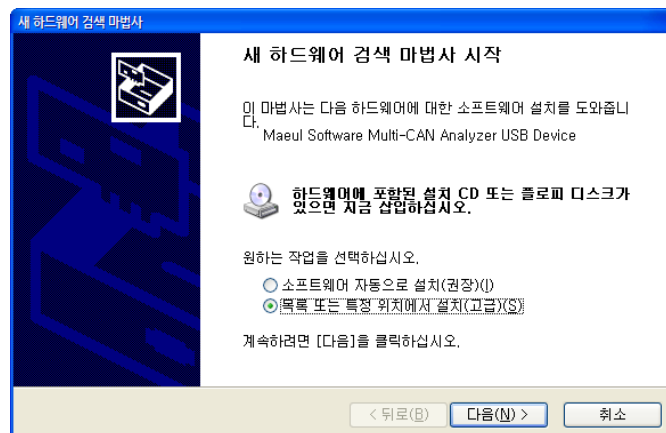
CAN 버스 모든 메시지를 모니터링 하거나 프로토콜을 분석할 때 사용되는 Windows 응용프로그램으로 6장에서 자세히 설명된다.

3.2 USB Device Driver

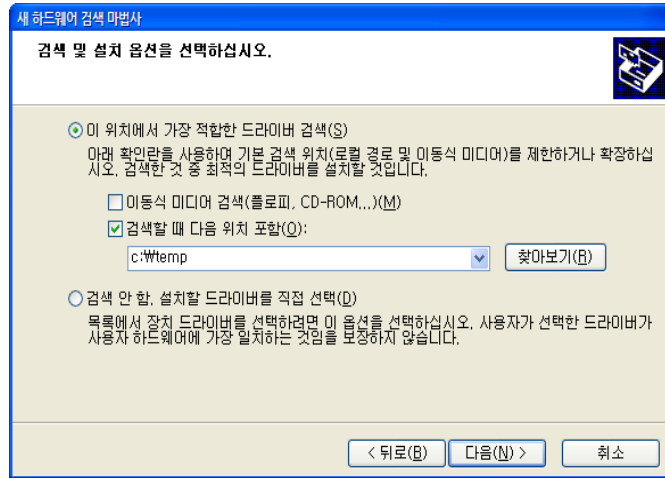
USB 인터페이스를 사용하는 경우 드라이버는 반드시 설치되어야 한다. 드라이버를 위해 필요한 파일은 다음과 같다.

1. mchpubs.sys : 32비트 용
2. mchpubs64.sys : 64비트 용
3. mchpubs.inf

드라이버를 설치하기 위해 위 3개 파일을 PC의 특정 디렉토리에 복사하고 PC에 Analyzer를 연결하면 다음과 같은 팝업창이 보이며 두번째 항목인 “목록 또는 특정 위치에서 설치(고급)(S)”를 선택하고 “[다음]”을 클릭한다.



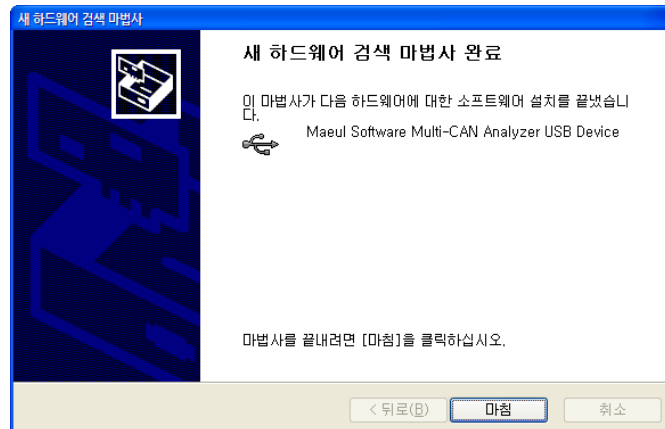
그럼 다음과 같은 창이 보이며 “검색할 때 다음 위치 포함”을 체크한 후 이전에 복사해 두었던 드라이버가 있는 위치의 디렉토리를 “[찾아보기]”를 클릭하여 설정한 후 “[다음]”을 클릭한다.



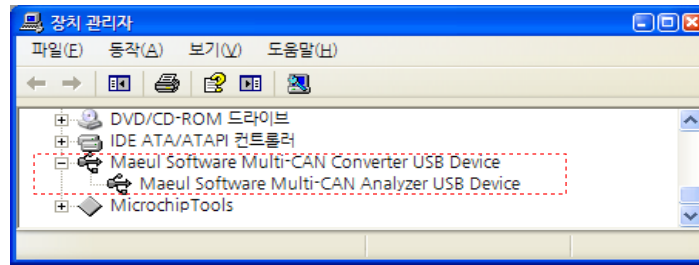
다음과 같은 화면이 보이면서 드라이버가 시스템으로 복사되면서 설치된다.



정상적으로 설치가 완료 되면 다음과 같은 화면을 볼 수 있으며 “[마침]” 을 클릭하여 완료한다.



드라이버 설치가 완료 되고 [장치관리자]를 띄우면 다음과 같이 드라이버가 설치된 모습을 볼 수 있다.

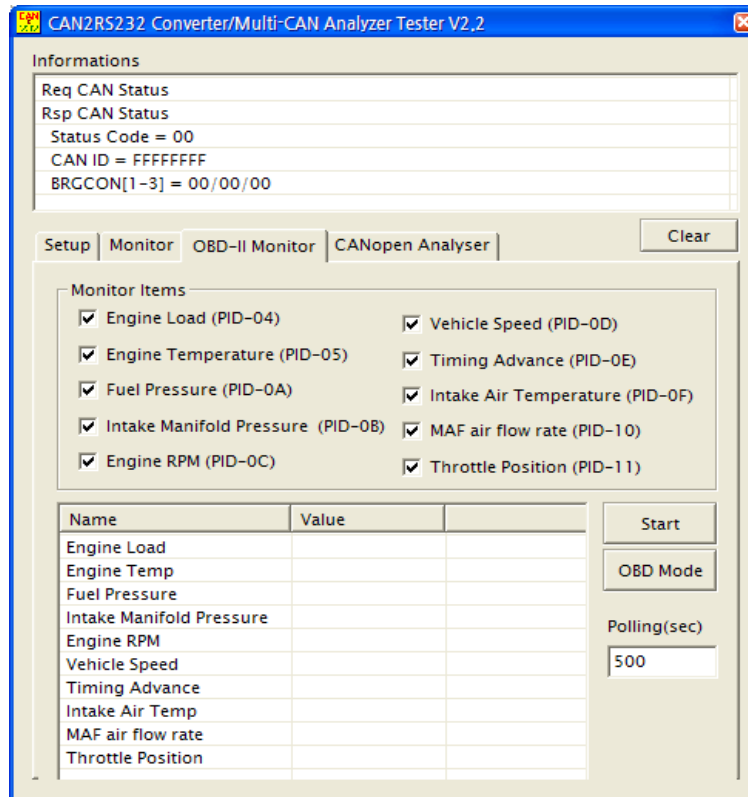


USB드라이버는 Enumeration시에 사용되는 Control을 포함하여 Bulk In / Bulk Out 두개의 Endpoint를 사용한다. 실제 데이터 통신용으로는 Control Endpoint는 사용하지 않는다.

3.3 CAN Tester Application (RS232/USB용, Ethernet용)

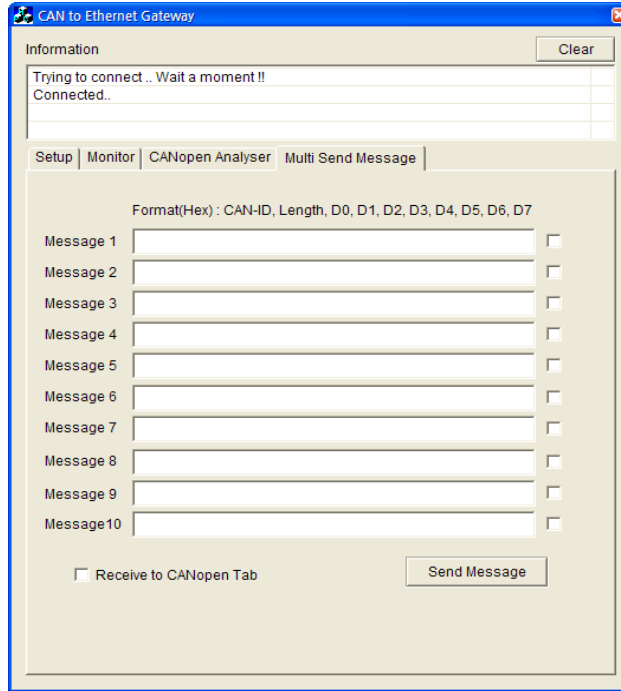
Analyzer의 동작을 테스트하거나 간단한 모니터링이나 설정을 위해 Tester 응용 프로그램을 사용할 수 있으나 프로토콜 분석용으로는 6장에서 설명되는 Protocol Analyzer 응용프로그램을 사용해야 한다.

Tester는 인터페이스 종류에 따라 두개의 응용프로그램을 구분해서 사용해야 한다. RS232와 USB 인터페이스를 사용하는 경우는 CAN2RS232 Tester 응용프로그램 사용해야 하며 다음과 같다.



본 응용프로그램은 간단한 기본적인 설정과 모니터링 기능을 포함하고 있으며, OBD나 CANopen 프로토콜에 대해 간단히 구현되어 있다. 자세한 사용법은 CAN2RS232 컨버터 매뉴얼을 참고한다.

Ethernet인터페이스를 사용하는 경우는 “CAN to Ethernet Gateway Tester” 응용프로그램을 사용해야 하며 다음과 같다.



본 프로그램도 CAN2RS232 Tester 응용 프로그램과 동일할 기능을 수행하며 추가적으로 10개의 메시지를 등록하여 보낼수 있는 기능을 지원한다. 자세한 설명은 “CAN to Ethernet Gateway” 사용 설명서를 참고한다.

3.4 Converter Active-X Control

패킷모드와 RS232통신을 지원하는 Active-X 컨트롤의 파일이름은 “can2rs232.ocx” 파일이다. 본 OCX를 이용하는 경우 VisualBasic에서 쉽게 컨버터 기능을 활용할 수 있다. 자세한 등록 및 사용법은 “CAN to RS232 Converter” 매뉴얼을 참고한다.

3.5 Microchip CAN Bit time calculator Windows Application

CAN 속도를 사용자 정의로 설정한 경우에 CAN의 Bit Timing 파라미터 값인 BRGCON1~BRGCON3를 계산하기 위한 응용프로그램으로 “CAN to RS232C Converter” 매뉴얼을 참고한다.

본 프로그램은 사용하기 위해서 미리 설치되어야 하며 설치 파일은 다음과 같이 구성되어 있다.

1. setup.exe

2. setup.lst
3. ics2510BitTime.CAB

3.6 Low-Level USB Interface API

사용자가 직접개발하는 응용프로그램에서 USB인터페이스를 사용하는 경우에 드라이버와 통신하는 기능을 API함수로 제공하는 DLL을 사용해야 한다. 사용자가 좀더 사용하기 쉽도록 class로 재정의 하였고 전체적으로 다음과 같은 파일로 구성된다. 단 본 모드는 단순 입출력을 위한 저수준의 API만을 제공하기 때문에 사용자가 직접 패킷모드에 대한 프로토콜을 구성과 해석에 대해 코딩해야 한다. 좀더 편리한 고수준의 API는 다음절에서 설명된다.

1. MpUsbAPI.h : class를 정의한 헤더 파일
2. MpUsb.h : DLL의 API를 정의한 헤더 파일
3. MpUsbAPI.cpp : class를 구현한 cpp파일
4. MpUsbApi.dll : 드라이버와 직접 통신하는 DLL파일
5. type.h : Analyzer에 사용되는 데이터 형 및 정의가 선언된 헤더 파일

사용자가 사용해야 할 class의 멤버함수와 설명은 다음과 같다.

3.6.1 BOOL LoadMpUsbAPIDll (HWND hwnd)

- DLL을 로딩하고 초기화 하는 멤버함수며 USB로부터 데이터를 수신하면 hwnd 핸들을 갖는 윈도우로 Window Message를 전송하기 때문에 본 함수를 호출할 때 반드시 수신된 데이터를 처리할 윈도우의 핸들을 전달해야 한다. 본 함수가 FALSE를 리턴하는 경우는 “MpUsbApi.dll” 파일을 여는데 실패한 경우이므로 파일이 현재 실행파일이 있는 디렉토리나 System32 디렉토리에 있는지 확인 한다.
- 전달되는 Window Message의 정의는 다음과 같다.

```
#define WM_CV_RSP_SIGNAL (WM_USER+3)
```

- Message의 WPARAM은 수신 데이터의 size이고, LPARAM은 수신데이터가 보관된 할당된 메모리 주소로 본 메시지를 받는 윈도우에서는 반드시 할당 메모리를 직접 free시켜줘야 한다. 다음은 전형적인 수신 메시지 함수의 형태다.

```

void OnUsbDataReceived (WPARAM wparam, LPARAM lparam)
{
    BYTE    *data = (BYTE*)lparam;
    UINT32  size = (UINT32)wparam;
    BYTE    buff[4096];

    if (data)
    {
        memcpy (buff, data, size);
        free (data);
        ProtocolRxHandler (buff, size);
    }
}

```

3.6.2 BOOL OpenPipe (void)

- USB드라이버를 열고 USB통신을 위한 Bulk In/Out Pipe에 대한 핸들을 얻기 위한 함수이다.
- 본 함수가 FALSE를 리턴하면 USB드라이버가 정상적으로 인식되지 않는 상황이므로 USB연결이 안되었는지와 드라이버가 설치되지 않았는지 확인한다.

3.6.3 void ClosePipe (void)

- Open한 드라이버를 더이상 사용하지 않아 종료하는 경우에 사용된다.

3.6.4 BOOL SendPacket (BYTE *data, UINT32 *pSize, UINT32 timeOut)

- USB로 데이터를 전송할 때 사용된다.
- 변수의 의미는 다음과 같다.
 - data는 전송할 데이터의 포인터
 - pSize는 전송할 크기가 들어있는 변수의 포인터이며 전송이 완료되면 실제 전송된 크기를 전달
 - 본 함수는 데이터 전송을 완료하기 전까지 block상태로 있어서 전송 유효기간인 timeOut 시간 (ms 단위)을 주어 지정된 시간안에 완료되지 않으면 block상태에서 즉시 깨어남

3.7 High-Level USB & RS232 Interface API

VB/VC/C# 등 다양한 언어에서 프로그램을 작성하는 경우 저수준의 DLL을 사용하면 사용자가 원하는 모

든 기능에 대해 매우 세밀하게 직접 제어할 수 있지만 사용자가 프로그래밍에 많은 노력을 들여야 한다. 이런 문제로 사용자가 매우 쉽고 편리하게 프로그래밍을 하기 위해 Analyzer와 입출력을 위한 프로토콜 생성과 해석에 대해 DLL에서 수행하고 사용자는 결과만 이용할 수 있도록 고수준의 API를 제공한다. 모든 함수의 리턴값은 다음과 같다. 0 이상의 양의 수는 에러가 아닌 함수의 전달값이다.

```
typedef enum
{
    ERR_OK = 0,
    ERR_DLL_NOT_LOADED = -1,
    ERR_DLL_NOT_EXIST = -2,
    ERR_DLL_LOAD_FAILED = -3,
    ERR_DRIVER_LOAD_FAILED = -4,
    ERR_DRIVER_NOT_INITIALIZED = -5,
    ERR_DEVICE_DISCONNECTED = -6,
    ERR_INVALID_PARAMETER = -7,
    ERR_NOT_ENOUGH_MEMORY = -8,
    ERR_GENERAL_FAILURE = -9,
    ERR_TIMEOUT = -10,
    ERR_ALREADY_DRIVER_OPENED = -11
}
ERR_CODE;
```

다음은 각 API에 대한 설명이다.

3.7.1 int DNK_InitDLL (void)

cantalker.dll 파일을 로딩하는 함수로 가장 먼저 수행해야 한다.

3.7.2 int DNK_GetDeviceCount (void)

CAN2USB Analyzer가 몇 대가 연결되어 있는지 값을 리턴 한다.

3.7.3 int DNK_InitUSBDriver (int DeviceHandle)

USB를 사용하는 경우 DNK_GetDeviceCount에서 얻어진 N개의 장치 중 하나를 골라 드라이버를 연결한다. DeviceHandle은 0 부터 N-1 사이 값을 지정한다. 단, 이미 RS232 사용을 위해 DNK_InitRS232Driver가 호출되어 RS232 드라이버가 열려 있는 경우는 사용할 수 없다. 즉 USB와 RS232는 동시 사용할 수 없다.

3.7.4 int DNK_InitRS232Driver (int port, int baudrate)

RS232를 사용하는 경우 시리얼 포트 드라이버에 연결 한다. 단, 이미 USB 사용을 위해 DNK_InitUSBDriver가 호출되어 USB드라이버가 열려 있는 경우는 사용할 수 없다. 즉 USB와 RS232는 동시 사용

할 수 없다.

3.7.5 int DNK_CloseDriver (void)

드라이버 연결을 종료한다. 프로그램 종료 시에 반드시 호출해야 한다.

3.7.6 int DNK_CloseDLL (void)

DLL 사용을 완료한다. 프로그램 종료 시에 반드시 호출해야 한다.

3.7.7 int DNK_SetCanInit (void)

Analyzer의 CAN Controller를 초기화 한다. 에러가 있는 경우는 에러 카운트가 모두 리셋 된다.

3.7.8 int DNK_SendCanData (CAN_FRAME *canFrame)

CAN Data Frame을 전송한다. CAN_FRMAE의 데이터 형식은 다음과 같다. 필드 중 timeStamp는 입력할 필요가 없다. 2.0B 29비트 모드와 RTR설정 부분은 4.5.5 절을 참고하여 CAN ID를 설정한다.

```
typedef struct
{
    UINT32    timeStamp;
    UINT32    canId;
    BYTE      dlc;
    BYTE      data[8];
}
CAN_FRAME;
```

3.7.9 int DNK_GetTotalRcvCanDataCount (void)

현재 수신 되어 메모리에 버퍼링 된 CAN Data Frame의 갯수를 리턴 한다.

3.7.10 int DNK_ClearRcvCanaData (void)

현재 수신 되어 버퍼링 되어 있는 CAN Data Frame을 모두 메모리에서 삭제 한다. 프로그램 종료 시에 삭제 한다.

3.7.11 int DNK_GetRcvCanData (CAN_FRAME *can)

DNK_GetTotalRcvCanDataCount로 1개 이상의 CAN Data Frame이 버퍼링 되어 있을 때 호출하면 1개 씩 버퍼에서 빼온다.

3.7.12 int DNK_SetClearCanError (void)

CAN Error가 있는 경우 리셋 한다.

3.7.13 int DNK_ReqCanStatus (CAN_STATUS *status)

현재 CAN 상태를 얻어 온다. CAN_STATUS 데이터 형은 다음과 같다.

```
typedef struct
{
    UINT32    canId;
    UINT8     status;
    UINT8     brgcon[3];
}
CAN_STATUS;
```

3.7.14 int DNK_ReqErrorCount (UINT8 *txErrCnt, UINT8 *rxErrCnt)

현재 Tx와 Rx에 대한 Error 수를 얻어 온다.

3.7.15 int DNK_SetCanID (BYTE extMode, UINT32 id)

수신하기 위한 CAN ID를 설정한다. 자세한 설정 법은 4.5.2 절을 참조 한다.

3.7.16 int DNK_SetCanBaudRate (UINT32 brate)

CAN 통신속도를 설정한다. 자세한 설정 법은 4.5.3 절을 참조 한다.

3.7.17 int DNK_ReqDeviceInfo (DEVICE_INFO *devInfo)

Analyzer의 모델명과 시리얼 번호를 가져온다. DEVICE_INFO 데이터 형은 다음과 같다. 내용은 8자리 문자로 구성되어 있다. 마지막에 NULL문자는 포함되어 있지 않다.

```
typedef struct
{
    UINT8     modelName[8];
    UINT8     serialNo[8];
}
```

```

}
DEVICE_INFO;

```

3.7.18 int DNK_SetRcvMode (UINT32 mode)

CAN Data Frame수신시 Time Stamp를 포함할 지를 설정한다. 자세한 설명은 4.5.28 절을 참조 한다.

3.7.19 int DNK_ReqRcvMode (UINT32 *rcvMode)

현재 설정된 수신 모드값을 가져온다. 수신모드에 대한 설명은 4.5.28 절을 참조 한다.

3.7.20 int DNK_ReqCanFilters (FILTER_INFO *filterInfo)

현재 설정된 CAN Mask와 Filter값을 가져 온다. FILTER_INFO의 데이터 형은 다음과 같다.

```

typedef struct
{
    UINT32    mask[2];
    UINT32    filter[6];
}
FILTER_INFO;

```

3.7.21 int DNK_ReqVersion (UINT8 *major, UINT8 *minor)

Analyzer의 버전을 가져 온다. 버전 표시는 “major.minor” 형식을 갖는다.

3.7.22 int DNK_ReqSpeedInfo (UINT32 *canSpeed, UINT8 *rs232SpeedIdx)

현재 설정된 CAN 과 RS232 통신 속도를 가져 온다. 자세한 설명은 4.5.21 절을 참조한다.

3.7.23 int DNK_ReqTcplInfo (TCPIP_INFO *pInfo)

현재 설정된 TCP/IP 정보를 가져 온다. TCPIP_INFO 데이터 형은 다음과 같다.

```

typedef struct
{
    UINT16    port;
    UINT8     ipMode;
    UINT8     ipAddr[4];
    UINT8     subnetMask[4];
    UINT8     gateway[4];
    UINT8     macAddr[6];
}

```

```
TCPIP_INFO;
```

3.7.24 int DNK_ReqCurTimeStamp (UINT32 *timeStamp)

현재 Analyzer의 Time Stamp를 가져 온다.

3.7.25 int DNK_SetCanBitTiming (BRGCON *brgcon)

사용자 지정 CAN 통신 속도를 설정한다. 자세한 설명은 4.5.10 절을 참조하고, BRGCON의 데이터 형은 다음과 같다.

```
typedef struct
{
    UINT8      brgcon[3];
}
BRGCON;
```

3.7.26 int DNK_SetFilters (FILTER_INFO *filterInfo)

CAN Mask와 Filter 를 설정한다. FILTER_INFO 데이터 형은 DNK_ReqCanFilters 설명을 참조한다.

3.7.27 int DNK_SetTcplpInfo (TCPIP_INFO *pInfo)

TCP/IP 설정 값을 변경 한다. TCPIP_INFO 데이터 형은 DNK_ReqTcplpInfo 설명을 참조한다.

3.7.28 int DNK_SeRs232Baudrate (UINT8 baudrate)

RS232 통신 속도를 변경 한다. 자세한 설명은 4.5.4 절을 참조한다.

3.7.29 int DNK_CloseC2EConnection (void)

Multi-CAN Analyzer에서 C2E TCP 연결을 끊는다.

4 Packet Mode

4.1 Overview

본 Analyzer는 두개의 통신 모드를 지원하는데 패킷모드와 터미널 모드이다. 패킷모드는 지정된 형식의 프로토콜로 바이너리 통신을 하며 터미널 모드는 사용자 대화 형식의 아스키(ASCII) 통신을 한다. 패킷모드는 데이터를 바이너리 형태로 보내기 때문에 사람의 눈으로 확인하고 직접 조작하기는 어려운 반면 최적의 데이터를 전송하므로 전송효율이 높은 편이다. 터미널 모드는 사용자의 눈으로 확인하고 직접 입력할 수 있는 모드로 일반적인 터미널 응용프로그램(예. 하이퍼터미널)으로 사용 할 수 있다. 터미널 모드에 대해서는 5장에서 자세히 기술된다.

4.2 Physical Setup

RS232C 통신을 하는 경우는 PC측에서 다음과 같이 설정되어야 한다.

1. Data : 8bit
2. Parity : None
3. Stop Bit : One
4. Handshake : None
5. Baudrate : 9600/19200/38400/57600/115200/230400/460800 bps

USB는 특별히 설정할 것이 없으며 Ethernet인 경우는 TCP/IP를 주소를 설정해 주어야 한다. IP를 설정하기 전에 Analyzer에 설정된 정확한 IP모드를 확인해야 하는데 IP Mode가 static인 고정 IP모드인 경우는 Analyzer와 함께 Network에서 사용할 정확한 IP/Gateway주소와 Subnet Mask를 설정해야하고 나머지 Dynamic IP모드와 DHCP server모드에서는 PC의 TCP/IP 설정을 IP자동 할당모드로 설정해야 한다.

Analyzer에 설정되는 기본 TCP/IP 설정값은 다음과 같다. (C2UA 제품은 제외)

1. IP Address : 192.168.10.100
2. Gateway Address : 192.168.10.1
3. Subnet Mask : 255.255.255.0
4. C2E Socket Port : 십진수 54321 (0xD431)

4.3 Handshake

데이터 전송의 무결성을 보장하기 위해 응답을 요청하는 REQ(Request)에 대해서는 관련된 RSP(Response)로 응답하고, 데이터를 요구하지 않는 SET 명령에 대해서는 ACK/NACK로 응답한다. 만약 NACK를 수신 받는 경우는 재전송하고, 이는 5회까지 반복할 수 있으며, 이후에도 NACK를 수신하면 패킷은 소멸 시킨다. REQ명령인 경우도 패킷내용이 비정상적이면 NACK를 받을 수 있다.

모든 응답은 200 ms 이내에 수행해야 하고, 시간이 초과되는 경우는 명령을 전송했던 쪽에서 재전송한다. 5회 재전송에도 응답하지 않으면 Fail로 간주 하고, 패킷은 소멸 시킨다. 여기서 응답시간, 200 ms는 패킷전송이 시작되는 시점이 아니고, 패킷전체가 전송된 후 부터의 시간을 의미 한다. 이는, 패킷마다 길이가 다를수 있고, UART속도에 따라 전송 시간이 달라질 수 있기 때문이다.

CAN 메시지를 송수신 하는 PID는 Handshake하지 않는다. 즉 Analyzer로 Send Packet하거나 PC로 Receive 하는 경우 ACK로 응답하지 않는다. 단 비정상 packet인 경우만 NACK로 응답한다. handshake를 하지 않는 이유는 고속 통신을 보장하기 위함이므로 handshake가 필요하다면 Application Level에서 직접 해주어야 한다.

RS232C 인터페이스를 사용하는 경우, 패킷중에 설정값을 NV메모리에 저장해야 하는 것들에 대해서는 최소 50ms 이상의 패킷간 간격을 두어야 한다. NV메모리에 기록하는 중에 다른 명령이 오는 경우 RS232통신에 영향을 주기 때문이다.

4.4 General Packet Format

다양한 종류의 명령들을 수행 하기 위해 전송되는 데이터들은 지정된 형식으로 추가 포장을 하여 전송하게 된다. 이렇게 포장된 전송단위를 Packet이라고 하고, 한 Packet은 다음과 같은 구조를 갖는다.

SYNC(0xF0)	PID(1byte)	Length(1byte)	Data(0~n byte)	Checksum (1byte)	EndMark(0xE0)
------------	------------	---------------	----------------	------------------	---------------

여기서 Sync과 EndMark는 Frame의 시작과 끝을 지시하는 값으로서 각각 1byte의 크기를 갖는다. 단 Ethernet 인터페이스에서는 이 필드가 사용되지 않는다.

PID는 각 Packet의 특성을 기술하는 값으로 Table 1에서 모든 PID 종류를 볼 수 있다.

Length는 순수 Data의 길이를 의미하며 1바이트 크기를 갖는다.

Data는 Length에서 지시한 만큼 전송되며 Length가 0인 경우는 Data가 존재하지 않고 즉시 CheckSum이 위치하게 된다. 데이터로 전달 되는 모든 데이터 형은 하위데이터가 먼저 위치하는 Little Endian으로 처리되어야 한다.

Checksum은 PID부터 Data까지 1바이트 형식으로 모두 더한 값을 의미 한다. 합계가 1바이트 크기가 넘는 경우는 하위 1바이트만 취한다.

Packet의 프로토콜을 처리하는 부분에서는 위에서 지정된 형식을 벗어나는 데이터가 수신되면 즉시 SYNC입력상태로 천이 하여 다음 SYNC가 들어올때까지 수신되는 모든 데이터는 무시한다.

No	PID	Description	Data Size (byte)	Data Structure	Direction
1	0x80	Set CAN Initialize	0		From PC
2	0x81	Set CAN ID	5		From PC
3	0x82	Set CAN Baudrate	1 or 4	CAN_BIT_RATE	From PC
4	0x83	Set RS232 Baudrate	1	UART_BAUDRATE	From PC
5	0x84	Send CAN Message (ACK Not required)	4~12		From PC
6	0x85	Receive CAN Message (ACK Not required)	4~12		To PC
7	0x86	Request CAN Status	0		From PC
8	0x87	Response CAN Status	8		To PC
9	0x88	Indicate CAN Error	-		To PC
10	0x89	Set CAN Baudrate Parameters	1 or 4		From PC
11	0x8A	Clear CAN Errors	0		From PC
12	0x8B	Set CAN Filters	32		From PC
13	0x8C	Request Version	0		From PC
14	0x8D	Response Version	2		To PC
15	0x90	Close C2E Connection	0		From PC
16	0x91	Request CAN Filters	0		From PC
17	0x92	Response CAN Filters	32		To PC
18	0x93	Request Device Information	0		From PC
19	0x94	Response Device Information	16		To PC
20	0x95	Request Speed Information	0		From PC
21	0x96	Response Speed Information	5		To PC
22	0x97	Set TCP/IP Information	21	IP_MODE, Port, IP Addr	From PC
23	0x98	Request TCP/IP Information	0		From PC
24	0x99	Response TCP/IP Information	21	IP_MODE, Port, IP Addr	To PC
25	0x9A	Request Time Stamp	0		From PC
26	0x9B	Response Time Stamp	4		To PC
27	0xA0	Receive CAN Message with time stamp	8~16		To PC
28	0xA1	Set CAN Receive Mode	4		From PC
29	0xA2	Request CAN Receive Mode	0		From PC
30	0xA3	Response CAN Receive Mode	4		To PC
31	0xA4	Request CAN Error Count	0		From PC
32	0xA5	Response CAN Error Count	2		To PC
33	0xC0	Acknowledge	0		All
34	0xD0	No Acknowledge	1	NACK_REASON	All

Table 4: PID List

4.5 Packet Formats

본 절에서는 각 PID에 대한 구체적인 패킷형식에 대해 정의하고 기능에 대해 기술한다. 앞서 설명된 것처럼 본절에서 설명되는 각 패킷의 SYNC(0xF0)와 END MARK(0xE0)는 Ethernet 인터페이스에서는 사용되지 않으니 사용에 주의해야 한다.

4.5.1 Set CAN Initialize (0x80)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x80	0	0x80	0xE0
------	------	---	------	------

CAN 컨트롤러를 초기화 한다. 본 명령은 사용초기나 CAN에러가 발생한 경우에 사용할 수 있다. Analyzer는 초기화 수행후 ACK를 PC로 전송하거나 비정상인 경우 NACK을 전송할 수 있다.

4.5.2 Set CAN ID (0x81)

Packet Frame (14 bytes) : PC → Analyzer

0xF0	0x81	5	Extend ID(1)	CAN ID(4)	Checksum	0xE0
------	------	---	--------------	-----------	----------	------

수신할 메시지에 대한 CAN ID를 설정한다. 모든 메시지를 수신하고자 하는 경우는 CAN ID를 [0xFFFFFFFF]로 설정해야 하고, 사용자 정의 Filter & Mask를 사용하여 메시지를 수신하고자 하는 경우에는 CAN ID를 [0xFFFFFFFFE]로 설정해야 한다.

Extend ID는 송수신하는 메시지의 CAN버전을 결정하는 부분으로 0인 경우 11비트 CAN ID가 사용되는 CAN2.0A이고, 1인 경우는 29비트 CAN ID를 사용하는 CAN2.0B 인 경우다. 그러나 본 명령은 기존 Converter들과의 호환을 위해 사용되는 것이며, Extend ID 설정값에 상관없이 CAN2.0A와 CAN2.0B를 항상 수신하거나 발송할 수 있다.

정상적으로 설정되면 Analyzer는 ACK를 전송하고 오류가 있는 경우는 NACK 을 전송한다.

4.5.3 Set CAN Btrrate (0x82)

Packet Frame (6 or 9 bytes) : PC → Analyzer

0xF0	0x82	1 or 4	Btrrate(1 or 4)	Checksum	0xE0
------	------	--------	-----------------	----------	------

CAN 버스의 송수신 속도 Bitrate를 설정한다. 데이터 길이는 1 또는 4를 갖을 수 있다. 길이가 1인 경우는 Bitrate가 다음과 같은 속도에 대한 enum type index값을 갖으며 지정할 수 있는 bitrate은 50 / 100 / 125 / 200 / 250 / 400 / 500 / 1000 kbps 이다.

```
typedef enum
{
    CAN_BIT_RATE_125 = 0,
    CAN_BIT_RATE_250,
    CAN_BIT_RATE_500,
    CAN_BIT_RATE_1000,
    CAN_BIT_RATE_50,
    CAN_BIT_RATE_100,
    CAN_BIT_RATE_200,
    CAN_BIT_RATE_400,
    CAN_BIT_RATE_USER,
}
CAN_BIT_RATE;
```

여기서 [CAN_BIT_RATE_USER = 8] 인 경우는 사용자 정의 bit time 값을 사용할 수 있다. 사용자 정의 bit time은 세밀하게 bit time의 조정이 필요한 경우에 사용되며 설정하는 방법은 4.5.10절을 참고한다.

데이터 길이가 4인 경우는 직접 bitrate을 bps단위로 지정할 수 있으며 최소 50000부터 1000000 사이의 값을 갖을 수 있다.

설정이 정상적으로 완료되면 Analyzer는 ACK을, 오류가 발생하는 경우는 NACK를 전송한다.

4.5.4 Set RS232 Baudrate (0x83)

Packet Frame (6 bytes) : PC → Analyzer

0xF0	0x83	1	Baud Rate(1)	Checksum	0xE0
------	------	---	--------------	----------	------

RS232C의 통신 속도인 Baud Rate를 설정한다. Baud Rate의 값은 다음과 같이 enum type index값을 갖는다.

```
typedef enum
{
    UART_BAUDRATE_9600 = 0,
    UART_BAUDRATE_19200,
    UART_BAUDRATE_38400,
    UART_BAUDRATE_57600,
    UART_BAUDRATE_115200,
    UART_BAUDRATE_230400,
    UART_BAUDRATE_460800,
    UART_BAUDRATE_TOTAL_NUM
}
UART_BAUDRATE;
```

115200bps를 초과하는 경우는 반드시 고속을 지원하는 장치나 USB2Serial Converter를 사용해야 한다.

Analyzer는 baudrate를 변경하기 전에 먼저 ACK를 전송하므로 PC측에서는 ACK를 수신한 후 RS232C의 baudrate를 변경해야 한다. 만약 본 패킷이 비정상이라면 Analyzer는 NACK를 전송한다. RS232C baudrate를 변경한 경우는 정상적인 동작을 위해 Analyzer의 전원을 꺾다 킨다.

4.5.5 Send CAN Message (0x84)

Packet Frame (9~17 bytes): PC → Analyzer

0xF0	0x84	4~12	CAN ID(4)	Data (0~8)	Checksum	0xE0
------	------	------	-----------	------------	----------	------

CAN 메시지를 전송하기 위한 패킷으로 CAN ID는 메시지를 수신받을 장치의 목적지 Id이다. CAN_ID의 각 비트별 의미는 다음과 같다.

bit31	bit 30	bit 29	bit28	bit0
Extend ID	RTR	reserved	Destination CAN ID	

31번째 비트는 Extend ID를 의미 하며 1로 설정하는 경우 29 bit CAN ID로 전송되며, 30번째 비트는 RTR를 전송하고자 하는 경우 1로 설정한다.

본 메시지는 실제 CAN 버스상에 전송 유무와 상관없이 Analyzer는 응답하지 않고 packet이 비정상인 경우만 NACK로 응답한다.

4.5.6 Receive CAN Message (0x85)

Packet Frame (9~17 bytes) : Analyzer → PC

0xF0	0x85	4~12	CAN ID(4)	Data (0~8)	Checksum	0xE0
------	------	------	-----------	------------	----------	------

수신한 CAN 메시지를 PC로 전달하는 패킷으로 CAN_ID의 각 비트별 의미는 Send CAN Message와 동일하게 다음과 같다.

bit31	bit 30	bit 29	bit28	bit0
Extend ID	RTR	reserved	Destination CAN ID	

31번째 비트는 Extend ID를 의미 하기 때문에 이 값이 0인 경우는 CAN ID의 11비트만, 1인 경우는 29비트만 유효하며, 30번째 비트는 RTR을 의미 한다.

본 메시지를 수신하더라도 PC에서는 ACK나 NACK를 응답하지 않는다. 본 메시지는 Receive Mode (0xA1) 명령에 의해 Time Stamp가 Disable되어 있는 경우만 수신할 수 있고, Enable된 경우는 Receive CAN Message

with time stamp (0xA0) 로 수신 된다.

4.5.7 Request CAN Status (0x86)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x86	0	0x86	0xE0
------	------	---	------	------

Analyzer에 현재 CAN 컨트롤러의 상태를 요청하는 패킷이다. 응답 데이터는 다음절의 [0x87] 로 전송되며 본 패킷이 비정상인 경우는 NACK[0xD0]로 응답한다.

4.5.8 Response CAN Status (0x87)

Packet Frame (13 bytes) : Analyzer → PC

0xF0	0x87	8	Status(1)	CAN ID(4)	BRGCON1(1)	BRGCON2(1)	BRGCON3(1)	Checksum	0xE0
------	------	---	-----------	-----------	------------	------------	------------	----------	------

CAN상태를 요청하는 [0x86] 에 응답하는 패킷이다. 1바이트 크기의 Status는 각 비트별로 다음과 같은 의미를 갖는다.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Extend ID	Tx Bus Off	Tx Bus Passive	Rx Bus Passive	Tx Warning	Rx Warning	Error Warning	Overrun

각 status 비트 값은 Figure 6와 같이 Tx와 Rx의 Error Count에 따라 State Machine에 의해 관리되며 설정된다. 즉, Rx또는 Tx의 에러수가 127보다 작으면 Error Active가 되고, 그 이상이 되면 Error Passive가 된다. Error Passive상태에서 또 Tx에러수가 255를 넘어서게 되면 Bus Off가 된다. Bus off상태에서는 연속적인 11개의 Resessive 비트가 128회 발생하면 (Bus off회복 시퀀스) Bus off상태를 회복하여 다시 Error Active상태로 된다. 여기서 실제 Tx와 Rx에 대한 Error count는 4.5.31 절의 [0xA4] 패킷으로 알아낼 수 있다.

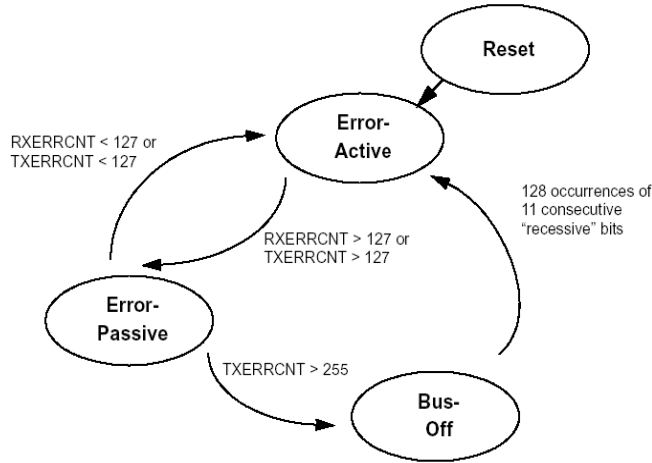


Figure 6: CAN Error State Machine

4.5.9 Indicate CAN Error (0x88)

Packet Frame (6 bytes) : Analyzer → PC

0xF0	0x88	1	Status(1)	Checksum	0xE0
------	------	---	-----------	----------	------

CAN컨트롤러에 Error가 발생한 경우에 비동기적으로 PC로 전송되는 패킷이나, 사용되지 않는다. Status 내용은 4.5.8 절의 내용을 참고한다.

4.5.10 Set CAN Bitrate Parameters (0x89)

Packet Frame (8 bytes) : PC → Analyzer

0xF0	0x89	3	BRGCON1(1)	BRGCON2(1)	BRGCON3(1)	Checksum	0xE0
------	------	---	------------	------------	------------	----------	------

사용자 정의 CAN Bitrate를 설정할 때 사용되며 설정이 정상적으로 완료되면 Analyzer는 ACK을, 오류가 발생하는 경우는 NACK를 전송한다. 본 Bit Time이 유효하기 위해서는 4.5.3 절의 CAN Bitrate0[CAN_BIT_RATE_USER] 로 설정되어 있어야 한다. BRGCON1~3은 직접 계산해도 되지만 Microchip CAN Bit Timing Calculator에서도 쉽게 얻을 수 있다. 각 BRGCON이 비트별 의미 하는 내용은 다음과 같다.

[BRGCON1]

bit7	bit6	bit5	bit0
SJW(2)		BRP(6)	

BRGCON2

bit7	bit6	bit5	bit3	bit2	bit0
SEG2PHTS(1)	SAM(1)	SEG1PH(3)	PRSEG(3)		

BRGCON3

bit7	bit6	bit5	bit4	bit3	bit2	bit0
-	WAKFIL	-	-	-	SEG2PH	

Microchip CAN Bit Timing Calculator 응용프로그램을 사용하여 BRGCON을 계산 하는 경우에는 Figure 7의 Oscillator Frequency를 80MHz로 해야 한다.



Figure 7: Microchip CAN Bit Timing Calculator

4.5.11 Clear CAN Errors (0x8A)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x8A	0	0x8A	0xE0
------	------	---	------	------

CAN Error, USB / RS232C의 Fifo Overrun 상태를 모두 리셋하는 패킷이다. CAN Error LED가 점등된 경우 이 패킷으로 Error가 해제되면서 소등 시킬 수 있다. 정상적으로 처리되면 Analyzer는 ACK을 그렇지 않은 경우는 NACK을 전송한다.

4.5.12 Set CAN Filters (0x8B)

Packet Frame (37 bytes) : PC → Analyzer

0xF0	0x8B	32	Mask1 (4)	Mask2 (4)	Filter1 (4)	Filter2 (4)	Filter3 (4)	Filter4 (4)	Filter5 (4)	Filter6 (4)	Checksum	0xE0
------	------	----	--------------	--------------	----------------	----------------	----------------	----------------	----------------	----------------	----------	------

특정 CAN ID를 수신하기 위한 사용자 Mask & Filter를 설정한다. 이 값이 활성화 되기 위해서는 4.5.2에서 설명된 것처럼 CAN ID가 [0xFFFFFFFF]로 설정되어 있어야 한다. Mask1과 Filter1/Filter2가 한쌍이고, Mask2와 Filter2/Filter3/Filter4/Filter5/Filter6 이 한쌍으로 동작된다. CAN ID, Filter와 Mask의 관계는 CAN ID를 Mask값으로 마스크(CAN ID & Mask)하여 나온 값이 Filter와 동일한 경우 그 메시지를 수신하는 것이다. 이런 조합을 2개 set으로 할 수 있다. 개념을 간단히 Pseudo Code로 작성하면 다음과 같고 보다 자세한 내용은 “CAN to RS232 Converter Manual”의 6장을 참고한다.

```

if((CAN_ID & Mask) == Filter) DoAccept()
else                          DoReject()

```

정상적으로 처리되면 Analyzer는 ACK를 그렇지 않은 경우는 NACK를 응답한다.

4.5.13 Request Version (0x8C)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x8C	0	0x8C	0xE0
------	------	---	------	------

Analyzer의 펌웨어 버전을 요청하는 패킷으로 [0x8D]로 응답하고 패킷에 오류가 있으면 NACK로 응답한다.

4.5.14 Response Version (0x8D)

Packet Frame (7 bytes) : Analyzer → PC

0xF0	0x8D	2	Major Version(1)	Minor Version(1)	Checksum	0xE0
------	------	---	------------------	------------------	----------	------

[0x8C]로 요청한 펌웨어 버전정보를 전달하는 패킷이다. 버전의 표시형식은 “MajorVersion . MinorVersion” 이다.

4.5.15 Close C2E Connection (0x90)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x90	0	0x90	0xE0
------	------	---	------	------

Ethernet인터페이스를 사용하면서 C2E 프로토콜로 연결된 경우 접속을 해제하기 위한 패킷이다. C2E 프로토

콜로 연결된 경우는 반드시 본 패킷으로 연결을 해제해 주어야 한다. Analyzer는 ACK나NACK를 전송한 후 접속을 해제한다. NACK를 전송하는 경우는 RS232C나USB인터페이스를 통해 본 패킷을 수신한 경우나 패킷의 형식이 잘못된 경우이다.

4.5.16 Request CAN Filters (0x91)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x91	0	0x91	0xE0
------	------	---	------	------

[0x8B]로 설정했던 Filter & Mask값을 요청하는 패킷으로 [0x92]로 응답한다.

4.5.17 Response CAN Filters (0x92)

Packet Frame (37 bytes) : Analyzer → PC

0xF0	0x92	32	Mask1 (4)	Mask2 (4)	Filter1 (4)	Filter2 (4)	Filter3 (4)	Filter4 (4)	Filter5 (4)	Filter6 (4)	Checksum	0xE0
------	------	----	--------------	--------------	----------------	----------------	----------------	----------------	----------------	----------------	----------	------

[0x91]로 요청했던 Filter & Mask값을 전달하는 패킷이다.

4.5.18 Request Device Information (0x93)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x93	0	0x93	0xE0
------	------	---	------	------

Analyzer의 모델명과 시리얼 번호를 얻기 위해 요청하는 패킷으로 [0x94]로 응답한다. 패킷 형식이 잘못된 경우는 Analyzer가 즉시 NACK로 응답한다.

4.5.19 Response Device Information (0x94)

Packet Frame (21 bytes) : Analyzer → PC

0xF0	0x94	16	Model Name(8)	Serial No(8)	Checksum	0xE0
------	------	----	---------------	--------------	----------	------

[0x93] 패킷요청에 최대 8글자로 구성된 Model Name과 최대 8숫자로 된 Serial No를 응답하는 패킷이다. 모두 문자형식으로 되어있으며 끝문자는 없고 빈칸은 SPACE (ASCII Code 0x20) 로 채워진다.

4.5.20 Request Speed Information (0x95)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x95	0	0x95	0xE0
------	------	---	------	------

Analyzer에 현재 설정된 CAN Bitrate와 RS232C Baudrate값을 요청하는 패킷으로 [0x96]으로 응답하며 비정상 패킷인 경우는 NACK로 응답한다.

4.5.21 Response Speed Information (0x96)

Packet Frame (10 bytes) : Analyzer → PC

0xF0	0x96	5	CAN BitRate(4)	RS232C BaudRate(1)	Checksum	0xE0
------	------	---	----------------	--------------------	----------	------

[0x95]로 요청한 속도정보에 대해 응답하는 패킷으로 CAN BitRate와 RS232C BaudRate로 구성된다. CAN BitRate가 4.5.3에서 기술된 CAN_BIT_RATE_USER(8)을 초과하는 경우는 실제 CAN BitRate를 의미 한다. RS232C BaudRate는 4.5.4에서 기술된 UAR_BAUDRATE를 참고한다.

4.5.22 Set TCP/IP Information (0x97)

Packet Frame (26 bytes) : PC → Analyzer

0xF0	0x97	21	C2E Port(2)	IP Mode(1)	IP Addr(4)	Subnet Mask(4)	Gateway(4)	MAC Addr(6)	Checksum	0xE0
------	------	----	-------------	------------	------------	----------------	------------	-------------	----------	------

Analyzer의 TCP/IP의 정보를 설정한다. 정상적으로 설정되면 ACK를 그렇지 않은 경우는 NACK를 응답한다. C2E Port는 C2E 프로토콜을 사용하기 위한 TCP Port번호를 의미하며 54321이 기본값이다. IP Mode는 다음과 같은 3가지 경우의 값을 갖는다.

```
typedef enum
{
    IP_MODE_STATIC = 0,
    IP_MODE_DYNAMIC,
    IP_MODE_DHCP_SERVER,
    IP_MODE_MAX
}
IP_MODE;
```

여기서, Static인 경우는 고정 IP를 의미한다. Dynamic인 경우는 Analyzer가 DHCP서버로부터 IP를 할당 받는다. IP할당중인 경우는 [Status LED]가 점멸하며 할당이 완료되면 점등상태를 유지 한다. DHCP Server인 경우는 Analyzer에 내장된 DHCP Server가 동작하는 경우로 자동할당으로 설정된 PC가 Analyzer와 직접 연결되어 있는 경우 Analyzer가 PC의 IP를 할당해 준다. 자동 IP를 사용하고 내부 네트워크에 DHCP server가 없는 경우에 유용하다.

4.5.23 Request TCP/IP Information (0x98)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x98	0	0x98	0xE0
------	------	---	------	------

[0x97]로 설정된 TCP/IP 정보를 요청하는 패킷으로 [0x99]로 응답한다.

4.5.24 Response TCP/IP Information (0x99)

Packet Frame (26 bytes) : Analyzer → PC

0xF0	0x99	21	C2E Port(2)	IP Mode(1)	IP Addr(4)	Subnet Mask(4)	Gateway(4)	MAC Addr(6)	Checksum	0xE0
------	------	----	-------------	------------	------------	----------------	------------	-------------	----------	------

[0x98]로 요청한 TCP/IP 정보를 응답하는 패킷으로 패킷의 각 필드 내용은 4.5.22 절을 참고한다.

4.5.25 Request Time Stamp (0x9A)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0x9A	0	0x9A	0xE0
------	------	---	------	------

현재 CAN Controller의 Time Stamp 값을 요청하는 패킷으로 0x9B로 응답한다. 본패킷이 비정상적인 경우는 NACK로 즉시 응답한다.

4.5.26 Response Time Stamp (0x9B)

Packet Frame (5 bytes) : Analyzer → PC

0xF0	0x9B	4	Time Stamp(4)	Checksum	0xE0
------	------	---	---------------	----------	------

[0x9A]로 요청한 CAN Time Stamp를 응답한다. Time Stamp의 단위는 10usec 단위이다.

4.5.27 Receive CAN Message with time stamp (0xA0)

Packet Frame (13~21 bytes) : Analyzer → PC

0xF0	0xA0	8~16	Time Stamp(4)	CAN ID(4)	Data(0~8)	Checksum	0xE0
------	------	------	---------------	-----------	-----------	----------	------

4.5.6 절의 [0x85]와 같이 수신된 CAN 메시지를 PC로 전달하는 패킷이지만 추가적으로 수신시의 TimeStamp까지 기록되어 있다. PC는 본 패킷을 받고 ACK / NACK 응답을 해서는 안된다. 본 메시지는 Receive Mode (0xA1) 명령에 의해 Time Stamp가 Enable되어 있는 경우만 수신할 수 있고, Disable된 경우는 Receive CAN Message (0x85) 로 수신된다.

4.5.28 Set CAN Receive Mode (0xA1)

Packet Frame (9 bytes) : PC → Analyzer

0xF0	0xA1	4	Received Mode(4)	Checksum	0xE0
------	------	---	------------------	----------	------

PC에서 수신할 CAN 메시지의 형식을 결정한다. Receive Mode의 최하위 비트가 1이면 수신 CAN 메시지는 Time Stamp를 포함하여 [0xA0]로 전달되며 0이면 [0x85]로 전달된다.

Analyzer는 정상적으로 설정되면 ACK를 그렇지 않은 경우는 NACK로 응답한다.

각 비트별 의미는 다음과 같다. 현재는 bit0만 사용된다.

bit31	Reserved	bit1	bit0
			Time Stamp Enable

4.5.29 Request CAN Receive Mode (0xA2)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0xA2	0	0xA2	0xE0
------	------	---	------	------

[0xA1]로 설정된 Receive Mode값을 요청하는 패킷으로 Analyzer는 [0xA3]로 응답한다. 본 패킷이 비정상적인 경우는 NACK로 즉시 응답한다.

4.5.30 Response CAN Receive Mode (0xA3)

Packet Frame (5 bytes) : Analyzer → PC

0xF0	0xA3	4	Received Mode(4)	Checksum	0xE0
------	------	---	------------------	----------	------

[0xA2]로 요청된 CAN Receive Mode에 대해 응답한다. Receive Mode는 4.5.28 절을 참고한다.

4.5.31 Request CAN Error Count (0xA4)

Packet Frame (5 bytes) : PC → Analyzer

0xF0	0xA4	0	0xA4	0xE0
------	------	---	------	------

현재 CAN Controller에 발생된 CAN TX/RX Error Count를 요청하는 패킷으로 Analyzer는 [0xA5]로 응답한다. 본 패킷이 비정상인 경우는 NACK로 즉시 응답한다.

4.5.32 Response CAN Error Count (0xA5)

Packet Frame (5 bytes) : Analyzer → PC

0xF0	0xA5	2	Rx Error Count(1)	Tx Error Count(1)	Checksum	0xE0
------	------	---	-------------------	-------------------	----------	------

[0xA4]로 요청된 CAN Error Count 요청에 응답한다.

4.5.33 Acknowledge (0xC0)

Packet Frame (5 bytes) : Analyzer → PC / PC → Analyzer

0xF0	0xC0	0	0xC0	0xE0
------	------	---	------	------

거의 대부분 SET 종류의 패킷에 대해 정상 처리된 경우ACK응답을 전송한다.

4.5.34 No Acknowledge (0xD0)

Packet Frame (5 bytes) : Analyzer → PC / PC → Analyzer

0xF0	0xD0	1	Reason(1)	Checksum	0xE0
------	------	---	-----------	----------	------

전송된 패킷의 형식이나 값의 범위가 잘못되거나 동작에 실패하는 경우는 NACK를 전송한다. NACK를 전송하는 이유가 되는 Reason의 종류는 다음과 같다.

```
typedef enum
{
    NACK_NO_REASON,
    NACK_INVALID_CSUM, /* Checksum오류 */
    NACK_INVALID_PID, /* 비정상 PID */
    NACK_ETC_ERROR, /* 기타 오류 */
    NACK_INVALID_PARAM, /* 비정상 값의 범위 */
    NACK_INVALID_LEN /* 비정상 패킷데이터 길이 */
}
NACK_REASON;
```

5 Terminal Mode

5.1 Overview

Analyzer는 간편하게 연결하여 조작할 수 있는 사용자 대화형 터미널 모드를 지원한다. 모든 데이터 또는 메시지들은 아스키(ASCII)코드로 전송되기 때문에 눈으로 직접확인할 수 있고 터미널에서 키보드로 입력된 값을 전달할 수 있다. RS232C의 경우는 일반적인 시리얼터미널 응용프로그램 (teraterm)으로 사용하면 되며, Ethernet인 경우는 Telnet을 통해 지원된다. USB는 RS232C와 같이 6장의 CANTALKER 응용프로그램의 Terminal 메뉴에서 Terminal기능을 사용 할 수 있다.

RS232C로 연결하는 경우 PC의 상황에 따라 Windows에서 제공하는 하이퍼터미널 응용프로그램은 비정상적으로 동작하는 경우가 있기 때문에 되도록 “TeraTerm”, “이야기” 또는 “Putty” 등을 이용하는것이 좋다.

5.2 How to enter terminal Mode

RS232C와 USB의 경우는 SPACE (ASCII 0x20, ‘ ’) 가 연속으로 5번 입력되면 즉시 진입된다. 즉 터미널 응용프로그램을 띄운 상태에서 키보드 스페이스바를 연속해서 5번 누르면 Table 5와 같은 Help 화면이 표시되면서 진입한다. Help화면이 보이지 않는다면 진입되지 않는것이므로 케이블 상태나 전원 상태를 확인해야 한다.

Ethernet을 사용하는 경우는 Telnet으로 터미널 모드를 진입할 수 있다. 초기 Telnet의 ID는 “root” 그리고, Password는 “can2eth” 이므로 이 값을 입력하여 로그인 하면 터미널 모드에 진입하고 마찬가지로 Help화면이 보이게 된다.

5.3 Terminal Commands

5.3.1 INI (CAN Initialize)

CAN 컨트롤러를 초기화 한다. CAN Error가 발생한 경우도 본 명령으로 clear할 수 있다. 정상적으로 처리 되면 OK, 그렇지 않은 경우는 ERROR가 표시된다.

5.3.2 EID (Extend ID Mode, 2.0B)

사용될 CAN ID의 bit수를 결정하는 명령으로, 11bit / CAN 2.0A와 29bit / CAN2.0B 를 선택할 수 있다. “EID” 를 입력하면 다음과 같은 선택 메뉴를 볼수 있으며 0 또는 1을 선택할 수 있다. 정상적으로 입력되면 OK 그렇지 않은 경우는 ERROR가 표시 된다.

```
>>EID
0: Standard ID(11bit) mode
1: Extended ID(29bit) mode
?
```

```
<< MULTI-CAN ANALYZER Terminal >>
+- syntax --+ function -----+
|INI          | CAN controller initialize|
|EID          | Set 11/29 bit ID mode   |
|SID [hex8]   | Set CAN ID (Hexa Code)  |
|CBR          | Set CAN baud rate      |
|SCP          | Set CAN parameters     |
|RBR          | Set RS232 baud rate    |
|RCV          | Start receiving message |
|SND          | Send message           |
|STS          | Request status         |
|CLR          | Clear error            |
|SIP          | Set IP Address         |
|SSM          | Set Subnet Mask        |
|SGW          | Set Gateway            |
|SMA          | Set MAC Address        |
|SIM          | Set IP Mode            |
|SPT [hex4]   | Set Port                |
|CPW          | Change Password        |
|DTS [hex1]   | Display Time Stamp(0/1) |
|VID [hex4]   | Change USB Vendor ID   |
|PID [hex4]   | Change USB Product ID  |
|EXT          | Exit terminal mode     |
+-----+
>>_
```

Table 5: Terminal Mode Help

위의 Terminal Mode Help에서 C2UA인 경우 Ethernet과 관련한 명령인, SIP / SSM / SGW / SMA / SIM / SPT / CPW 표시 되지 않고 동작되지 않으며, 제목도 “CAN-USB ANALYZER Terminal”로 표시된다.

5.3.3 SID (Set CAN ID)

수신받을 CAN ID를 설정한다. 본명령이 수행되면 지정된 CAN_ID를 갖는 메시지만 수신된다. 단 CAN ID가 0xFFFFFFFF 인 경우는 모든 CAN 메시지가 수신되고, 0xFFFFFFFFE인 경우는 이미 설정된 Filter & Mask에 의해 CAN 메시지가 필터링 되어 수신된다. 입력형식은 다음과 같이 16진수 8자리 ID를 입력한다.

```
>>SID 12345678
```

```
OK
>>
```

11비트 ID모드에서는 입력된 32비트 값중에 하위 11비트만 유효하고, 29비트 ID모드에서는 하위 29비트만 유효하다.

5.3.4 CBR (CAN Bit Rate)

CAN의 Bitrate을 설정하는 명령으로 “CBR”을 입력하면 다음과 같이 bitrate가 나열되고 1에서 8사이 값을 입력 할 수 있다. 8번 “User”를 선택한 경우는 “SCP”에서 설정한 Bit Timing에 맞게 속도가 설정되므로 “SCP”명령을 이용하여 미리 BRGCON1~3을 입력해야 한다.

```
>>CBR
0 : 125 kbps
1 : 250 kbps
2 : 500 kbps
3 : 1000 kbps
4 : 50 kbps
5 : 100 kbps
6 : 200 kbps
7 : 400 kbps
8 : User
Press the No.[0-8] ?
```

정상처리되면 OK, 그렇지 않은 경우는 ERROR가 표시된다.

5.3.5 SCP (Set CAN Parameter)

“CBR” 명령에 의해 CAN Bit Rate가 “User”모드로 설정된 경우 사용자가 직접 Bit Timing값을 설정하기 위한 명령이다. 본 명령에 의해 입력되는 파라미터는 BRGCON1, BRGCON2, BRGCON3로 다음과 같이 16진수 형식으로 ‘ ’ 구분자로 3개를 연속 입력한다. 정상처리되면 OK, 그렇지 않은 경우는 ERROR가 표시된다.

```
>>SCP
BRGCON1(2) BRGCON2(2) BRGCON3(2) ? 07 1F 3A
OK
>>_
```

5.3.6 RBR (RS232 Baud Rate)

RS232C의 Baud Rate를 변경하는 명령으로 “RBR”을 입력하면 다음과 같은 화면이 표시되고 0과 6사이 값

을 입력할 수 있다. 정상처리되면 OK, 그렇지 않은 경우는 ERROR가 표시된다.

```
>>RBR
0 : 9600 bps
1 : 19200 bps
2 : 38400 bps
3 : 57600 bps
4 : 115200 bps
5 : 230400 bps
6 : 460800 bps
Press the No.[0-6] ?
```

RS232C baudrate를 변경한 경우는 정상적인 동작을 위해 Analyzer의 전원을 껐다 켜다.

5.3.7 RCV (CAN Message Receive)

수신되는 CAN메시지를 실시간으로 표시해준다. DTS에 의해 TimeStamp으로 표시 설정에 따라 Time Stamp가 표시될 수도 있다. 표시형식은 다음과 같다. 첫번째 컬럼은 10us단위 Time Stamp, 두번째는 CAN ID, 세번째는 데이터 길이, 마지막은 데이터 이다.

```
>>RCV
[[ Data Logging Start ]]
132357553:BA:8:D0 D1 D2 D3 D4 D5 D6 D7
132357654:BB:8:D8 D9 DA DB DC DD DE DF
132357753:BC:8:E0 E1 E2 E3 E4 E5 E6 E7
132370854:F8:8:C0 C1 C2 C3 C4 C5 C6 C7
[[ Data Logging Stop ]]

>>_
```

표시되는 CAN ID의 최상의 31번째 비트는 29비트 CAN ID를 의미하고, 30번째 비트는 RTR을 의미 하므로 주의해야 한다. 예를 들어 CAN ID가 0x80000123이면 29비트 CAN ID 0x123을 의미 한다.

수신 모드를 종료하려면 [Enter] 키를 입력한다.

5.3.8 SND (CAN Message Send)

CAN 메시지를 전송하는 명령으로 “SND”를 입력하면 다음과 같은 화면이 표시된다.

```
>>SND
FORMAT (Hexa code) : ID(8) data1(2) data2(2) ... data8(2)
?00000123 11 22 33 44 55
OK
```

```
>>SND
FORMAT (Hexa code) : ID(8) data1(2) data2(2) ... data8(2)
? 80000001 01 02 03 44 55
OK
>>
```

모든 값은 16진수로 해야하며, CAN ID는 8자리, Data는 2자리 단위로 “ ”(SPACE)로 구분하여 입력한다. 데이터 길이는 입력된 수만큼 자동으로 계산된다. CAN ID의 31번째 비트를 1로 설정한 값(예 80000001)을 입력하면 29비트 CAN ID로 전송된다.

5.3.9 STS (CAN Status)

현재 Analyzer의 각종 상태 및 설정값들을 표시하는 명령이다.

```
>>STS
Model          MSMCA100
Serial No      00000000
Version        1.0
USB VID/PID    04D8/FFEE
CAN_ID         FFFFFFFF (All Accepted)
CAN BaudRate   1000 kbps
CAN Param      00/00/00
RS232 BaudRate 115200 bps
Error Warning  0
Rx/Tx Warning  0/0
Rx/Tx Bus passive 0/0
Bus off        0
Overrun        0 (HW) :0 (RX FIFO) :0 (TX FIFO)
Error Count    0 (RX) /0 (TX)
Extend ID mode 0
Fifo Overrun   USB (RX:0/TX:0) , UART (RX:0/TX:0)
Mask1,2        00000000,00000000
Filter1-2      00000000,00000000
Filter3-6      00000000,00000000,00000000,00000000
IP Address     192.168.10.100
Subnet Mask    255.255.255.0
Gateway        192.168.10.1
MAC Address    00-04-A3-14-6E-1C
Port Number    D431
IP mode        Static
CAN Time Stamp Enable
```

>>_

5.3.10 CLR (Clear CAN Error)

CAN Error가 발생한 경우 본 명령으로 Clear할 수 있다.

5.3.11 SIP (Set IP Address)

TCP/IP의 IP 주소를 설정하는 명령으로 “SIP”를 입력하면 다음과 같은 화면이 표시된다. (C2UA 제품은 해당없음)

```
>>SIP
FORMAT Example (Decimal) : 192.168.1.1
?192.168.10.100
```

값의 입력형식은 십진수를 “.”으로 구분하여 4자리를 입력한다. 본 명령은 Telnet이나 C2E 가 연결된 경우는 다음과 같이 표시되면서 설정되지 않기 때문에 반드시 Telnet이나 C2E를 접속하지 않은 상태에서 RS232C나 USB로만 설정해야 한다.

```
>>SIP
You cannot change IP address when telnet or c2e is connected !
>>_
```

5.3.12 SSM (Set Subnet Mask)

TCP/IP의 Subnet Mask를 설정하는 명령으로 “SIP” 명령과 동일한 형식 취하므로 “SIP”명령을 참고한다. (C2UA 제품은 해당없음)

5.3.13 SGW (Set Gateway Address)

TCP/IP의 Gateway Address를 설정하는 명령으로 “SIP” 명령과 동일한 형식 취하므로 “SIP”명령을 참고한다. (C2UA 제품은 해당없음)

5.3.14 SMA (Set MAC Address)

Analyzer의 MAC Address를 설정하는 명령으로 “SMA”를 입력하면 다음과 같은 화면이 표시된다. (C2UA 제품은 해당없음)

```
>>SMA
FORMAT (Hexa code) : XX.XX.XX.XX.XX.XX
? 00.1A.00.00.00.00
OK
>>_
```

MAC Address는 모두 16진수로 입력하며 각 필드는 “.”으로 구분하여 6자리를 입력한다. MAC Address의 첫번째 바이트는 특별한 의미가 있으므로 가급적이면 0x00으로만 설정한다.

5.3.15 SIM (Set IP Mode)

IP모드를 설정하는 명령으로 “SIM”을 입력하면 다음과 같은 선택화면을 볼수 있으며 0-2사이 값을 입력할 수 있다. (C2UA 제품은 해당없음)

```
>>SIM
0. Static IP mode
1. Dynamic IP mode(DHCP)
2. DHCP Server (IP=192.168.10.100)
?
```

Static IP모드는 고정 IP로 “SIP”등의 명령으로 입력된 IP를 사용하며, Dynamic IP모드는 Network에 존재하는 DHCP server로부터 IP를 할당받는다. DHCP Server모드는 Analyzer에 내장된 DHCP server에 의해 Analyzer가 아닌 PC와 같은 다른 장치들의 IP를 할당해 준다. 이 모드는 주로 PC와 Analyzer가 직접 연결된 경우나, 네트워크 상에 DHCP서버가 없는 경우에 사용될 수 있다. DHCP Server모드에서 Analyzer의 IP주소는 “SIP” 명령으로 지정된 IP가 사용되며 위 화면 처럼 “2. DHCP Server (IP=Analyzer Ip주소)” 형식으로 표시 된다.

5.3.16 SPT (Set C2E TCP Port No)

C2E 프로토콜을 사용하는 경우 사용될 TCP Port번호를 설정하는 명령으로 “SPT 포트번호” 형식으로 입력한다. 포트번호는 반드시 4자리 16진수로 입력해야 한다. 다음은 입력 예이다. (C2UA 제품은 해당없음)

```
>>SPT D431
OK
>>_
```

5.3.17 CPW (Change Telnet Password)

Telnet의 비밀번호를 변경하는 명령이다. 초기값은 “can2eth” 임 “CPW”를 입력하면 다음과 같은 화면이 표시되고 최대 8자리의 비밀번호 문자열을 입력한다. (C2UA 제품은 해당없음)

```
>>CPW
New password (Max 8 characters) ? *****
OK
>>_
```

본 터미널에서는 입력된 비밀번호를 재확인 하지 않으므로 입력시 주의를 요한다.

5.3.18 DTS (Display Time Stamp)

“RCV” 명령에 의해 CAN 메시지가 수신될 때 Time Stamp를 표시할지를 결정하는 명령으로 “DTS 1” 형식으로 입력한다. 입력 인자가 1인 경우는 표시하고, 0인 경우는 표시하지 않는다.

5.3.19 VID (Change USB Vendor ID)

PC에서 인식되는 USB장치의 VID를 변경한다. VID를 변경해야 하는 특별한 경우만 사용하고 그외에는 절대 변경하지 않는다. VID나 PID를 변경하는 경우는 드라이버 설치파일중 “*mchpusb.inf*” 파일의 내용중에 다음 부분을 변경해야 한다. 파일은 텍스트 편집기같은 “*notepad.exe*” 등으로 편집할 수 있다.

```
[DeviceList]
%DESCRIPTION%=DriverInstall, USB\VID_04D8&PID_FFEE

[DeviceList.ntamd64]
%DESCRIPTION%=DriverInstall64, USB\VID_04D8&PID_FFEE
```

위 내용중 “VID_” 다음과 “PID_” 다음의 16진수 4자리 숫자를 변경하려는 값으로 수정해야 한다.

(C2EA 제품은 해당없음)

5.3.20 PID (Change USB Product ID)

PC에서 인식되는 USB장치의 PID를 변경한다. PID를 변경해야 하는 특별한 경우만 사용하고 그외에는 절대 변경하지 않는다. 드라이버 관련 변경 사항은 5.3.19 절을 참고한다. (C2EA 제품은 해당없음)

5.3.21 EXT (Exit Terminal)

터미널 모드를 종료하는 명령이다. 터미널 모드를 종료하지 않으면 패킷모드로 동작되지 않으므로 반드시 터미널 사용이 완료되면 종료해야 한다.

6 Protocol Analyzer Application (CANTALKER)

6.1 Overview

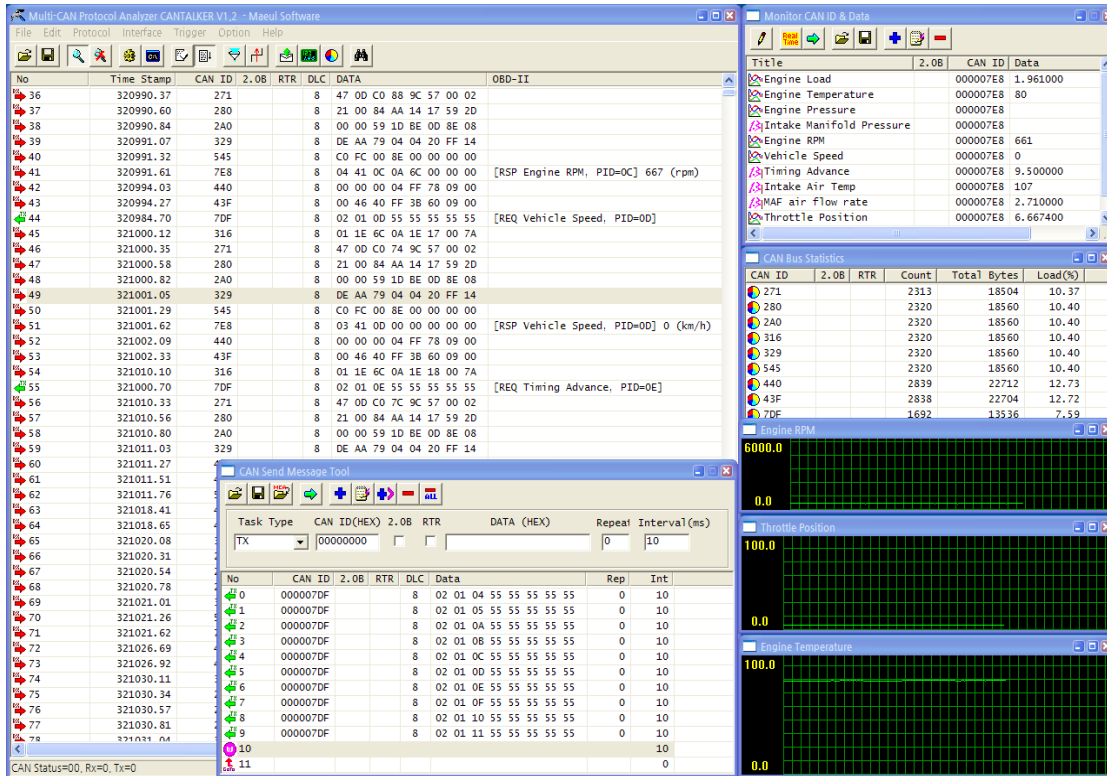


Figure 8: Protocol Analyzer Application

Multi-CAN Protocol Analyzer의 기능을 최대한 이용할 수 있는 응용프로그램으로 Figure 8 과 같은 화면을 볼 수 있다. 본 CANTALKER는 Windows전용으로 다른 OS는 지원하지 않으며 USB인터페이스를 이용하는 경우는 반드시 사용전 USB드라이버가 설치되어 있어야 한다. CANTALKER는 기본적인 CAN 메시지의 송수신 기능 이외에 CAN 상위 프로토콜인 CANopen, OBD-II, J1939 그리고 DeviceNet일부 프로토콜 분석까지 지원한다. 추가 적으로 메시지 수신조건을 결정할 수 있는 Filter와 Trigger기능과 CAN Bus 통계 및 모니터링을 지원한다.

CANTALKER는 Analyzer가 지원하는 모든 인터페이스를 지원하여 휴대성 및 장치 접근성이 매우 용이 하다. Window는 크게 [Menu], [Tool Bar], [Status Bar], [메시지 리스트]로 구성되어 있으며 다음 절부터 설명된다.

6.2 CAN Message List

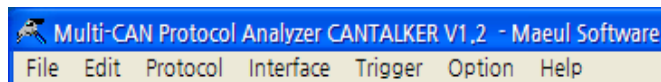
리스트는 다음과 같은 항목으로 구성되어 있다.

No	Time Stamp	CAN ID	2.0B	RTR	DLC	DATA	OBD-II
----	------------	--------	------	-----	-----	------	--------

각 항목별 의미는 다음과 같다.

1. No : 1씩 자동 증가하는 숫자이다. 앞에 표시되는 아이콘은 메시지의 방향을 표시한다.
2. Time Stamp : 메시지가 송수신 될때의 ms 단위의 Time값을 의미 하며 0.01ms 해상도를 갖는다.
3. CAN ID : 송수신 되는 CAN ID를 표시 한다.
4. 2.0B : 29비트 CAN ID인 경우 “v” 포 표시 된다.
5. RTR : Remote Frame 메시지인 경우 “v”로 표시 된다.
6. DLC : CAN메시지의 Data 길이를 의미 한다.
7. DATA : CAN메시지의 Data를 표시 한다.
8. CANopen/OBD-II/... : 선택된 프로토콜로 해석된 내용이 표시된다. (C2UA/C2EA 제품은 해당없음)

6.3 Menu



CANTALKER에서 지원하는 Menu는 위와 같으며 다음절 부터 하위 메뉴에 대해 설명된다.

6.3.1 File

1. Open
 - “mca” 확장명을 갖는 저장된 CAN메시지를 읽어온다.
 - [CTRL+O] 키입력으로 동일한 기능을 수행한다.
2. Save
 - 현재 표시되고 있는 수신된 CAN 메시지를 확장명 “mca”로 저장한다.
 - [CTRL+S] 키 입력으로도 동일한 기능을 수행한다.
3. Save As
 - 새로운 이름을 지정하여 “Save” 메뉴를 수행한다.
4. Export
 - Text파일 형식으로 저장한다. 저장형식은 다음과 같다.

No	TimeStamp	DIR	CAN_ID	2B	RTR	DLC	DATA	CANopen
00000	6368575.86	->	00000000			8	00 01 02 03 04 05 06 07	[NMT] Unknown Service (CS=0) (Node ID=1)
00001	6368576.98	->	00000001			8	08 09 0A 0B 0C 0D 0E 0F	[NMT] Unknown Service (CS=8) (Node ID=9)

00002	6368577.94	->	00000002	8	10	11	12	13	14	15	16	17	[NMT]	Unknown	Service (CS=16)	(Node ID=17)
00003	6368578.94	->	00000003	8	18	19	1A	1B	1C	1D	1E	1F	[NMT]	Unknown	Service (CS=24)	(Node ID=25)

- Protocol 이 선택된 경우에는 관련 정보도 추가로 표시된다.

5. Exit

- CANTALKER를 종료한다.

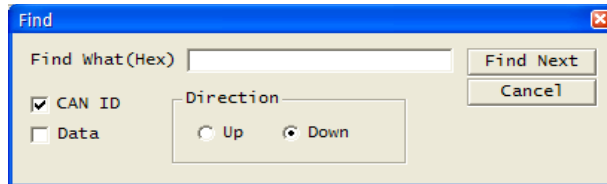
6.3.2 Edit

1. Copy

- CAN메시지 리스트에서 선택된 항목을 클립보드로 복사 한다.
- [CTRL+C] 키입력으로 동일한 기능을 수행한다.
- 복사되는 형식은 “Export” 메뉴의 저장 형식과 동일 하다.

2. Find

- 아래와 같은 창이 표시되면서 수신 메시지 리스트에서 CAN_ID를 기준으로 또는 Data를 기준으로 검색할 수 있다. 데이터는 메시지의 일부만 입력해도 되며 모든 값은 16진수로 입력해야 한다.



- Direction에 따라 검색 방향이 변경된다.
- [CTRL+F]키로 동일한 기능을 수행한다.

3. Filter

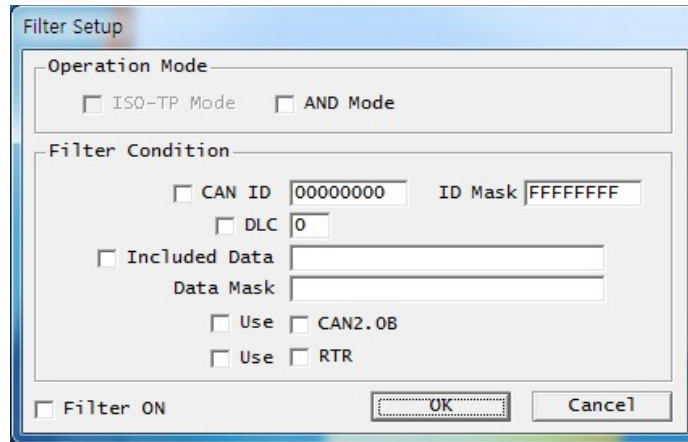
- 수신되는 메시지에 대해 필터링 기능을 제공한다. 본 Filter는 CAN 설정에 있는 Filter & Mask와는 다른 개념으로 Filter & Mask를 통해 하드웨어적으로 1차 필터링된 모든 메시지에 대해 소프트웨어적으로 필터링을 수행한다. 따라서 가급적 Analyzer로 사용하는 경우는 CAN의 HW필터 기능을 사용하지 않고 모든 메시지를 수신하게 한다.
- 본 기능은 이미 수신된 메시지는 적용되지 않고 설정된 이후 부터 적용된다. 그러나, Filter On된 상태에서 MCA파일을 Open 하는 경우는 필터링되어 로드된다.
- AND Mode가 체크되면 Filter Condition에서 선택된 항목들이 모두 조건이 맞을 때 Filtering되고, 체크되지 않으면 OR조건으로 선택된 항목중 한가지라도 맞을 때 Filtering된다. 여기서 Filtering된것은 Cantalker에 기록되고 그렇지 않은것은 버려진다.

- CAN ID 와 ID Mask의 동작하는 개념은 수신된 CAN프레임의 CAN ID와 Mask를 비트 AND시키고 그 값이 CAN ID와 맞는지를 확인한다. 수식으로 표현하면 다음과 같다.

$$(\text{수신_CAN_ID} \ \& \ \text{ID_Mask}) == \text{CAN_ID}$$

예를들어 CAN_ID가 7E0~7EF인 모든 CAN메시지를 수신하려면 다음과 같이 설정한다.

$$\text{CAN_ID}=0x7E0, \ \text{ID_MASK}=0xFF0$$



- Included Data는 수신된 CAN 프레임의 데이터중 Included Data가 포함되어 있는지를 확인하는 것으로, CAN ID와 비슷한 개념으로 다음과 같이 필터링을 수행한다.

$$(\text{CAN 일부데이터} \ \& \ \text{DATA_MASK}) == \text{INCLUDED_DATA}$$

예를들어 데이터에 61 E0 ~ 61 EF 까지 16가지 경우의 2바이트 데이터가 포함되어 있는 것을 수신하려면 다음과 같이 설정한다.

$$\text{INCLUDED_DATA}=61 \ \text{E0}, \ \text{DATA_MASK}=FF \ \text{F0}$$

- [CTRL+L] 키입력으로 동일한 기능이 수행된다.
4. Clear Display
- 메시지 리스트에 표시되는 모든 내용을 지운다.
 - [CTRL+X] 키입력으로 동일한 기능이 수행된다.

6.3.3 Protocol

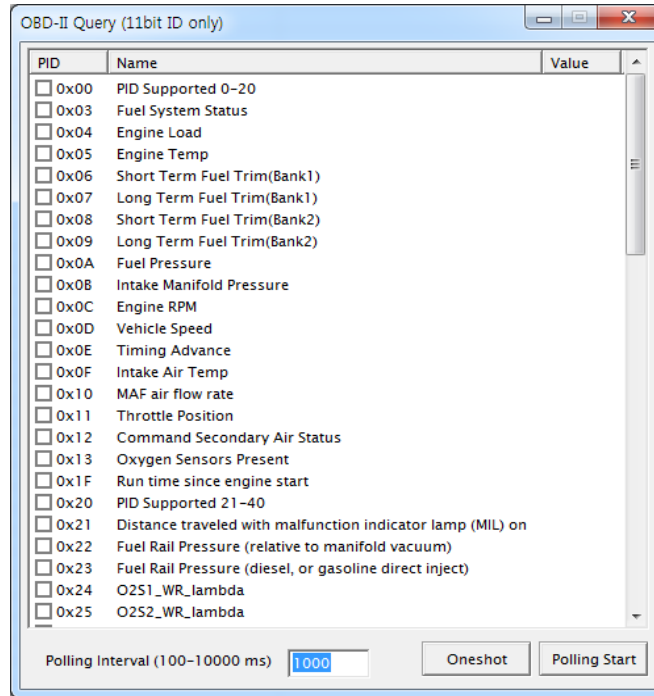
메시지 수신 리스트에 표시되는 항목과 관련된 해석될 프로토콜을 선택하는 메뉴 이다.

1. CAN Raw Data

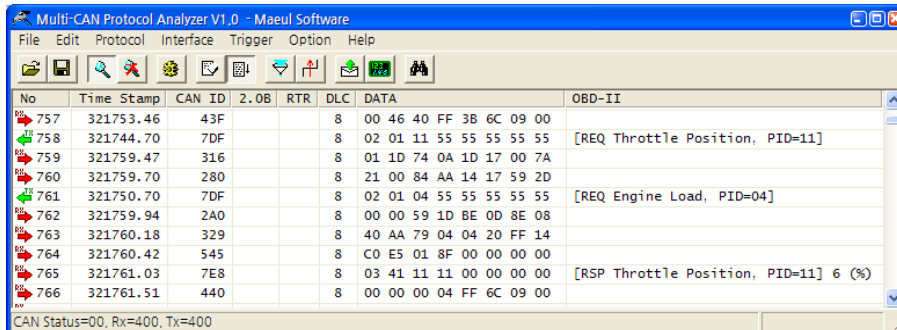
- 항상 선택되기 때문에 리스트에도 CAN 데이터는 항상 표시된다.

2. OBD-II

- 수신된 메시지를 OBD-II 프로토콜로 해석하여 리스트의 마지막 컬럼에 표시한다.
- 본 메뉴를 선택하면 다음과 같은 창이 표시된다.



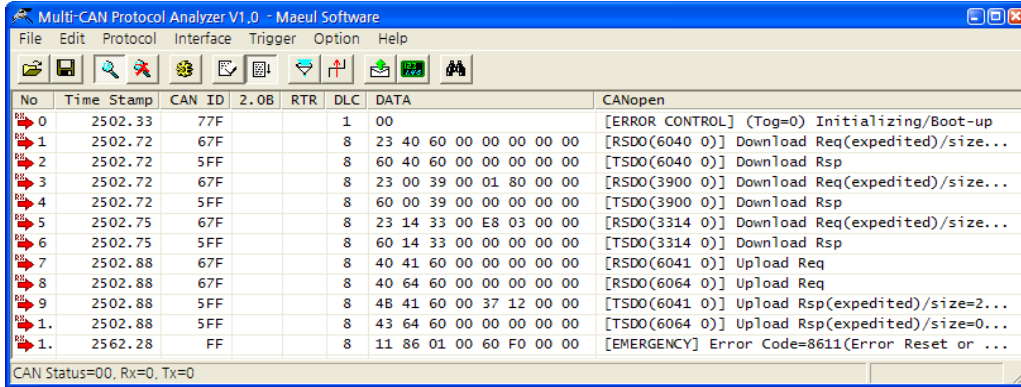
- ECU로 전달할 폴링 아이템을 선택하고 [Polling Start]버튼을 클릭하면 ECU로 정해진 주기로 CAN 메시지를 송신한다. 수신 메시지는 다음과 같이 메시지 리스트에 표시된다.



- Polling 주기는 100ms에서 10000ms 사이의 값으로 설정할 수 있다. (C2UA/C2EA 제품은 해당없음)
- 한번만 전송하려면 [Oneshot] 버튼을 클릭한다.

3. CANopen

- 수신된 메시지를 CANopen 프로토콜로 해석하여 메시지 리스트의 마지막 컬럼에 표시한다.



- 다음은 수신에 이다. (C2UA/C2EA 제품은 해당없음)

4. DeviceNet

- 수신된 메시지를 DeviceNet 프로토콜로 해석하여 메시지 리스트의 마지막 컬럼에 표시한다.
- 아직은 DeviceNet의 모든 프로토콜을 지원하지 않는다.

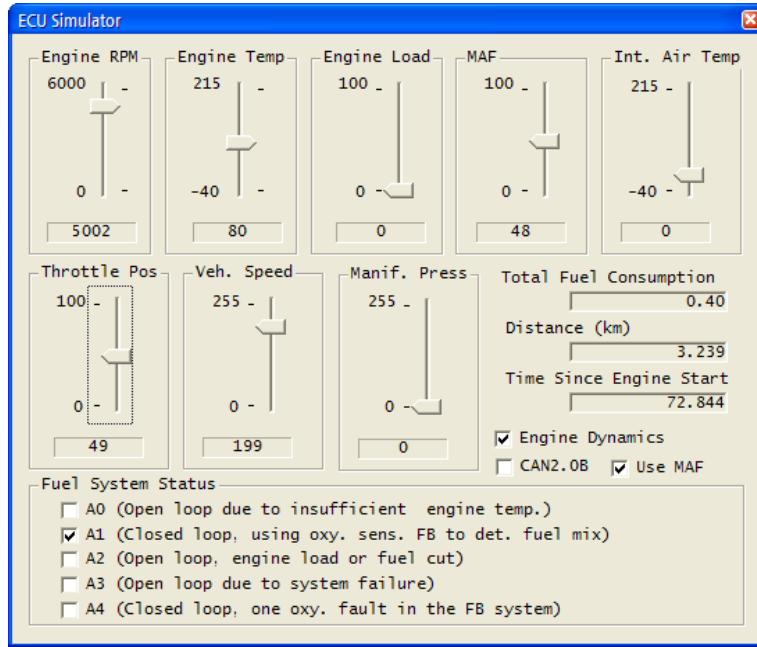
5. J1939

- 수신된 메시지를 J1939 프로토콜로 해석하여 메시시 리스트의 마지막 컬럼에 표시한다.
- 가로로 길게 표시되나, Copy & Paste 를 하거나 [File] 메뉴의 [Export] 로 저장하는 경우 다음과 같이 보기 좋게 표시 된다.

No	TimeStamp	DIR	CAN_ID	2B	RTR	DLC	DATA	J1939
00000	2562.28	->	18FFB200	v		8	39 00 FF FF FF FF 1A FF	Prio=6, DP=0, PGN=65266, SA=0, EngFuelRate=2.850 (L/hr) EngInstantaneousFuelEconomy=127.990 (km/L) EngAverageFuelEconomy=127.990 (km/L) EngThrottlePos=10.400 (%) EngThrottle2Pos=102.000 (%)

6. ECU Simulator

- OBD-II PID 명령에 대해 응답하는 ECU 기능을 에뮬레이션 한다. 실제 차량에 연결하기 전에 본 Analyzer에 연결하고 ECU Simulator로 동작시켜 장치가 정상적으로 동작하는지 확인 할 수있다.
- 기본적으로 각각의 슬라이드 값을 변경할 수도 있지만, [Engine Dynamics]를 체크하면 간단하게 엔진 동작을 시뮬레이션 할 수 있다. 이때에 입력가능한 값은 [Throttle Position]이며, 이 값의 변화에 따라 다른 값들이 자동으로 변경되며, 다른값은 사용자가 변경할 수 없다.
- 본 기능이 선택되면 다음과 같은 윈도우가 표시된다.



- [CAN2.0B] 를 선택하면 OBD-II PID 명령 전송시 CAN ID를 0x18DAF110로 보내며 0x18DB33F1 CAN ID에 수신 처리 한다. [Use MAF]를 선택하면 Supported ID에 대해 MAF센서의 지원 유무가 설정된다.
- 본 Simulator에서 지원되는 Mode는 1,2,3이고, 지원되는 PID는 다음과 같다.

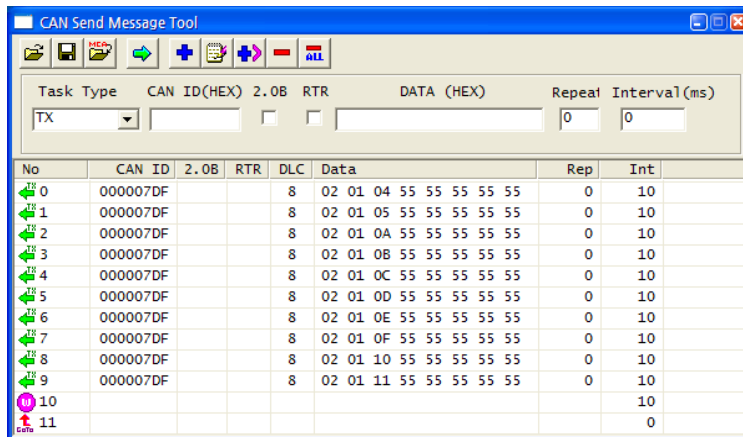
PID	Description	Mode 1 Value	Mode2 Value
0x00	Support PID 01-20	0xBE7FB013	0x70188001
0x01	Monitor status	0x8107EF80	-
0x02	Freeze DTC	-	0x0301
0x03	Fuel System Status	0x0201	0x0100
0x04	Engine load	Engine RPM Slider	0x17
0x05	Engine Temperature	Engine Temp Slider	-
0x06	Short Term Fuel Trim: Bank 1	0x3c	-
0x07	Long Term Fuel Trim: Bank 1	0x46	-
0x0A	Fuel Pressure	128	-
0x0B	Intake Manifold Pressure	Manif. Press. Slider	-
0x0C	Engine RPM	Engine RPM Slider	0x32C8
0x0D	Vehicle Speed	Veh. Speed Slider	0x27
0x0E	Timing Advance	128	-
0x0F	Intake Air Temperature	Int. Air Temp. Slider	-
0x10	MAF air flow rate	MAF Slider	-
0x11	Throttle Position	Throttle Pos. Slider	0x72
0x13	Location of Oxygen Sensors	1	-
0x14	Oxygen Sensor Voltage and Short	0x8080	-

	term fuel trim		
0x1C	OBD Type	1	-
0x1F	Time Since Engine Start	Internal Timer	-
0x20	Support PID 21-40	0x80022001	0x81800000
0x21	Distance Traveled While MIL is Activated OBD Type	Distance calculated from Veh. Speed	0x01F2
0x2C	Commanded EGR	-	0x22
0x2D	EGR Error	-	0x84
0x2F	Fuel Level Input	128	-
0x33	Barometric pressure	100	-
0x40	Support PID 41-60	0x44008010	-
0x42	Control module voltage	0x2EE0	-
0x46	Ambient air temperature	60	-
0x51	Fuel Type	5 (LPG)	-
0x5C	Engine oil temperature	130	-

Mode3 에 대한 응답값은 0x030000000000 이다.

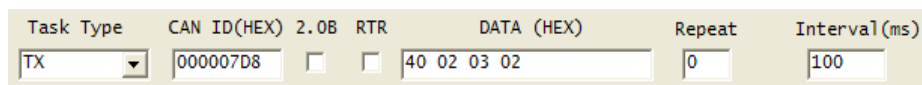
7. CAN Send Message

- CAN메시지를 이미 입력된 패턴에 맞게 전송하기 위한 기능을 제공한다.
- 메뉴를 선택하면 다음과 같은 창이 표시 된다.



- Task Type에 맞게 작업내용을 기록한 후 **+** (Add)버튼을 클릭하여 작업 리스트에 추가한다. Task Type 은 CAN메시지 전송을 위한 [TX], 대기 위한 [WAIT], 반복을 위한 [LOOP], 특정 위치로 이동하기 위한 [GOTO]가 있으며 각 Type에 따라 정보내용과 입력 방법이 달라진다.

1. [TX]



- 보내려는 메시지의 정보를 입력하고 [Add]버튼을 클릭하면 리스트에 추가 된다. 여기서 [Repeat]는 1회를 초과한 추가로 반복전송할 횟수를 의미하고, [Interval]은 반복 전송시 메시지의 시간간격을 의미 한다. 시간 단위는 millisecond 이다.
- CAN ID와 DATA는 모두 16진수로 입력한다.

2. [WAIT]

Task Type	CAN ID(HEX)	2.OB	RTR	DATA (HEX)	Repeat	Wait Time(ms)
WAIT	000007D8	<input type="checkbox"/>	<input type="checkbox"/>	40 02 03 02	0	100

- Wait Time 시간 동안 대기한다. 시간 단위는 millisecond 이다.

3. [LOOP].









Task Type	CAN ID(HEX)	2.OB	RTR	DATA (HEX)	Loop count	Start Idx
LOOP	000007E8	<input type="checkbox"/>	<input type="checkbox"/>	40 02 03 02	100	1000

- Start Index (No) 위치에서 Loop가 있는 위치까지 Loop Count만큼 반복 수행한다.

4. [GOTO]

Task Type	CAN ID(HEX)	2.OB	RTR	DATA (HEX)	Repeat	Goto Idx
GOTO		<input type="checkbox"/>	<input type="checkbox"/>		0	0

- Goto Index (No) 위치로 무조건 이동하여 수행한다.

-  (Modify)는 현재 작업리스트에서 더블클릭으로 선택된 항목에 대해 수정한 내용을 반영한다.
-  (Insert)버튼은 특정 위치에 메시지를 끼워넣을 때 사용하며 특정 위치는 리스트에서 마우스로 선택한다.
-  (Remove)는 리스트에서 선택된 항목을 삭제 한다.
-  (Remove All)은 모든 리스트 내용을 삭제 한다.
-  (Load) /  (Save)는 확장명이 "csm"인 Send Message 리스트파일을 저장하거나 읽어 올때 사용된다.
-  (Import)는 CAN 메시지 로그 파일인 "*.mca" 파일을 읽어 RX 메시지만 읽어와 리스트에 추가한다. 이 기능은 캡처된 CAN 버스 메시지 흐름을 그대로 시뮬레이션 하여 출력할 때 유용하다.
-  (Go)는 리스트에서 프로그래밍 된대로 전송을 수행한다.
- CANTALKER와 같이 제공되는 파일중 "obd_query.csm" 파일은 OBD-II Query명령에 대해 프로그램된 것이므로 참고한다. 이것과 연관되어 OBD-II Monitor 파일인 "obd_ii.mon" 파일도 있으므로 "Monitor CAN ID & Data" 툴에서 읽어 같이 사용하면 ECU모니터링에 효과적이다.

8. Monitor CAN ID & Data

- 특정 CAN ID에 대해 데이터를 모니터링 하는 기능으로 데이터의 가공까지 가능하다.
- 메뉴를 선택하면 Figure 9 과 같은 모니터 화면을 볼 수 있다.
- [Monitor Condition] 은 모든 수신되는 CAN메시지로부터 모니터링 하고자 하는 CAN ID와 데이터의 일부가 원하는 형태가 되는 메시지에 대해 데이터 중 어떤부분을 이용하여 어떻게 가공할 것인지를 결정한 후 리스트에 추가하기 위한 프레임이다.
- [Item Title]은 모니터링 하고자 하는 정보에 대한 이름을 입력한다.
- [CAN ID to monitor] 는 모니터링하고자 하는 CAN ID를 16진수로 입력한다. [CAN2.0B] 체크 박스는 29비트 CAN ID를 사용하는 경우 체크 한다.

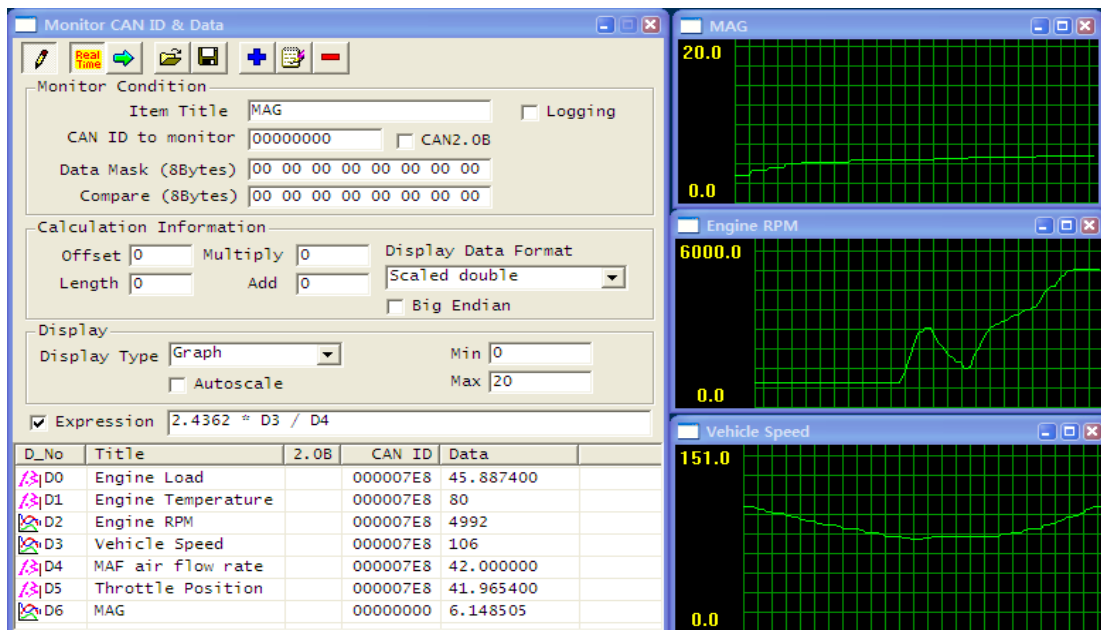


Figure 9: Monitor CAN ID & Data

- [Data Mask]와 [Compare]는 수신되는 데이터에 대해 [Data Mask]로 비트 AND를 취하고, 그 결과가 [Compare]와 맞는 경우의 데이터를 취한다. Mask와 Compare는 반드시 8바이트 16진수 형태로 입력해야 한다. 만약에 수신된 CAN메시지 데이터의 첫번째 바이트 중에 상위 4비트 값이 4인 경우를 모니터링하고자 한다면 [Data Mask]는 “F0 00 00 00 00 00 00 00”이고, Compare는 “40 00 00 00 00 00 00 00”이 된다. 개념적인 식으로 표현하면 다음과 같다









$$[\text{CAN Data}] \& [\text{Data Mask}] == [\text{Compare}]$$
- [Logging]이 체크된 경우 CALTalker가 실행중인 디렉토리에 [Item Title]을 이름으로 하고 확장명이 “log”인 “Item Title.log”형식의 파일로 Time Stamp와 최종 계산되는 Data가 기록 저장되어 진다. 로그 파일은 (Monitor Start)버튼이 클릭되면 새롭게 생성되어지므로 주의해야 한다.

- [Offset]은 최대 8바이트를 갖는 데이터에서 몇번째 바이트위치부터 데이터를 사용할 것인지를 의미하는 값으로 0~7 범위의 값을 갖는다.
- [Length]는 [Offset] 떨어진 위치로부터 몇개의 바이트 데이터를 사용할지를 의미하는 값으로 [Offset+Length]는 8을 초과해서는 안된다.
- [Big Endian]은 [Offset]에서 [Length] 만큼 얻은 바이트 배열을 Big Endian(좌->우)형식으로 인식할지, Little Endian(우->좌)으로 인식할지를 결정하는 것으로 체크되면 Big Endian으로 인식된다. 이렇게 얻어진 최종 값을 [X]라 한다.
- Multiply는 [X]에 대해 곱해지는 값을 의미하고, Add는 더해지는 값이된다. 따라서 [Scaled] 가 붙는 [Display Data Format]이 선택된 경우 $[Y = X * Multiply + Add]$ 와 같은 계산을 거쳐 최종 [Y]값이 사용되어진다.
- [Display Data Format]은 다음과 같은 종류가 있다.
 1. All Data : 데이터를 아무런 가공없이 한바이트씩 띄어서 표시됨
 2. Hex : [X]를 16진수로 표시
 3. Binary : [X]를 2진수로 표시
 4. Scaled Integer : [Y]를 부호 있는 정수형으로 표시
 5. Scaled Unsigned Integer : [Y]를 부호 없는 정수형으로 표시
 6. Scaled Double : [Y]를 부호 있는 부동소수로 표시
 7. Scaled Unsigned Double : [Y]를 부호 없는 부동소수로 표시
- [Display Type]은 다음과 같은 종류가 있다.
 1. Number ; [Display Data Format]에서 지정된 형태의 숫자로 표시
 2. Bar ; Display Data Format이 [Scaled xx]형식인 경우 Bar 형태로 표시(향후지원)
 3. Graph ; Display Data Format이 [Scaled xx]형식인 경우 Graph 형태로 표시
 4. Gauge : Display Data Format이 [Scaled xx]형식인 경우 Gauge 형태로 표시(향후지원)
- [Auto Scale]은 Graph/Gauge/Bar 형태일 때 Min/Max 입력값에 상관없이 실시간으로 표시되는 값의 최대/최소값에 맞춰 좌표축의 최대/최소값이 자동으로 갱신되는 기능이다.
- [Min] [Max]는 Graph/Gauge/Bar 형태일 때 좌표축에 대한 최소/최대값을 나타낸다.
- [Expression]이 체크 되어 있는 경우는 CAN ID 기반으로 데이터를 계산하지 않고 이미 리스트에 계산된 결과값을 기반으로 한 계산식이 사용된다. 따라서 이 기능이 체크되어 있으면 CAN 관련 정보는 입력할 필요없이 D_No 컬럼에 해당하는 기호값과 숫자 그리고 +, -, *, / 연산자를 사용하여 수식을 입력

할 수 있다. 예를 들어 Figure 9 의 D6라인의 MAG의 수식은 다음과 같다.

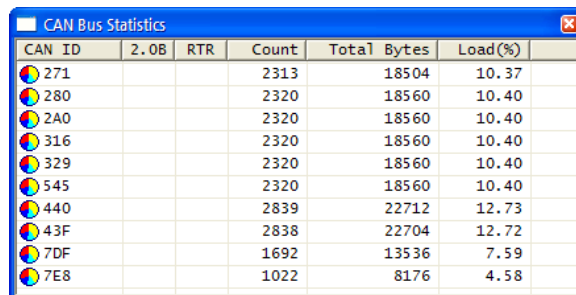
$$2.4362 * D3 / D4$$

여기서, D3는 Vehicle Speed, D4는 MAF air flow rate 값을 의미 한다. 수식은 연산자의 우선순위와 종류에 상관없이 맨 앞에서부터 차례로 계산된다. 수식의 입력시 주의할 점은 모든 항목에 대해 반드시 스페이스키로 띄어 쓰기를 해야 한다. 예를들어 위 수식은 “2.4362 *_D3/_D4” 와 같은 식이 되어야 한다. 최대로 추가할 수 있는 Item갯수는 128로 제한되기 때문에 수식에 사용될 수 있는 D_No는 D0 ~ D127 까지만 사용 가능하다.

-  (Add)는 [Monitor Condition]에 입력된 값을 리스트에 추가하는 버튼이다.
-  (Modify)는 리스트에서 더블클릭으로 선택된 항목에 대해 수정한 내용을 적용하는 버튼이다.
-  (Remove)는 현재 리스트에서 선택된 항목을 삭제하는 버튼이다.
-  (Load)는 저장된 Monitor Condition리스트를 읽어오는 버튼이다.
-  (Save)는 현재 리스트에 등록된 Monitor Condition을 저장하는 버튼이다.
-  (Monitoring Start) 또는 (Stop)버튼은 수신되는 CAN메시지로부터 모니터링을 시작하거나 중지하는 버튼이다.
-  (Realtime)이 체크된 경우는 실시간으로 수신되는 메시지에 대해 처리하여 실시간으로 표시 및 로그 파일로 저장하며, 체크되지 않은 경우는, 이미 수신되거나 Main Toolbar에서  (Load) 버튼으로 Load된 “mca” 파일의 데이터에 대해 후처리하여 표시 및 로그파일로 저장한다. 이미 Monitoring이 시작된 경우는 [RealTime]을 변경할 수 없다. **(View Editor)는 Item을 추가/삭제/변경하기 위한 편집창을 보여주는 버튼이다. 체크된 경우는 편집창이 보이고, 그렇지 않은 경우는 Item리스트만 보인다.**
- from Statistics) 는 CAN Bus Statistics창에 표시되는 모든 CAN ID를 추가할 때 사용한다.

9. CAN Bus Statistics

- CAN Bus에 송수신 되는 모든 메시지들의 CAN ID 분포와 그것의 메시지 갯수등 버스의 통계정보가 아래와 같이 표시되며, 1초 단위로 갱신된다.



CAN ID	2.0B	RTR	Count	Total Bytes	Load(%)
271			2313	18504	10.37
280			2320	18560	10.40
2A0			2320	18560	10.40
316			2320	18560	10.40
329			2320	18560	10.40
545			2320	18560	10.40
440			2839	22712	12.73
43F			2838	22704	12.72
7DF			1692	13536	7.59
7E8			1022	8176	4.58

- 표시 항목으로 CAN ID, 각 CAN ID에 대한 수신 메시지수 및 전체 수신량(bytes), 그리고 전체 메시지

수신량에 대한 상대 비율 (Load) 이 있다. 표시 내용은 마우스나 키보드로 선택할 수 있고 [CTRL+C] 키를 입력하면 클립보드로 복사되며, 텍스트 편집기 등에서 [CTRL+V] 를 입력하여 복사한 내용을 붙여넣기 할 수 있다.

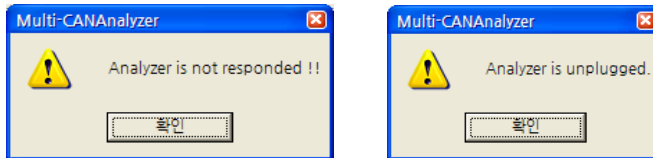
10. C Interpreter

- 7 장을 참고한다.

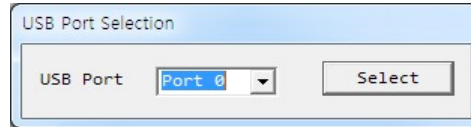
6.3.4 Interface

1. Connect

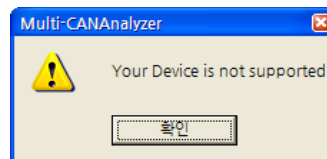
- 선택된 인터페이스로 접속을 시도한다. 접속이 원활하지 않거나 사용중 인터페이스 컨넥터나 플러그를 제거하면 다음과 같은 창이 표시된다. USB인 경우는 오른쪽 창이 표시된다.



- USB인 경우는 여러 장치가 꽂힌 경우 장치를 구분 및 선택하기 위해 다음과 같이 창이 먼저 표시되므로, 사용자는 해당 장치의 포트를 선택해야 한다.



- LAN선을 꼽은 후 즉시 [Connect]를 클릭하는 경우는 PC에서 Ethernet라인을 확인중에 있기 때문에 연결이 안될 수 있다. 따라서 LAN선을 꼽고 10초 정도 지난후에 [Connect]하는게 바람직 하다.
- CANTALKER는 Analyzer H/W와 초기 접속시에 CANTALKER와 Analyzer에 대해 각각 인증을 수행하는데 두가지 인증에 성공해야지 정상적으로 CANTALKER를 사용할 수 있다. 따라서 전용 H/W가 아닌 경우 CANTALKER동작이 불가능하며, 향후 H/W 모델에 따라 S/W의 동작이 달라지거나 동작하지 않을 수 있다. 이러한 경우는 H/W에 맞는 SW버전을 사용해야 한다.
- 인증에 실패하는 경우는 다음과 같은 창을 볼수 있다.



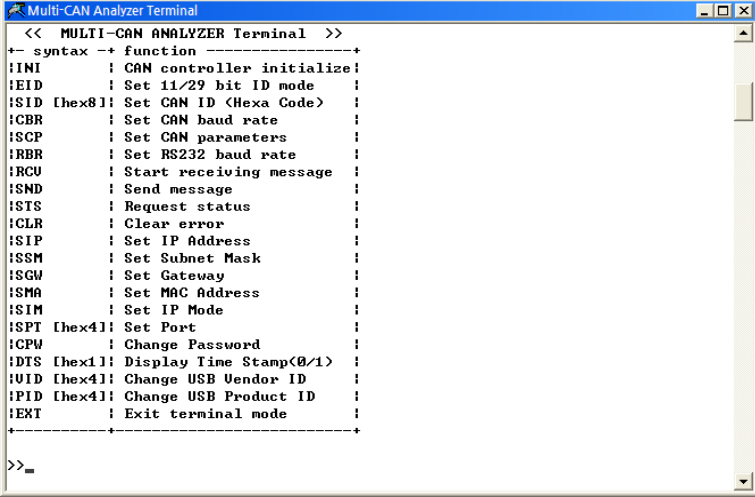
2. Disconnect

- 접속을 해제 한다. 사용중 사용자가 접속을 해제 하지 않은 상태에서 인터페이스 플러그 또는 컨넥터

를 제거하여도 팝업 메시지와 함께 자동으로 접속이 해제된다.

3. Terminal

- CANTALKER내에서 Terminal 기능을 사용 할 수 있고 메뉴가 선택되면 다음과 같은 화면이 표시된다. 단 Ethernet인터페이스를 사용하는 경우는 지원하지 않으며 Telnet을 이용해서 Terminal을 사용할 수가 있다.



```

Multi-CAN Analyzer Terminal
<< MULTI-CAN ANALYZER Terminal >>
+- syntax -+ function
!INI      ! CAN controller initialize!
!EID      ! Set 11/29 bit ID mode    !
!SID [hex8]! Set CAN ID (Hexa Code) !
!CBR      ! Set CAN baud rate      !
!SCP      ! Set CAN parameters     !
!RBR      ! Set RS232 baud rate     !
!RCU      ! Start receiving message !
!SND      ! Send message            !
!STS      ! Request status          !
!CLR      ! Clear error             !
!SIP      ! Set IP Address          !
!SSM      ! Set Subnet Mask         !
!SGW      ! Set Gateway             !
!SMA      ! Set MAC Address         !
!SIM      ! Set IP Mode             !
!SPT [hex4]! Set Port              !
!CPW      ! Change Password        !
!DTS [hex1]! Display Time Stamp(0/1) !
!UID [hex4]! Change USB Vendor ID  !
!PID [hex4]! Change USB Product ID !
!EXT      ! Exit terminal mode     !
-----
>>=

```

- Terminal모드와 Packet모드가 동시사용할 수 없는 이유 때문에 CANTALKER에서 Terminal기능을 사용하게 되면 다른 기능은 전혀 사용할 수 없으므로 CANTALKER의 모든 기능을 사용하기 위해서는 반드시 Terminal에서 “ext” 와 [Enter] 키를 입력하거나 [CTRL+C]나 [ESC] 키를 입력하여 빠져 나와야 한다.

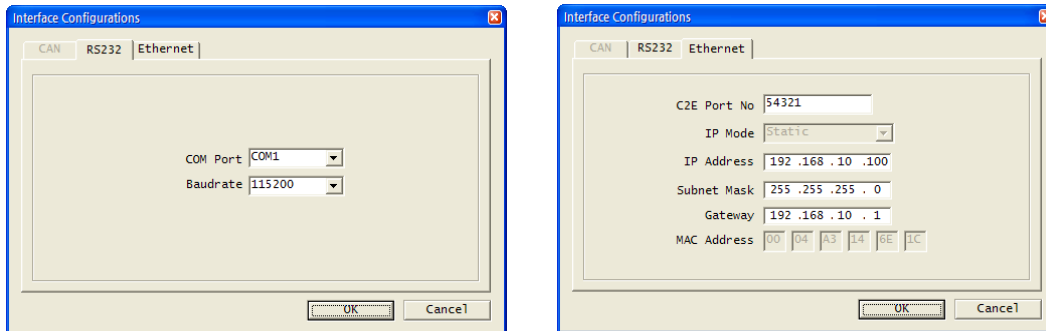
4. Interface Type

- USB / Ethernet / RS232C 중에 하나를 선택할 수 있다.
- Ethernet이나 RS232C인 경우는 [Connect]를 수행하기 전에 [Configuration] 메뉴에서 관련된 설정이 선행 되어야 한다.
- 접속이 잘되지 않는 경우 각 인터페이스별로 다음을 참고한다.
 - USB : 드라이버가 정상적으로 설치되었는지, USB plug가 정상적으로 꼽아 있는지, 그리고 [장치관리자] 에서 드라이버가 정상적으로 보이는 지 확인한다.
 - Ethernet : LAN선이 정상적으로 꼽아져 LAN소켓의 녹색LED가 켜져 있는지, TCP/IP설정이 정상적으로 맞는지, Analyzer에서는 Static IP(고정IP)모드로 되어 있는 경우 PC도 고정 IP로 설정되어 있는지 (자동할당(DHCP) IP 아님) 확인한다.

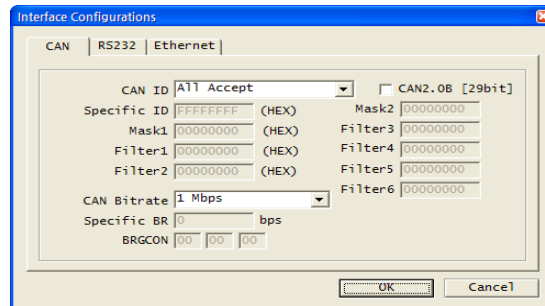
- RS232C : 1:1 RS232C케이블을 연결한것인지, 통신 속도를 서로 맞췄는지, 그리고 Analyzer에서 가능한 통신속도인지 확인한다.

5. Configuration

- CAN / RS232C / Ethernet 인터페이스를 설정하는 기능이다. Connect이전에는 CAN을 설정할 수 없고, RS232C는 Analyzer와 연결할 통신포트와 속도를 결정하고, Ethernet인 경우는 C2E프로토콜로 연결할 Analyzer의 IP주소와 TCP포트를 결정할 수 있다.



- Connect이후에는 다음과 같은 화면을 볼수 있으며 CAN통신 설정을 할수 있으며 RS232C의 경우는 Baud Rate가 Analyzer의 통신 속도를 의미한다.



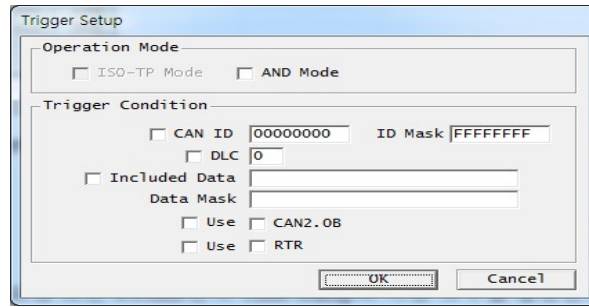
- Ethernet인터페이스로 연결된 경우는 Ethernet 설정값을 변경 할 수 없다.
- RS232C 인터페이스로 연결된 경우는 RS232C 설정값을 변경 할 수 없다.

6.3.5 Trigger

1. Trigger On

- Trigger Setup에서 설정된 환경으로 Trigger를 시작한다. 모든 송수신 메시지는 Trigger조건이 맞기 전 까지 Drop한다.

2. Trigger Setup



- 필터와 동일한 형식을 취한다. 여기서 설정된 조건은 [Trigger On] 메뉴가 선택되어야 Trigger기능이 비로소 수행된다.

6.3.6 Option

1. CAN H/W Time Stamp

- Analyzer의 CAN 컨트롤러에서 관리되고 있는 0.01ms 단위 Time Stamp의 사용 유무를 결정한다. 이 기능이 사용되지 않으면 Time Stamp는 Windows의 1ms 단위 멀티미디어 Timer값을 이용한다.

2. CAN Initialize

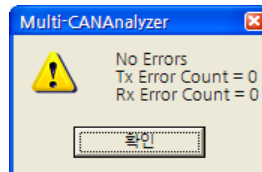
- CAN 컨트롤러를 초기화 한다.

3. CAN Error Clear

- CAN Error가 발생한 경우 초기화 한다.

4. CAN Status

- 현재 CAN 컨트롤러의 상태를 표시하며 다음과 같은 창이 표시 된다. CAN Error가 있으면 관련 정보가 구체적으로 보이며 Tx와 Rx에 대한 Error Count값도 표시 된다.



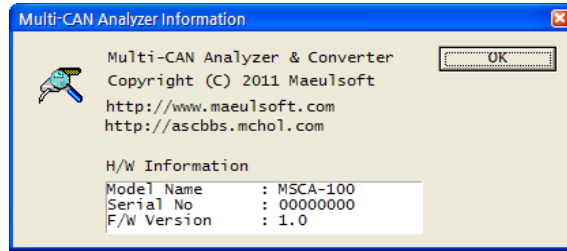
5. Auto Scroll

- 메시지 리스트의 자동 스크롤 조건을 선택한다. 선택된 경우는 가장 마지막으로 송수신 되는 메시지가 표시되도록 스크롤 된다.

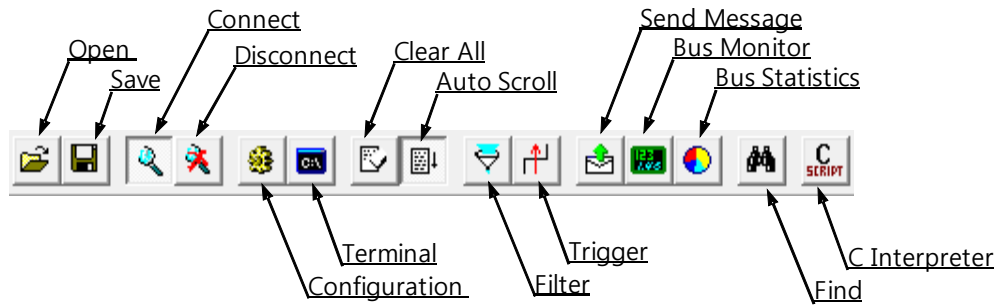
6.3.7 Help

1. About

- 다음과 같이 CANTALKER와 Analyzer의 기본정보를 표시 한다.

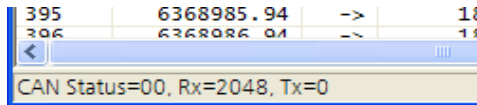


6.4 Tool Bar



각 버튼 별 의미 하는 내용은 위 그림과 같고 실제 기능은 관련된 메뉴를 참고한다.


6.5 Status Bar

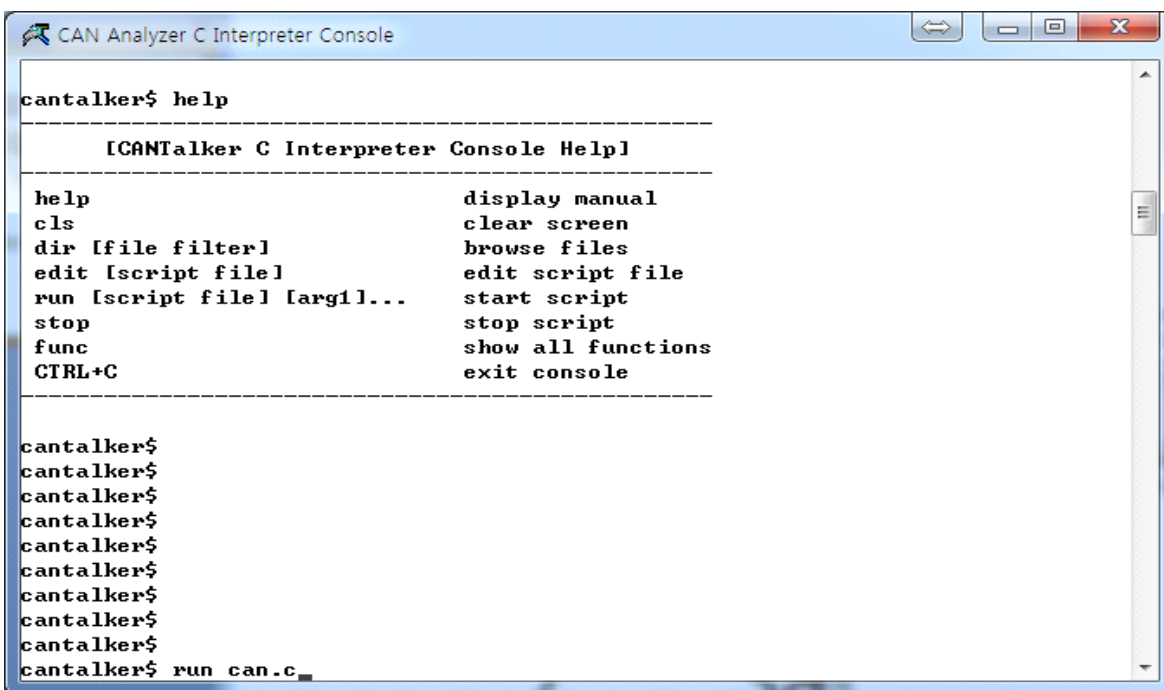


Status Bar에는 현재 16진수 CAN Status값과 송수신 메시지의 총 갯수가 1초주기로 표시 된다. 총 메시지 수는 [Clear All]이 수행되면 0으로 리셋된다.

7 C Interpreter

7.1 Overview

CANTalker환경에서 CANTalker에서 지원하는 기능들을 컴파일 하지 않고 C언어로 자유롭게 실시간 프로 그래밍 하여 보다 세밀하게 CAN입출력 제어 할 수 있다. 지원하는 C 언어는 표준 ANSI-C 규격을 대부분 따르고 있기 때문에 C언어에 대한 경험이 있다면 쉽게 사용할 수 있다. C Interpreter를 사용하기 위해서는 메뉴에서 [Protocol] → [C Interpreter Console] 을 선택하거나, Toolbar에서  버튼을 클릭하여 실행 할 수 있으며, 다음과 같은 console창이 표시된다.



```

CAN Analyzer C Interpreter Console
cantalker$ help
-----
[CANTalker C Interpreter Console Help]
-----
help                display manual
cls                 clear screen
dir [file filter]  browse files
edit [script file] edit script file
run [script file] [arg1]... start script
stop                stop script
func                show all functions
CTRL+C              exit console
-----
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$
cantalker$ run can.c

```

Figure 10: C Interpreter Console Window

7.2 Limitations

표준 ANSI-C를 지원하지만 CANTalker환경내에서 Interpreter로서 동작하기 때문에 다음과 같은 제약 조건 이 있다.

1. 함수 포인터는 사용 할 수 없다.
2. 이전 label에 대해 goto를 사용 할 수 없다.
3. #define은 함수 밖에서만 사용 가능하며 일반적인 expression만 사용 가능 하다.
4. #if defined는 지원 하지 않는다.

5. 구조체는 함수 밖에서만 선언 가능하며 비트 필드는 지원하지 않는다.
6. static, extern, volatile, register, auto 키워드는 지원하지 않는다.
7. 표준 입출력 함수로 출력하는 경우 Console창에서 Terminal ANSI Code는 동작하지 않는다.
8. Heap과 Stack이 공통으로 사용하는 메모리 크기는 4MB 이다.

7.3 Console Keys and Commands

7.3.1 Keys

Console창에서 입력 가능한 키는 알파벳, 숫자, 기호, UP, DOWN, LEFT, RIGHT, DEL, BS, END, HOME, TAB 이다. 입력했던 문장은 최대 16개까지 기억하며, [UP/DOWN] 키로 입력했던 문장을 재 표시할 수 있다. 입력한 라인의 문장을 편집하기 위해 DEL, BS, END, HOME 등의 키를 사용할 수 있다. TAB키는 [script] 디렉토리내에 입력한 글자로 시작하는 파일명을 차례로 자동 표시하여, 파일 입력을 간편히 할 수 있다.

[script] 디렉토리는 cantalker.exe 파일이 위치한 디렉토리에 “script” 란 이름으로 존재 한다.

7.3.2 Commands

1. “help “
 - 사용 가능한 명령에 대한 도움말을 보여준다.
2. “cls”
 - console 창의 화면의 글자를 모두 지우고, 커서를 가장 상위/좌측에 위치 시킨다.
3. “dir”
 - [script] 디렉토리 안의 C 파일들을 보여준다. “dir” 만 입력하면 확장자가 “.c” 파일만 보여주며, “*.txt” 와 같이 파일 필터를 이용하면 지정한 파일들을 보여 준다.
4. “edit”
 - [script] 디렉토리에 이미 존재하는 파일을 편집하거나, 새롭게 생성할 때 사용하는 명령이다. “edit C파일명”과 같은 형식으로 입력한다. 편집기는 notepad가 사용된다. 파일이 존재 하지않으면 새롭게 생성한다.
5. “run”
 - “script” 디렉토리에 존재하는 지정한 C파일을 읽어 해석 및 수행하기 시작한다. 이미 실행중에는 다시 실행할 수 없으며 반드시 “stop”으로 종료 후 실행해야 한다.
6. “stop”

- “run” 명령으로 실행중인 스크립트를 중지한다.

7. “func”

- 모든 내장 함수의 원형을 보여준다. 모든 함수는 7.4절의 “내장함수 목록”에서 설명된다.

8. [CTRL+C]

- Control키와 C키를 동시에 입력하면 Console창을 닫을 수 있다. Console창이 닫히면 자동으로 수행중인 스크립트는 중지 된다. 또한 Analyzer장치와 연결이 종료되는 경우도 Console창은 닫힌다.

7.4 Integrated Function List

7.4.1 Standard Input / Output Functions

ANSI-C의 표준 입출력함수와 동일하므로 특별한 설명은 생략한다. 표준 입출력은 Console창이며, FILE 함수를 이용하여 파일로의 입출력도 가능하다.

- FILE *fopen(char *, char *);
- FILE *freopen(char *, char *, FILE *)
- int fclose(FILE *)
- int fread(void *, int, int, FILE *)
- int fwrite(void *, int, int, FILE *)
- int fgetc(FILE *)
- int getc(FILE *)
- char *fgets(char *, int, FILE *)
- int fputc(int, FILE *)
- int fputs(char *, FILE *)
- int remove(char *)
- int rename(char *, char *)
- void rewind(FILE *)
- FILE *tmpfile()
- void clearerr(FILE *)

- `int feof(FILE *)`
- `int ferror(FILE *)`
- `int fileno(FILE *)`
- `int fflush(FILE *)`
- `int fgetpos(FILE *, int *)`
- `int fsetpos(FILE *, int *)`
- `int ftell(FILE *)`
- `int fseek(FILE *, int, int)`
- `void perror(char *)`
- `int putc(int, FILE *)`
- `int putchar(int)`
- `int fputc(int)`
- `void setbuf(FILE *, char *)`
- `void setvbuf(FILE *, char *, int, int)`
- `int puts(char *)`
- `char *gets(char *)`
- `int printf(char *, ...)`
- `int fprintf(FILE *, char *, ...)`
- `int sprintf(char *, char *, ...)`
- `int snprintf(char *, int, char *, ...)`
- `int scanf(char *, ...)`
- `int fscanf(FILE *, char *, ...)`
- `int sscanf(char *, char *, ...)`
- `int vprintf(char *, va_list)`
- `int vfprintf(FILE *, char *, va_list)`

- int vsprintf(char *, char *, va_list)
- int vsnprintf(char *, int, char *, va_list)
- int vscanf(char *, va_list)
- int vfscanf(FILE *, char *, va_list)
- int vsscanf(char *, char *, va_list)

7.4.2 Standard Library Functions

ANSI-C의 표준 라이브러리 함수와 동일하므로 자세한 설명은 생략한다.

- float atof(char *)
- float strtod(char *,char **)
- int atoi(char *)
- int atol(char *)
- int strtol(char *,char **,int)
- int strtoul(char *,char **,int)
- void *malloc(int)
- void *calloc(int,int)
- void *realloc(void *,int)
- void free(void *)
- int rand()
- void srand(int)
- void abort()
- void exit(int)
- char *getenv(char *)
- int system(char *)
- int abs(int)

- int labs(int)

7.4.3 Standard String Functions

ANSI-C의 표준문자열 함수와 동일하므로 설명은 생략한다.

- void *memcpy(void *,void *,int)
- void *memmove(void *,void *,int)
- void *memchr(char *,int,int)
- int memcmp(void *,void *,int)
- void *memset(void *,int,int)
- char *strcat(char *,char *)
- char *strncat(char *,char *,int)
- char *strchr(char *,int)
- char *strrchr(char *,int)
- int strcmp(char *,char *)
- int strncmp(char *,char *,int)
- int strcoll(char *,char *)
- char *strcpy(char *,char *)
- char *strncpy(char *,char *,int)
- char *strerror(int)
- int strlen(char *)
- int strspn(char *,char *)
- int strcspn(char *,char *)
- char *strpbrk(char *,char *)
- char *strstr(char *,char *)
- char *strtok(char *,char *)

- int strxfrm(char *,char *,int)
- char *strdup(char *)

7.4.4 Standard Time Functions

ANSI-C의 표준 시간함수와 동일한 부분은 설명이 생략되고, 그외 sleep, timeus, timesec, timemin, timehour, timeday, timemon, timeyear 함수가 추가되었다.

- char *asctime(struct tm *)
- time_t clock()
- char *ctime(int *)
- double difftime(int, int)
- struct tm *gmtime(int *)
- struct tm *localtime(int *)
- int mktime(struct tm *ptm)
- int time(int *)
- int strftime(char *, int, char *, struct tm *)
- void sleep(int ms) : 지정된 ms단위 시간만큼 대기
- int timeus(void) : us 단위 시간을 얻음
- int timems(void) : ms 단위 시간을 얻음
- int timesec(void) : 현재 시간의 초를 얻음 (0~59)
- int timemin(void) : 현재 시간의 분을 얻음 (0~59)
- int timehour(void) : 현재 시간의 시를 얻음 (0~23)
- int timeday(void) : 달력의 일을 얻음
- int timemon(void) : 달력의 월을 얻음
- int timeyear(void) : 달력의 년을 얻음

7.4.5 Standard Mathematics Functions

ANSI-C의 표준 수학적함수와 동일하므로 설명은 생략한다.

- float acos(float)
- float asin(float)
- float atan(float)
- float atan2(float, float)
- float ceil(float)
- float cos(float)
- float cosh(float)
- float exp(float)
- float fabs(float)
- float floor(float)
- float fmod(float, float)
- float frexp(float, int *)
- float ldexp(float, int)
- float log(float)
- float log10(float)
- float modf(float, float *)
- float pow(float, float)
- float round(float)
- float sin(float)
- float sinh(float)
- float sqrt(float)
- float tan(float)
- float tanh(float)

7.4.6 CANTalker High Level Functions

자세한 함수 설명과 데이터 형은 3.7 High-Level USB & RS232 Interface API 을 참고한다. 접속과 DLL 초기화 등의 함수를 제외하도 대부분의 함수가 사용가능하다.

- int DNK_SetCanInit(void)
- int DNK_SendCanData (CAN_FRAME *canFrame)
- int DNK_GetTotalRcvCanDataCount (void)
- int DNK_ClearRcvCanData (void)
- int DNK_GetRcvCanData (CAN_FRAME *can)
- int DNK_SetClearCanError (void)
- int DNK_ReqCanStatus (CAN_STATUS *status)
- int DNK_ReqErrorCount (UINT8 *txErrCnt, UINT8 *rxErrCnt)
- int DNK_SetCanID (BYTE extMode, UINT32 id)
- int DNK_SetCanBaudRate (UINT32 brate)
- int DNK_ReqDeviceInfo (DEVICE_INFO *devInfo)
- int DNK_SetRcvMode (UINT32 mode)
- int DNK_ReqRcvMode (UINT32 *rcvMode)
- int DNK_ReqCanFilters (FILTER_INFO *filterInfo)
- int DNK_ReqVersion (UINT8 *major, UINT8 *minor)
- int DNK_ReqSpeedInfo (UINT32 *canSpeed, UINT8 *rs232SpeedIdx)
- int DNK_ReqTcpIpInfo (TCPIP_INFO *pInfo)
- int DNK_ReqCurTimeStamp (UINT32 *timeStamp)
- int DNK_SetCanBitTiming (BRGCON *brgcon)
- int DNK_AddLog (BYTE bRx, CAN_FRAME *canFrame) : CANTalker의 로그창에 CAN메시지를 추가함
- 프로그램 예제

```

/* OBD PID Scan Test */
#define OBD_REQ_CAN_ID    0x7DF
#define OBD_RSP_CAN_ID    0x7E8
void main (void)
{
    CAN_FRAME can, rcan;
    UINT8      *cmd = can.data;
    UINT8      pid;
    printf ("Set Rcv CAN ID %04X\n", OBD_RSP_CAN_ID);
    DNK_SetCanID (0, OBD_RSP_CAN_ID);
    DNK_ClearRcvCanData ();
    cmd[0] = 0x02;
    cmd[1] = 0x01;
    cmd[3] = 0x55;
    cmd[4] = 0x55;
    cmd[5] = 0x55;
    cmd[6] = 0x55;
    cmd[7] = 0x55;
    can.canId = OBD_REQ_CAN_ID;
    can.dlc = 8;
    for (pid = 0; pid <= 64; pid++)
    {
        printf ("Query PID %d --> ", pid);
        can.data[2] = pid;
        DNK_SendCanData (&can);
        sleep (50);
        if (DNK_GetTotalRcvCanDataCount() > 0)
        {
            DNK_GetRcvCanData (&rcan);
            if (rcan.canId == OBD_RSP_CAN_ID)
            {
                cmd = rcan.data;
                printf ("Rsp : %02X %02X %02X %02X %02X %02X %02X %02X\n",
                    cmd[0], cmd[1], cmd[2], cmd[3], cmd[4], cmd[5], cmd[6], cmd[7]);
            }
        }
        else
        {
            putchar ('\n');
        }
    }
}

```

7.4.7 TCP/UDP Network Functions

TCP / UDP Socket 통신을 간단한 함수로 구현할 수 있다. 데이터를 수신하기 위해서는 GetRcvDataSize로 수신데이터가 있는지 확인 후 GetData함수로 수신 데이터를 가져 올 수 있다. GetData는 Non-Blocking함수로 데이터가 없다면 즉시 리턴한다. 접속이 끊기거나 오류가 발생하는 경우는 (-1) 리턴값을 갖는다.

- int TCP_Open (char *ipAddr, int port)
- void TCP_Close (void)
- int TCP_SendData (char *buff, int size)
- int TCP_GetRcvDataSize (void)

- int TCP_GetData (char *buff, int size)
- void UDP_Open (char *ipAddr, int rxPort, int txPort) : txPort는 Remote Port를 의미함.
- void UDP_Close (void)
- int UDP_SendData (char *buff, int size)
- int UDP_GetRcvDataSize (void)
- int UDP_GetData (char *buff, int size)
- 예제 프로그램

```

void main (void)
{
    int bCon = TCP_Open ("127.0.0.1", 21);
    char buff[128];
    int size, count = 0;
    int pt, ct;

    if (!bCon)
    {
        printf ("TCP connection failed\n");
        return;
    }
    pt = timems ();
    while (1)
    {
        size = sizeof (buff) - 1;
        if (TCP_GetRcvDataSize ())
        {
            int i;
            size = TCP_GetData (buff, size);
            buff[size] = 0;
            printf ("s=%d, %s\n", size, buff);
        }
        ct = timems ();
        if (ct - pt >= 1000)
        {
            pt = ct;
            size = sprintf (buff, "count=%d\n", count++);
            TCP_SendData (buff, size);
        }
    }
    TCP_Close ();
}

```

7.4.8 Serial Communication Functions

Serial 통신을 간단한 함수로 구현할 수 있다. 데이터를 수신하기 위해서는 GetRcvDataSize로 수신데이터가 있는지 확인 후 GetData함수로 수신 데이터를 가져 올 수 있다. GetData는 Non-Blocking함수로 데이터가 없다

면 즉시 리턴한다.

- int SER_Open (int port, int baudRate) : 성공인 경우는 리턴값이 1, 실패인 경우는 0
- void SER_Close (void)
- int SER_SendData (char *buff, int size)
- int SER_GetRcvDataSize (void)
- int SER_GetData (char *buff, int size)
- 예제프로그램

```
void main (void)
{
    int size;
    char buff[128];
    int count = 0;
    int pt, ct;
    if (!SER_Open (2, 115200))
    {
        printf ("serial port open error\n");
        return;
    }
    pt = timems ();
    while (count < 100)
    {
        if (SER_GetRcvDataSize ())
        {
            size = sizeof (buff);
            size = SER_GetData (buff, size);
            buff[size] = 0;
            printf (buff);
        }
        ct = timems ();
        if (ct - pt >= 1000)
        {
            pt = ct;
            size = sprintf (buff, "count = %d\n", count++);
            SER_SendData (buff, size);
        }
    }
    SER_Close ();
}
```

8 Firmware Upgrade

8.1 Overview

본 Analyzer 및 CANTalker Windows응용 프로그램은 기능 향상을 위해 지속적으로 업그레이드될 예정이다. 이를 위해 사용자가 직접 업그레이드 할 수 있는 방법 및 기능이 지원된다. 최신 매뉴얼 및 CANTalker 는 다음 링크에서 최신버전을 받을 수 있다.

1. 최신매뉴얼 링크 : http://dnkict.dothome.co.kr/can/MCA_100_Manual.pdf
2. 최신 CANTalker링크 : <http://dnkict.dothome.co.kr/can/cantalker.zip>

펌웨어는 CANTalker내에 내장되어 있으므로 CANTalker만 받으면 최신 펌웨어로 업그레이드 할 수 있다.

8.2 Preparations

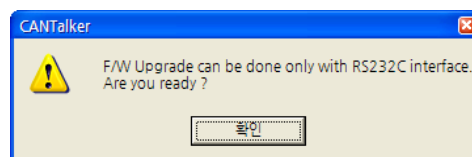
펌웨어업그레이드는 RS232C 인터페이스만 지원하기 때문에 반드시 사전에 RS232C 케이블 연결은 되어 있어야 한다. 통신속도는 115,200 bps이기 때문에 사용자는 CANTalker의 [Interface] → [Configurations] 메뉴를 선택하여 [RS232C]에서 통신 포트와 통신속도를 정확하게 설정한다. 이때 CANTalker뿐만 아니라 Analyzer의 RS232C 통신속도도 115,200 bps으로 설정되어 있어야 하므로 USB나 Ethernet인터페이스로 연결하여 RS232C통신 속도를 변경해 놓는다.

Analyzer의 Error LED가 전원 인가시에 점멸하는 경우는 펌웨어에 문제가 있거나 없는 경우이기 때문에 사용자는 반드시 펌웨어업그레이드를 수행해야 한다.

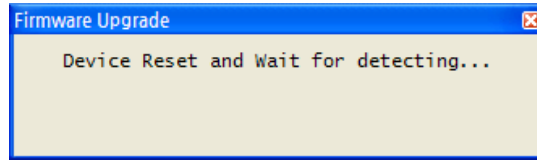
모든 준비가 끝났으면 다음절의 절차에 따라 펌웨어 업그레이드를 수행한다.

8.3 Procedure

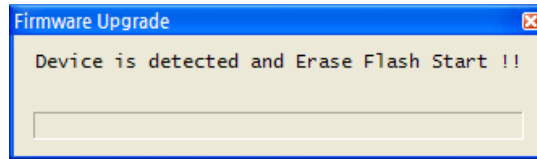
1. CANTalker를 실행하고 준비가 된 상태에서 [Help] 메뉴의 하위 메뉴인 [Firmware Upgrade]를 선택한다. 이때 가급적 Analyzer와는 [Connect] 되어 있지 않는것이 바람직 하다. 메뉴를 선택하면 무조건 업그레이드가 진행되므로 주의 깊게 선택해야 한다. 선택되면 다음과 같은 창이 표시되며 준비가 완료되었으면 [확인] 버튼을 클릭한다.



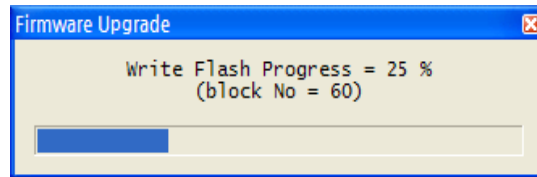
2. 펌웨어 업그레이드가 시작되면 Device와 동기를 맞추기 위한 작업이 수행되며 아래와 같은 창이 표시된다.



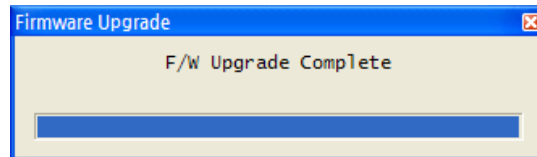
3. 동기가 맞으면 Flash를 지우는 작업이 수행되며 다음과 같은 창이 표시된다.



4. Flash 지움에 완료되면 Flash에 쓰기를 시작하며 progress바가 0부터 100까지 변경된다



5. 정상적으로 Write가 완료되면 Verify를 수행하고 문제가 없으면 아래와 같은 창이 표시된다. 만약 오류가 있으면 "Verify Failed" 가 표시 된다.



6. 모든 작업이 완료되면 Analyzer는 리셋되어 다운받은 펌웨어로 부팅한다.
7. 다운로드 중 전원이 없어지거나 비정상적으로 종료되는 경우 펌웨어가 비정상이다. 이 경우는 Analyzer의 전원을 껐다 키고 Error LED가 점멸할 때 다시 처음 과정을 반복한다.

9 Appendix1. Multi-CAN Analyzer를 이용한 차량 ECU 모니터링

9.1 들어가며

CAN (Controller Area Network)은 멀티마스터(Multi-Master) 브로드캐스트 시리얼(Broadcast Serial) 버스 표준으로 실시간 제어 응용시스템 내에 있는 센서나 기동장치 등과 같은 주변장치들을 서로 연결해 주는 메시지 통신방식으로 마이크로 제어기용 네트워크를 가리킨다. CAN에서는 이더넷 등에서 사용되는 것과 같은 주소 지정 개념은 사용되지 않으며, 메시지는 해당 네트워크에서의 고유한 식별자를 사용하여 네트워크 내에 있는 모든 노드들에게 동시에 뿌려진다. 개개의 노드들은 이 식별자에 기반하여 해당 메시지를 처리할 것인지의 여부를 결정하며, 버스 접근 순서 역시 경쟁원리에 따라 메시지 우선순위가 배정되어 충돌이 없어 중단 없는 데이터 전송이 가능하다. 또한 differential twisted pair방식을 사용하여 노이즈에 매우 강한 특성이 있다. 보다 자세한 CAN에 대한 역사나 기술적인 내용은 본고에서는 생략하기로 하고 참고자료를 참고한다.

위와 같은 CAN 특성으로 최근 매우 안정성을 요구 하는 차량의 내부 네트워크로 CAN 통신을 사용하고 있는 추세다. 즉 주제어기인 ECU (Electronic Control Units) 끼리의 통신이나 각종 센서 및 작동기 등과 안정적인 통신을 위해 CAN이 사용되고 있는 것 이다. 따라서 CAN 통신을 이용하여 ECU가 이해할 수 있는 프로토콜로 통신을 하게 되면 차량 네트워크로부터 차량의 다양한 정보를 얻어 낼 수 있게 된다.

본고에서는 CAN 메시지를 송수신하고 메시지에서부터 프로토콜을 해석할 수 있는 Analyzer를 소개하고, 그것을 통해 실제 차량 ECU로부터 다양한 정보를 모니터링 하는 방법에 대해 소개한다.

9.2 Multi-CAN Analyzer 소개



D&K ICT는 실시간 시뮬레이션 및 네트워크 통신장치 제조 전문업체로 그간 산업용 제어 분야의 경험을 기반으로 다양한 CAN 제품을 출시하였다. CAN 경험이 없는 일반 사용자들을 위해 보다 쉽게 접근할 수 있는 CAN2RS232 컨버터와, 인터넷을 통한 원격 제어를 위한 CAN2Ethernet 게이트웨이 등이 그것이다. 이 제품들은 주로 제어기 용도로 개발되었기 때문에 필터 설정을 통해 필요로 하는 메시지들만 CAN 버스로부터 수신하고 전송하는 제품으로 컨버터나 게이트웨이가 처리할 수 있는 용량 이상으로 CAN버스 메시지를 모두 처리 할 수는 없었다. 또한 절연되지 않아 현장 상황에 따라 컨버터가 파손되는 경우도 있었다. 이런 이유로 Analyzer 용도로는 적합하지 못해 High-End급 고성능 절연 CAN-USB Analyzer / CAN2Ethernet Analyzer 와 Multi-CAN Analyzer를 개발 출시 하게 되었다.

본고에서 소개될 Multi-CAN Analyzer는 이름에서 유추 할 수 있듯이 동시에 여러 인터페이스 방식을 사용할 수 있는데, 그것은 RS232C, USB 그리고 Ethernet 이다. 따라서 어떠한 환경이나 조건에도 쉽게 Analyzer를 쓸 수 있는 장점이 있다. RS232C는 최대 460800bps의 속도 까지 사용할 수 있으며 인터페이스가 UART밖에 없는 소형 제어기와도 쉽게 연결된다. USB는 PC와 외부 전원 없이 연결할 수 있기 때문에 노트북으로 현장에서 디버깅하는데 매우 유용하다. Ethernet은 현장에 설치된 CAN버스의 문제점을 원격지 사무실에서 모니터링 하는데 매우 적합하다.

본 Analyzer는 제공되는 전용 프로토콜을 통해 직접 Analyzer를 제어기 등과 연결하여 고성능 컨버터 처럼 사용할 수도 있지만, Windows 응용프로그램으로 제공되는 전용 분석기인 CANTALKER를 통해 추가적인 소프트웨어 개발 없이 고수준의 CAN 프로토콜을 분석할 수 있다. 기본적으로 제공되는 기능으로 OBD-II, CANopen, DeviceNet, 그리고 J1939 표준 프로토콜 해석과 ECU Simulator 기능이 있다.

No	Time Stamp	CAN ID	2.OB	RTR	DLC	DATA	CANopen
0	2502.33	77F			1	00	[ERROR CONTROL] (Tog=0) Initializing/Boot-up
1	2502.72	67F			8	23 40 60 00 00 00 00 00	[RSDO(6040 0)] Download Req(expedited)/size=0/data=0
2	2502.72	5FF			8	60 40 60 00 00 00 00 00	[TSDO(6040 0)] Download Rsp
3	2502.72	67F			8	23 00 39 00 01 80 00 00	[RSDO(3900 0)] Download Req(expedited)/size=0/data=...
4	2502.72	5FF			8	60 00 39 00 00 00 00 00	[TSDO(3900 0)] Download Rsp
5	2502.75	67F			8	23 14 33 00 E8 03 00 00	[RSDO(3314 0)] Download Req(expedited)/size=0/data=3E8
6	2502.75	5FF			8	60 14 33 00 00 00 00 00	[TSDO(3314 0)] Download Rsp
7	2502.88	67F			8	40 41 60 00 00 00 00 00	[RSDO(6041 0)] Upload Req
8	2502.88	67F			8	40 64 60 00 00 00 00 00	[RSDO(6064 0)] Upload Req
9	2502.88	5FF			8	48 41 60 00 37 12 00 00	[TSDO(6041 0)] Upload Rsp(expedited)/size=2/data=1237
10	2502.88	5FF			8	43 64 60 00 00 00 00 00	[TSDO(6064 0)] Upload Rsp(expedited)/size=0/data=0
11	2562.28	FF			8	11 86 01 00 60 F0 00 00	[EMERGENCY] Error Code=8611(Error Reset or No Error...

CAN Status=00, Rx=0, Tx=0

그림 1 CANTALKER Windows Application

그 밖에 CANTALKER가 지원하는 주요 기능은 다음과 같다.

- 10us 단위 Timestamp
- CAN 송신 메시지 프로그래밍
- CAN 송수신 메시지 읽기 및 저장 (텍스트 파일로 export지원)
- 특정 CAN 메시지 모니터링 및 산술 계산 / 그래픽 표시 / 로깅(실시간/비실시간 지원)
- CAN 수신 메시지 소프트웨어 필터링 / 트리거
- CAN 송수신 메시지에서 특정 메시지 찾기 / 복사
- CAN 버스 통계자료 제공

- CAN 컨트롤러 상태 확인 및 제어
- 터미널
- 펌웨어 업그레이드

본고에서 ECU모니터링에 사용될 장비와 프로그램은 Multi-CAN Analyzer와 CANTALKER 이다.

9.3 ECU 모니터링

CAN통신을 지원하는 차량의 대부분은 OBD-II PID 프로토콜을 지원한다. ECU와 통신하기 위해서는이 프로토콜을 반드시 숙지 해야 하는데 그다지 복잡하기 않기 때문에 쉽게 구현될 수 있다. 그럼 OBD-II PID가 무엇인지 다음절에서 알아 본다.

9.3.1 OBD-II PID란 ?

OBD-II PID는 On Board Diagnostics Parameter ID의 약자로 정확하게 말하면 프로토콜이 아니라 코드를 의미 하며 SAE J1979 표준의 한 부분으로 1996년부터 북미의 모든 차량에 구현되었다. 이 코드는 고장진단장치가 차량으로부터 데이터를 얻는데 사용되는데 데이터를 요청할 때 OBD-II PID 코드를 정해진 프로토콜을 이용하여 차량으로 요청하여 응답을 얻는다. 중요 PID종류와 관련 응답 데이터 형식은 표 6과 같다. 모든 PID는 모든 차량에서 지원하는 것은 아닌데, 지원여부는 PID 00의 응답 값으로부터 알 수 있다.

표 6 의 공식에서 사용되고 있는 알파벳 A,B,C,D 등은 수신된 데이터의 바이트 순서와 일치한다. 즉 첫번째 바이트는 A, 두번째는 B, 세번째는 C, 그리고 네번째는 D이다.

표 6. PID 목록 및 설명

Mode	PID	수신 크기	설명	Min	Max	단위	공식
00	00	4	PID Supported [01-20]				Bit encoded [A7..D0]==[PID 0x01..PID 0x20]
01	04	1	Calculated engine load value	0	100	%	$A*100/255$
01	05	1	Engine Coolant Temperature	-40	215	°C	$A-40$
01	0C	2	Engine RPM	0	16383.75	rpm	$((A*256)+B)/4$
01	0D	1	Vehicle speed	0	255	km/h	A
01	0E	1	Timing Advance	-64	63.5	relative	$A/2-64$

						to #1 cylinder	
01	0F	1	Intake air Temperature	-40	215	°C	A-40
01	10	2	MAF air flow rate	0	655.35	g/s	((A*256)+B)/100
01	11	1	Throttle Position	0	100	%	A*100/255

표 6의 Mode는 각 숫자별로 의미하는 내용은 표 7에서 알 수 있다. ECU의 각종 정보 데이터를 얻는 경우는 대부분 현재 데이터 표시용 Mode 0x01만 이용하면 되며, 차량의 고장진단 코드를 얻고자 하는 경우는 Mode 0x03를 사용한다.

표 7. Mode 별 설명

Mode	설명
0x01	Show current data
0x02	Show freeze frame data
0x03	Show stored Diagnostic Trouble Codes
0x04	Clear Diagnostic Trouble Codes and stored values
0x05	Test results, oxygen sensor monitoring (non CAN only)
0x06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
0x07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
0x08	Control operation of on-board component/system
0x09	Request vehicle information
0x0A	Permanent DTC's (Cleared DTC's)

차량으로부터 데이터를 얻기 위해 ECU로 PID를 전달하기 위한 질의 CAN 메시지 형태는 각 메시지 데이터 바이트 별로 다음과 같은 형식을 갖는다. CAN ID는 11비트를 사용하는 경우 반드시 0x7DF로 해야 한다.

Byte 0	Byte1	Byte 2	Byte3	Byte4	Byte5	Byte6	Byte7
추가데이터 바이트 수	Mode	PID Code	0x55	0x55	0x55	0x55	0x55
0x02	0x01=show current data						

CAN메시지 전체 8바이트 중 첫번째 바이트(Byte0)는 몇 바이트의 정보가 오는지 의미 하는 것으로 그 뒤로 Mode와 PID Code 가 각각 1바이트씩 2바이트가 오기 때문에 2가 되며, 세번째 바이트(Byte2)는 PID코드가 위치하고 그 이상은 사용되지 않으므로 0x55로 채워진다.

이상과 같이 생성된 CAN 메시지를 ECU로 전송하면 ECU는 다음과 같은 형태의 CAN메시지로 응답한다. 이때 응답으로 오는 CAN ID는 주로 0x7E8이며, 0x7E9나 0x7EA가 될 수 도 있다.

Byte 0	Byte1	Byte 2	Byte3	Byte4	Byte5	Byte6	Byte7
추가데이터	Custom Mode	PID	Value0	Value1	Value2	Value3	0x55
바이트 수	0x40+Mode	Code	(A)	(B)	(C)	(D)	
3~6	0x41=show current data			(Option)	(Option)	(Option)	

여기서 첫번째 바이트는 질의 메시지와 동일하게 이후에 올 추가 데이터의 크기이다. 두번째는 질의 메시지에서 사용된 Mode값에 0x40을 더한값과 같다. 내번째 부터는 질의한 PID에 대한 응답데이터가 온다. 데이터 크기는 1바이트부터 4바이트까지만 갖을 수 있어 마지막 Byte7은 사용되지 않아 0x55로 채워진다. 응답데이터는 첫번째 바이트부터 A,B,C,D로 명명되고 PID 표의 공식 부분에서 참조된다.

9.3.2 ECU 모니터링을 위한 전송 메시지 설정

이제 ECU로 전달해야 할 CAN메시지 형식을 알게 되었기 때문에 Analyzer에서 질의 메시지를 작성하여 보내 보고자 한다. 보낼 PID는 표 6 에 나와 있는 것으로 하고, 각 PID별로 보내는 시간 간격은 10ms로 한다. 모든 PID 를 모두 보내면 10ms 만큼 대기하고 다시 처음부터 PID전송을 반복 한다. 이것을 Analyzer의 “CAN Send Message Tool” 에서 설정하는 방법은 다음과 같다.

1. Engine RPM 질의를 위한 CAN메시지 예는 다음과 같다.

Task Type	CAN ID(HEX)	2.0B	RTR	DATA (HEX)	Repeat	Interval(ms)
TX	000007DF	<input type="checkbox"/>	<input type="checkbox"/>	02 01 04 55 55 55 55 55	0	10

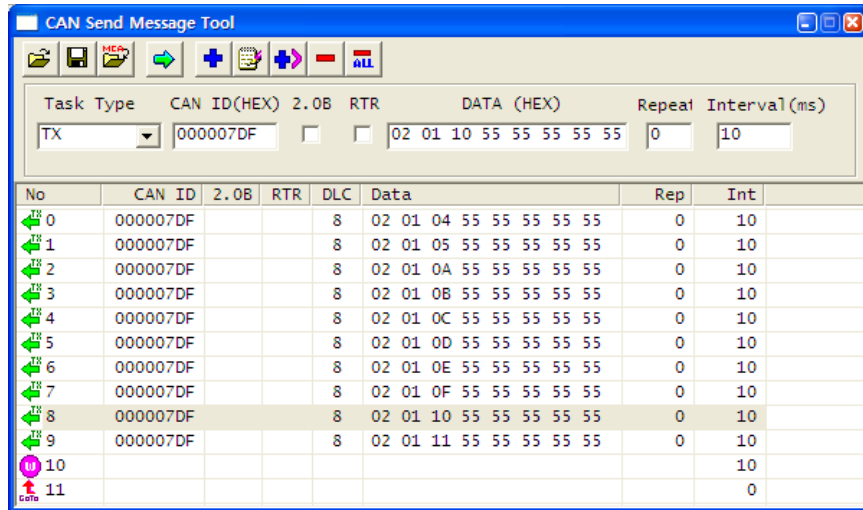
1. 전송하므로 Task Type은 “TX”
2. 11비트 질의이므로 CAN ID는 0x7DF, 2.0B와 RTR은 설정하지 않음
3. DATA는 “02 01 0C 55 55 55 55 55”로 함, 여기서 0x0C Engine RPM 질의 PID임.
4. 추가 반복하지 않으므로 Repeat는 “0”
5. 10ms이후에 다음 메시지 전송하므로 Interval은 “10”
2. 위와 같은 형식으로 나머지 PID에 대해서 모두 추가 한다.
3. 모든 PID가 전송된 후 10ms대기를 위해 Task Type이 “WAIT” , Wait Time을 10으로 설정한다.

Task Type	CAN ID(HEX)	2.0B	RTR	DATA (HEX)	Repeat	Wait Time(ms)
WAIT	000007DF	<input type="checkbox"/>	<input type="checkbox"/>	02 01 04 55 55 55 55 55	0	10

4. 다시 처음부터 반복 전송을 위해 Task Type을 “GOTO” , Goto Index는 “0” 으로 설정한다.

Task Type	CAN ID(HEX)	2.OB	RTR	DATA (HEX)	Repeat	Goto Idx
GOTO	00000000	<input type="checkbox"/>	<input type="checkbox"/>		0	0

이렇게 모두 입력되면 “CAN Send Message Tool” 창은 다음과 같이 보여지며 전송준비는 완료되며 (GO) 버튼을 클릭하면 CAN메시지가 전송되기 시작한다. 이때 대부분 차량내에 사용되는 CAN통신 속도는 500kbps 이므로 사전에 CAN통신 속도는 맞춰져 있어야 한다. 그렇지 않은 경우는 Analyzer의 Error LED가 켜질 것이다.



9.3.3 ECU 모니터링을 위한 수신 메시지 설정

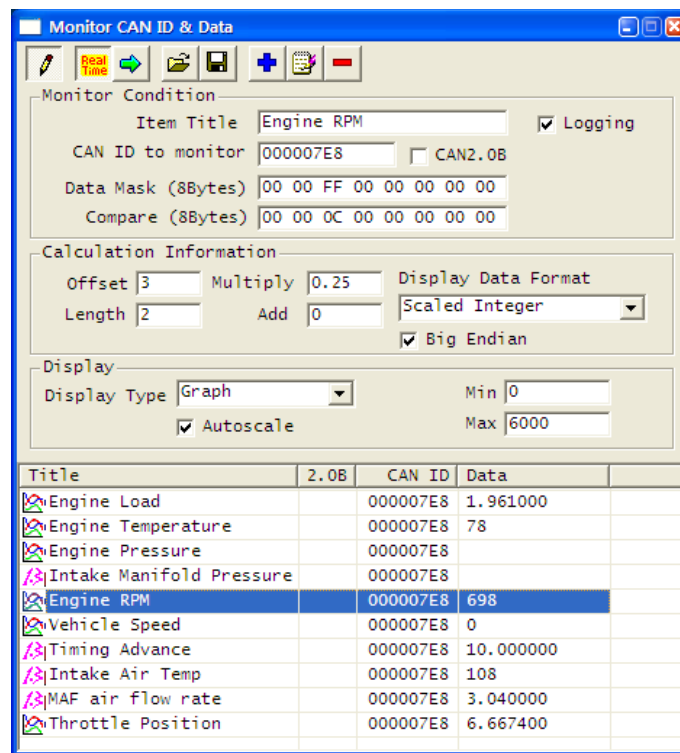
9.3.1절에서 OBD-II PID 가 포함된 질의 메시지에 대한 ECU 응답 메시지에 대한 형식을 알아 봤다. 이 정보를 토대로 CANTALKER에서 지원하는 “Monitor CAN ID & Data” 기능을 이용하여 응답 메시지로부터 실제 물리적인 값들을 추출하여 그래프로 표시하여 보자.

응답 메시지 형식과 표 6 의 공식 부분을 참고 하여 “Monitor Condition” 을 작성하여 보자. “Engine RPM” 작성예를 보면 다음 과 같다.

- Item Title은 제목인 “Engine RPM”을 기록한다.
- “CAN ID To Monitor”는 모니터링 하고자 하는 CAN ID를 의미 하므로 0x7E8이 된다.
- “Engine RPM”의 경우 PID가 0x0C인데, CAN메시지 8바이트 중 세번째 바이트가 PID이므로 이 값이 0x0C인 메시지를 찾기 위해 Mask와 Compare는 다음과 같이 설정한다.
- Mask는 세번째 바이트만 관심 대상이므로 0xFF로 설정하고, Compare값의 세번째 바이트를 0x0C로 설정한다. 이유는 “MSG & Mask == Compare” 공식에 의해 각 바이트에 대해 모두 참이 되면 수신 메시지를 사용하게 된다.

- 이렇게 수신 조건에 의해 걸러진 “Engine PRM”메시지에서 Engine PRM에 관련된 데이터를 추출하기 위해 다음과 같이 설정한다.
- Engine RPM값은 CAN 메시지 8바이트 중 4번째 바이트부터 2바이트 크기를 갖기 때문에 “Offset”은 3, “Length”는 2가 된다. 추출된 2바이트에서 첫번째가 상위 두번째가 하위 바이트가 되는 Big Endian형식이므로 “Big Endian”에 체크한다. 또한 공식에 의해 2바이트 값을 4로 나누거나 또는 0.25로 곱해야 하므로 “Multiply”에 0.25를 입력한다. 최종 생성되는 값은 산술 계산된 정수형이므로 “Display Data Format”을 “Scaled Integer”로 선택한다.
- 결과적으로 생성된 물리값은 그래프로 표시하고 싶으므로 “Display Type”을 “Graph”로 선택하고 최대 최소값을 각각 0과 6000으로 한다.

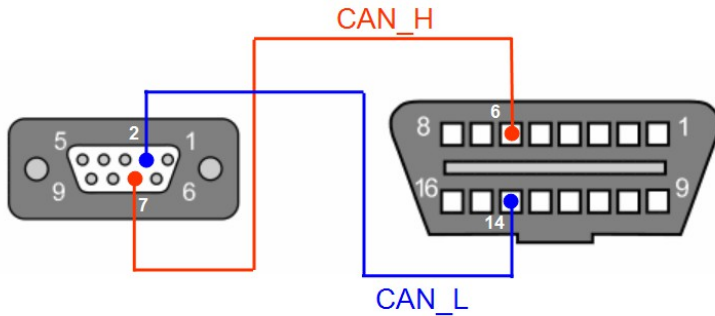
그 밖의 다른 Item들도 위와 같은 형식에 의해 공식을 참고하여 비슷한 형식으로 아래 그림과 같이 모두 입력한다.



9.3.4 실차량에서 모니터링 수행


9.3.2절에서 전송할 메시지들을 생성하고 9.3.3절에서 추출할 수신메시지 형식과 표시형식을 모두 설정하였으므로 실제차량에서 ECU에서 정보가 실제로 오는지 확인해 보자. 먼저 CAN 방식의 OBD-II가 지원 되는 차량을 준비해야 한다. 대부분 최근에 생산된 차량은 거의 CAN 버스와 OBD-II를 지원하고 있다. 간혹 CAN은 지원하는데 OBD-II PID메시지에 대해 무응답하는 차량도 있으니 미리 확인해야 한다. Analyzer를

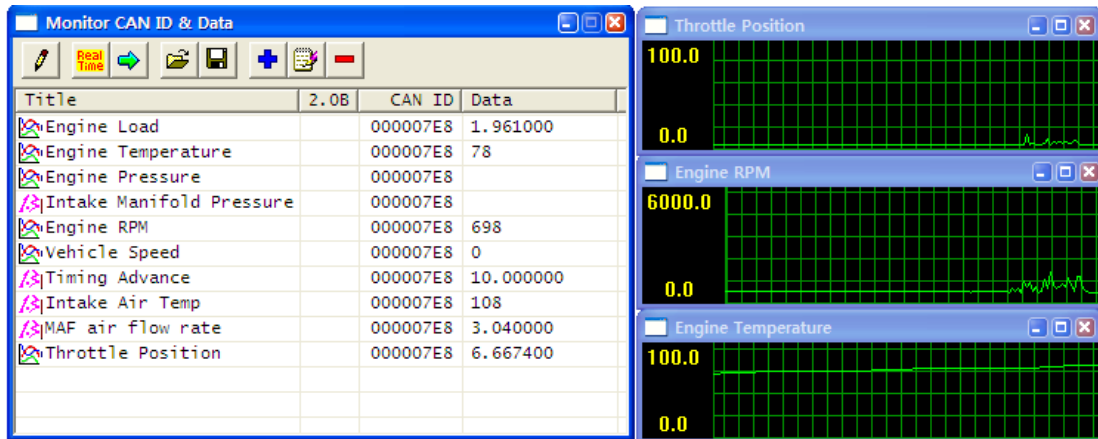
차량의 OBD-II 단자에 연결하기 위해서는 OBD-II 단자를 핀배열을 확인한 후 CAN 컨넥터와 맞게 케이블을 다음과 같이 제작해야 한다. OBD-II 컨넥터는 디바이스마트에서 쉽게 구매할 수 있다.



위 그림에서 OBD-II Male Connector의 PIN6은 CAN_H, PIN2는 CAN_L 이므로 Analyzer DB9 Connector의 PIN7번과 PIN14번에 각각 연결한다. CAN_H와 CAN_L은 잘못 연결되어서는 절대 안되므로 주의 깊게 작업해야 한다.

컨넥터 작업이 완료되면, Analyzer를 차량의 OBD-II 단자에 연결한다. 차량에 전원이 들어가거나 시동이 걸려있어야 CAN 통신이 되므로 사전에 전원을 넣거나 시동을 걸어 놓는다. 만약 컨넥터를 연결하자마자 Analyzer의 Error LED가 점등 된다면, (1) 500kbps 속도 설정이 제대로 안되었거나, (2) CAN_H, CAN_L 라인 결선에 문제가 있거나 (3) 종단저항 연결이 ON되어 있는 경우일 수 있으므로 다시 한번 확인해 본다.

모든 준비가 완료되면, “CAN Send Message Tool” 에서  (Go) 버튼을 클릭하여 ECU에 질의 메시지를 전송시킨다. 그러면 아래와 같이 질의에 해당하는 응답이 수신되면서 그 값과 그래프가 표시된다.



9.3.5 맺음말

이상과 같이 차량의 ECU 정보값을 모니터링 하기 위해 OBD-II PID와 관련한 질의와 응답 프로토콜에 대해 확인하였고, 그 내용을 Analyzer에 적용하여 각각의 데이터를 수신하고 그래프까지 출력해 보았다.

필자도 그러했지만 많은 사람들은 정보가 없었을 때 차량의 ECU와 통신을 해서 각종 정보를 얻어내는 것은 자동차 회사에서 프로토콜을 공개하지 않는 이상 불가능 한 것으로 여겨 왔었다. 그러나 정말 단순한 OBD-II PID 프로토콜만 인터넷 검색으로 찾아본다면 차량의 각종 상태나 고장진단 등을 Analyzer로 매우 쉽게 해낼 수 있다는 것을 알게 되었다.

그 밖에 Analyzer에서 제공하는 CANopen프로토콜로 CANopen프로토콜을 지원하는 다양한 장비들의 상태 체크나 고장진단 등에 활용할 수 있으며, 표준 프로토콜을 사용하지 않는 각종 장비나 차량 부품도 CAN버스 통계 정보로 사용되는 모든 CAN ID를 찾아 낼 수 있고, 찾아낸 CAN ID에 대해 각각 실시간 모니터링 함으로써 그 메시지의 특성이나 프로토콜을 쉽게 알아 낼 수 있다.

9.3.6 참고자료

1. <http://terms.co.kr/CAN.htm>
2. http://en.wikipedia.org/wiki/Controller_area_network
3. <http://en.wikipedia.org/wiki/OBD-II#OBD-II>
4. http://en.wikipedia.org/wiki/OBD-II_PIDs
5. Multi-CAN Analyzer & Converter User's Manual
(http://dnkict.dothome.co.kr/can/MCA_100_Manual.pdf)