강좌<10>: CAN 통신 사용

목표: 이번 강좌에서는 STM32F 내부의 CAN 통신에 대해서 다루어봅니다

RealSYS

작성일자: 2010.6.2

LCD & 버튼 보드 사용 시험

BT1 : 표준 형태, “0011..” 전송

BT2 : 확장 형태, “0011..” 전송

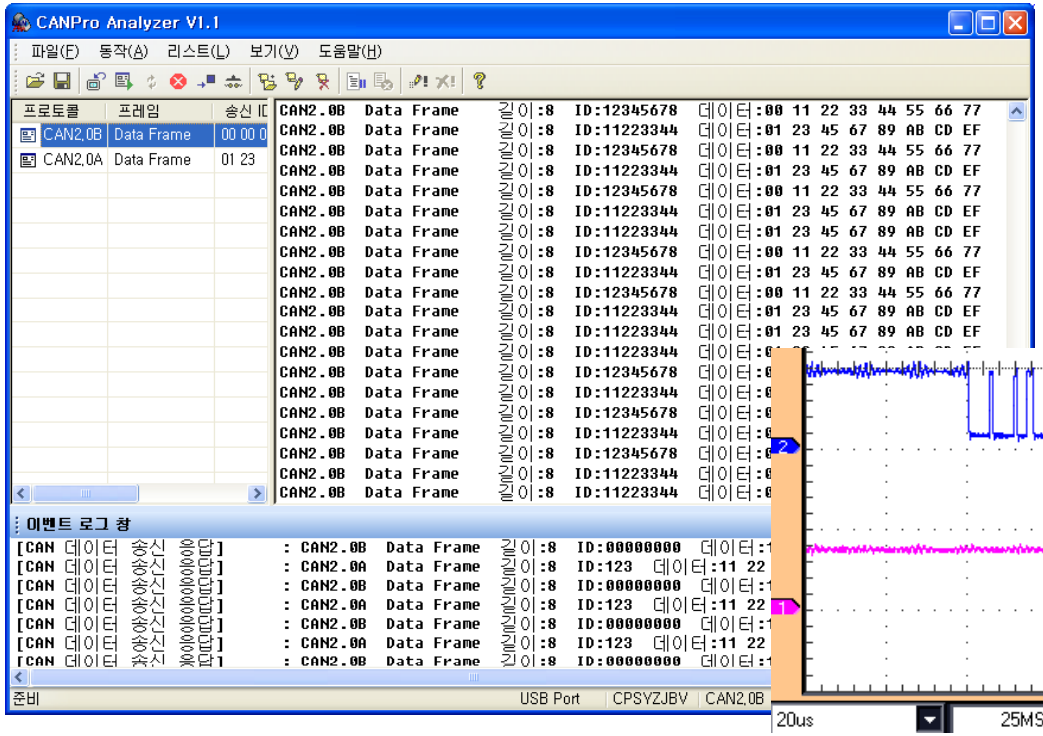
BT3 : 확장 형태, “0123..” 전송

BT4 : 표준 형태, “0123..” 전송

BT5 : 메시지 지움

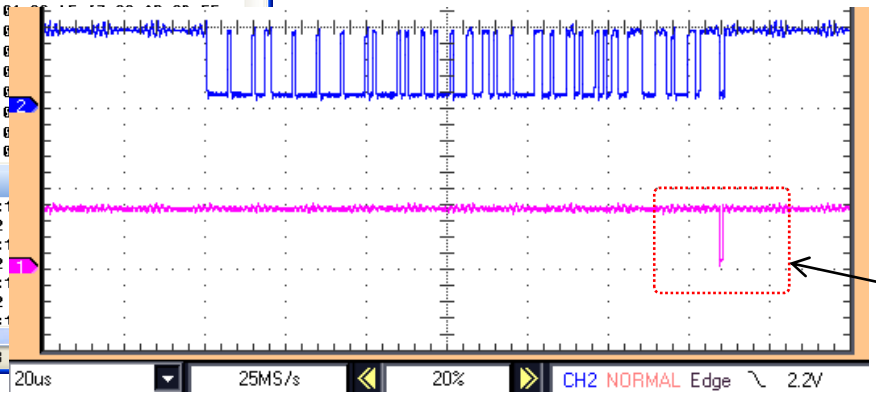
CAN 통신은 자동차 내부 전장 및 산업용 제어기에 많이 사용됩니다.
노이즈에 강하고, 프레임을 하드웨어적으로 처리하므로 소프트웨어 처리가 비교적 단순합니다.
본 강좌에서는 CAN 통신의 기본적인 설명과 실행 예제를 첨부하였으므로 사용자가 프로그램을 수정 첨가하면서 사용하시면 됩니다.

CAN 통신 시험 장비: CANPro Analyzer



CAN 수신 데이터 LCD 표시 예

CAN 통신 파형 관찰



ACK 신호

본 자료는 원본 손상 없이 배포 활용 가능합니다.

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

CAN 통신 특징 살펴보기

RealSYS

CAN 통신 이력:
- 1983 : Bosch사에서 자동차 용 네트워크 프로젝트 시작
- 1986 : Bosch사에서 CAN 통신 프로토콜 소개
- 1987 : Intel 및 Philips사에서 첫 번째 CAN controller 제작
- 1991 : Bosch에서 CAN 스펙 2.0 발표
- 1992 : CAN in Automation (CiA) 유저 그룹 결성
  CiA에서 CAL(CAN Application Layer) 프로토콜 발표
  Benz사에서 CAN 통신을 사용한 첫 번째 자동차 출시
- 1993 : ISO 11898 표준 제정
- 1994 : 1st international CAN Conference (iCC) by CiA
  DeviceNet 프로토콜 소개(Allen-Bradley사)
- 1995 : ISO 11898 확장 포맷 발표
  CANopen 프로토콜 발표(CiA)
- 2000 : time-triggered communication protocol for CAN (TTCAN) 개발

응용 분야:

- Passenger cars
- Trucks and buses
- Off-highway and off-road vehicles
- Passenger and cargo trains
- Maritime electronics
- Aircraft and aerospace electronics
- Factory automation
- Machine control
- Lifts and escalators
- Building automation
- Medical equipment and devices
- Non-industrial control
- Non-industrial equipment

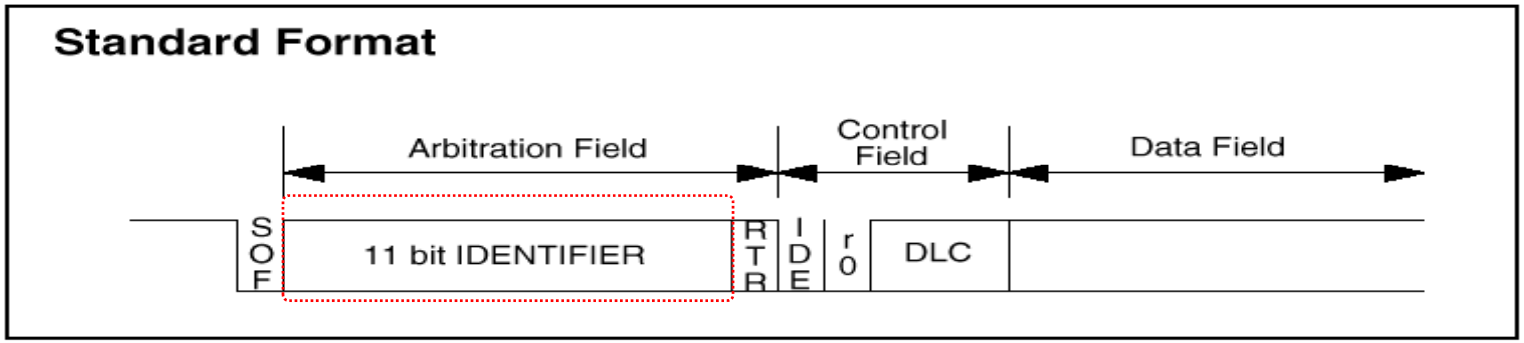
CAN 통신 특징:
- Multi-Master 구조:
  통신 신호 충돌 대책이 있음(CSMA/CA).
  메시지 ID간 우선 순위 있음.
  데이터 송신 충돌 정지 시, 선로가 비어 있을 때 자동 재 전송 기능.
- 통신 속도 : 최대 1 MBPS까지 통신 가능
- 통신 데이터 수 : 8바이트 (산업용/차량용 제어 장치에 적합)
- 통신 프로토콜/에러 처리를 Hardware적으로 처리.
- 다수의 장치(Standard: 11bit, Extended:29bit ID구분)간 통신 가능.
- 노이즈 환경에 강함(응용 분야 예: 자동차 내부 신호간 통신)
- 비용이 경제적 임 : 여러 회사의 MCU 내에 기본 장착이 늘고 있음.
  예) Motorola, Philips, TI, Hitachi, Dallas, Intel 등

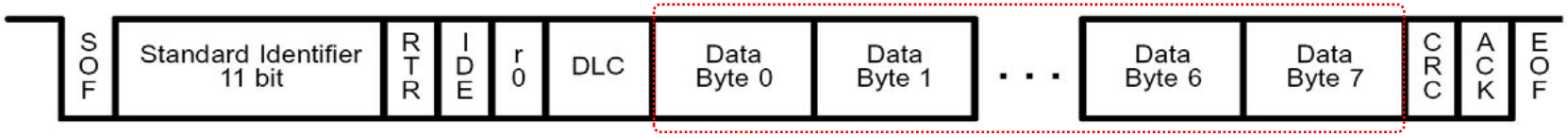
- CAN 통신을 개발한 회사: 독일의 Robert Bosch 사

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

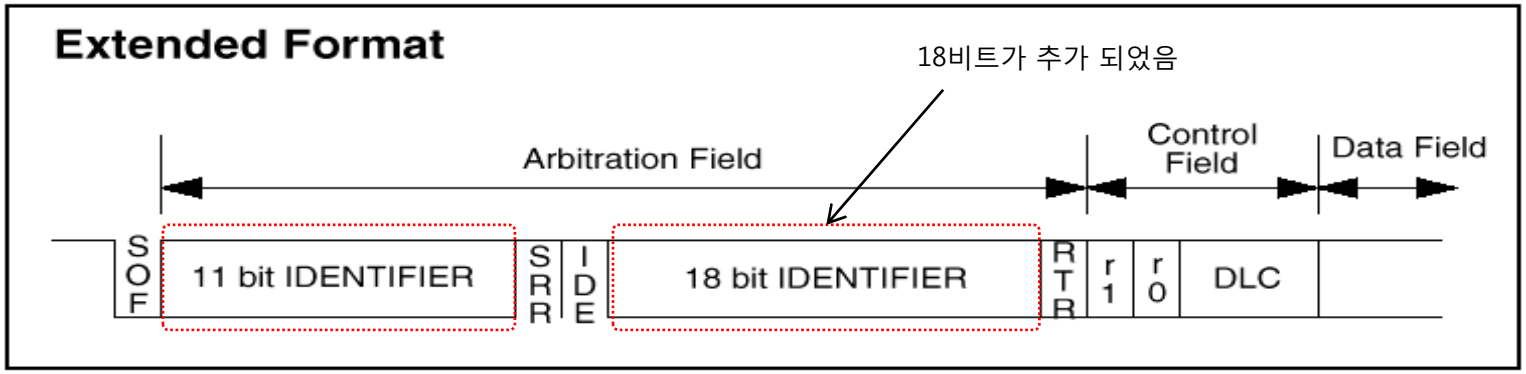
CAN 통신 포맷

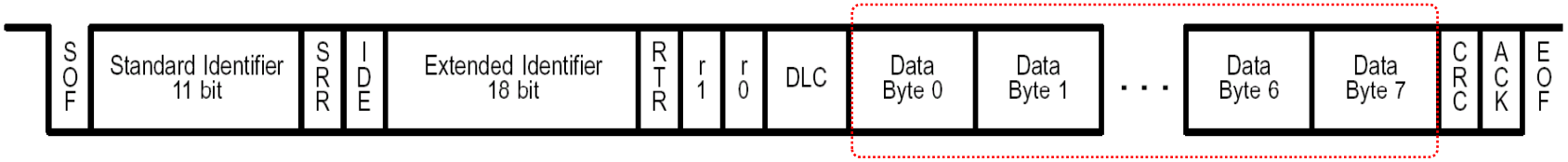
CAN 통신의 편리한 점: 통신 포매 관리를 프로그램에서 하지 않고 하드웨어적으로 처리
CAN 통신의 최대 데이터 수는 8바이트 임
표준 형태(2.0A)는 ID가 11비트이고, 확장형태(2.0B)는 ID가 29비트 임
통신 속도는 최대 1Mbps 임





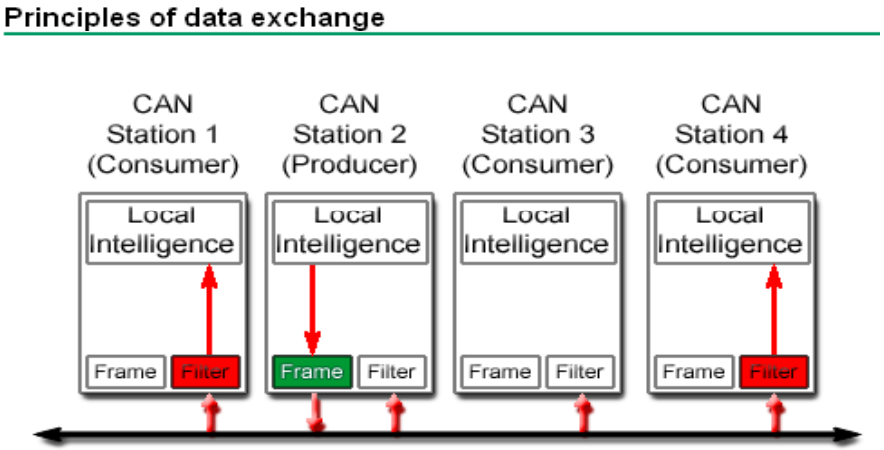
데이터 수는 최대 8바이트 임, 더 많은 양의 데이터를 보내려면 ID를 사용하여 여러 프레임으로 나누어 전송하면 됨



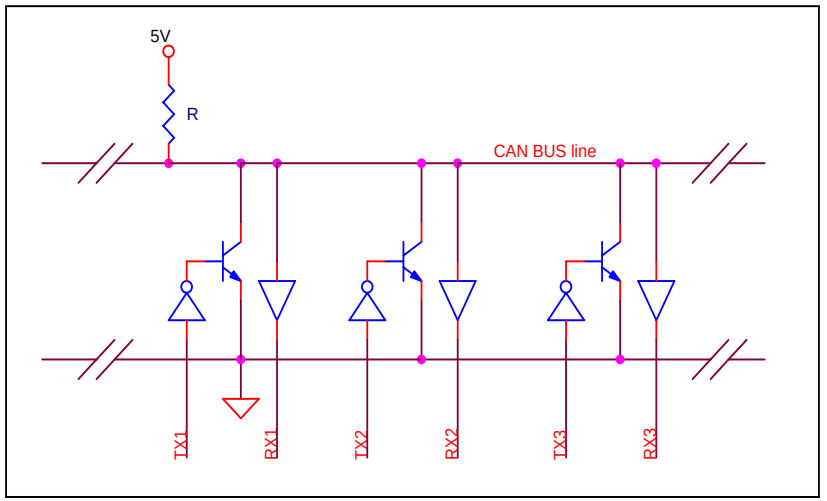


리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

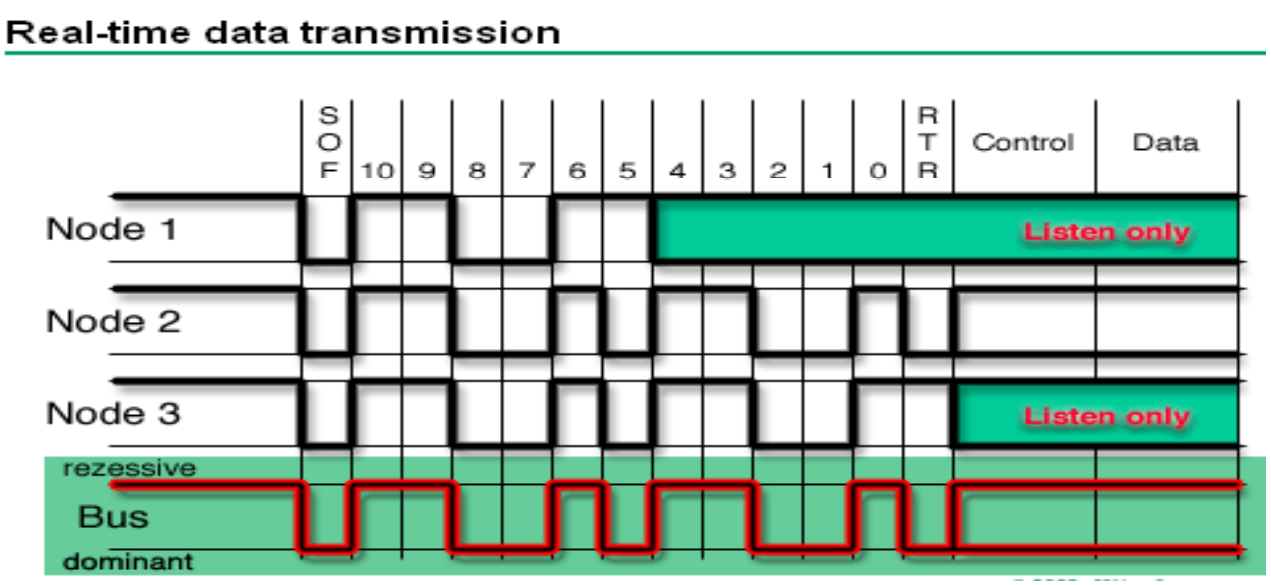
CAN 통신 동작 방식



하나의 통신 선로에 여러 장치 연결 가능



통신 신호를 오픈 컬렉터 형태 신호로 이해 바람



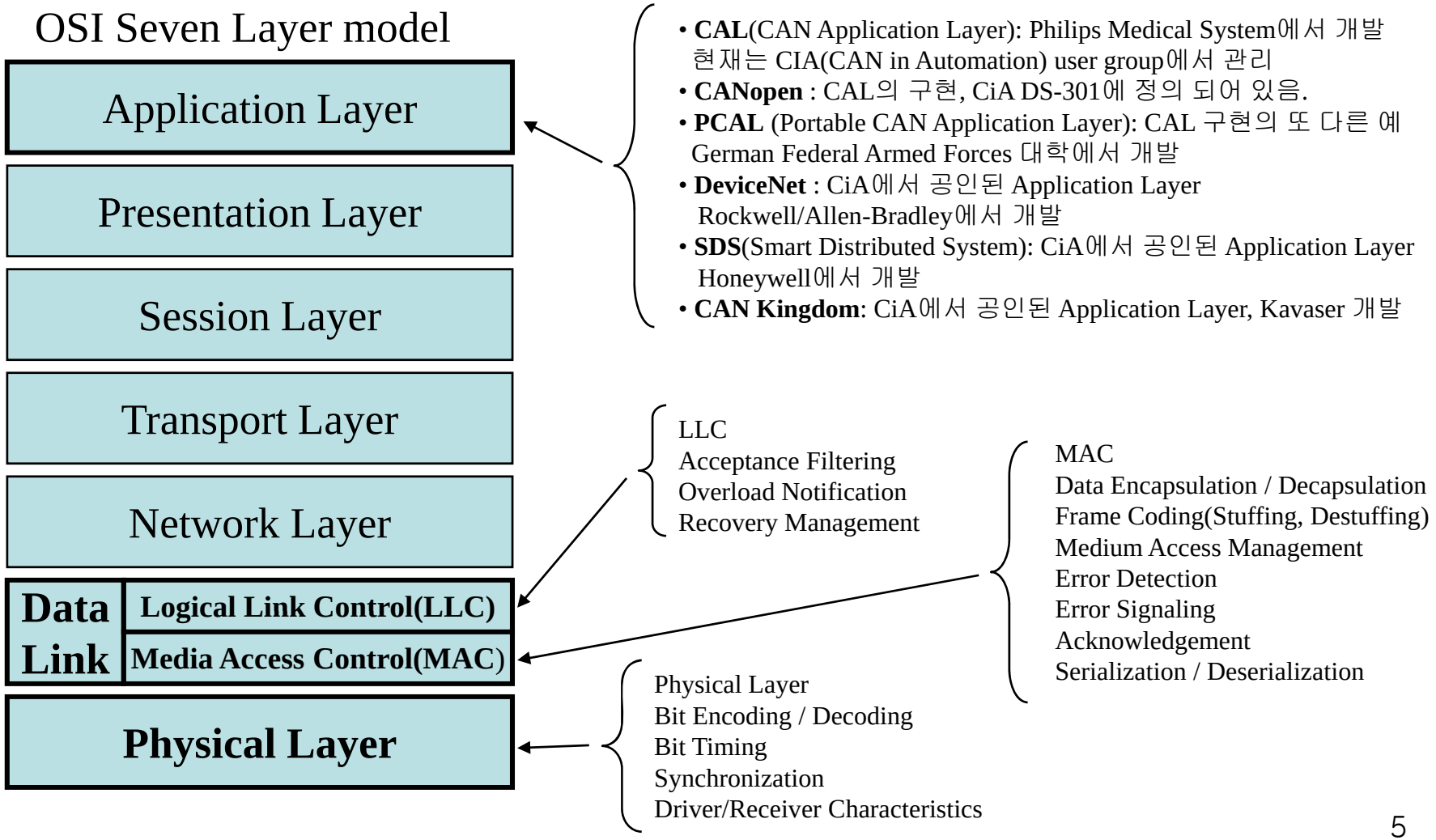
여러 장치가 신호를 함께 보내고 있는데 신호 상태를 피드백 받으면서 진행, 보낸 신호와 동일하면 다음 비트 진행
그런데 내가 High 상태를 보내고 있는데 다른 장치가 Low를 보내면 선로 상태는 Low가 됨. 그러면 본인은 전송을 즉시 중지 함.
따라서 Low값을 보낸 장치가 우선 순위가 높게 진행.
중간에 전송을 중지한 장치는 다음에 선로가 비어 있을 때 재전송 시도.
전송을 잘 끝마친 장치는 송신 OK !!!
뭐 ~ 그런 원리로 진행합니다.
이해가 안되시면 왼쪽 파형을 잘~~ 살펴보세요. 이해될 때까지.

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

통신의 6계층: CAN 통신은 어떻게 분류?

RealSYS

통신을 그래도 좀 한다 하시는 분들은 좀 유식하게 레이어를 6계층으로 나누어 다루는 거 같아요. (저도 잘 모르는데, 아는 체 하는 거 임 ~)
아래 왼쪽과 같은 6계층이 있는데, 가장 아래 층은 실제 통신의 매체, 예를 들면 RS485 방식 또는 LAN 통신과 같은 차동 또는 펄스트랜스 사용, 멘체스터 방식, 바이폴라 방식 등 그런 뭐 그런 부류가 있고,
제일 위층에는 응용 계층이 있는데, Ethernet에서는 그래도 6계층으로 어느 정도 나누어 설명하는데,
CAN 통신은 비교적 간단하므로 3계층으로 나누어 다루는 거 같습니다



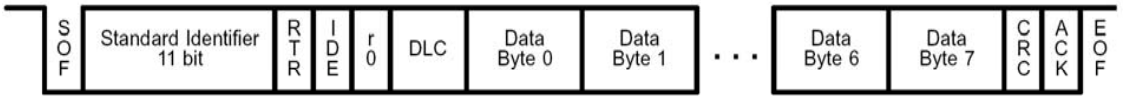
CAN 통신 프레임: 데이터 프레임

CAN의 통신 프레임에는 크게 4개의 프레임이 있습니다.

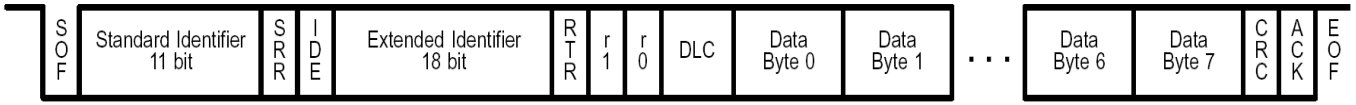
데이터 프레임(DATA FRAME), 리모트 프레임(REMOTE FRAME), 에러 프레임(ERROR FRAME), 오버로드 프레임(OVERLOAD FRAME)

데이터 프레임:
데이터 프레임은 아래와 같은 데이터를 전송하는 포맷이죠.
버전에 따라 2.0A, 2.0B나 나누어 지고, 8바이트까지 데이터를 가지고 있고…

Standard
Data Frame(2.0A)



Extended
Data Frame(2.0B)



기본 항목을
살펴봅시다.

SOF (Start of Frame) : 프레임의 시작 의미

Identifier : 표준 프레임(11 비트), 확장 프레임(11+18 = 29 비트).
이 ID로 메시지 우선 순위 결정 및 메시지 수신 여부 결정.

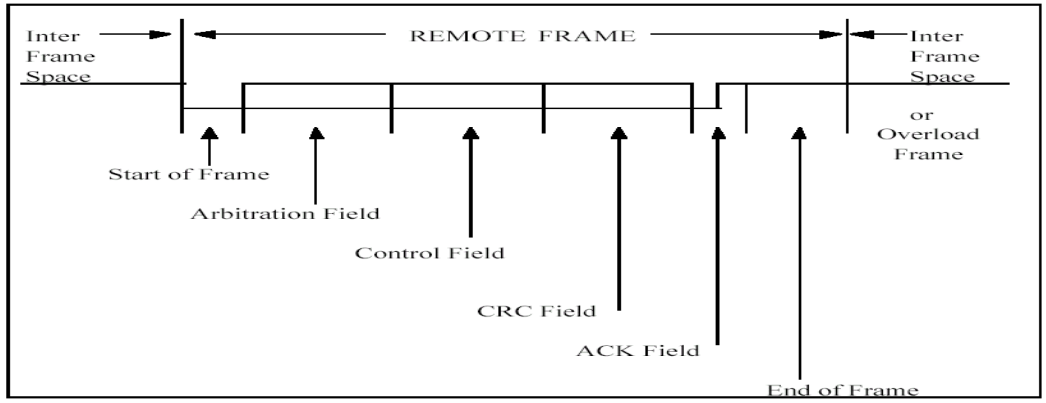
RTR (Remote Transmission Request) : 데이터 프레임과 리모트 프레임 구분
데이터 프레임은 항상 0 (dominant) 상태 임.

SRR (Substitute Remote Request) : 표준 프레임의 RTR 위치 점유
IDE (Identifier Extension) : 표준 프레임과 확장 프레임 구별
r0,r1 : 예약
DLC (Data Length Code) : 데이터 프레임의 데이터 바이트 수, 범위: 0 ~ 8
Data : 8 바이트 크기, MSB(Most Significant Bit)부터 송신 됨
CRC (Cyclic Redundancy Check) : 16 비트 체크섬(checksum)
ACK (Acknowledgement) : 응답
EOF (End of Frame) : 프레임의 끝 의미

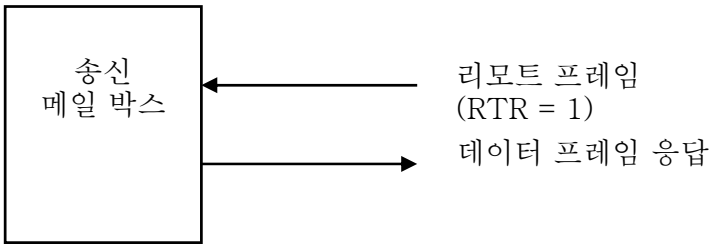
CAN 통신 프레임: 리모트/에러/오버로드 프레임

리모트 프레임:

리모트 프레임은 데이터 프레임에서 RTR 비트가 1상태이고 데이터 영역이 없는 형태를 취합니다.
리모트 프레임은 데이터를 요구하는데 사용되며, 리모트 프레임의 요구를 받은 장치는 데이터 프레임으로 응답하겠죠.

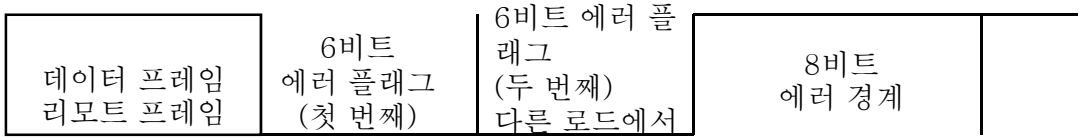


리모트 프레임에는 데이터 영역이 없음



송신 메일 박스가 자동 응답 기능이
설정되어 있고 ID가 일치해야 응답 함

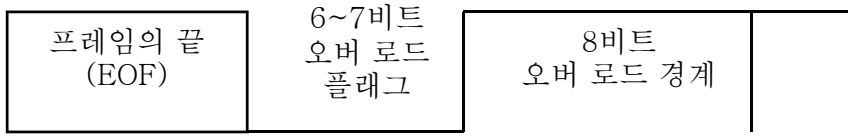
에러 프레임



에러프레임은 프레임에 프레임에 에러가 발견된 경우, 일부러 규칙을 어기는(스터핑 에러) 프레임을 날려, 에러 있음을 알립니다.
에러 프레임은 0 상태의 6~12비트 크기의 에러 플래그와 1 상태의 8비트 크기 에러 종료 상태로 이루어져 있으며, 현재의 송신 프레임은 에러가 있으므로 다시 전송하라는 의미로 비트 stuffing 규칙을 일부러 손상 시키며, 따라서 현재 전송 중인 장치는 에러로 인해 현재의 프레임이 실패 하였음을 인식하고 즉시 중지하여 다음의 재 전송 기회를 기다리겠죠?
뭐 ~ 좀 복잡 한 거 같은데, 사용자 입장에서는 이 프레임을 신경쓰지 않아도 됩니다. 다 ~~ 하드웨어적으로 처리되므로…

오버로드 프레임

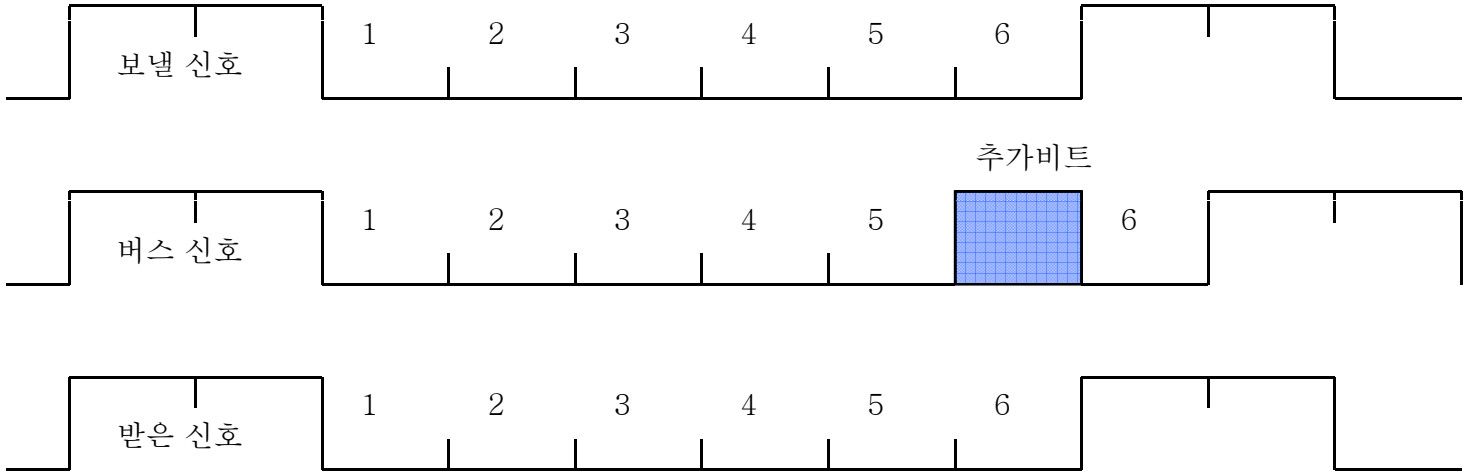
오버로드 프레임은 CAN 제어기가 아직 이전 메시지의 처리를 마치지 못한 경우 다음 메시지의 시작을 지연시키기 위해서 사용되며, 오버 로드 프레임은 에러카운터 수에 영향을 미치지 않아요.



마찬가지로 사용자는 몰라도 됩니다.
다 ~~ 하드웨어가 알아서 하므로 ~~

비트 스터핑이 무엇인가?

CAN 통신에서는 비동기 통신에서와 같이 NRZ(Non Return to Zero) 코딩 방법을 사용합니다.
다시 말해 1비트 동안은 같은 값을 유지하죠. 따라서 같은 상태가 계속 이어질 수도 있고, 이 경우 수신 측에서는 비트의 경계를 알 수 없으므로 오차가 누적될 수 있겠죠.
비동기 통신의 경우에는 바이트 단위로 구분(시간이 짧음)하고 , 스타트, 스톱 비트 등이 있으므로 클럭의 동기를 쉽게 잡을 수 있지만, CAN의 경우에는 프레임이 여러 바이트로 구성되어 길이가 길어지므로 비트 극성이 바뀌는 시점이 길어져서 에러가 발생할 수 있으므로 이에 대한 대책이 필요합니다.
이러한 대책으로 CAN에서는 5 비트 동안 같은 상태가 유지되면 반대 상태를 1 비트 추가하여 송신하고, 수신 장치는 이를 고려하여 원래대로 다시 복원합니다. 이것이 바로 비트 스터핑입니다.
그러면 연속하여 5비트 이상 동안 0 또는 1 상태가 지속되지 않겠죠. 그러면 ~~ 샘플링 위치 에러가 적어질 수 있다는…
비트 스터핑이 이해가 안 되시는 분은 그냥 몰~~라도 돼요. 다 ~~ 하드웨어가 알아서 처리하므로…



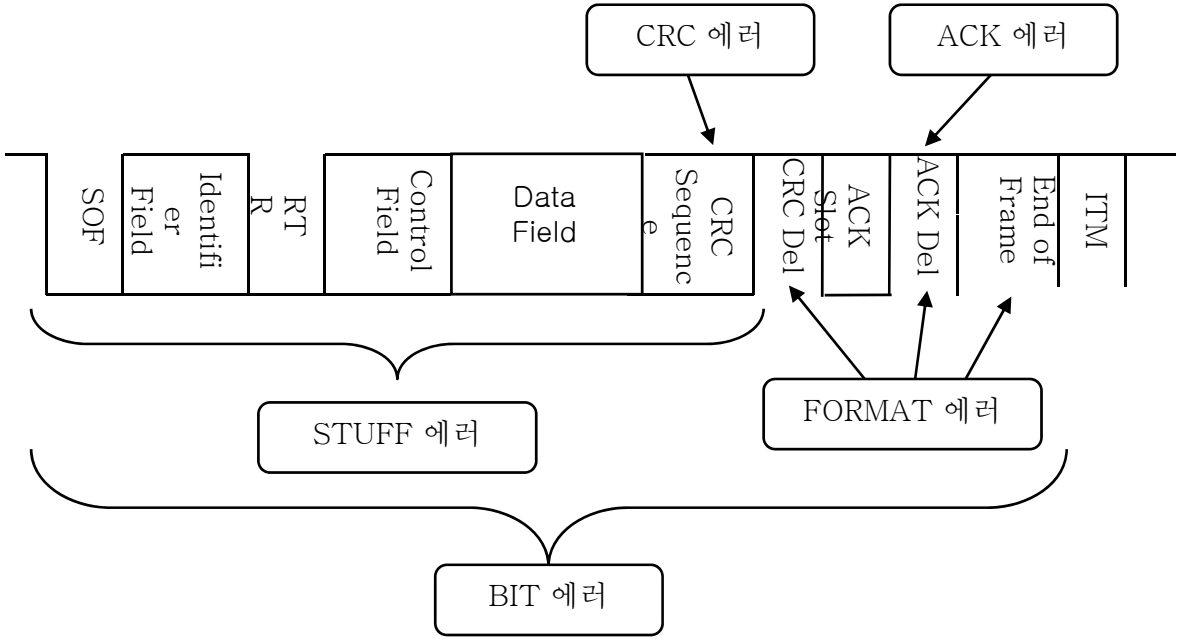
리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

CAN 통신에는 어떤 종류의 에러들이 있나?

CAN 통신이 자동차와 같이 노이즈가 심한 환경에서도 충분히 잘 동작하여 그 동안 신뢰성이 검증된 이유중에 하나가 에러 처리라도 생각도 되는데, 생명을 맡기고 동작시키는 시스템에서 통신 에러가 발생하여 오작동을 한다… 끔찍하겠죠… 뭐 ~ 바로 퇴출이 되겠죠.
CAN 통신은 훌륭한 에러 검출 능력이 있습니다. 아래그림에서와 같이 철저히 에러를 검출하죠.
그래서 CAN 통신은 신뢰성이 높고, 많은 소자에서 계속 채택되어 사랑을 받고 있어요…

송신 장치에서 발생하는 에러:
BIT 에러 : 보낸 비트와 받은 비트가 서로 다르면 비트 에러 발생
ACK 에러 : ACK 슬롯 동안 수신 장치가 0을 보내 주지 않을 때, 이 에러는 모든 수신 장치가 접속되어 있지 않을 때 (통신 선로가 끊겼을 때 등) 발생.

수신 장치에서 발생하는 에러:
CRC 에러 : CRC 가 맞지 않을 때 에러 발생
STUFF 에러 : 스터핑(stuffing) 규칙이 맞지 않을 때 에러 발생
FORMAT 에러 : CRC Del, ACK Del, EOF의 영역에 0(dominant) 값이 나타날 때 발생



만일 에러가 발생되면 즉시 에러 프레임이 전송된다.

단 CRC 에러가 발생된 경우 EOF의 첫번째 비트가 나올 때까지 에러 프레임을 보내지 않는다.

그 이유는 모든 다른 장치들이 ACK를 받을 때까지 기다리도록 하기 위함이다.

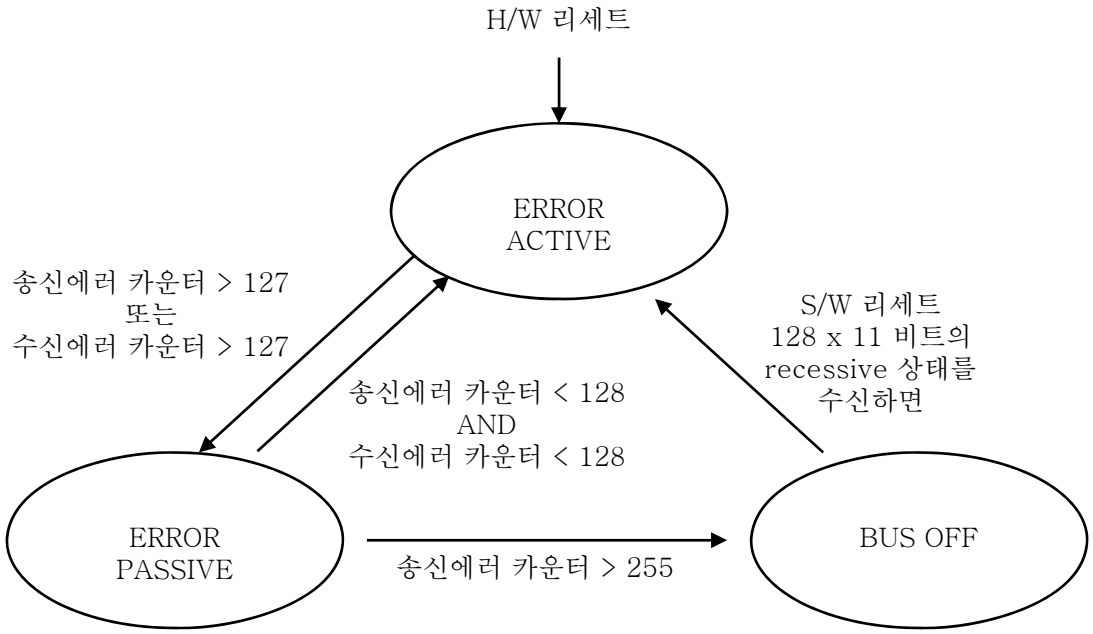
리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

CAN 통신의 에러 상태 이동 살펴보기

한국 사람들로 꼼꼼(좀 섬세하고, 철두철미의 의미, 이거~ 나만 쓰는 사투리인가?)하지만, 독일 사람들도 매우 꼼꼼한 거 같아요. CAN 통신도 독일 사람들이 만들었어요.

좀 머리가 아프지만 좀 설명을 해보죠.
CAN 제어기 내부에 송신 에러 카운터(TX_CNT)와 수신 에러 카운터(RX_CNT)가 있으며, 이들 카운터는 에러가 검출되면 1씩 증가하고, 에러 없이 성공하면 1씩 감소하며, 이들 에러 카운터 수에 따라 노드의 상태가 달라 집니다.
초기 상태는 에러 액티브(ERROR ACTIVE) 상태이며, 에러 카운터 값이 127보다 크면 에러 패시브(ERROR PASSIVE) 상태가 됩니다.
이 때에는 액티브 에러 프레임(6비트 0상태)을 보내지 못하고, 패시브 에러 프레임(6비트 1상태)을 보낼수 있어요.
즉 에러 프레임을 발생하는 권리를 제한 시키는데, 그 이유는 자꾸 이상한 에러를 발생하여 전체 통신망을 마비 시킬 수 있기 때문이죠.
또한 두 개의 연속적인 메시지 전송을 못하도록 부가적인 지연시간을 갖도록 합니다. 즉 에러 발생이 많은 노드는 전송 회수를 제한해요.
만일 송신 에러 카운터가 255를 넘으면 버스 오프(BUS OFF) 상태가 되어 이 노드는 CAN 통신망에서 접속을 끊어버립니다(이 장치는 많은 에러가 발생하므로 뭔가 문제가 많다… 문제아야~~)
이 버스 오프 상태에서 벗어나는 방법은 소프트웨어 리세트 또는 하드웨어 리세트를 걸거나 부가적인 웨이트 시간(128 x 11 비트 1 (recessive) 상태)을 기다려야 합니다.

에러 상태 변화 도는 아래 그림과 같습니다.



리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

RealSYS

CANopen

- **Features**
  - CANopen a subset from CAL (CAN Application Layer) developed by CiA!
  - Auto configuration the network
  - Easy access to all device parameters
  - Device synchronization
  - Cyclic and event-driven data transfer
  - Synchronous reading or setting of inputs, outputs or parameters
- **Applications**
  - Machine automatisation
- **Advantages**
  - Accommodating the integration of very small sensors and actuators
  - Open and vendor independent
  - Support s inter-operability of different devices
  - High speed real-time capability

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

RealSYS

DeviceNet

- **Features**
  - Created by Allen-Bradley (Rockwell Automatisation nowadays), now presented by the users group ODVA (Open DeviceNet Vendor Association)
  - Power and signal on the same network cable
  - Bus addressing by: Peer-to-Peer with multi-cast & Multi-Master & Master-Slave
  - Supports only standard CAN
- **Applications**
  - Communications link for industrial automatisation: devices like limit switches, photo-electric sensors, valve manifolds, motor starters, process sensors, bar code readers,variable frequency drives, panels...
- **Advantages**
  - Low cost communication link and vendor independent
  - Removal and replacement of devices from the network under power

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

RealSYS

CAN Kingdom

- CAN Kingdom is more then a HLP: A Meta protocol
  - Introduced by KVASER, Sweden
  - A 'King' (system designer) takes the full responsibility of the system
  - The King is represented by the Capital (supervising node)
  - World wide product identification standard EAN/UPC is used for
- Applications
  - Machine control, e.g. industrial robots, weaving machines, mobile hydraulics, power switchgears, wide range of military applications
- Advantages
  - Designed for safety critical applications
  - Real time performance
  - Scalability
  - Integration of DeviceNet & SDC modules in CAN Kingdom possible

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

RealSYS

OSEK/VDX

Offene Systeme und deren Schnittstellen fuer die Elektonik im Kraftfahezeug/Vehicle Distributed eXecutive)

- **Initialized by:**
  - BMW, Bosch, DaimlerChrysler, Opel, Siemens,VW & IIIT of the University of Karlsruhe / PSA and Renault
- **OSEK/VDX includes:**
  - Communication (Data exchange within and between Control Units)
  - Network Management (Configuration determination and monitoring)
  - Operating System (Real-time executive for ECU software)
- **Motivation:**
  - High, recurring expenses in the development and variant management of non-application related aspects of control unit software
  - Compatibility of control units made by different manufactures due to different interfaces
- **Goal:** Portability and re-usability of the application software
- **Advantages:** Clear saving in costs and development time

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

RealSYS

SAE J1939

- Features
  - Developed by Society of Automotive Engineers heavy trucks and bus division (SAE)
  - Use of the 29 identifiers
  - Support of real-time close loop control
- Applications
  - Light to heavy trucks
  - Agriculture equipment e.g. tractors, harvester etc...
  - Engines for public work

Smart Distributed System (SDS)

- Features
  - Created by Honeywell
  - Close to DeviveNet, CAL & CANopen

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

bxCAN main features

- Supports CAN protocol version 2.0 A, B Active
- Bit rates up to 1 Mbit/s
- Supports the Time Triggered Communication option

Transmission

- Three transmit mailboxes
- Configurable transmit priority
- Time Stamp on SOF transmission

Reception

- Two receive FIFOs with three stages
- Scalable filter banks:
  - 28 filter banks shared between CAN1 and CAN2 in connectivity line devices
  - 14 filter banks in other STM32F10xxx devices
- Identifier list feature
- Configurable FIFO overrun
- Time Stamp on SOF reception

Time-triggered communication option

- Disable automatic retransmission mode
- 16-bit free running timer
- Time Stamp sent in last two data bytes

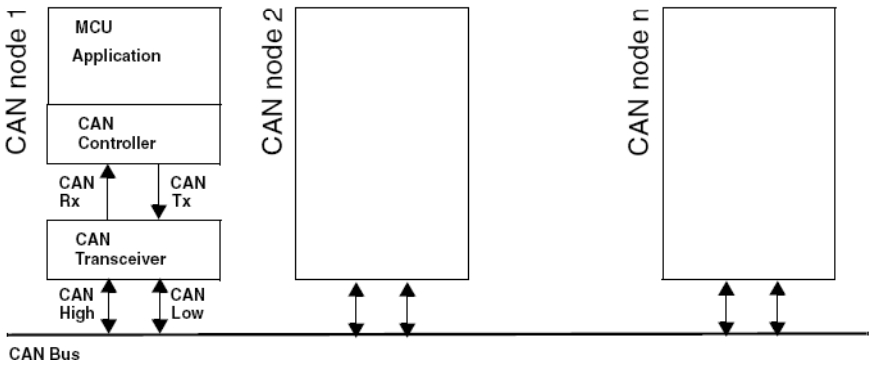
Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

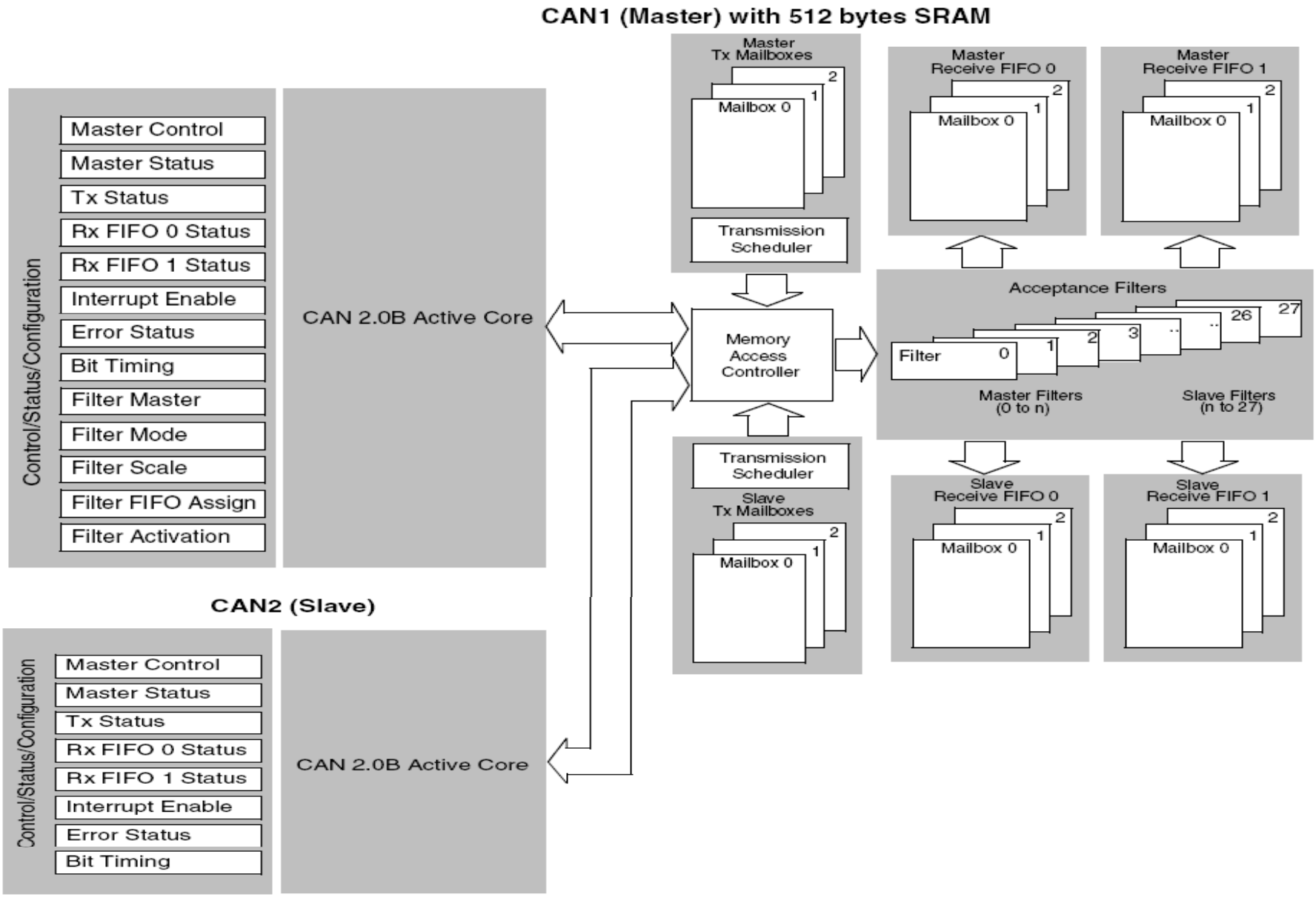
이제 STM32F시리즈에 있는 CAN 통신에 대해서 살펴봅니다.
회사마다 CAN이름을 조금씩 다르게 붙이는데, 기본 동작은
거의 같고, 조금씩 차이는 있는 거 같습니다.
아무튼 STM32에서는 bxCAN이라 하네요.
Basic Extended CAN(Controller Area Network)

주요 특징은 왼쪽과 같은데,
• 2.0A 및 2.0B 지원
• 1Mbps까지 지원
• 3개의 송신 메일박스
• 2개의 수신 FIFO (3 stages)
등등..

하나의 통신라인에 여러 장치 접속 가능

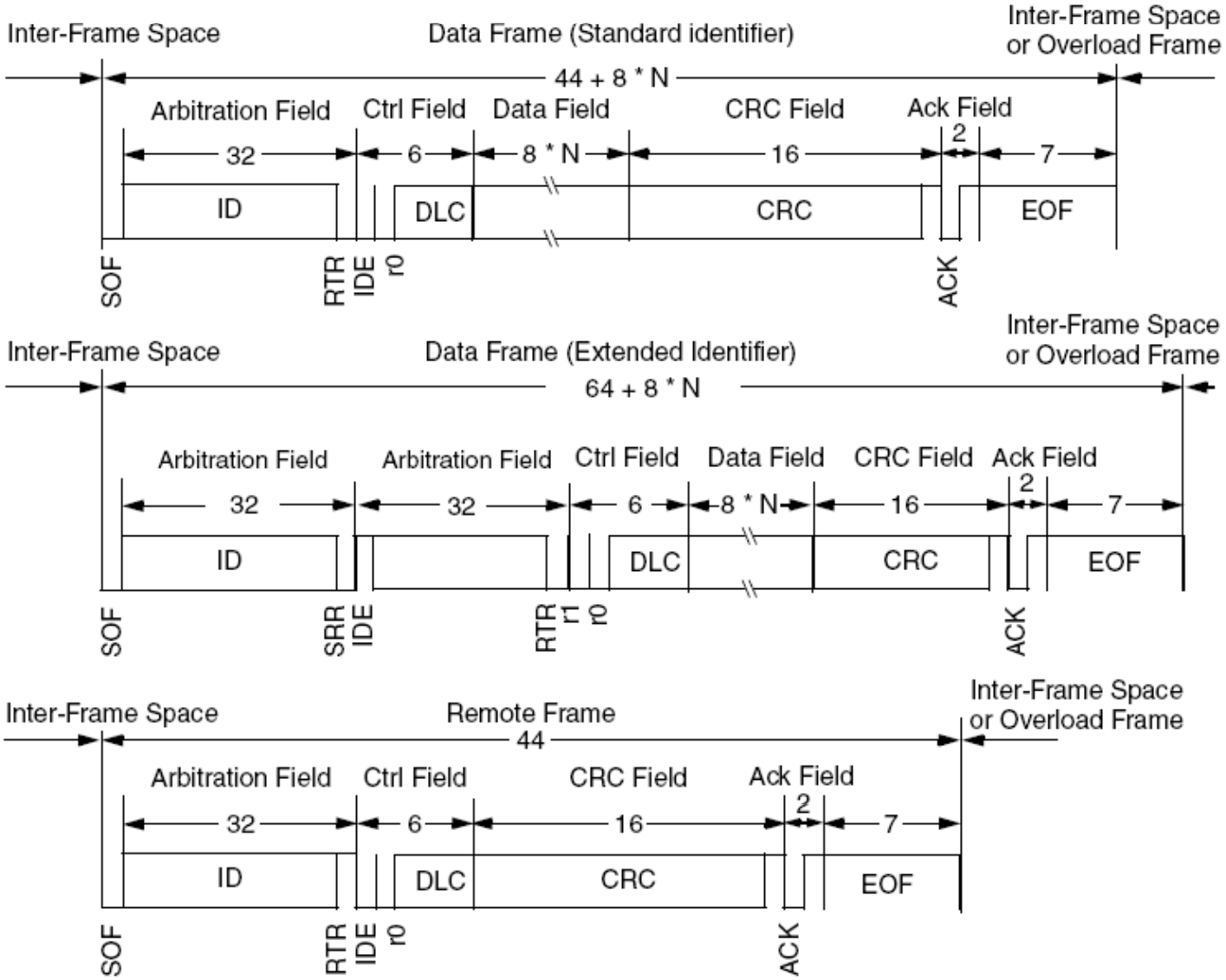


Dual CAN 구성도(connectivity devices)



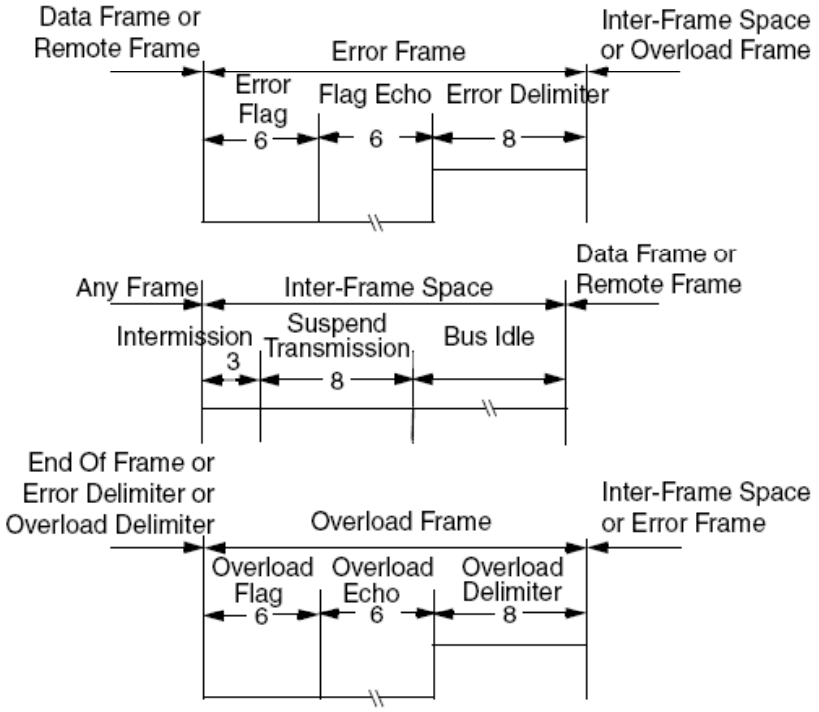
리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

STM32의 CAN 통신 프레임



리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

STM32의 CAN 통신 프레임



Notes:

0 <= N <= 8

SOF = Start Of Frame

ID = Identifier

RTR = Remote Transmission Request

IDE = Identifier Extension Bit

r0 = Reserved Bit

DLC = Data Length Code

CRC = Cyclic Redundancy Code

Error flag: 6 dominant bits if node is error active else 6 recessive bits.

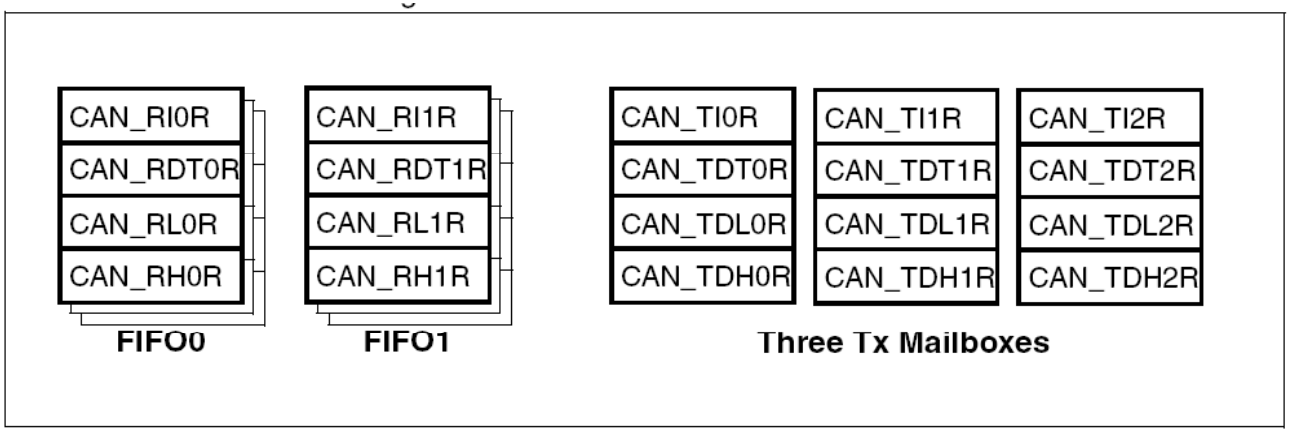
Suspend transmission: applies to error passive nodes only.

EOF = End of Frame

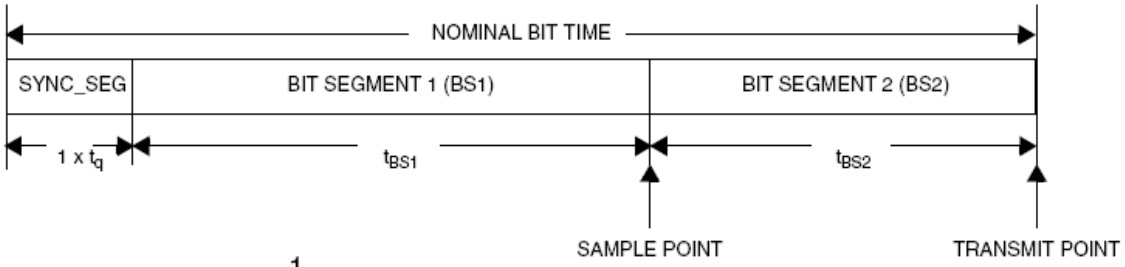
ACK = Acknowledge bit

Ctrl = Control

메일박스의 레지스터 구성



CAN 통신 비트 타이밍 계산



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

with:

$t_{BS1} = t_q \times (TS1[3:0] + 1)$,

$t_{BS2} = t_q \times (TS2[2:0] + 1)$,

$t_q = (BRP[9:0] + 1) \times t_{PCLK}$

where t_q refers to the Time quantum

t_{PCLK} = time period of the APB clock,

BRP[9:0], TS1[3:0] and TS2[2:0] are defined in the CAN_BTR Register.

기타 상세 내용은 매뉴얼 참조:
April 2010 Doc ID 13902 Rev 11 1/1072
RM0008
Reference manual
STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs

에서 23. Controller Area Network(bxCAN) 장 참조 바랍니다.

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

CAN 예제 살펴보기: 초기화

```
// 2010.6.2 RealSYS KCO
// can.c
#include "common.h"
#include "can.h"

u8 can_rx_flag=0,can_rx_cnt=0;
CanTxMsg TxMessage;
CanRxMsg RxMessage;

void init_can(void){
GPIO_InitTypeDef  GPIO_InitStructure;
CAN_InitTypeDef CAN_InitStructure;
CAN_FilterInitTypeDef CAN_FilterInitStructure;
NVIC_InitTypeDef  NVIC_InitStructure;
    /* GPIOA clocks enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA , ENABLE);

    /* CAN1 Periph clocks enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_CAN1, ENABLE);

    /* Configure CAN1 RX pin */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure CAN1 TX pin */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
```

CAN 통신 초기화 함수

CAN 클럭 Enable

입출력 핀 초기화

CAN 레지스터 초기화

CAN 수신 인터럽트 Enable

```
    /* CAN register init */
    CAN_DeInit(CAN1);
    CAN_StructInit(&CAN_InitStructure);
    /* CAN1 cell init */
    CAN_InitStructure.CAN_TTCM = DISABLE;
    CAN_InitStructure.CAN_ABOM = DISABLE;
    CAN_InitStructure.CAN_AWUM = DISABLE;
    CAN_InitStructure.CAN_NART = DISABLE;
    CAN_InitStructure.CAN_RFLM = DISABLE;
    CAN_InitStructure.CAN_TXFP = DISABLE;
    CAN_InitStructure.CAN_Mode = CAN_Mode_Normal;
    CAN_InitStructure.CAN_SJW = CAN_SJW_1tq;
    CAN_InitStructure.CAN_BS1 = CAN_BS1_3tq;
    CAN_InitStructure.CAN_BS2 = CAN_BS2_5tq;
    CAN_InitStructure.CAN_Prescaler = 4;
    CAN_Init(CAN1, &CAN_InitStructure);
    /* CAN1 filter init */
    CAN_FilterInitStructure.CAN_FilterNumber = 0;
    CAN_FilterInitStructure.CAN_FilterMode = CAN_FilterMode_IdMask;
    CAN_FilterInitStructure.CAN_FilterScale = CAN_FilterScale_32bit;
    CAN_FilterInitStructure.CAN_FilterIdHigh = 0x0000;
    CAN_FilterInitStructure.CAN_FilterIdLow = 0x0000;
    CAN_FilterInitStructure.CAN_FilterMaskIdHigh = 0x0000;
    CAN_FilterInitStructure.CAN_FilterMaskIdLow = 0x0000;
    CAN_FilterInitStructure.CAN_FilterFIFOAssignment = 0;
    CAN_FilterInitStructure.CAN_FilterActivation = ENABLE;
    CAN_FilterInit(&CAN_FilterInitStructure);
    /* IT Configuration for CAN1 */
    CAN_ITConfig(CAN1, CAN_IT_FMP0, ENABLE);
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);
    NVIC_InitStructure.NVIC_IRQChannel = USB_LP_CAN1_RX0_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

본 예제의 사용 환경:
소자 : STM32F103R8T6 사용
CAN 포트: USB포트와 공용 핀 사용
  수신 핀: PA11_CANRX_USBDM
  송신 핀: PA12_CANTX_USBDP
CAN Driver칩: 82C251(5V 동작)
  RXD 신호에 완충용 330옴 저항 연결 사용

CAN 통신 설정:
통신 속도: 1Mbps
인터럽트: 수신 인터럽트 사용
송신 포맷: 2.0A 및 2.0B 가능

CAN 예제 살펴보기

인터럽트 처리 함수

```
void USB_LP_CAN1_RX0_IRQHandler(void){
    can_rx_cnt++;
    CAN_Receive(CAN1, CAN_FIFO0, &RxMessage);
    can_rx_flag = 1;
}
```

수신 데이터를 저장하고,
플래그만 체크

메인 함수에서

```
while(1){
  if(can_rx_flag){
    can_rx_flag = 0;
    lcd_gotoxy(0,2);
    lcd_printf("Rx_cnt=%02X",can_rx_cnt);
    if(RxMessage.IDE==CAN_ID_STD){
      lcd_printf(",S:%03X      ",RxMessage.StdId);
    }
    else{
      lcd_printf(",E:%08X",RxMessage.ExtId);
    }
    lcd_gotoxy(0,3);
    lcd_printf("%d: %02X%02X%02X%02X %02X%02X%02X%02X",RxMessage.DLC,
               RxMessage.Data[0],RxMessage.Data[1],RxMessage.Data[2],RxMessage.Data[3],
               RxMessage.Data[4],RxMessage.Data[5],RxMessage.Data[6],RxMessage.Data[7]);
  }
```

메인 루프에서
플래그를 체크하여
CAN 수신 데이터를
LCD에 표시

CAN 수신 데이터 LCD 표시 예


```
void can_tx_data_s(u16 id, u8 *data){
u8 i;
  /* Transmit */
  TxMessage.StdId = id;  // between 0 to 0x7FF
  TxMessage.RTR = CAN_RTR_DATA; // CAN_RTR_DATA or CAN_RTR_REMOTE
  TxMessage.IDE = CAN_ID_STD;   // CAN_ID_EXT or CAN_ID_EXT;
  TxMessage.DLC = 8;
  for(i=0;i<8;i++) TxMessage.Data[i] = data[i];
  CAN_Transmit(CAN1, &TxMessage);
}
```

CAN 데이터 송신 함수: 표준(2.0A) 형태

```
void can_tx_data_e(u32 id, u8 *data){
u8 i;
  /* Transmit */
  TxMessage.ExtId = id; // between 0 to 0x1FFFFFFF
  TxMessage.RTR = CAN_RTR_DATA; // CAN_RTR_DATA or CAN_RTR_REMOTE
  TxMessage.IDE = CAN_ID_EXT;   // CAN_ID_EXT or CAN_ID_EXT;
  TxMessage.DLC = 8;
  for(i=0;i<8;i++) TxMessage.Data[i] = data[i];
  CAN_Transmit(CAN1, &TxMessage);
}
```

CAN 데이터 송신 함수: 확장(2.0B) 형태

CAN 예제 살펴보기

RealSYS

LCD(20 x 4) & 버튼(5개) 보드로 통신 시험

버튼을 누를 때, 데이터를 전송하고
송수신 상태를 LCD에 표시

BT1 : 표준 형태, “0011..” 전송

BT2 : 확장 형태, “0011..” 전송

BT3 : 확장 형태, “0123..” 전송

BT4 : 표준 형태, “0123..” 전송

BT5 : 메시지 지움

버튼을 누를 때 송신 동작:

```
void    key_down_proc(BYTE c){
    switch(c){
        case 0:
            can_txd2[0]=0x00;
            can_txd2[1]=0x11;
            can_txd2[2]=0x22;
            can_txd2[3]=0x33;
            can_txd2[4]=0x44;
            can_txd2[5]=0x55;
            can_txd2[6]=0x66;
            can_txd2[7]=0x77;
            can_tx_data_s(0x000, can_txd2); // send std. type
            lcd_gotoxy(0,2);    lcd_puts("CAN2.0A(Std)ID=0x000");
            lcd_gotoxy(0,3);    lcd_puts("Txd:0011223344556677");
            break;
```

BT1 누름

```
        case 1:
            can_txd2[0]=0x00;
            can_txd2[1]=0x11;
            can_txd2[2]=0x22;
            can_txd2[3]=0x33;
            can_txd2[4]=0x44;
            can_txd2[5]=0x55;
            can_txd2[6]=0x66;
            can_txd2[7]=0x77;
            can_tx_data_e(0x12345678, can_txd2); // send ext. type
            lcd_gotoxy(0,2);    lcd_puts("CAN2.0B,ID=0x1234567");
            lcd_gotoxy(0,3);    lcd_puts("Txd:0011223344556677");
           break;
```

BT2 누름

```
        case 2:
            can_txd2[0]=0x01;
            can_txd2[1]=0x23;
            can_txd2[2]=0x45;
            can_txd2[3]=0x67;
            can_txd2[4]=0x89;
            can_txd2[5]=0xab;
            can_txd2[6]=0xcd;
            can_txd2[7]=0xef;
            can_tx_data_e(0x11223344, can_txd2); // send ext. type
            lcd_gotoxy(0,2);    lcd_puts("CAN2.0B,ID=0x11223344");
            lcd_gotoxy(0,3);    lcd_puts("Txd:0123456789abcdef");
            break;
```

BT3 누름

버튼 동작은 메인 루틴에서 일정 간격으로 처리 함

```
        case 3:
            can_txd2[0]=0x01;
            can_txd2[1]=0x23;
            can_txd2[2]=0x45;
            can_txd2[3]=0x67;
            can_txd2[4]=0x89;
            can_txd2[5]=0xab;
            can_txd2[6]=0xcd;
            can_txd2[7]=0xef;
            can_tx_data_s(0x7ff, can_txd2); // send std. type

            lcd_gotoxy(0,2);    lcd_puts("CAN2.0A(Std)ID=0x7ff");
            lcd_gotoxy(0,3);    lcd_puts("Txd:0123456789abcdef");
            break;
```

BT4 누름

```
        case 4:
            lcd_gotoxy(0,2);
            lcd_puts("                    ");
            lcd_gotoxy(0,3);
            lcd_puts("                    ");
            clear_can_rx_message(&RxMessage);
            break;
        default:
            break;
    }
}
```

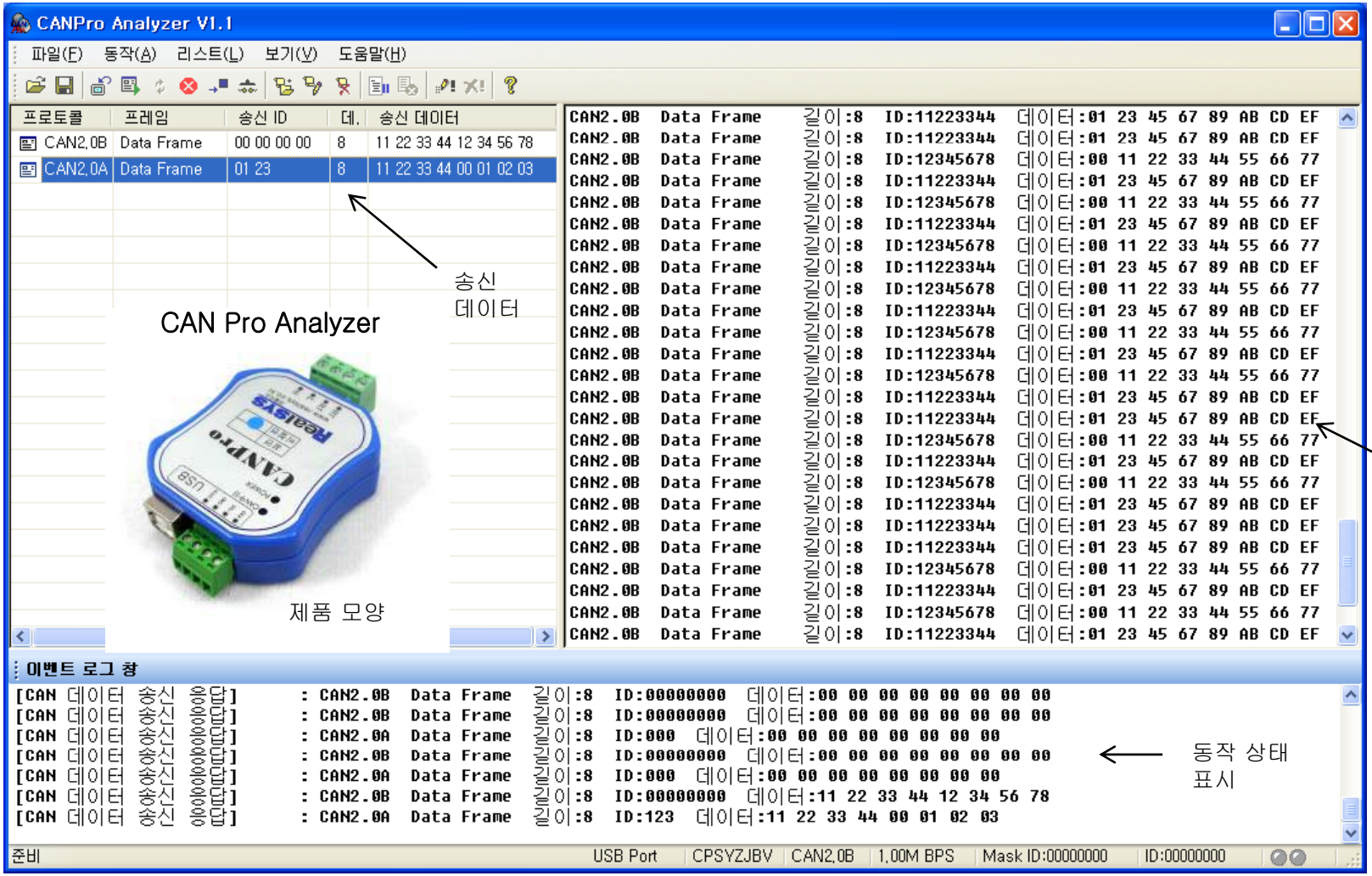
BT5 누름

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

CAN 통신 데이터 관찰

CAN 통신은 1개의 장치로 시험이 어렵고, 2대 이상 접속하여 시험해야 하는데, 본 강좌에서는 리얼시스(www.realsys.co.kr)사의 CANPro Analyzer를 사용하여 데이터 수신 및 송신 동작을 했습니다.

CANPro 실행 화면



제품 관련 정보는:
http://www.realsys.co.kr/goods/main.asp?cate=357

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호