FlexRay

소개 - 분석 - **Rest Bus** 시뮬레이션

Dipl.-Ing. Roman M. F. Hofmann
IXXAT, Weingarten

목차

- FlexRay 통신 소개
  애플리케이션, 프로토콜 원리, 동기화(synchronization),
  그리고 시동(start-up) 위상들
- FlexRay 측정 장비를 기존 FlexRay 네트워크에 연결하는
  방법. 제약, 필수 요건, 요령
- FlexRay 네트워크의 구성, FlexRay 분석과 시뮬레이션 장치
- MultibusAnalyser를 이용한 FlexRay 네트워크의 분석
- FlexRay 시스템의 Rest Bus 시뮬레이션을 위한 요건과
  해법

FlexRay 배경 [1/2]

- 미래 차량에서의 전기와 통신 요건의 양적인 면에서의 획기적인 증가
- 안전, 신뢰성, 편의와 관련된 필수 요건들
- 향후 차량-내 제어 애플리케이션들의 추가 요건들은 현재의 통신 프로토콜들로 처리될 수 없습니다

⇒ FlexRay 는 차세대 통신 시스템입니다:

- 보다 빠른 데이터 속도 (10 Mbit/s)
- 시간 결정적 행동
- 오류 안정(fault-tolerance) 지원
- 대역폭과 시스템 확장에서의 유연성

FlexRay 배경[2/2]

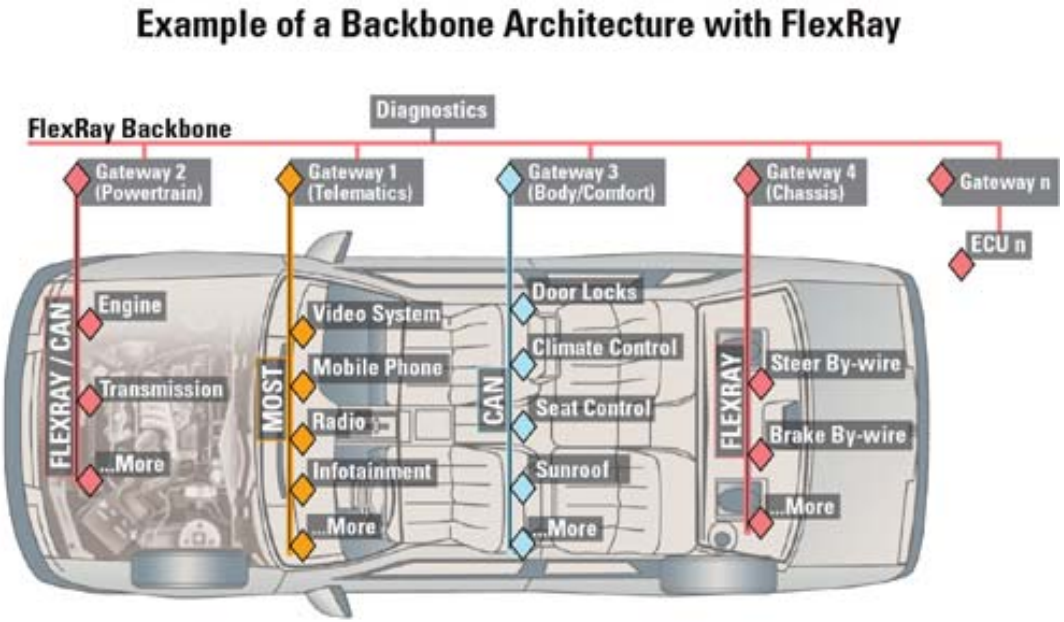
- FlexRay 프로토콜은 FlexRay 컨소시엄 (코어 회원: BMW, DC, GM, VW, Freescale, Philips, Bosch)에 의해 개발되었습니다
- FlexRay 는파워-트레인, 섀시, X-by-wire 시스템과 같은 차세대 고속 제어 애플리케이션들을 위한 통신 기반 구조를 제공합니다

실리콘 상태:

- Silicon protocol chip Freescale MFR 4200A, Freescale MFR4300 Bosch IP ERay; Freescale EagleRay
- Bus Driver Philips TJA1080N1D

FlexRay 사용 범위

- 보다 높은 대역폭이 요구되는 경우 CAN 대체
- 루프 제어 애플리케이션에서, 실시간
- Back Bone
- 안전 관련 시스템
- X-by wire 시스템



FlexRay 애플리케이션 상황

- 거의 모든 차 제조업체들이 FlexRay 컨소시엄의 회원입니다
- DC, VW, GM 은 FlexRay 애플리케이션의 개발을 시작했습니다
- BMW는 2006년에 FlexRay 차량의 생산을 시작합니다
  new X5 model (제동 시스템)
  new 7 series with FlexRay power train network
- Audi는 2007년 후반에 FlexRay 시리즈를 이용한 새로운 A8 차량을 계획하였습니다 (power train)
- Toyota 는 2007년에 FlexRay 도입을 계획하였습니다 (JasPar)
- FlexRay는 또한 경주용 차량에서도 사용됩니다: BMW 레이싱, 포뮬러 1 의 페라리

FlexRay 프로토콜의 핵심 특징

Time and event triggered 통신 프로토콜들의 장점 결합으로 FlexRay 프로토콜은 다음을 제공합니다:

- Global time base 를 통한 오류 안정(fault-tolerant) 클럭 동기화 지원
- Collision free 버스 접속
- 보증된 메시지 대기 시간
- 식별자를(identifiers) 통한 메시지 중심 주소지정
- 단일 또는 이중 채널 구현 지원을 통한 확장 시스템 오류 안정

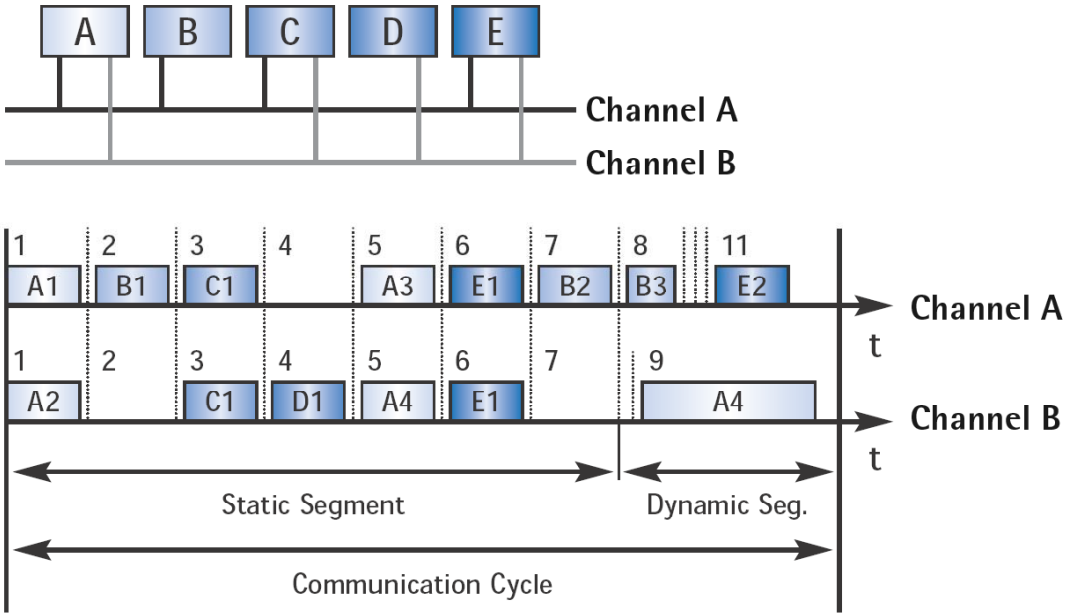
향후 독립적인 Bus Guardian 이 추가의 오류 견제를 제공할 것입니다 (옵션)

FlexRay 프로토콜 서비스

다음 유형의 서비스들이 제공됩니다 (프로토콜 엔진에 의해 구현):

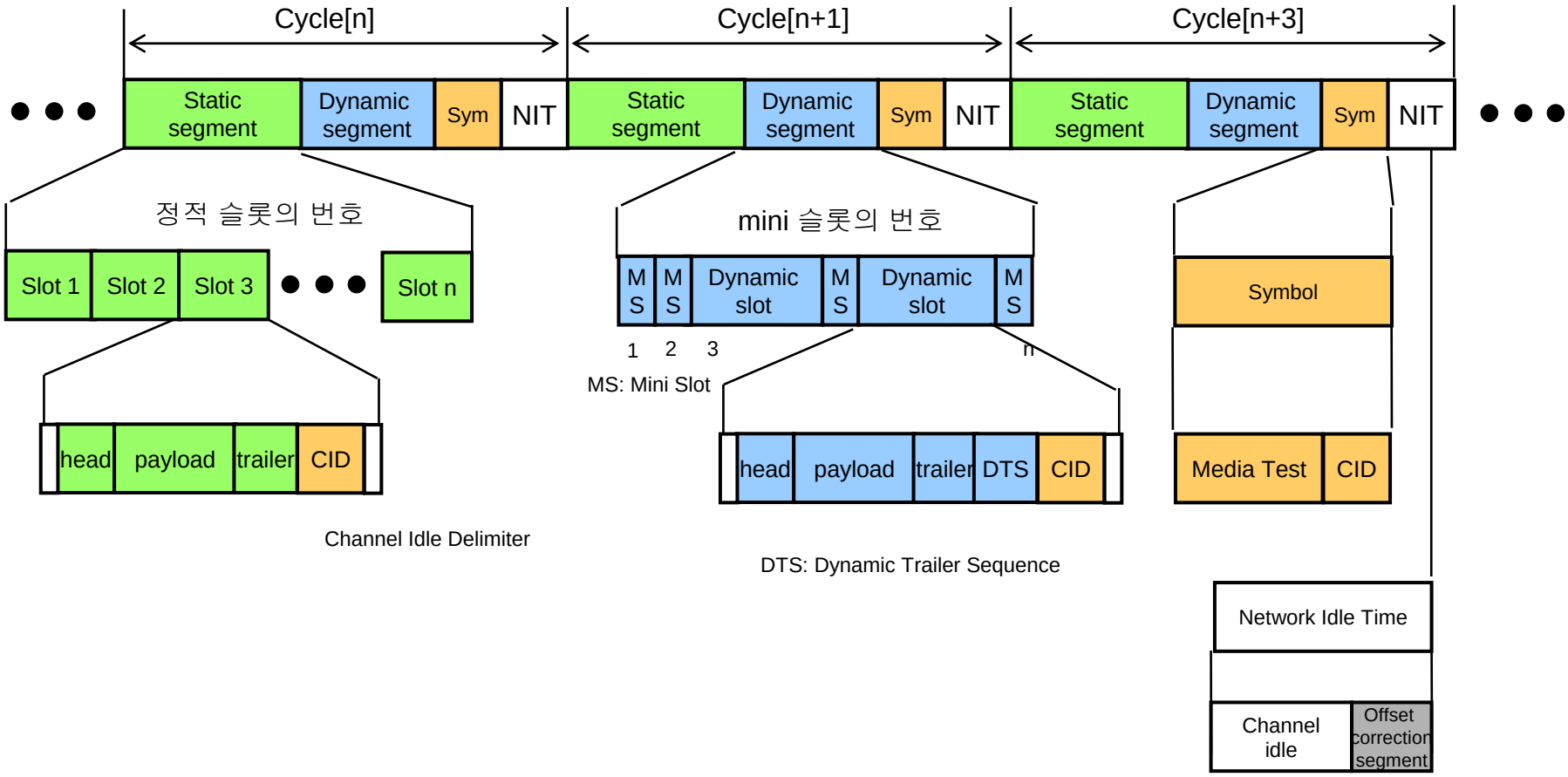
- Media Access — 정적 메시지의 경우 TDMA ,
  동적 메시지의 경우 Mini 슬롯
- Synchronization — 모든 노드들의 클럭들은 정의된 정밀함대로 동기화됩니다
- Cluster Start-Up — Fault tolerant: 임의의 "cold start node" 가 start-up을 개시, 다른 노드들이 뒤따릅니다
- Frame processing — 구문 오류, 내용 오류, 경계 위반,
  전송 충돌의 검출
- Symbol — 시동(start-up) 동안의 충돌 회피와
  매체 테스트를 위한 심볼
- Wake-Up — 통신 채널 경유:
  power down 에서 power up 으로 변환 요청

FlexRay 통신 주기

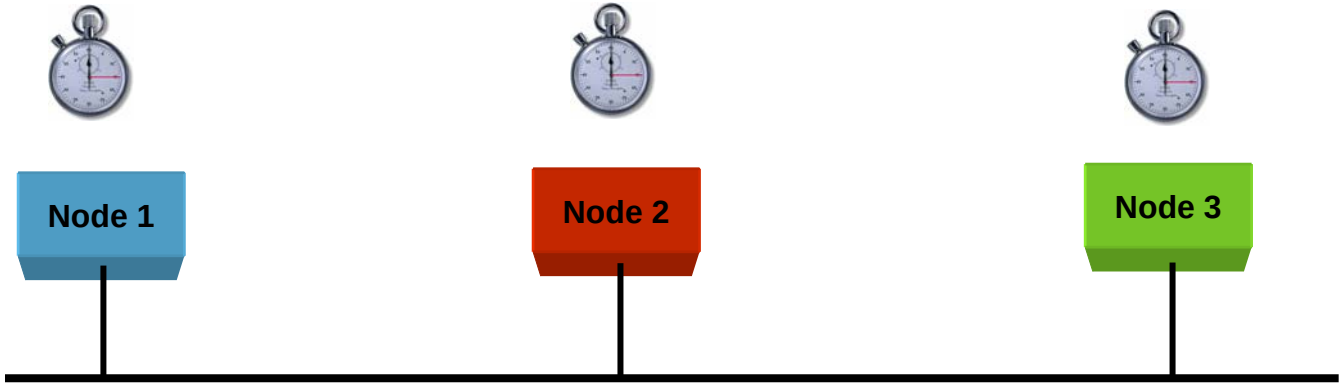


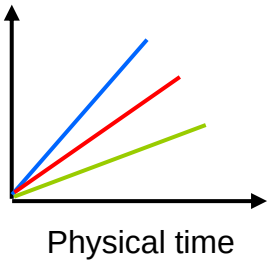
- 노드 A, B, C, D, E 가 있는 FlexRay cluster
- 채널 A 와 B 의 노드 A, C, E 는 sync 프레임과 data 프레임을 생성합니다
- 노드 B 와 D 는 data 프레임만을 생성합니다
- 정적 세그먼트에 배치된 대역폭은 항상 소비됩니다
- 동적 세그먼트에 배치된 대역폭은 필요한 경우만 소비됩니다

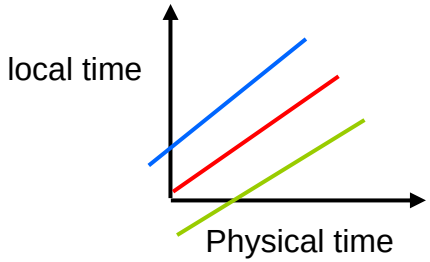
FlexRay TDMA 구조



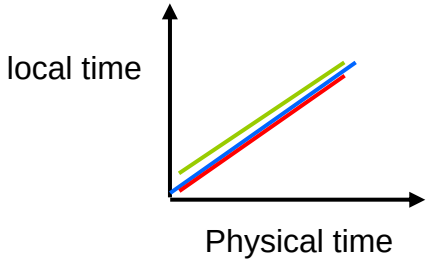
FlexRay 동기화 원칙 [1/2]





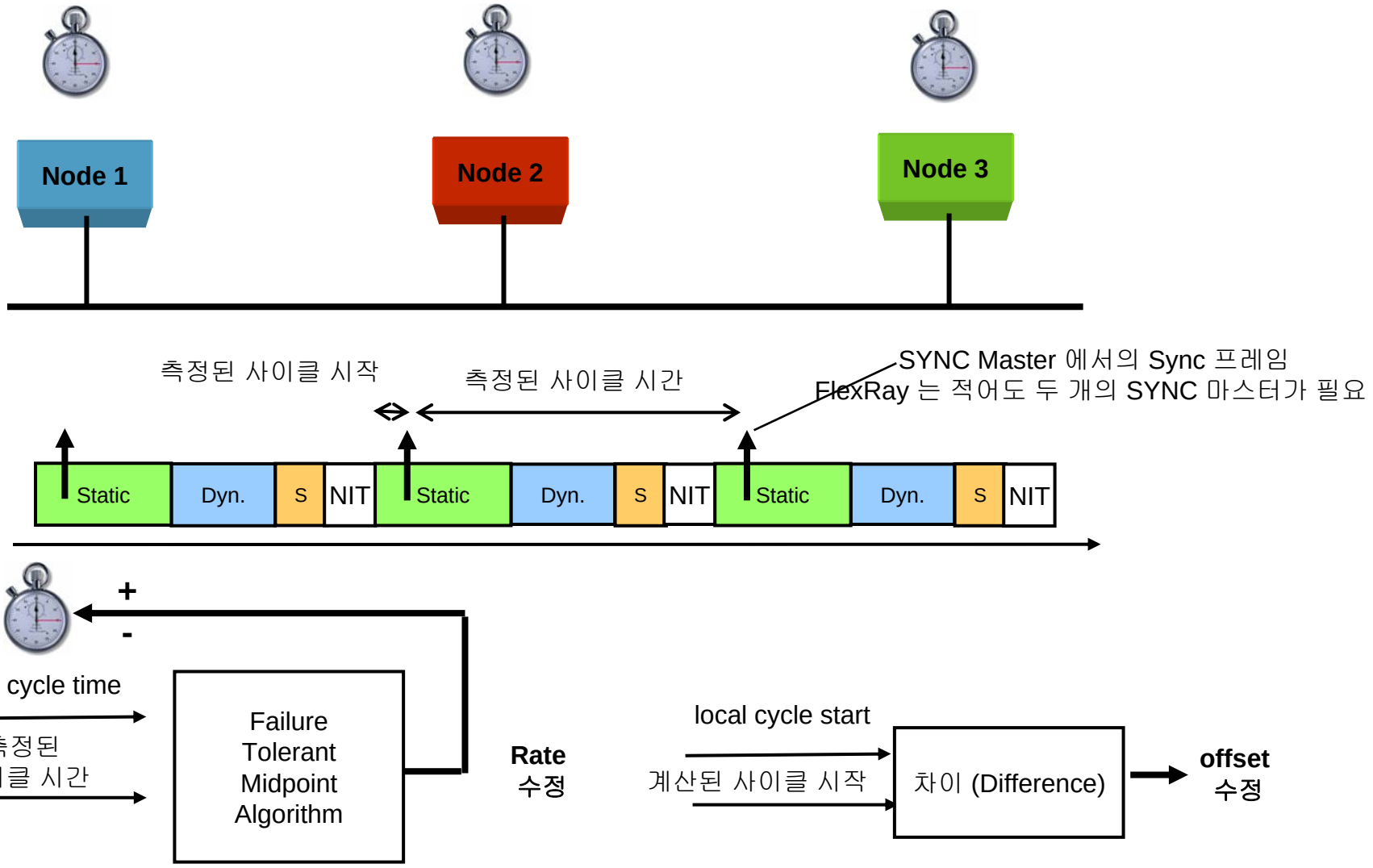


Rate 수정

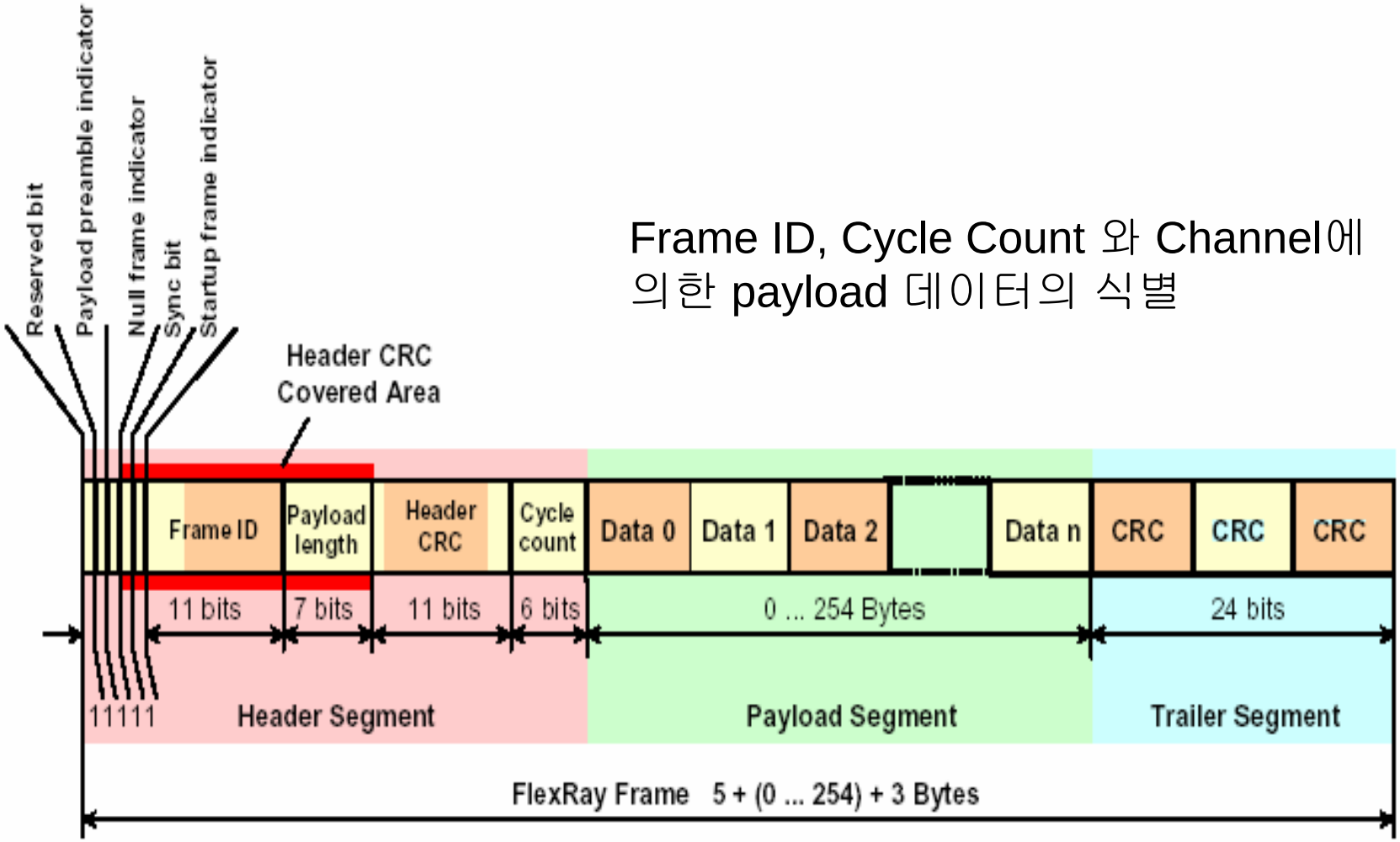


Offset 수정

FlexRay 동기화 원칙 [2/2]

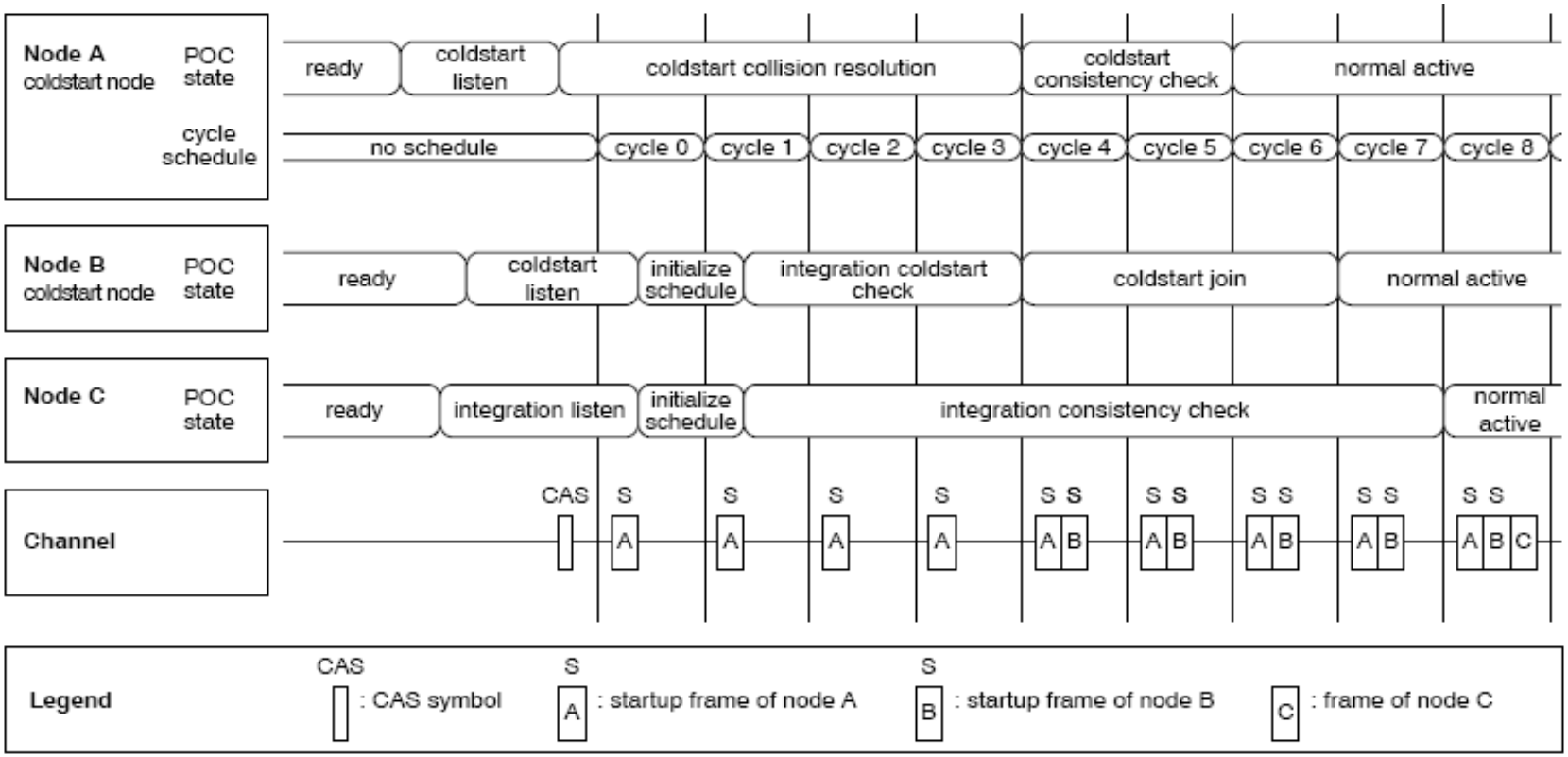


FlexRay 프로토콜 프레임 형식



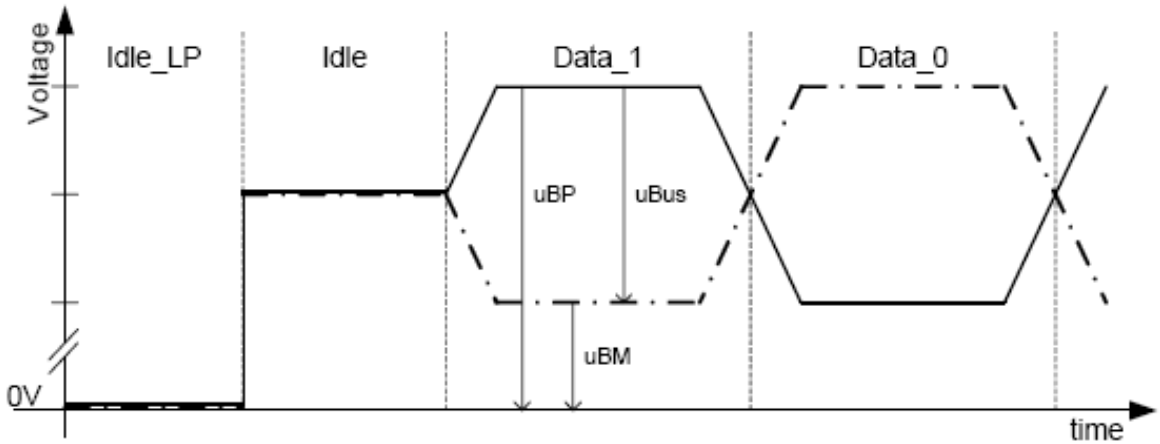
Frame ID, Cycle Count 와 Channel에 의한 payload 데이터의 식별

FlexRay Cluster Start-Up



-, 0, 1, 2, 3:	노드 A 가 CAS 심볼과 동기 프레임을 전송합니다
0, 1, 2, 3:	노드 B 가 노드 A에 동기화, 2 사이클 동안 sync 검사
4, 5:	노드 A 와 B 가 sync 프레임을 전송합니다
6, 7, 8, ..	"normal active" 상태의 노드 A
7, 8, ..	"normal active" 상태의 노드 B

FlexRay 전기적 신호방법



Idle_LP: 저 전력: BP와 BM 모두를 GND로 Biasing

Idle: BP와 BM 모두를 특정 전압 레벨로 Biasing

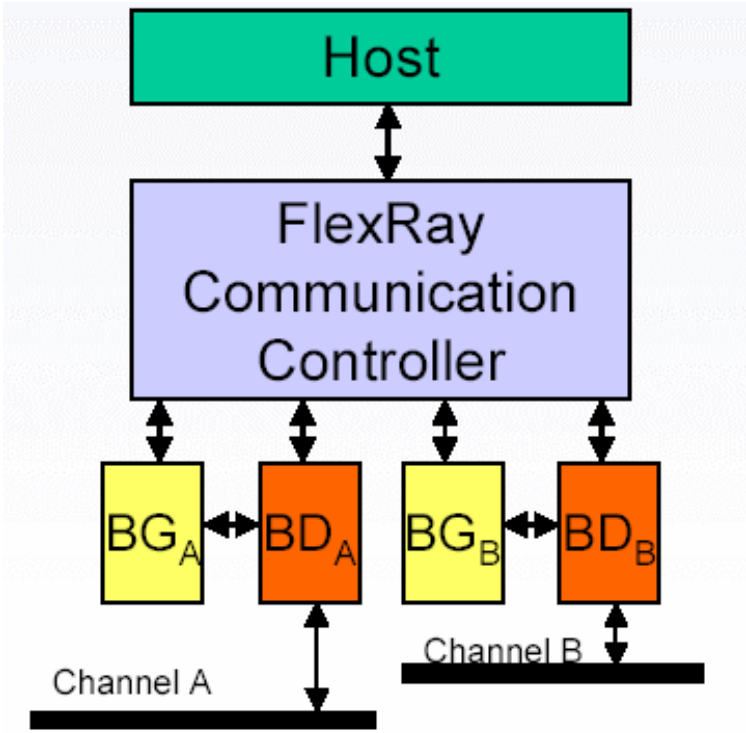
Data_1: 버스 드라이버가 양의 차이 전압 BM – BP로 끌어당김

Data_0: 버스 드라이버가 음의 차이 전압 BM – BP로 끌어당김

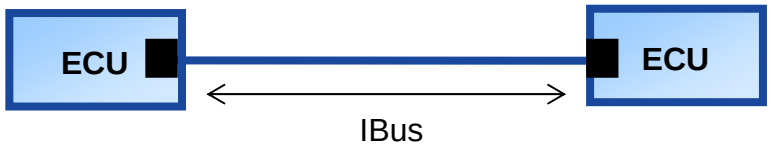
BM – BP: 최소 송신기 출력 ± 1200 mV
최소 수신기 입력 ± 800 mV

FlexRay 노드의 구조

FlexRay 노드는 호스트 컨트롤러 (e.g. PowerPC555, STAR12), 통신 컨트롤러 (e.g. MFR4200A), 버스 드라이버 (e.g. TJA1080N1C) 그리고 Bus Guardian (개발중) 으로 구성됩니다

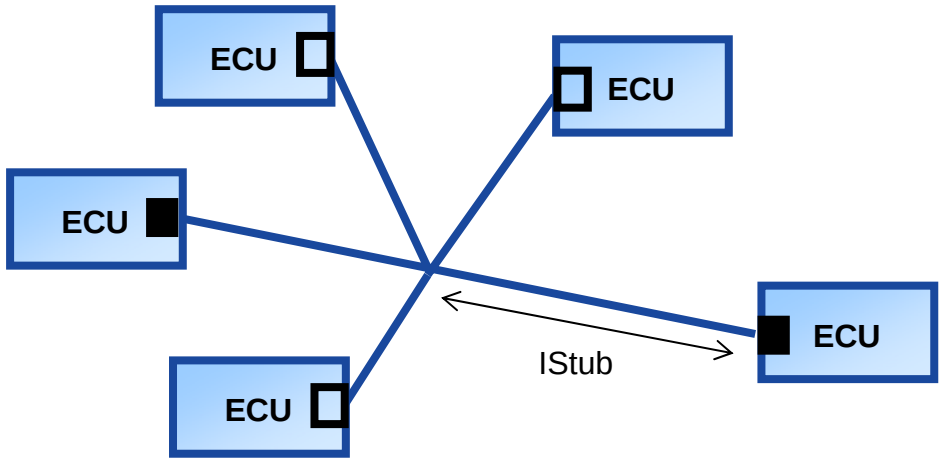


FlexRay Topologies [1/2]



Point-to-Point connection

Name	Description	Values
Ibus	Bus Length	Max 24m



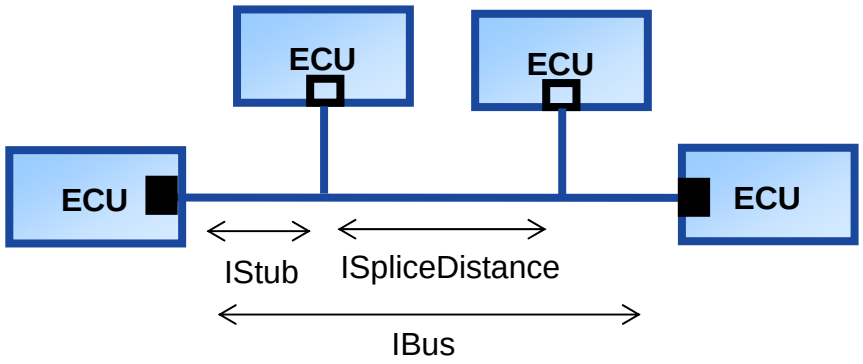
Passive Star

Name	Description	Values
Istbn+Istubm	Cable length between any two ECU	Max 24m
N Stubs	Number of stubs	3..22
Istub	Max Stub length	Depends on N and max length between any two ECUs

ECU terminated (Rt)

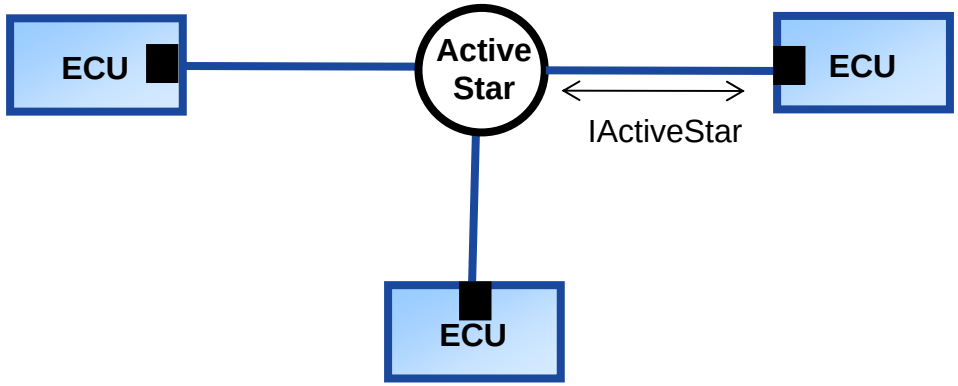
ECU not terminated

FlexRay Topologies [2/2]



Linear passive Bus

Name	Description	Values
iBus	Bus Length	Max 24m
N Stub	Number of stubs	4..22
N Splice	Number of Splices	Min 2

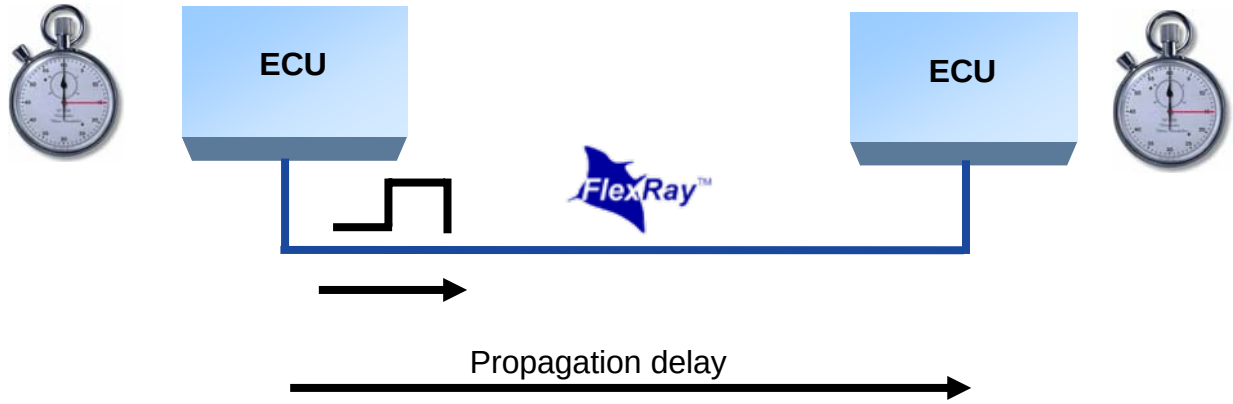


Active Star

Name	Description	Values
IActiveStar	Length of a branch from ECU N to star	Max 24m
N active Branches	Number of branches at an active star	Min 2

기존의 FlexRay 네트워크 어디에 측정 장치를 연결하는가 [1/2]

동기적 통신의 원칙은 정밀한 global clock 입니다.



global time 의 위상은 네트워크에서의 그들 위치에 따라 네트워크 요소의 전파 지연의 구성에 의해 계산됩니다.

추가적인 FlexRay 노드를 (측정을 위해) 배선한다는 것은 전파 지연을 변경한다는 것을 의미합니다

기존의 **FlexRay** 네트워크 어디에 측정 장치를 연결하는가 [2/2]

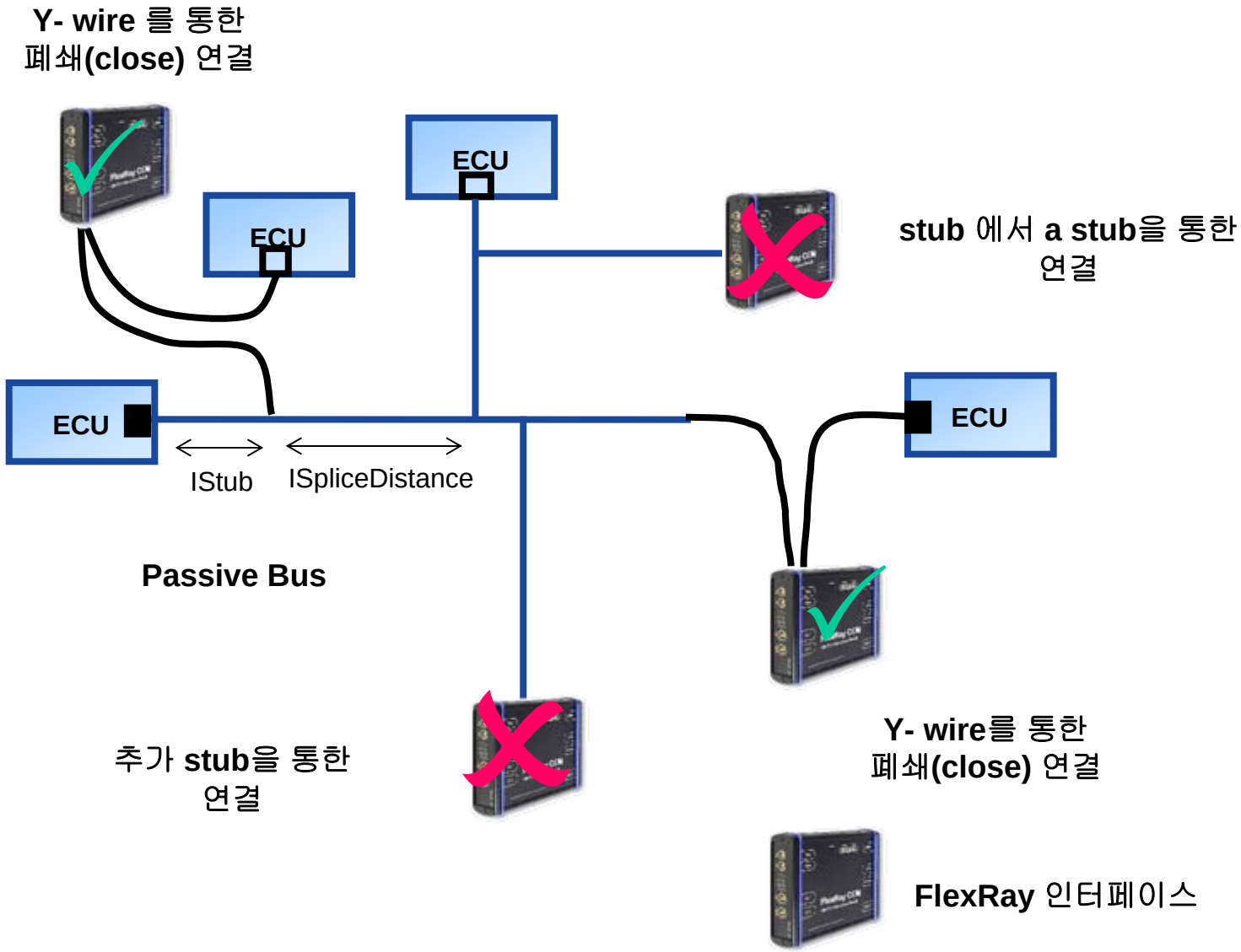
가장 중요한 네트워크 파라매터

- gdMaxPropagationDelay = f (Line_length, Max_nStar Path, $d_{pdTransmitter}$,d_{pdline}, d_{pdStar})

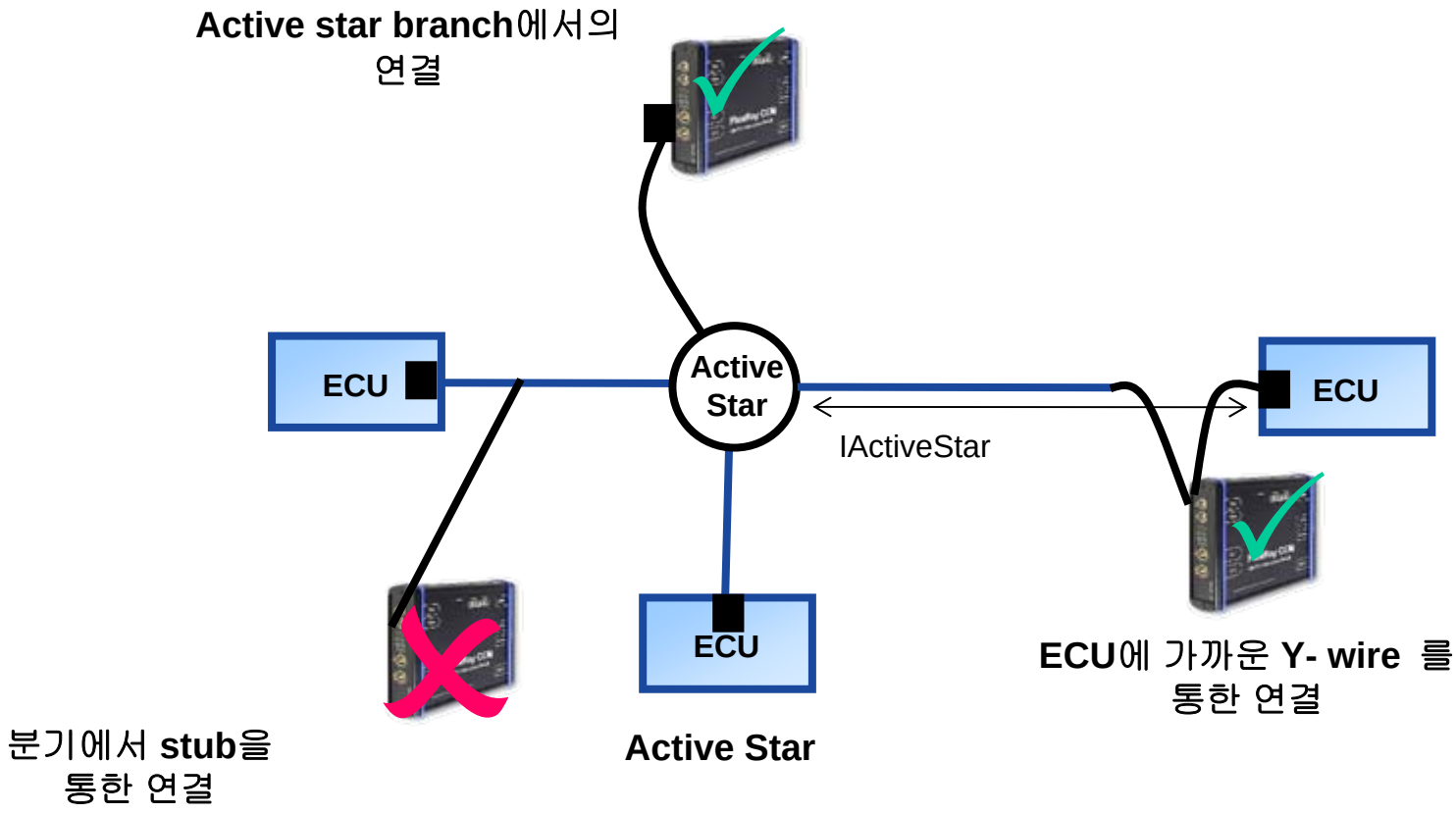
Affected constrains: 3, 11, 12, 13, 15, 16, 23, 33, 34, 37

측정 장치에 의해 야기된 변화는 가능한 작아야만 합니다.
그렇지 않으면 전체 네트워크의 재구성이 필수적입니다

선형 버스 위상(Linear Bus Topology)에 연결하기

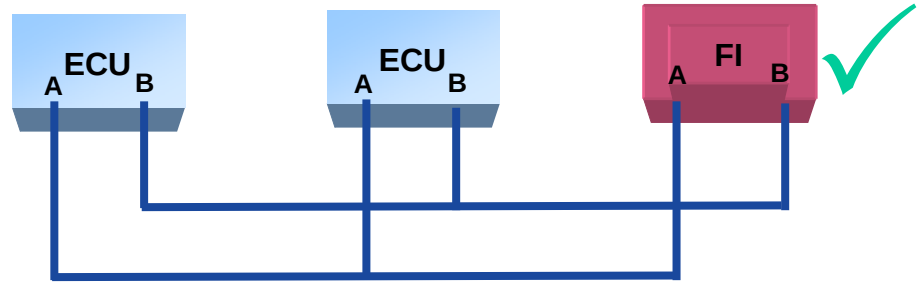


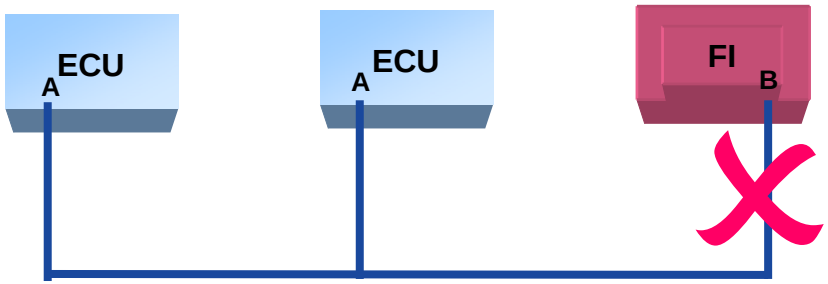
Active Star 위상에 연결하기



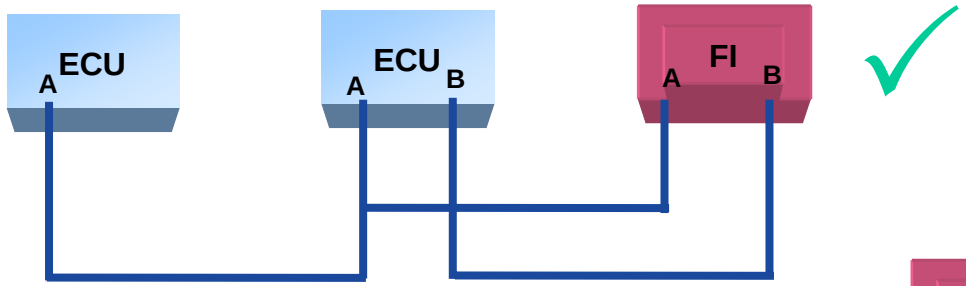
FlexRay 인터페이스

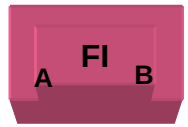
정확한 채널 연결하기





채널은 Header CRC에서 코드가 됩니다
잘못 연결된 채널들은 수신 오류를 야기합니다





FlexRay 인터페이스

해야 할 것과 하지 말아야 할 것들

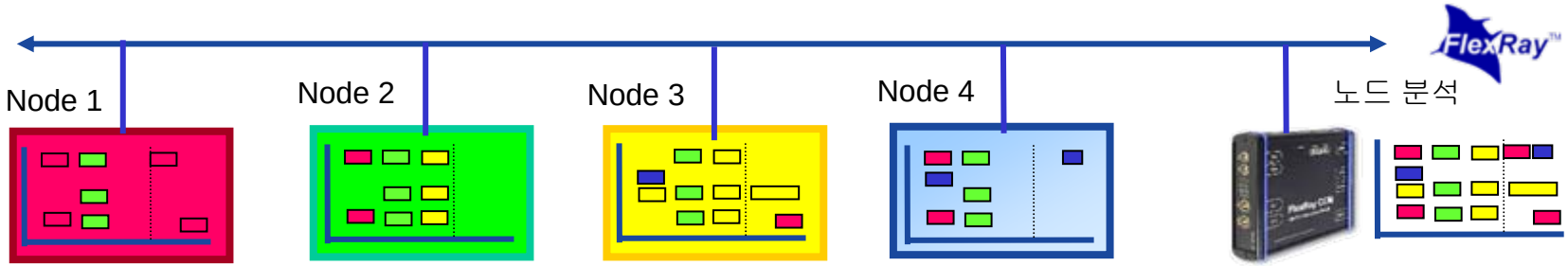
- ECU 에서 가능한 가깝게 연결
- stubs에서 stubs을 갖지 말 것
- 추가적인 splices / branches 회피
- 가능한 짧은 Y-라인들을 사용
  - FlexRay 측정 케이블들을 사용: △ Z < 50 Ohm/cm
  - shield interruption 허용 금지
- 정확한 채널들을 연결: channel A to A, channel B to B
- active star 위상에 관해서는 terminated 라인을 사용
- 물리 네트워크에서의 모든 변경들은 안전 폭을 감소시킵니다

⇒ 측정 장치는 네트워크에 가깝게 연결할 수 있어야 합니다.

Configuration

FlexRay의 측정 디바이스 구성하기

구성(Configuration)의 원칙[1/2]



Node 1
전송
Message A
Message D
Message E

수신
Message C

Node 2
전송
Message C

수신
Message A
Message F
Message G

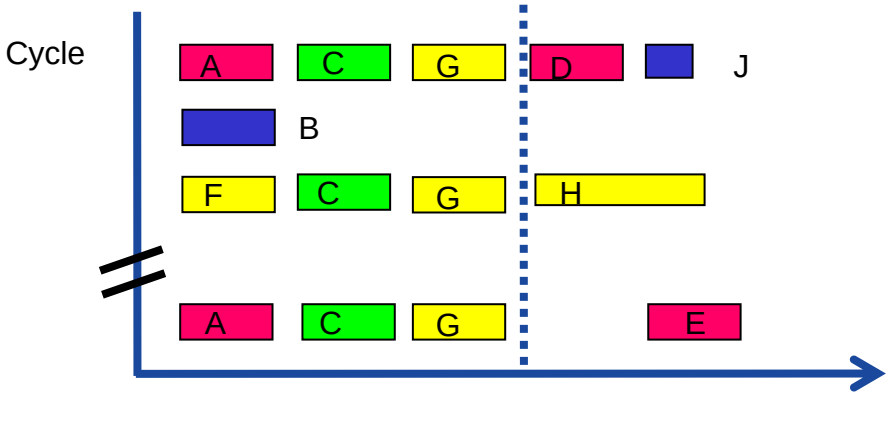
Node 3
전송
Message F
Message G
Message H

수신
Message B
Message C
Message E

Node 4
전송
Message B
Message J

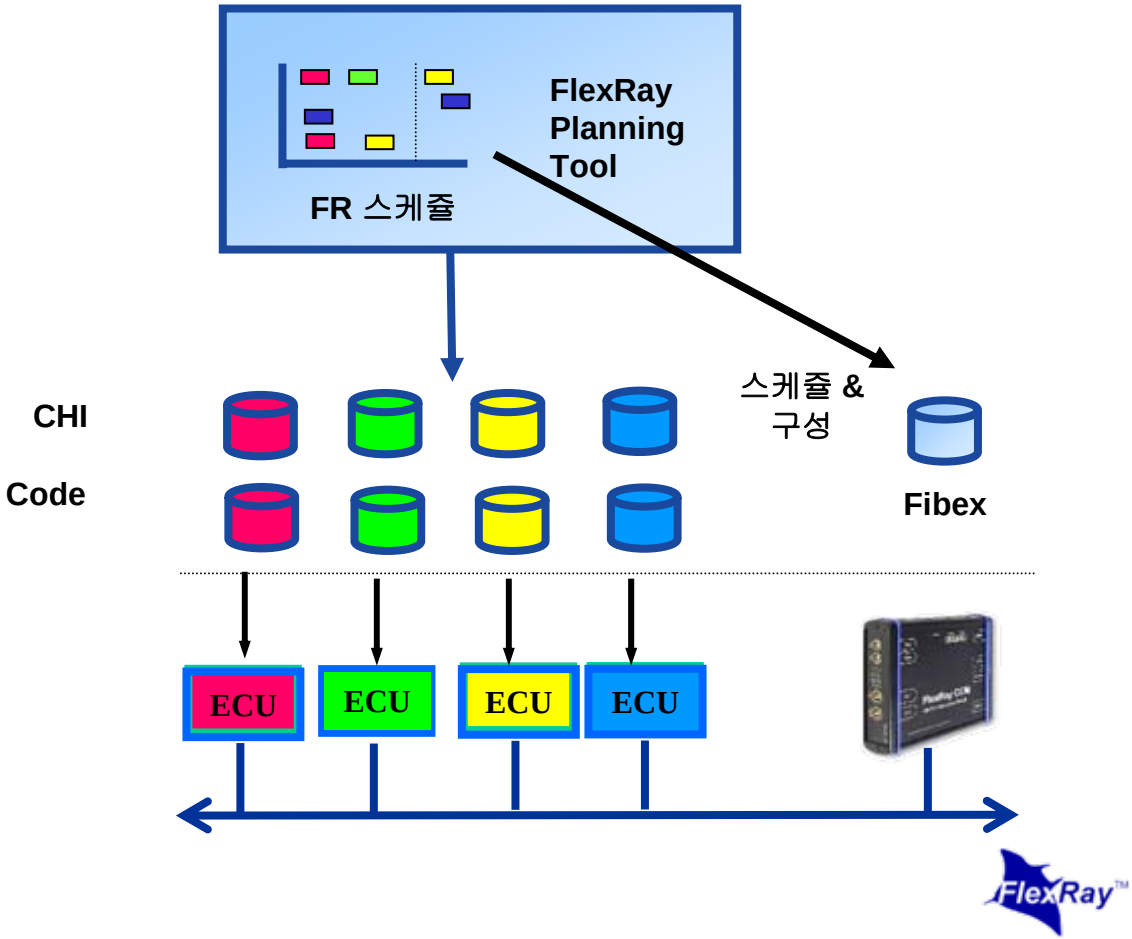
수신
Message A
Message C

노드 분석
수신
모든 메시지들

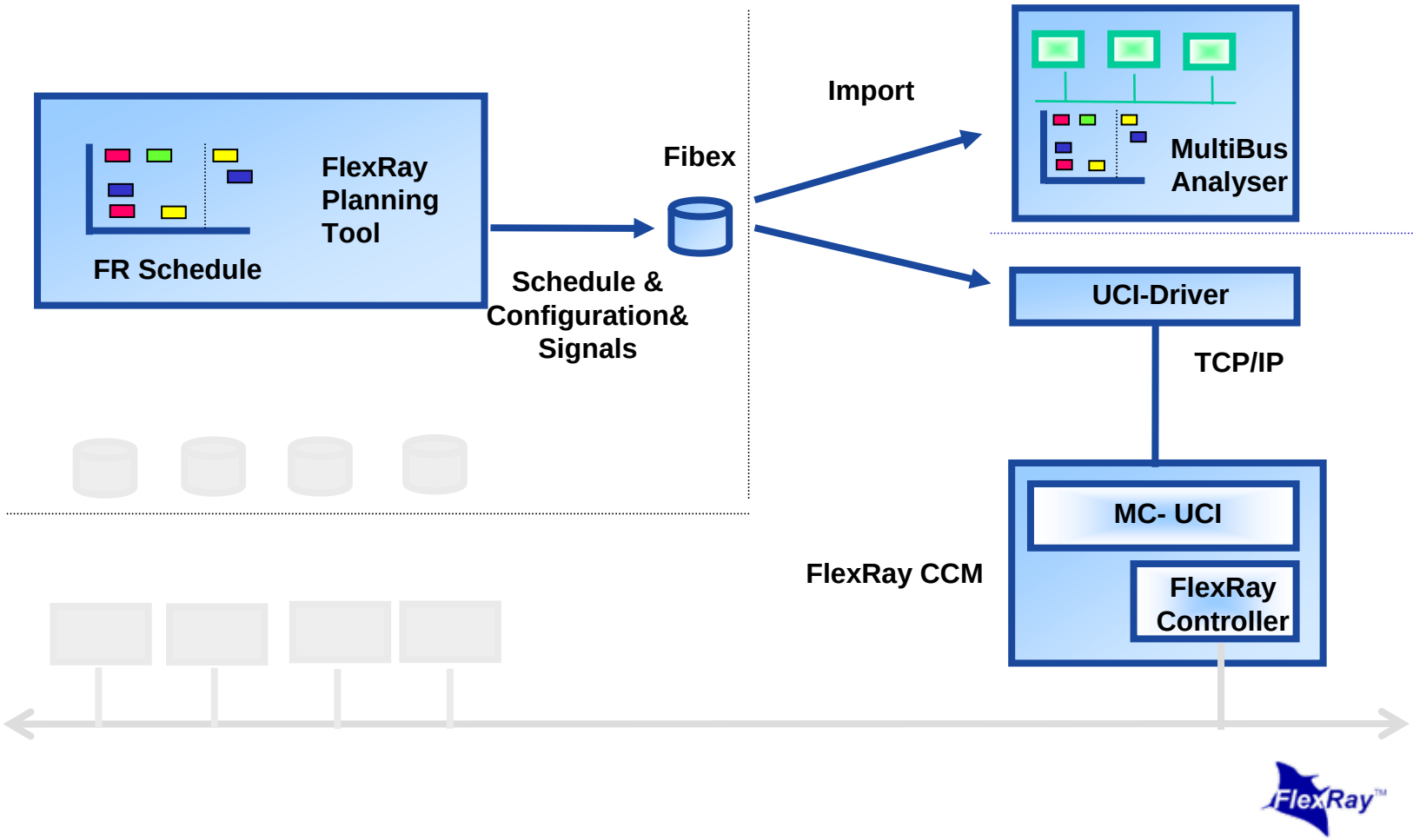


FlexRay 통신 스케쥴

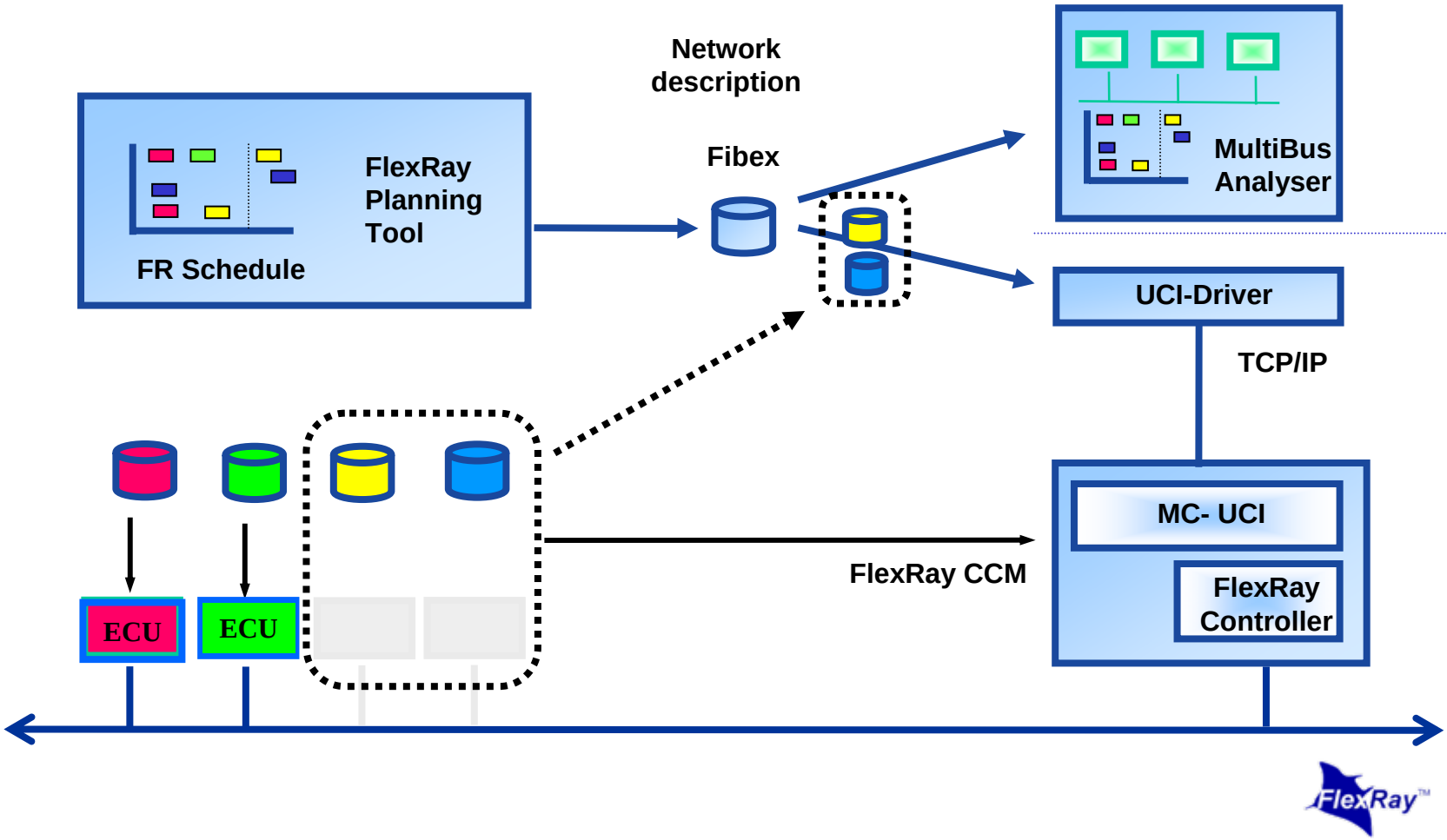
구성(Configuration)의 원칙 [2/2]



분석 노드(Analyzing Node)의 구성



시뮬레이션 노드의 구성



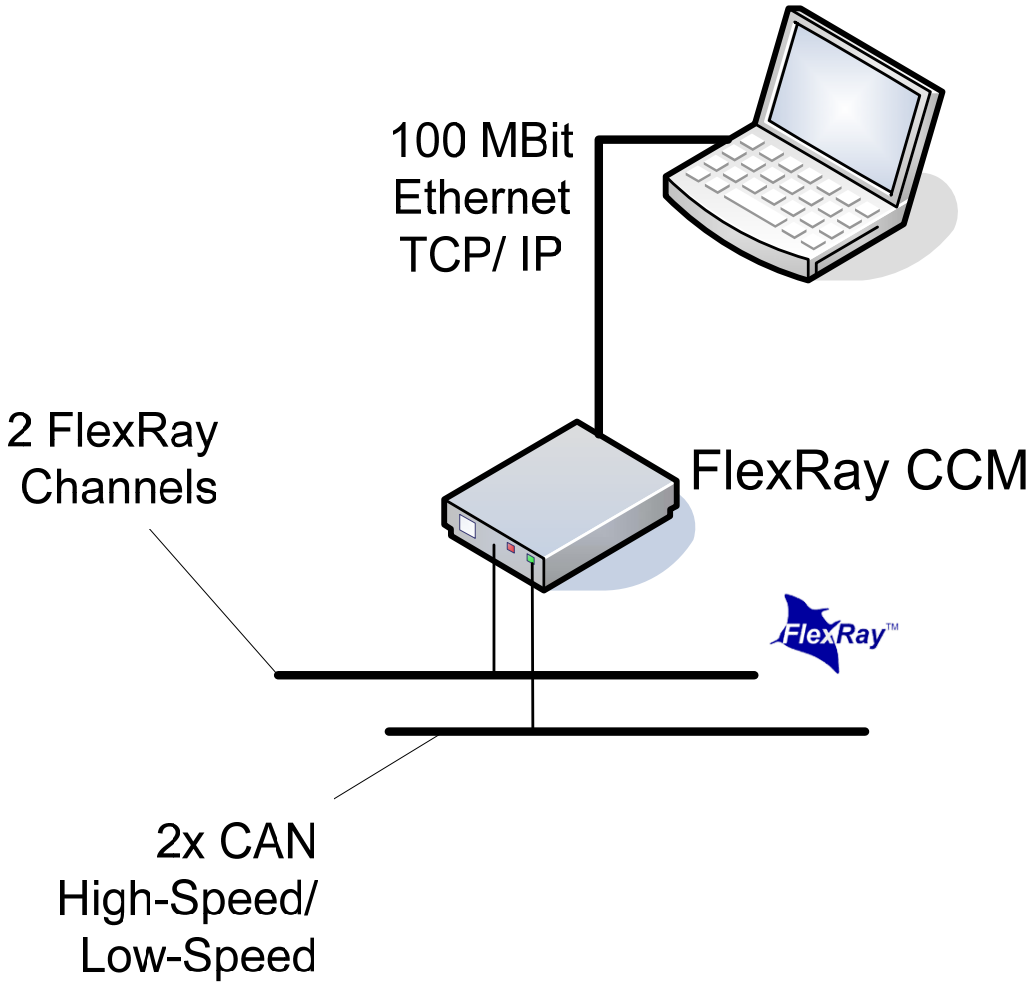
Analyzing

FlexRay 네트워크의 분석을 위한 필수 요건

HW 플랫폼 분석
FlexRay의 요건

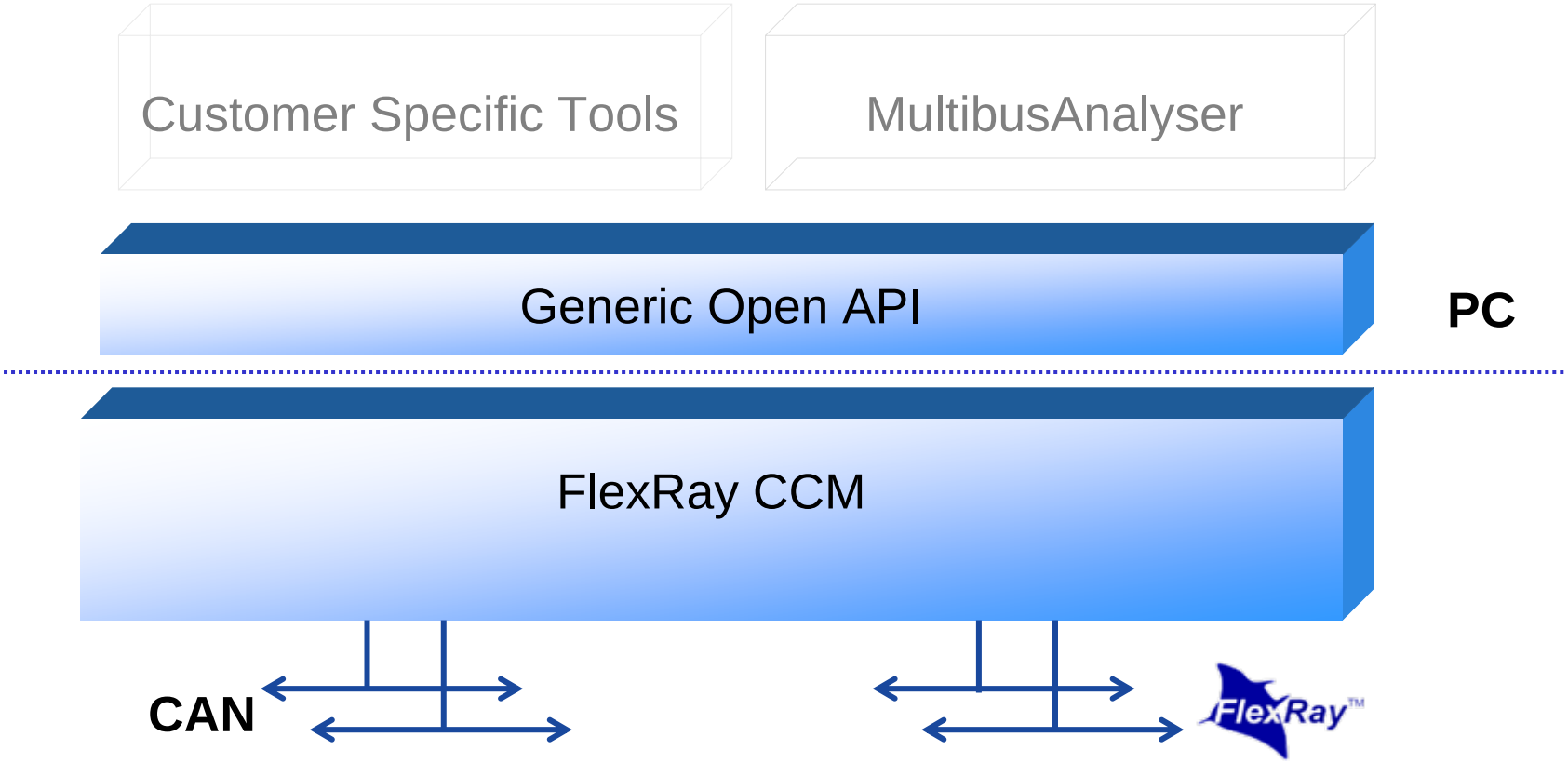
- 2 x 10 MBit/s FlexRay, 100% Busload, 모든 메시지들
- 서로 다른 FlexRay 프로토콜 버전들 지원
- ISO 11898-2 와 ISO 11898-3에 따른 2 x 1 MBit/s CAN
- FlexRay 와 CAN 네트워크의 시간 동기 분석
- 한 개 PC로 제어되는 하나 또는 그 이상의 디바이스들
- 높은 대역폭을 지닌 PC 인터페이스
- 자동차 사용을 위한 설계
- 비동기화된 FlexRay 네트워크의 분석
  (비동기 모드)
- 개방형 일반 Application Programming Interface

HW 플랫폼을 분석하는 FlexRay

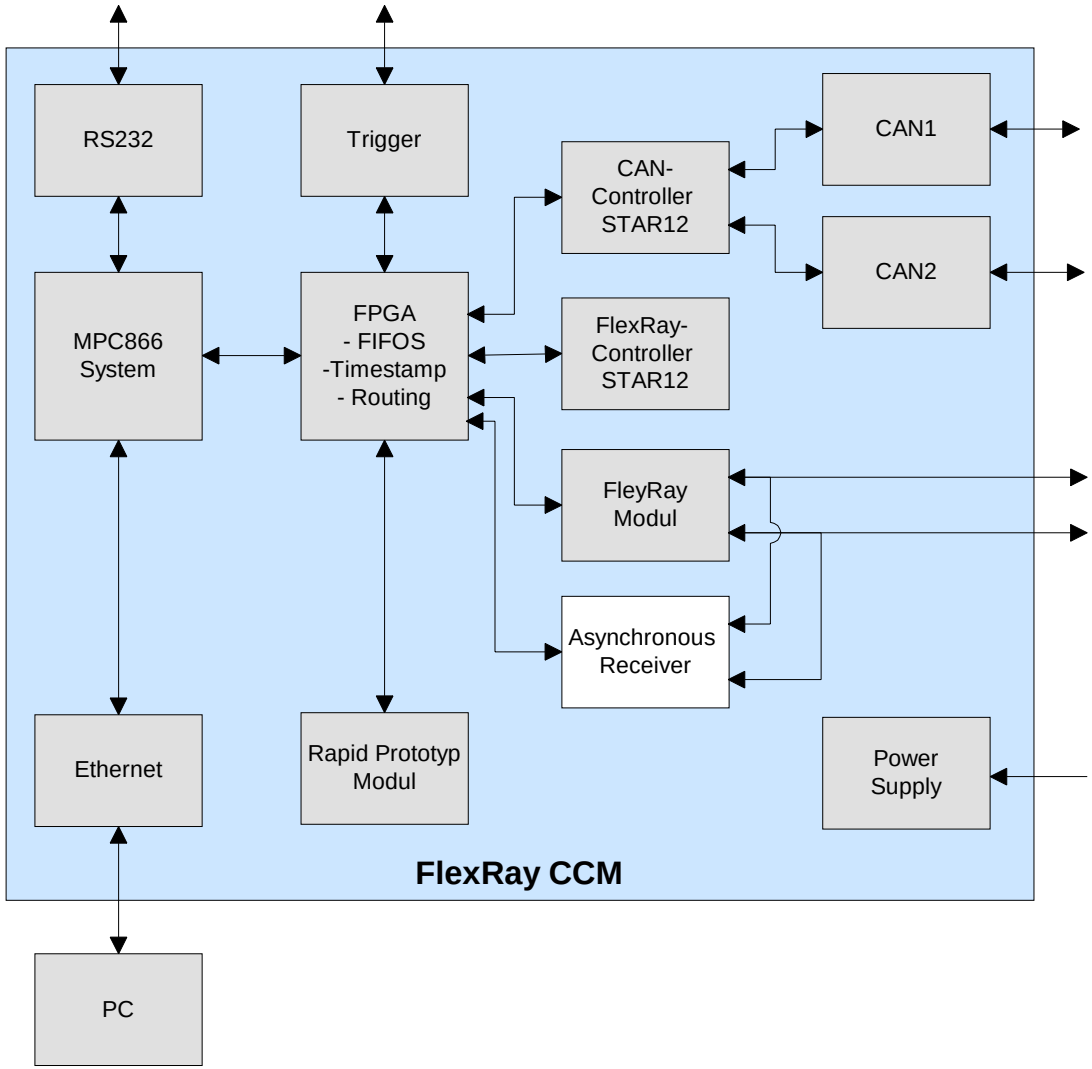


FlexRay CCM 기본 구조

FlexRay와 **CAN**을 위한 **PC** 인터페이스



FlexRay CCM: 하드웨어 구조



FlexRay CCM: API

일반적인 **Application Programming Interface**로 다음이 가능합니다:

- 보드 초기화
- FlexRay 와 CAN 컨트롤러 레지스터의 초기화
- FlexRay 와 CAN 컨트롤러를 제어. 예. start, stop
- FlexRay 와 CAN 메시지를 수신
- FlexRay 와 CAN 메시지를 전송
- 트리거(trigger) 조건 정의

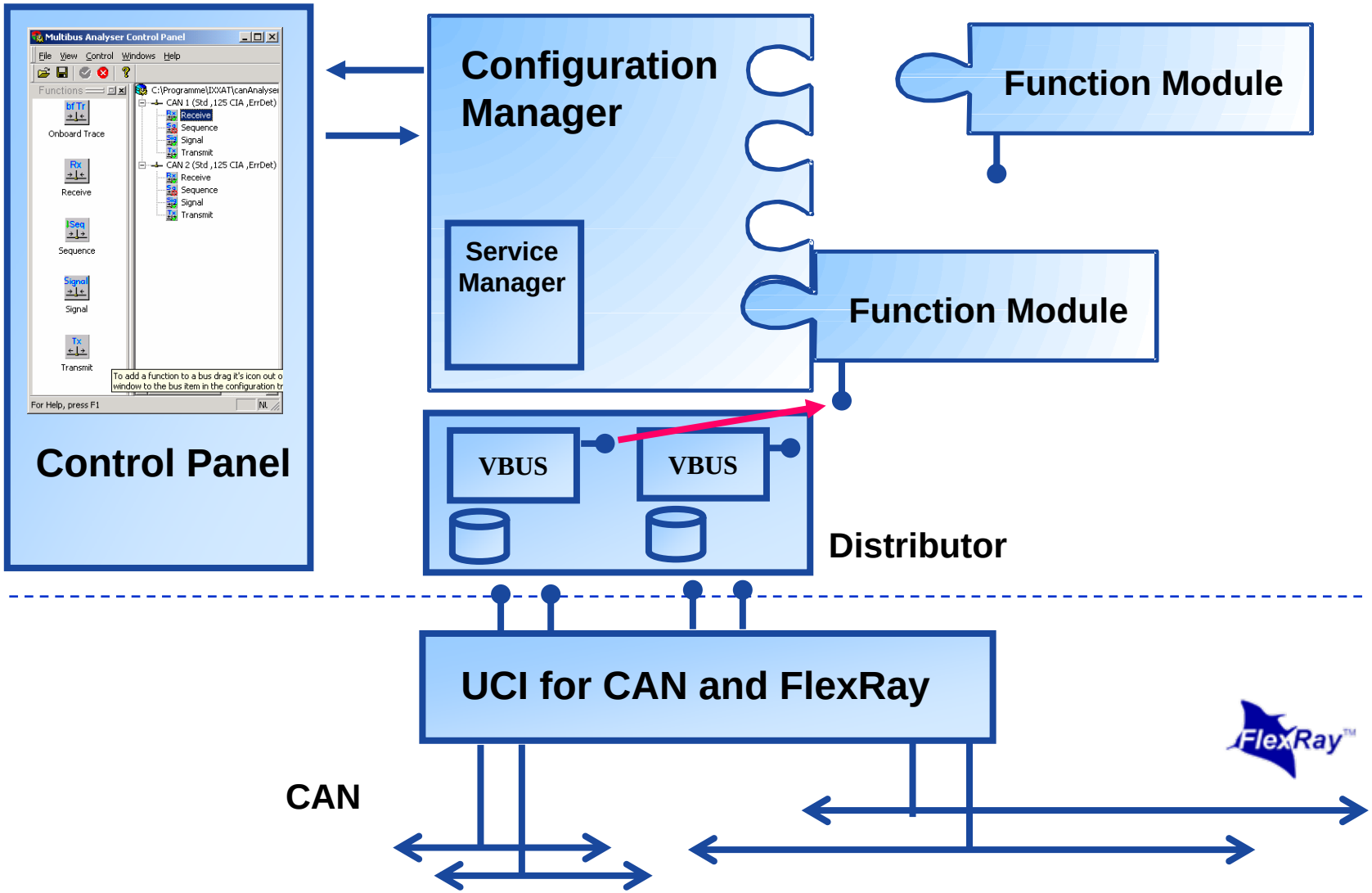
분석 소프트웨어의 요건

- 서로 다른 네트워크(FlexRay, CAN, ...)의 분석
- 서로 다른 네트워크에서의 시간-동기적 분석
- 메시지와 신호들의 다양하게 배치 가능한 디스플레이
- Monitoring, Tracing 과 Stimulation 기능들
- 비동기부터 “normal active”에 이르는 네트워크의 연속적 분석
- 사용자-특정 소프트웨어 모듈의 간편한 구현을 위한 개방형 API
- 스크립트 기본 모듈의 구현을 위한 Scripting Host 로 노드 또는 특정 사용자 인터페이스의 시뮬레이션이 가능
  (C#, support of panels)
- Fibex 와 CANdB data descriptions의 받아들임(import)

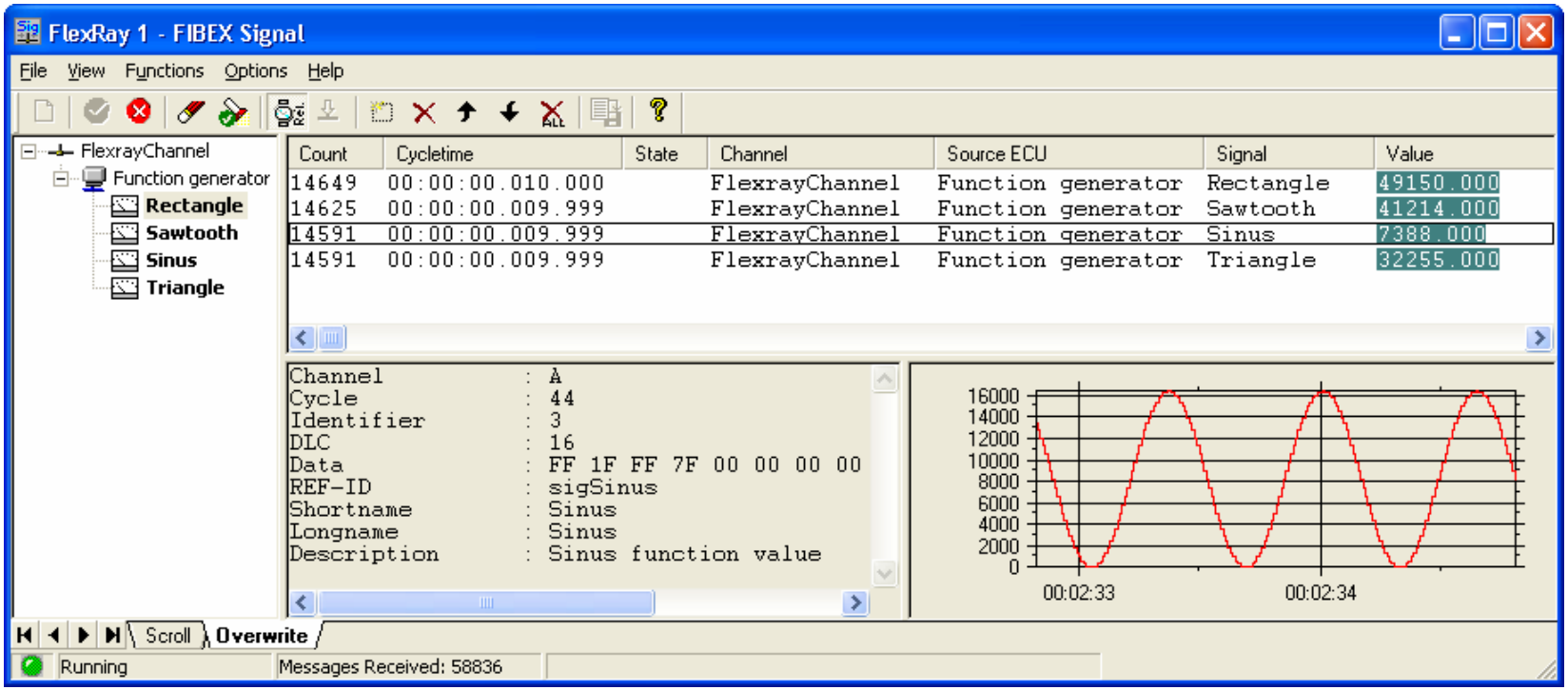
MultibusAnalyser: 모듈

Receive	메시지들의 연대기순 또는 메시지 중심 디스플레이. 시간 측정, 메시지 계수(counting), 주기 시간 모니터링
Signal	데이터베이스의 signal description을 바탕으로 물리 신호들의 연대기순 또는 신호 중심 디스플레이 (data interpretation) FIBEX, CANdb, 등의 가져오기
Graphic	물리 신호들의 그래픽 표시 (signal recorder)
Trace	PC의 하드 드라이브 안으로 메시지 추적. Trace View, Trace footprint
Statistic	bus load, 메시지와 오류에 대한 통계
Transmit	단일 또는 주기적 메시지들의 편집과 전송
	... 다양한 추가 CAN 모듈들

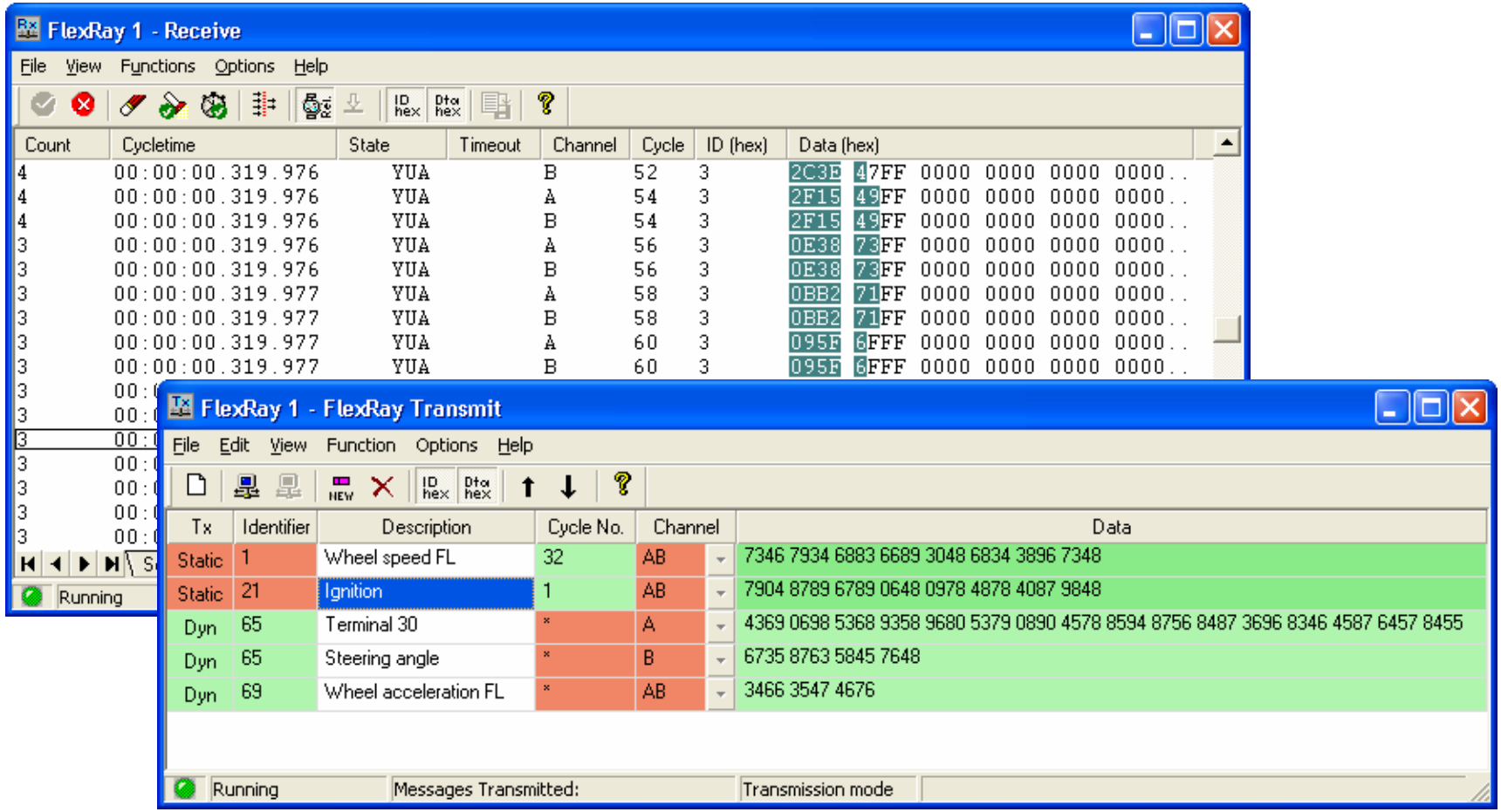
MultibusAnalyser: 구조



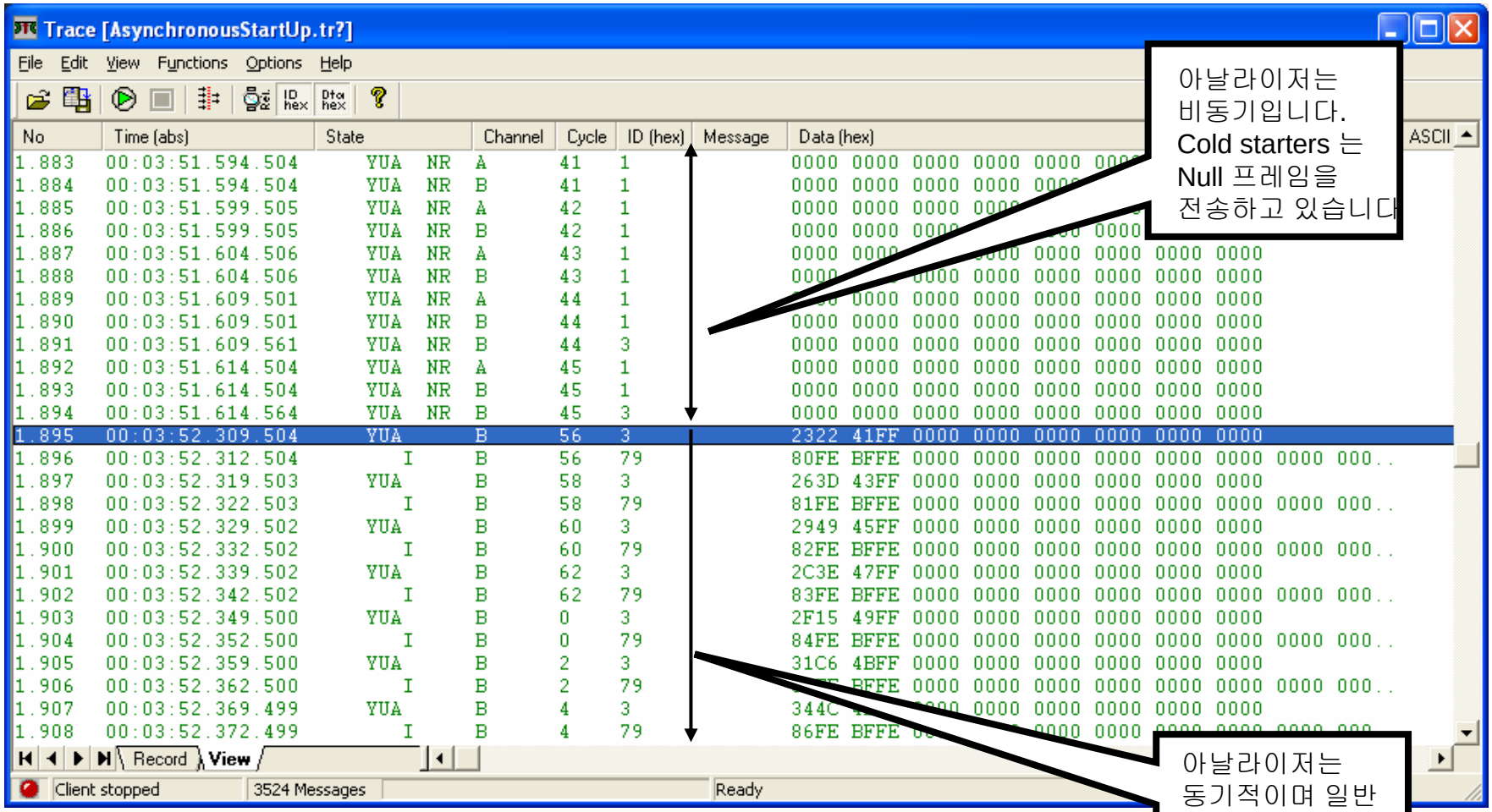
MultibusAnalyser: 단일 모듈



MultibusAnalyser:
수신과 전송 모듈



MultibusAnalyser: Analyzing Start Up Example



Data Description Standard FIBEX

FIBEX ... FieldBus Exchange format

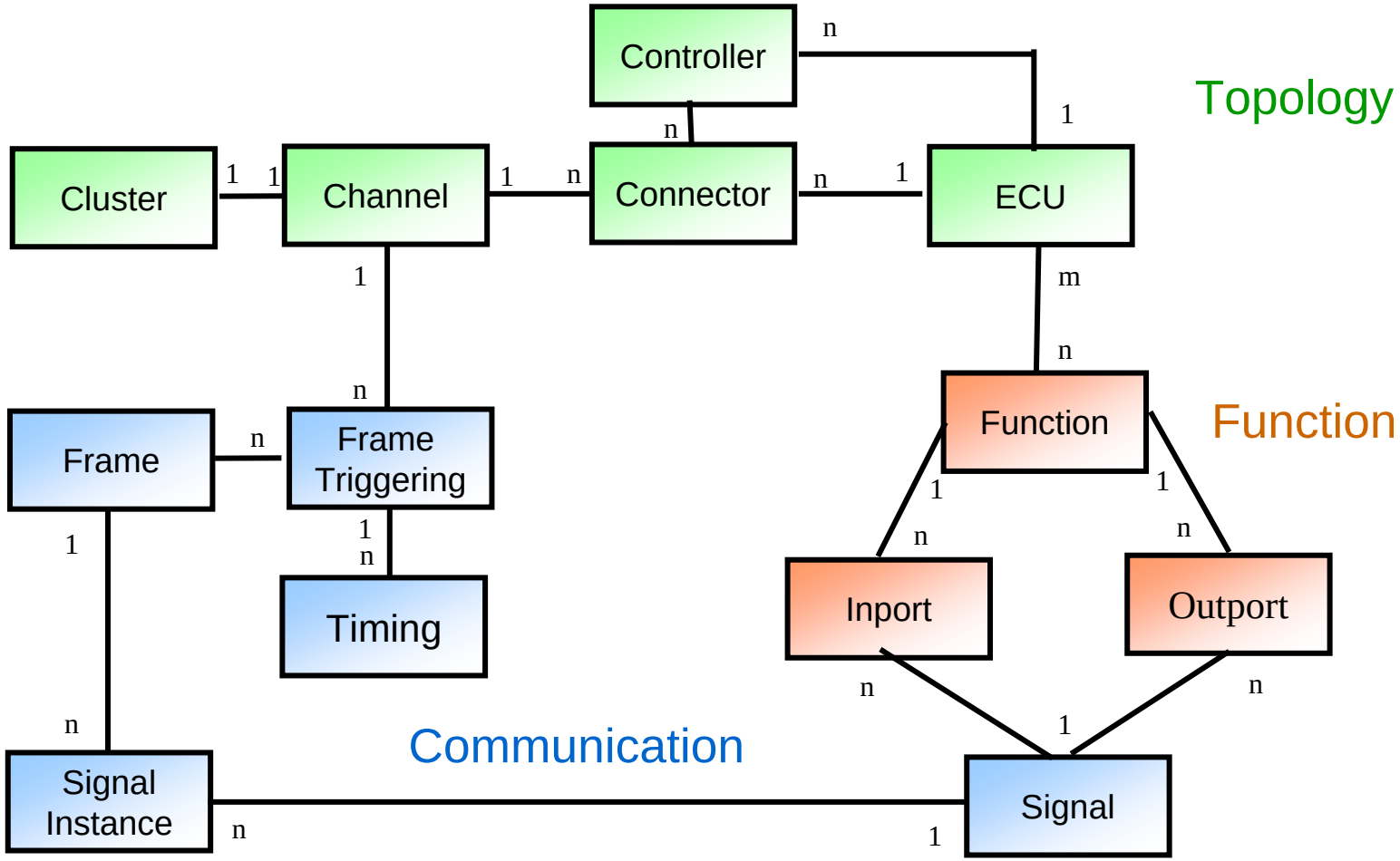
자동차 제조업체들과 부품 제조업체 그룹들에 의해 개발, ASAM에 의해 표준화 신청. 다음 정의에 관한 XML-을 바탕으로 하는 제조자-독립 데이터 교환 표준

- 기능적 네트워크 (functions 과 signals)
- 시스템 위상
- 통신 속성 (프레임, 타이밍 매개변수 등.)

CAN, FlexRay, MOST 같은 메시지 지향 통신 시스템을 바탕으로 하는 자동차 시스템

향후 개발, 분석, 그리고 서비스 툴에서의 표준

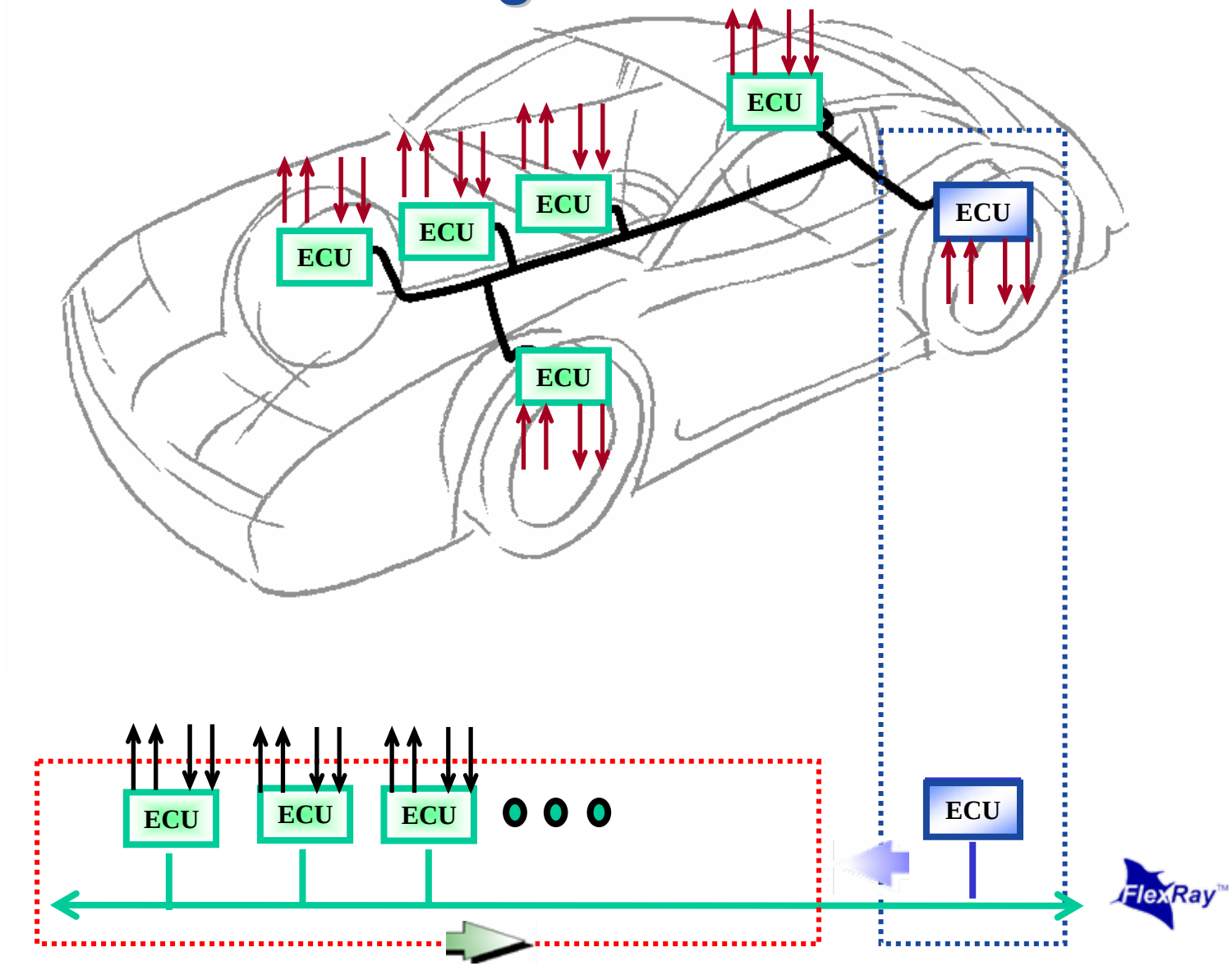
FIBEX Example ER-Diagram



Rest Bus의 시뮬레이션

FlexRay의 요건과 해법

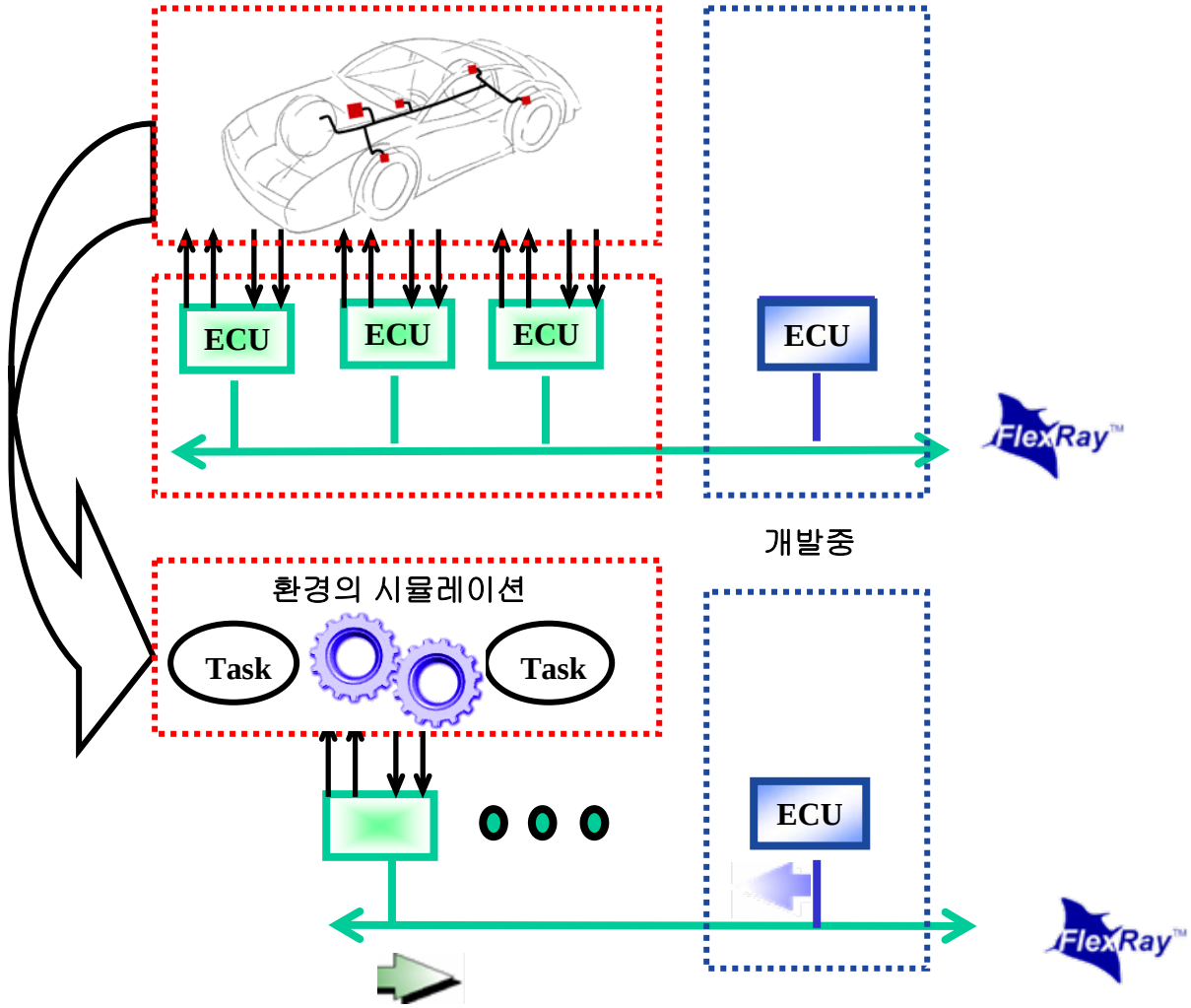
Rest Bus = Remaining Bus



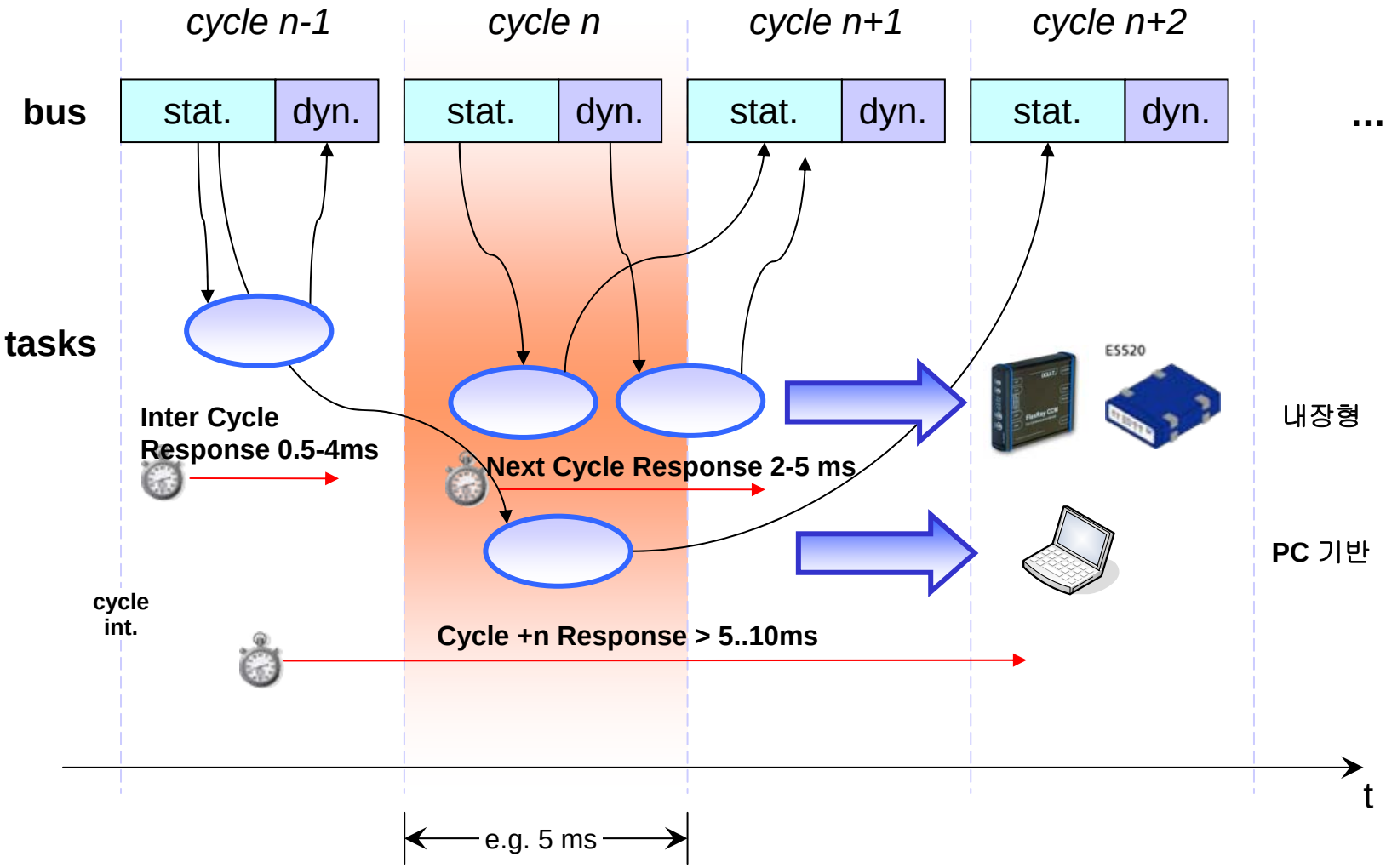
파란색 **ECU**의 **Rest Bus**

Rest Bus의 시뮬레이션 [1/3]

- 서브시스템의 메시지 제공
- Startup-Nodes 제공: 최소 두 개의 Cold-starters가 필요합니다
- 메시지의 응답
- 소프트웨어에서 기술 처리와 I/O의 시뮬레이션



Rest Bus의 시뮬레이션 [2/3]



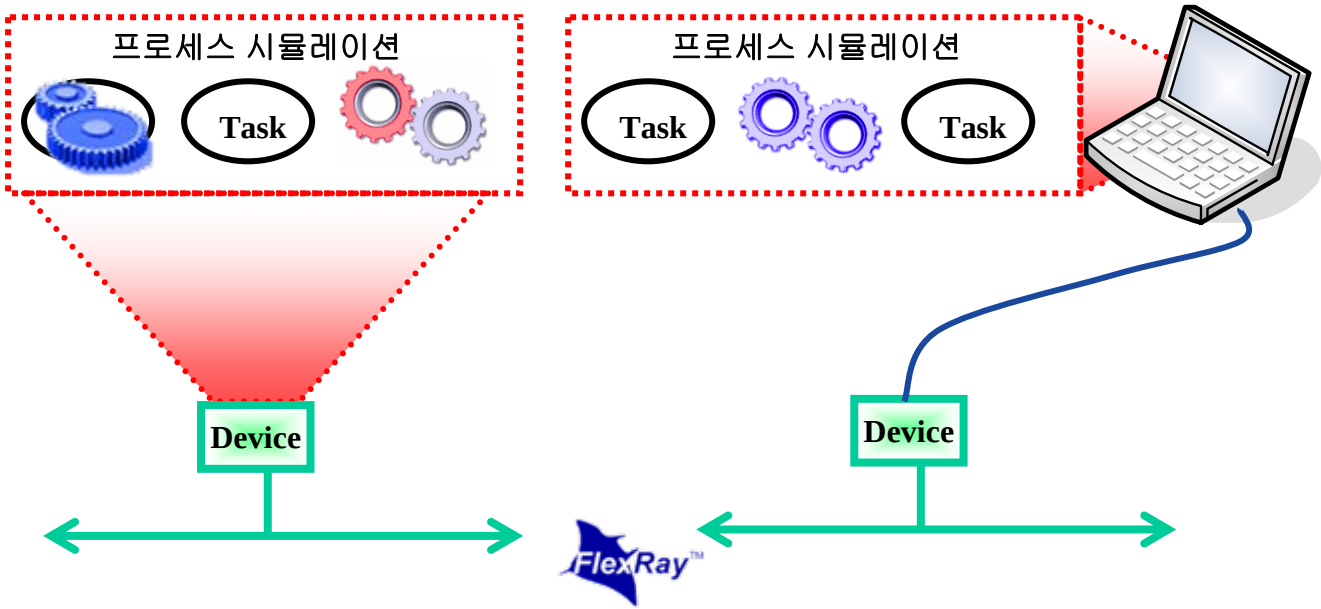
Rest Bus의 시뮬레이션 [3/3]

임베디드 실행 환경

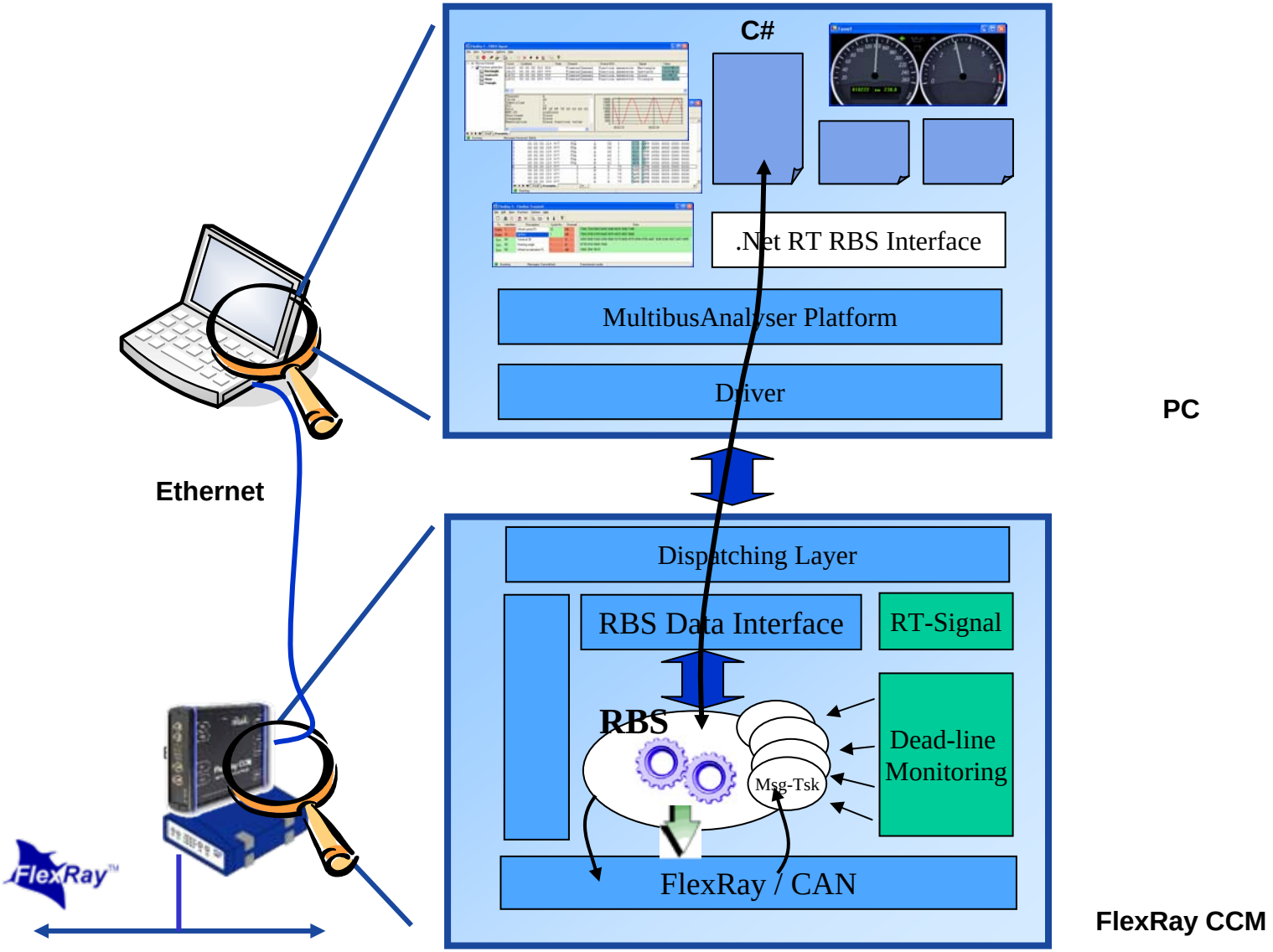
- In cycle & next cycle response
- Realtime Tasks

PC-기반 환경

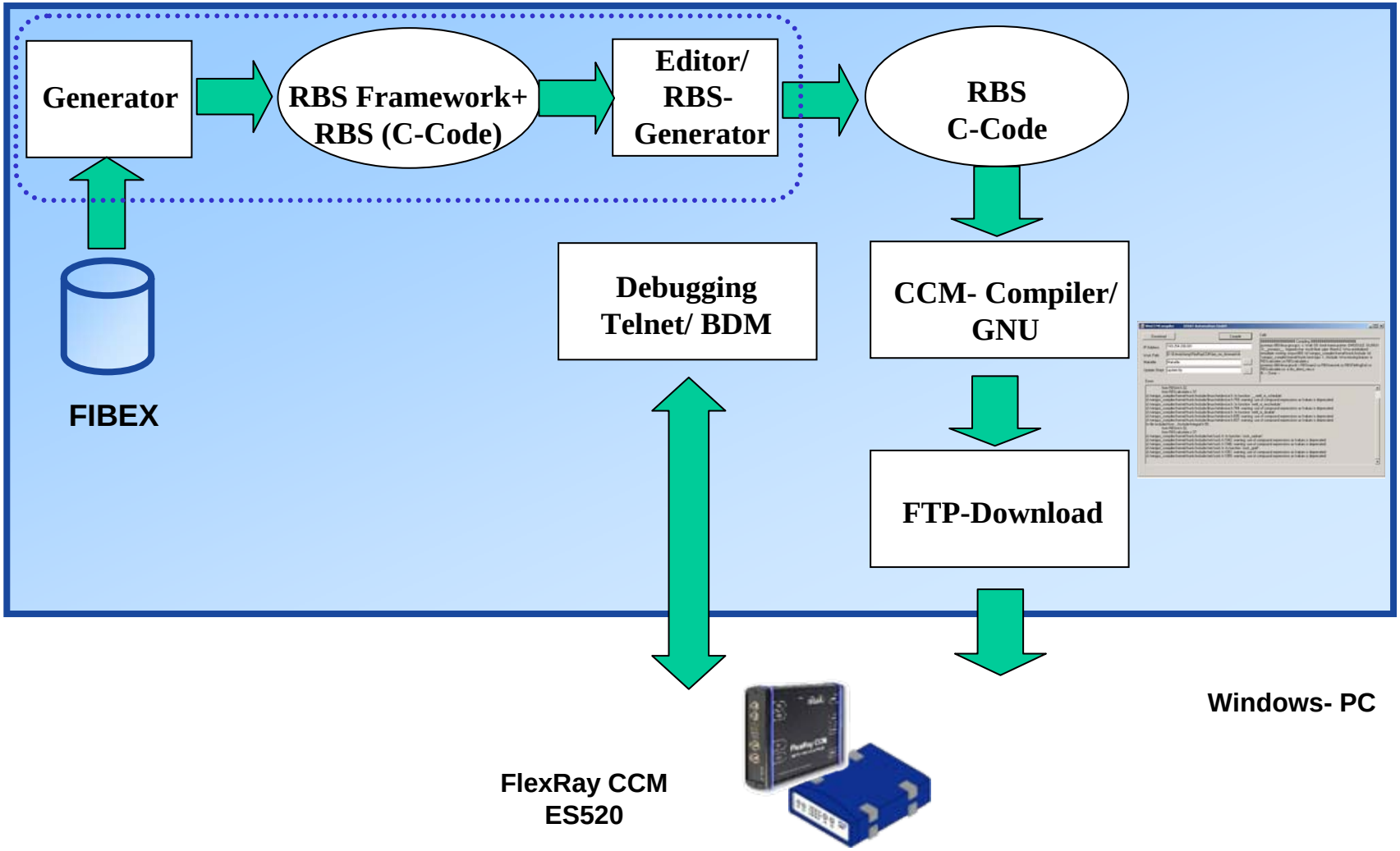
- 낮은 응답 시간 요건의 작업들
- 사용자 인터페이스
- 임베디드 환경을 위한 제어 인터페이스



Rest Bus 시뮬레이션



임베디드 **Rest Bus** 시뮬레이션, 코딩



PC-기반 Rest Bus 시뮬레이션