(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0019544
(43) 공개일자 2018년02월26일

(51) 국제특허분류(Int. Cl.)
*G06F 13/38* (2006.01) *G06F 13/42* (2006.01)
(52) CPC특허분류
*G06F 13/385* (2013.01)
*G06F 13/426* (2013.01)
(21) 출원번호 10-2017-7035534
(22) 출원일자(국제) 2016년06월20일
심사청구일자 없음
(85) 번역문제출일자 2017년12월08일
(86) 국제출원번호 PCT/US2016/038312
(87) 국제공개번호 WO 2016/205776
국제공개일자 2016년12월22일
(30) 우선권주장
62/181,533 2015년06월18일 미국(US)
15/185,257 2016년06월17일 미국(US)

(71) 출원인
**마이크로칩 테크놀로지 인코포레이티드**
미국 85224-6199 아리조나 챈들러 웨스트 챈들러 블러바드 2355
(72) 발명자
**사무엘, 로샨**
미국, 아리조나 85249, 챈들러, 사우스 밍거스 플레이스, 5065
**애버루에트, 잔마이클**
미국, 아리조나 85297, 길버트, 이스트 마샬 애비뉴, 4477
*(뒷면에 계속)*
(74) 대리인
**특허법인세신**

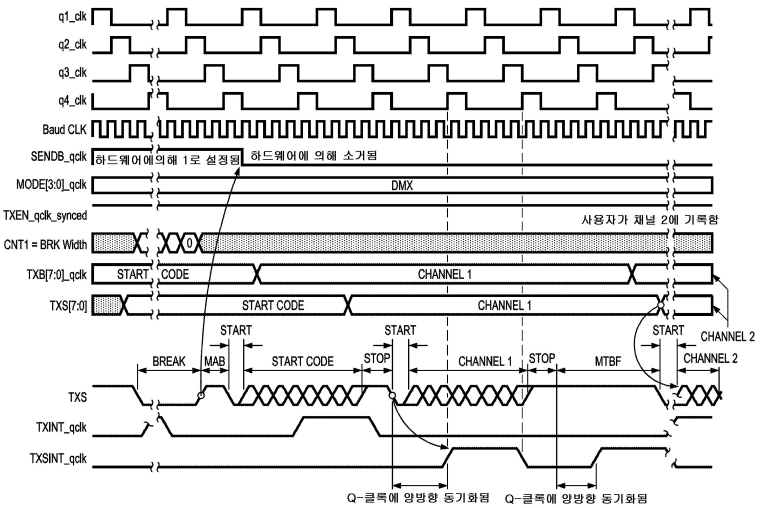
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **자동화된 프로토콜을 구비한 ＵＡＲＴ**

## *(57) 요 약*

범용 비동기 송수신기(UART) 인터페이스가 개시된다. UART 인터페이스는, 환경설정가능한 비동기 송수신기 유닛 및 환경설정가능한 상태 머신을 포함할 수 있고, 여기서, 상기 상태 머신은 상기 송수신기 유닛이 다양한 보드 속도(baud rate)들을 지원하고 또한 시작 비트 및 정지 비트 환경설정을 제공하도록 환경설정하는 것을 가능하게 하고, 여기서 상기 상태 머신은 또한 복수의 통신 프로토콜들을 자동으로 지원하도록 환경설정할 수 있다.

## *대 표 도*



(52) CPC특허분류
*G06F 13/4295* (2013.01)
(72) 발명자
**브라운, 워드 알.**
미국, 아리조나 85249, 챈들러, 사우스 빅 혼 플레이스, 5011
**드사이, 친탄**
미국, 아리조나 85286, 챈들러, 웨스트 시걸 드라이브, 1100
**아이비, 브랜트**
미국, 아리조나 85248, 챈들러, 웨스트 인디고 드라이브, 1303
**도치아, 라즈반**
러시아, 부커레스트 062057, 블록 비27, 타구르 네암츠 14에이

# 명 세 서

## 청구범위

### 청구항 1

범용 비동기 송수신기(UART) 인터페이스로서,

환경설정가능한 비동기 송수신기 유닛; 및

환경설정가능한 상태 머신을 포함하고,

상기 상태 머신은, 상기 송수신기 유닛이 다양한 보드 속도(baud rate)들을 지원하고 또한 시작 비트 및 정지 비트 환경설정을 제공하도록 환경설정하는 것을 가능하게 하고,

상기 상태 머신은, 복수의 통신 프로토콜들을 자동으로 지원하도록 환경설정할 수 있는, UART 인터페이스.

### 청구항 2

제1항에 있어서,

상기 복수의 통신 프로토콜들은 DMX 프로토콜을 포함하는, UART 인터페이스.

### 청구항 3

제1항에 있어서,

상기 복수의 통신 프로토콜들은 DALI 프로토콜을 포함하는, UART 인터페이스.

### 청구항 4

제1항에 있어서,

상기 복수의 통신 프로토콜들은 LIN 프로토콜을 포함하는, UART 인터페이스.

### 청구항 5

제1항 내지 제4항 중 어느 한 항에 있어서,

환경설정 레지스터에 의해(through) 동작 모드가 설정되는, UART 인터페이스.

### 청구항 6

제1항 내지 제5항 중 어느 한 항에 있어서,

상기 상태 머신은, 제2 모드에서, BREAK, MAB, 프레임 폭, MTBF 및 MTBP를 위한 자동 지원을 제공하는, UART 인터페이스.

### 청구항 7

제1항 내지 제6항 중 어느 한 항에 따른 UART 인터페이스를 포함하는 마이크로컨트롤러.

### 청구항 8

제7항에 있어서,

상기 마이크로컨트롤러는 8-비트 마이크로컨트롤러, 16-비트 마이크로컨트롤러, 및 32-비트 마이크로컨트롤러로 구성되는 그룹으로부터 선택되는, 마이크로컨트롤러.

### 청구항 9

제7항에 있어서,

상기 UART 인터페이스에 결합되는 타이머를 더 포함하는, 마이크로컨트롤러.

**청구항 10**

제1항 내지 제9항 중 어느 한 항에 있어서,

상기 동작 모드는 맨체스터(Manchester) 모드를 포함하는, UART 인터페이스.

**청구항 11**

환경설정가능한 비동기 송수신기 유닛과 환경설정가능한 상태 머신을 포함하는 범용 비동기 송수신기(UART) 인터페이스 - 상기 상태 머신은 상기 송수신기 유닛이 다양한 보드 속도들을 지원하고 시작 비트 및 정지 비트 환경설정을 제공하도록 환경설정하는 것을 가능하게 하고, 상기 상태 머신은 또한 복수의 통신 프로토콜들을 자동으로 지원하도록 환경설정할 수 있음 -;

상기 UART 인터페이스를 포함하는 마이크로컨트롤러; 및

상기 UART 인터페이스를 통해 데이터를 송수신할 수 있는, 상기 마이크로컨트롤러에 결합된 복수의 컴포넌트를 포함하는, 마이크로컨트롤러 시스템.

**청구항 12**

제11항에 있어서,

상기 복수의 통신 프로토콜들은 DMX 프로토콜을 포함하는, 마이크로컨트롤러 시스템.

**청구항 13**

제11항에 있어서,

상기 복수의 통신 프로토콜들은 DALI 프로토콜을 포함하는, 마이크로컨트롤러 시스템.

**청구항 14**

제11항에 있어서,

상기 복수의 통신 프로토콜들은 LIN 프로토콜을 포함하는, 마이크로컨트롤러 시스템.

**청구항 15**

제11항 내지 제14항 중 어느 한 항에 있어서,

환경설정 레지스터에 의해 동작 모드가 설정되는, 마이크로컨트롤러 시스템.

**청구항 16**

제11항 내지 제15항 중 어느 한 항에 있어서,

상기 상태 머신은, 제2 모드에서, BREAK, MAB, 프레임 폭, MTBF 및 MTBP에 자동 지원을 제공하는, 마이크로컨트롤러 시스템.

**청구항 17**

제11항 내지 제16항 중 어느 한 항에 있어서,

상기 마이크로컨트롤러는 8-비트 마이크로컨트롤러를 포함하는, 마이크로컨트롤러 시스템.

**청구항 18**

제11항 내지 제17항 중 어느 한 항에 있어서,

상기 UART 인터페이스에 결합되는 타이머를 더 포함하는, 마이크로컨트롤러 시스템.

**청구항 19**

제11항 내지 제18항 중 어느 한 항에 있어서,

상기 동작 모드는 맨체스터 모드를 포함하는, 마이크로컨트롤러 시스템.

**청구항 20**

제11항 내지 제19항 중 어느 한 항에 있어서,

상기 복수의 컴포넌트들은 복수의 시청각(audio-visual) 컴포넌트들을 포함하는, 마이크로컨트롤러 시스템.

## *발명의 설명*

### *기 술 분 야*

[0001] 본 출원은, 2015년 06월 18일자로 출원된 동일 출원인에 의한 미국 가출원 제62/181,533호의 우선이익을 주장하며; 상기 가출원은 모든 목적을 위해 본 출원에 참조함으로써 통합된다.

[0002] 본 개시는 직렬 인터페이스들에 관한 것으로서, 특히 자동화된 프로토콜들을 구비한 범용 비동기 송수신기(UART) 인터페이스에 관한 것이다.

### *배 경 기 술*

[0003] UART들은 잘 알려져 있으며 통신 채널을 제공하기 위해 마이크로컨트롤러들에 널리 사용된다. UART 인터페이스는 병렬 데이터를 직렬 송신 형태로 변환한다. 다양한 유형의 프로토콜들이 현존하며 이들은 전자 공업 협회(Electronic Industries Alliance) 표준들인 RS-232, RS-422 및 RS-485와 같은 다양한 통신 표준들에 의해 정의되는 UART 통신에 사용된다.

## *발명의 내용*

### *해결하려는 과제*

[0004] 복수의 자동화된 프로토콜들을 지원할 수 있는 범용 비동기 송수신기 인터페이스의 제공이 필요하다.

### *과제의 해결 수단*

[0005] 범용 비동기 송수신기(UART) 인터페이스가 개시된다. UART 인터페이스는, 환경설정가능한 비동기 송수신기 유닛; 및 환경설정가능한 상태 머신을 포함할 수 있고, 여기서 상기 상태 머신은, 상기 송수신기 유닛이 다양한 보드 속도(baud rate)들을 지원하고 또한 시작 비트 및 정지 비트 환경설정을 제공하도록 환경설정하는 것을 가능하게 하고, 여기서 상기 상태 머신은, 복수의 통신 프로토콜들을 자동으로 지원하도록 추가로 환경설정할 수 있다.

[0006] 일부 실시예들에서, 상기 복수의 통신 프로토콜들은 DMX 프로토콜을 포함할 수 있다. 동일한 또는 대안적인 실시예들에서, 상기 복수의 통신 프로토콜들은 DALI 프로토콜을 포함할 수 있다. 동일한 또는 대안적인 실시예들에서, 상기 복수의 통신 프로토콜들은 LIN 프로토콜을 포함할 수 있다.

[0007] 일부 실시예들에서, 동작 모드는 환경설정 레지스터에 의해 설정된다. 일부 실시예들에서, 상기 상태 머신은, 제2 모드에서, BREAK, MAB, 프레임 폭, MTBF 및 MTBP를 위한 자동 지원을 제공한다.

[0008] 일부 실시예들에서는, 또한, 마이크로컨트롤러 시스템이 개시된다. 마이크로컨트롤러 시스템은 UART를 포함하는 마이크로컨트롤러를 포함할 수 있다. 이러한 실시예들에서, 마이크로컨트롤러는 8-비트 마이크로컨트롤러, 16-비트 마이크로컨트롤러, 및 32-비트 마이크로컨트롤러로 구성되는 그룹으로부터 선택될 수 있다. 이러한 실시예들에서는, 마이크로컨트롤러가 UART 인터페이스에 결합되는 타이머를 더 포함할 수 있다.

[0009] 일부 실시예들에서, 동작 시스템은 맨체스터 모드를 포함할 수 있다.

### *도면의 간단한 설명*

[0010] 도 1은 본 개시의 특정 실시예들에 따른, 제1 내지 제512 바이트의 예시적인 송신을 도시한다.

도 2는 본 개시의 특정 실시예들에 따른, 예시적인 통신(corresponding) DMX 수신 프로토콜을 도시한다.

도 3은 본 개시의 특정 실시예들에 따른, 특정 데이터 사이즈(예를 들어, 레지스터 P1=2(바이트))로 설정된 메모리 부분을 구비한 예시적인 16-비트 DALI 디바이스를 도시한다.

도 4는 본 개시의 특정 실시예들에 따른, DALI 송신의 예시적인 타이밍도를 도시한다.

도 5는 본 개시의 특정 실시예들에 따른, LIN 마스터/슬레이브 모드의 예시적인 타이밍도를 도시한다.

도 6은 본 개시의 특정 실시예들에 따른, LIN 마스터/슬레이브 모드의 예시적인 타이밍도를 연속하여 도시한다.

도 7은 본 개시의 특정 실시예들에 따른, LIN 슬레이브 전용 모드의 예시적인 타이밍도를 도시한다.

도 8은 본 개시의 특정 실시예들에 따른, LIN 슬레이브 전용 모드의 예시적인 타이밍도를 연속하여 도시한다.

도 9는 본 개시의 특정 실시예들에 따른, 송신용 맨체스터 디코딩 방식의 예시를 도시한다.

도 10은 본 개시의 특정 실시예들에 따른, 수신용 맨체스터 디코딩 방식의 예시를 도시한다.

도 11은 본 개시의 특정 실시예들에 따른, 수신용 맨체스터 디코딩 방식의 예시를 연속하여 도시한다.

도 12는 본 개시의 특정 실시예들에 따른, 수신용 맨체스터 디코딩 방식의 예시를 계속 연속하여 도시한다.

도 13은 본 개시의 특정 실시예들에 따른, 수신용 맨체스터 디코딩 방식의 예시를 더욱 계속 연속하여 도시한다.

## *발명을 실시하기 위한 구체적인 내용*

[0011] 다양한 실시예들에 따르면, DMX(Digital Multiplex), LIN(Local Interconnect Network), DALI(Digital Addressable Lighting Interface) 등과 같은 다단(multi-step) 프로토콜들이 단지 데이터 상호작용(data interactions)만을 사용하여 구현될 수 있게 하는 범용 비동기 송수신기(UART)가 제공될 수 있다. 소프트웨어는 브레이크(BRK) 문자, START 비트, STOP 비트, 체크섬 등을 어디에 배치할지를 걱정할 필요가 없다. 이러한 엘리먼트들의 모든 물리 계층 배치는, 자동으로 처리될 수 있다. 소프트웨어는 링크를 통해서(over) 데이터를 판독하고 기록하는 것에만 책임질 수 있다.

[0012] 각 프로토콜에서, 사용자는 송신되거나 수신될 데이터를 판독하고 기록할 수 있다. 본 개시를 위하여, "사용자"는 임의의 적절한 전자 디바이스 및/또는 상기 디바이스의 조작자를 가리킬 수 있다. 한편, UART는 START, STOP, IDLE 주기들, 체크섬 계산, 데이터 삽입, 패리티 비트들 등을 자동으로 삽입할 수 있다. 이로써, 적어도 다음의 프로토콜들: 자동 DMX 프로토콜, 자동 DALI 프로토콜, 자동 LIN 프로토콜이 구현될 수 있다.

[0013] 특정의 종래의 UART들은 비트들에 대한 상대적으로 저레벨의 제어만을 제공할 수 있다. 대부분의 마이크로컨트롤러들은 이용가능한 많은 소프트웨어 사이클들을 가지기 때문에, 고레벨의 문제(issue)들(예를 들어, START 및 STOP 비트(들), 체크섬 등을 배치할 장소)은 소프트웨어로 처리될 수 있다. 하지만, 8-비트 마이크로컨트롤러들과 같은 소형 솔루션들에서는, 이용가능한 총 소프트웨어 사이클들이 적어서, 하드웨어 블록들을 활용하여(leverage) 소프트웨어의 부담을 제거(take off)할 수 있다.

[0014] 다양한 실시예들에 따르면, 범용 프로토콜 소프트웨어 설정(setup)이 제공될 수 있다. 예를 들면, 사용자로 하여금 복수의 이용가능한 프로토콜 모드들 중 하나를 선택할 수 있게 하는 복수의 비트들이 제공될 수 있다. 이것은, 예를 들면, UART 환경설정 레지스터의 제어 비트를 설정함으로써(예를 들어, USE_PROT=1) 달성될 수 있다. 더욱이, 필수(required) 클록 속도가 선택된 클록에서 분주될 수 있도록, 보드 속도값(baud rate value)(예를 들어, 보드 속도 발생기 비트 "BRGH/L")이 설정될 수 있다. 이러한 클록 속도들은 단지 예시로서 제공된다. 하드웨어는 클록 속도에 어떤 제한들도 가하지 않을 수 있다. 클록 속도들은 서로 다른 프로토콜들에 제공될 수 있는데, 예를 들면, DMX는 5-250 KHz 사이의 클록 속도를 사용할 수 있고, DALI는 2180-2666Hz의 클록 속도(이것은 핀(pin)에서 맨체스터 1090-1333 보드로 변환됨)를 사용할 수 있고, 그리고 LIN은 100Hz-20KHz의 클록 속도를 사용할 수 있다.

[0015] 일부 실시예들에서는, 워터마크값이 필수 워터마크로 설정(set up)될 수 있다. 예를 들면, 송신 선입선출(Transmit FIFO)이 3에 도달할 때 인터럽트를 원하면, 특정의 환경설정은 TINTFLG[2:0]을 3으로 설정할 수 있다. 본 개시의 범위에서 벗어나지 않고, 다른 환경설정값들이 사용될 수도 있다. 예를 들면, 특유의(appropriate) 신호 핀들의 로직 레벨들은 프로토콜에 의존하여 원하는 값으로 설정될 수 있는데; 송신 극성 비트(transmit polarity bit)가 설정될 수 있고; 모델 선택 신호가 원하는 모드로 설정될 수 있고; 송신-인에이블

신호가 설정될 수 있고; 수신-인에이블 신호가 설정될 수 있다

[0016] BRK(예를 들어, DMX 및 LIN)를 구비한 프로토콜을 포함하는 일부 실시예들에서, 만일 BRK가 패킷의 중간에서 수신되면(예상되지 않은 이른 BRK), 상태 머신은 마치 새로운 패킷이 시작된 것처럼 행동할 것이다. 수신 버퍼들의 컨텐츠는 버려지지(flush) 않을 것인데; 사용자는 이들을 판독해야만 하거나 (예를 들어, RXFLSH 비트를 활성화함으로써) 버퍼 컨텐츠를 의도적으로 버려야만 할 것이다.

[0017] 일부 실시예들에서, 체크섬값들과 패리티 내부 가산기들(그리고 그것의 워킹 레지스터들 중 일부)은 모든 프로토콜 모드들의 개시시 소거된다. 하지만, 체크섬과 패리티 상태 비트들은 소거되지 않는다. 일부 실시예들에 따르면 이들은 사용자에 의해 소거될 수 있다.

[0018] **DMX 프로토콜 상태 머신**

[0019] 다양한 실시예들에서, 자동화된 프로토콜을 구비한 UART는 DMX 프로토콜을 이용할 수 있다. 이러한 실시예들에서, 특정의 프로토콜은 사용자로 하여금 복수의 이용가능한 프로토콜 모드들 중 하나를 선택할 수 있게 하는 복수의 비트들에 의해 선택될 수 있다. 예를 들면, 특정의 환경설정은 "USE_PROT" 신호를 로직 하이로 설정함으로써 DMX 프로토콜을 선택할 수 있다. DMX는 스테이지 및 쇼 장비(Stage and Show Equipment)에서 이용되는 프로토콜이다. 스테이지 및 쇼 장비는 조명, 안개 장치 등을 포함한다. 상기 프로토콜은 명령들을 송출하는 제어 콘솔과 이러한 명령들을 수신하는 극장 조명과 같은 장비로 이루어진다. 상기 프로토콜은 일방향이며, 상기 콘솔은 명령들을 송신하고 상기 장비는 명령들을 수신한다. 장비는 어떤 것도 송신하지 않으며, 콘솔들은 어떤 것도 수신하지 않는다. 또한, DMX 환경설정들은 에러 제어들 또는 재송신 메커니즘을 포함하지 않는다.

[0020] DMX(또는 DMX-512로도 널리 알려질 수 있음)는 512 채널들의 "집합(Universe)"으로 이루어진다. 이것은, 하나의 콘솔이 하나의 DMX 링크에 512 바이트까지 출력할 수 있음을 의미한다. 라인상의 각 장비는 이 바이트들 중에서 하나 이상을 청취(listen)하도록 프로그램된다. 예를 들면, 집합들 중 하나에 연결된 안개 장치는 바이트 번호 10에서 시작을 수신하고 4바이트 카운트들을 수신하도록 프로그램될 수 있고, 조명 유닛은 바이트 번호 22에서 시작을 수신하고 1바이트 카운트를 수신하도록 프로그램될 수 있다.

[0021] 각 DMX 송신은 "시작 코드(Start Code)"로 지칭되는 바이트가 뒤따르는 BREAK를 이용하여 개시될 수 있다. BREAK의 폭은 특정 레지스터, 예를 들면, P1 레지스터를 이용하여 설정된다. BREAK에는 MAB(Make After Break)가 뒤따른다. 도 1은 본 개시의 특정 실시예들에 따른 제1 내지 제512 바이트의 예시적인 송신을 도시한다. 송신된 각각의 바이트는, 특정 신호(예를 들어, USE_FIFO)가 특정 로직 레벨(예를 들어, 0)로 설정될 때, 인터럽트 이벤트를 생성할 것이다. 인터럽트 주파수(frequency)는, 해당 신호가 변경될 때에, 변경될 수 있다. 예를 들면, 신호가 반대 로직 레벨(예를 들여, 1)로 설정될 때, 인터럽트 주파수는 워터마크 설정(Watermark Setting)에 의존할 수 있다.

[0022] 일부 실시예들에서, UART가 수신기로서 구성될 때, UART는 미리결정된 길이(예를 들어, 적어도 11 클록 폭)의 BREAK 문자를 청취(listen for)할 수 있다. BREAK 직후에 UART는 MAB를 확인(see)할 것이다. 이 공간(space)은 UART에 의해 무시될 수 있다. 이어서 UART는 시작 코드를 확인할 것이다. 시작 코드는 항상 클로킹(clock)되어 UART의 특정 메모리 부분(예를 들어, 수신 버퍼)에 저장될 것이다. 시작 코드는 항상 FIFO 점유(occupancy)에 관계없이 인터럽트를 생성할 것이다. 만일 인터럽트 주파수 신호가 특정 로직 레벨로 설정되면(예를 들어, USE_FIFO=1), 수신 인터럽트는 워터마크 설정에 따라 플래그될 것이다.

[0023] 시작 코드 이후에, UART는 제1 내지 제512 바이트를 수신할 것이다. 수신된 바이트들을 위해 512개의 인터럽트들을 갖기보다는, UART는 관심있는 하나가 수신될 때까지 수신된 바이트들을 무시할 수 있다. 이것은, UART의 다른 메모리 부분들(예를 들어, P2 및 P3 레지스터들)을 사용하여 수행된다. 하나의 메모리 부분(예를 들어, P2 레지스터)은, 수신 프로세스를 시작하기 위한 바이트 번호의 값을 유지할 수 있다. 상기 열거된 안개 장치의 경우, 예를 들면, 사용자는 P2 레지스터를 10으로 프로그램할 수 있다. 앞서 언급된 조명 기구는 번호 22로 프로그램될 수 있다. 또하나의 메모리 부분(예를 들어, P3 레지스터)은, 수신할 바이트들의 개수에 대한 값을 유지할 수 있다. 안개 장치의 경우, UART가 바이트 10, 11, 12 및 13, 총 4 바이트를 수신할 수 있도록 다른 메모리 부분(예를 들어, P3)은 값 4를 유지할 것이다. 도 2는 본 개시의 특정 실시예들에 따른, 예시적인 DMX 수신 프로토콜을 도시한다.

[0024] **송신 소프트웨어 모델**

[0025] 다양한 실시예들에서, 자동화된 DMX 프로토콜을 구비한 UART는 또한, 송신 소프트웨어 모델을 포함할 수 있다. 송신 소프트웨어 모델은 특정 신호를 어서트함으로써 설정될 수 있다. 예를 들면, 핀 신호(예를 들어, P1H/L)는

송신된 BREAK를 위해 원하는 구간(duration)으로 설정될 수 있다. 전형적인 값은 13이다.

[0026] 일부 실시예들에서, 상기 모델은 이어서 시작 코드를 통신 버스(예를 들어, TXB[7:0])에 기록할 수 있다. 이것은, 제1 워드가 FIFO에 기록될 때, UART가 BREAK를 송신하게 할 수 있다. BREAK는 원하는 구간 동안(예를 들어, 'P1H/L에 설정된 클록 카운트들의 개수에 의해) 지속될 수 있다. UART는 이어서 잔여 채널들을 통신 버스(예를 들어, TXB[7:0])에 기록할 수 있다. 시작 코드 이후에, 다음 채널들이 버스 FIFO의 외부로 송신된다. 송신 인터럽트 신호(예를 들어, int_tx)는, 워터마크에 도달할 때마다(예를 들어, USE_FIFO=1) 또는 버스가 가득찰 때(USE_FIFO=0), 설정될 수 있다.

[0027] 만일 잔여 채널들의 어떤 것도 기록되지 않으면, 송신 출력은 하이로 될 수 있고 더많은 버스 컨텐츠가 이용가능할 때까지 하이를 유지할 수 있다. 이로 인해, 결과적으로 버스에 더많은 공간이 이용가능하게 될 때까지 송신 타이머 리셋 신호(예를 들어, uart_timer_reset_tx)를 토글링하지 않을 수 있다. 타이머 2와 같은 외부 타이머가, DMX 연결이 시간만료하였는지 여부를 결정하기 위해, 상기 토글 사이의 시간을 카운트할 수 있다. 만일 그렇다면, 소프트웨어는, 송신 상태 머신을 재시작하기 위해서, (예를 들어, TXEN=0으로 설정하고 이어서 TXEN=1로 설정함으로써) 송신 인에이블 신호를 교대시킬 수 있다. 이것은 또한 버스를 비울(flush) 것이다. 일부 실시예들에서, UART는 MAB와 2-STOP 비트들을 자동으로 처리할 수 있다.

[0028] **수신 소프트웨어 모델**

[0029] 다양한 실시예들에서, 자동화된 DMX 프로토콜을 구비한 UART는 또한, 수신 소프트웨어 모델을 포함할 수 있다. 수신 소프트웨어 모델은 특정 신호를 어서트함으로써 설정될 수 있다. 예를 들면, 제1 핀 신호(예를 들어, P2H/L)는 데이터 수신을 시작하기 위해 원하는 바이트 카운트로 설정될 수 있다. 추가로, 제2 핀 신호(예를 들어, P3H/L)는 제1 핀 신호에서 시작을 수신하기 위한 바이트들의 원하는 카운트로 설정될 수 있다. 일부 실시예들에서, UART는 라인에 BREAK가 나타날 때까지 기다릴 수 있다. BREAK가 나타날 때, 샘플링 클록은 동기화될 수 있다.

[0030] 일부 실시예들에서, 수신된 제1 바이트는 START 코드이고 항상 FIFO에 로드된다. 수신 워터마크 인터럽트 신호(예를 들어, int_rx)는, 워터마크/FIFO 설정과 상관없이 START 코드가 수신될 때에는 항상 플래그된다. 시작 코드 이후에 상기 바이트가 시작되면, 바이트 카운트(어드레스)는 제1 신호에 의해 지정되어 매칭된다. 일단 매칭되면, 제2 핀 신호의 값이 FIFO에 로드될 수 있다. 수신 워터마크 인터럽트 신호(예를 들어, int_rx)는 워터마크에 도달할 때마다 인터럽트를 설정할 수 있다. 만일 수신 라인이 미리결정된 시간(예를 들어, 1초)보다 긴 시간 동안 유휴상태(idle)이면, DMX 링크는 죽은 것으로 여겨질 수 있다. 만일 UART가 미리결정된 시간까지 카운트할 만큼 충분히 큰 타이머를 가지고 있지 않다면, 외부 타이머가 이러한 미리결정된 시간을 재도록 사용될 수 있다.

[0031] **DALI 프로토콜 상태 머신**

[0032] 다양한 실시예들에서, 자동화된 프로토콜을 구비한 UART는 DALI 프로토콜을 이용할 수 있다. 이러한 실시예들에서, 특정 프로토콜이, 사용자로 하여금 복수의 이용가능한 프로토콜 모드들 중 하나를 선택할 수 있게 하는 복수의 비트들에 의해 선택될 수 있다. 예를 들면, 특정의 환경설정은 "USE_PROT" 신호를 로직 하이로 설정함으로써 DALI 프로토콜을 선택할 수 있다. DALI는 사무실들 및 공장들과 같은 대형 빌딩들의 조명을 제어하는 데에 사용되는 프로토콜이다. 그것은 "디바이스"와 "기어(Gear)"의 2가지 모드들로 이루어진다. "디바이스"는 조명 기구들에 명령들을 송출하는 메인 컨트롤러이다. 조명 기구 자체는 기어로 알려져 있다. 모든 비트 송신은 맨체스터 인코딩(Manchester Encoding)에서 수행될 수 있는데, 이는 하드웨어에 의해 수행된다. DALI 와이어(wire)는 하프-듀플렉스(half-duplex)이고; 송신 라인과 수신 라인은 AC 변압기(transformer)를 통해 전기적으로 서로 결합(tie)되어 있다.

[0033] 다른 모든 프로토콜들과는 다르게, DALI는 최상위 비트(MSb)가 먼저 송신된다. UART 트랜잭션은, "디바이스"가 송신을 시작할 때, 개시될 수 있다. 디바이스 송신은 DALI 1.0에서는 2바이트로 그리고 DALI 2.0에서는 3바이트로 구성된다. 첫번째 바이트는 'YAAAAAAS'로 지정된 제어 바이트이고, 이어서 하나 또는 개의 데이터 바이트가 뒤따른다. 디바이스로부터 나온 이 바이트들은 "포워드 프레임(Forward Frame)"으로 지칭된다. 도 3은 본 개시의 특정 실시예들에 따른, 메모리 부분이 특정 데이터 사이즈(예를 들어, 레지스터 P1=2(바이트))로 설정되어 있는 예시적인 16-비트 DALI 디바이스를 도시한다. 송신된 각각의 바이트는 USE_FIFO=0일 때 인터럽트 이벤트를 생성할 것이다. USE_FIFO=1일 때, 인터럽트 주파수는 워터마크 설정에 의존할 수 있다. 일부 환경설정들에서, 디바이스가 USE_FIFO=0으로 설정되어 있었다면, 단지 하나의 송신 버스(예를 들어, TXB)와 하나의 시프터만이

이용가능할 것이기 때문에, 3개의 워드들을 연속으로(back to back) 기록하는 것이 가능하지 않을 수 있다. 따라서, 3개의 바이트가 송신되어야 하는 DALI 2.0 디바이스를 위해 코드를 기록할 때에는, 소프트웨어가 인터럽트 신호(예를 들어, int_tx 또는 TXBE/TXBF)와 인터럽트 충돌 신호(예를 들어, TXMTIF)를 모니터링해야만 한다. 송신 버스가 이용가능하게 되자마자 그리고 시프터가 비워지기 전에, 제3 바이트가 버스에 기록되어야 한다. 이것은 3개의 바이트들이 인터럽트없이 연속으로 송신되는 것을 보장할 것이다.

[0034] 라인상의 모든 기어는 이 포워드 프레임을 수신할 수 있다. 기어들 중 하나는 '백 프레임(Back Frame)'이라고 지칭되는 하나의 바이트의 답신으로 이것에 응답할 수 있다. DALI 프로토콜은, 이 백 프레임이 11 보드 클록 사이클들 내에서 수신되기 시작해야함을 요구한다. 만일 백 프레임의 시작이 수신되면, 디바이스에는 백 프레임의 종료 이후에 또하나의 11 클록 사이클들을 기다릴 것이 요구된다. 이 시간 이후에, 디바이스는 자유롭게 또하나의 포워드 프레임을 송신한다. 도 4는 본 개시의 특정 실시예들에 따른, 예시적인 DALI 송신용 타이밍도를 도시한다. 수신된 각각의 바이트는 USE_FIFO=0일 때 인터럽트 이벤트를 생성할 것이다. 만일 USE_FIFO=1이면, 수신 인터럽트가 워터마크 설정에 따라 플래그될 것이다.

[0035] DALI 모드에 있을 때에는, UART가 현재의 바이트가 유효 수신으로 간주되기 전에 STOP 비트들 둘 다가 정확하게 수신되었음을 확인해야만 한다. 만일 어느 하나의 STOP 비트라도 정확하게 수신되지 않았으면, 수신 바이트는 시프터로부터 RXB로 송신되기 전에 폐기된다. UART 하드웨어는 STOP 비트들의 필수 개수를 검출할 것이고, 상기 개수가 검출되지 않으면 사용자에게 에러라고 응답할 것이다.

[0036] **DALI 디바이스 소프트웨어 모델**

[0037] 다양한 실시예들에서, 자동화된 DALI 프로토콜을 구비한 UART는 디바이스 소프트웨어 모델을 포함할 수 있다. 디바이스 소프트웨어 모델은 특정 신호를 어서트함으로써 설정될 수 있다. 예를 들면, 핀 신호(예를 들어, P1H/L)가 포워드 프레임들 사이의 클록 사이클들의 소망 개수로 설정될 수 있다. 전형적으로, 이 값은 11 또는 그보다 클 것이다. UART는 YAAAAAAS에 이어서 즉시 데이터를 (예를 들어, X[7:0]에) 송신할 수 있다. 따라서, 바이트들 둘 다는 송신이 개시되기 전에 이용가능해야만 한다. UART는 이어서 'YAAAAAAS' 헤더의 값을 송신 버스의 일부(예를 들어, TXB[7:0])에 기록할 수 있다. 이것은 버스로의 첫번째 기록이기 때문에, 그것은 즉시 송신 시프터로 복사될 수 있으며, 버스상의 하나의 위치를 해제(free up)할 것이다.

[0038] UART는 이어서 송신 데이터로부터의 (예를 들어, X[7:0]의) 데이터 바이트의 값을 송신 버스에(예를 들어, TXB[7:0]에) 기록할 수 있다. 만일 24-비트 DALI 2.0 송신이 요구되면, 제3 바이트는 송신 시프터가 비워지기 전에 버스에 기록되어야만 한다. 만일 버스가 비워진 상태로 유지되면, 송신 라인은 하이로 되고 하이를 유지할 것이다. UART는 이어서 송신 타이머 리셋 신호(예를 들어, uart_timer_reset_tx 출력)를 어서트할 수 있다. 일부 실시예들에서, 외부 타이머가 타임아웃을 모니터링하는 데에 사용될 수 있다. 만일 그렇다면, 소프트웨어는, 송신 상태 머신을 재시작하기 위해서, (예를 들어, TXEN=0으로 설정하고 이어서 TXEN=1로 설정함으로써) 송신 인에이블 신호를 토글해야만 한다. 소프트웨어는 또한 송신 버스를 비울지를 선택할 수 있다.

[0039] 첫번째 프레임 이후에, UART는 핀 신호(예를 들어, P1H/L)에 지정된 프레임들 사이의 클록들의 개수를 기다릴 수 있다. 이 시간 동안, 송신 출력은 하이로 유지된다. DALI에 있어서, 이 값은 11이고, 이는 UART가 백 프레임을 기다리기 위해 11 클록을 기다리게 할 것이다. UART는 이어서 송신 버스 FIFO로부터 다음 채널들을 송신할 수 있다. 송신 워터마크 인터럽트 신호(예를 들어, int_tx)는 워터마크에 도달할 때마다 인터럽트를 트리거할 수 있다. 일부 실시예들에서, UART는 헤더-데이터 바이트 쌍이 각 프레임을 위해 송신을 시작하기 전에 존재함을 확인할 수 있다. 일부 실시예들에서, UART는 자동으로 맨체스터 타이밍을 생성할 수 있다.

[0040] **DALI 기어 소프트웨어 모델**

[0041] 다양한 실시예들에서, 자동화된 DALI 프로토콜을 구비한 UART는 기어 소프트웨어 모델을 포함할 수 있다. 기어 소프트웨어 모델은 특정 신호를 어서트함으로써 설정될 수 있다. 예를 들면, 핀 신호(예를 들어, P1H/L)는 백 프레임을 시작하기 위해서 포워드 프레임들 이후에 클록 사이클들의 최대 소망 개수로 설정될 수 있다. 전형적으로 이 값은 11(예를 들어, 22의 절반-비트의 횟수)보다 작아야 한다. UART가 기어 모드로 설정될 때, UART는 수신된 포워드 프레임을 청취(listen)하는 것을 시작할 수 있다. 이것은, 포워드 프레임이 백 프레임이 송신되기 전에 수신되어야하기 때문에 요구된다. 포워드 프레임이 수신될 때, 포워드 프레임은 애플리케이션 소프트웨어에 의해 처리된다. 백 프레임의 값은 송신 버스(예를 들어, TXB[7:0])의 부분에 기록된다.

[0042] 버스의 상기 부분이 비워지지 않았을 때, UART는 포워드 프레임 이후에 클록 사이클들의 최대 소망 개수보다 적은 개수가 통과했음을 보장할 수 있다. 만일 상기 시간이 만료하기 전에 버스 워드가 이용가능하게 되었다면,

그것은 백 프레임으로서 송신된다. 만일 클록 사이클의 최대 소망 개수보다 많은 개수가 통과되었다면, 시프터의 컨텐츠는 폐기된다. 상기 워드는 버스로부터 송신 시프터로 전달될 것이지만, 그것은 송신되지 않을 것이다. 다음 백 프레임 동안에, 새로운 버스 워드가 사용자에 의해 버스에기록되어야 한다.

[0043] 첫번째 프레임 이후에, UART는 또하나의 포워드 프레임을 기다릴 수 있다. 이 시간 동안, 송신 출력은 하이로 유지될 수 있다. UART는 이어서, 다음 포워드 프레임이 수신될 때, 버스 FIFO로부터 다음 채널들을 송신할 수 있다. 송신 워터마크 인터럽트 신호(예를 들어, int_tx)는 워터마크에 도달할 때마다 인터럽트를 트리거할 수 있다.

[0044] **LIN 프로토콜 상태 머신**

[0045] 다양한 실시예들에서, 자동화된 프로토콜을 구비한 UART는 LIN 프로토콜을 이용할 수 있다. 이러한 실시예들에서, 특정의 프로토콜이 사용자로 하여금 복수의 이용가능한 프로토콜 모드들 중 하나를 선택할 수 있게 하는 복수의 비트들에 의해 선택될 수 있다. 예를 들면, 특정의 환경설정은 "USE_PROT" 신호를 로직 하이로 설정함으로써 LIN 프로토콜을 선택할 수 있다. LIN은 자동차 애플리케이션들에 사용되는 프로토콜이다. LIN 네트워크는 "마스터" 프로세스와 "슬레이브" 프로세스의 2종류의 소프트웨어 프로세스들로 이루어진다. 각 네트워크는 오직 하나의 마스터 프로세스만을 가지며, 하나 이상의 슬레이브 프로세스들을 가진다. 마스터 프로세스는 하나의 프로세서 UART에 상주한다. 네트워크상의 모든 그밖의 UART들은 슬레이브 소프트웨어 프로세스들에 의해 구동된다. 물리 계층의 관점에서 볼 때, 하나의 프로세서의 UART 매크로(macro)는, 하나의 마스터 프로세스가 하나의 네트워크상에 존재하는 한, 마스터 및 슬레이브 프로세스 둘 다에 의해 구동될 수 있다.

[0046] 일부 실시예들에서, UART는, 마스터 프로세스가 Break를 송신하여 송신을 시작할 때 개시되며, 이어서 구획문자 비트(Delimiter bit), 그리고 싱크 필드(Sync Field) 및 보호된 식별자(PID; Protected identification) 바이트가 뒤따라서 송신된다. PID는, 어느 슬레이브 프로세스가 마스터에 응답하도록 예정되었는지를 결정한다. PID 바이트가 인터바이트 공간(Interbyte Space)이면, 그것의 길이는 UART의 메모리 부분(예를 들어, P3 레지스터)에 의해 설정된다. 인터바이트 공간이 만료되기 전에, 하나 이상의 슬레이브 프로세스들은 마스터 프로세스에 응답할 수 있다. 만일 인터바이트 공간 내에 어떠한 응답도 없으면, 마스터는 자유롭게 또하나의 송신을 시작한다.

[0047] 슬레이브 소프트웨어 프로세스는 동일한 UART를 마스터로서 구동시킬 수 있거나 또는 또하나의 UART에 또는 또하나의 프로세서 디바이스에 함께 상주할 수도 있다. 슬레이브 프로세스는 마스터 프레임을 기다림으로써 시작한다. 만일 PID가 매칭되면, 슬레이브 프로세스는 아무것도 하지 않음으로써 또는 요청된 응답을 송신함으로써 응답한다. 일부 실시예들에서, 최대 8바이트들이 슬레이브에 의해 송신될 수 있다. 이들 바이트들의 끝에서, 슬레이브 UART는 송신된 워드들에 대해 체크섬을 자동으로 계산하고, 그것을 이들 바이트들의 끝에(예를 들어, 제9 데이터 바이트에) 데이터 바이트로서 송신하고, 이로써 LIN 트랜잭션이 완료된다. 체크섬은 한번에 추가 비트들(예를 들어, 8비트들)을 추가함으로써 그리고 그 결과에 캐리 비트를 추가함으로써 계산될 수 있다. 이 체크섬은 이어서 송신 데이터와 함께 송신될 수 잇다.

[0048] 데이터가 수신 UART에 의해 수신될 때, 체크섬은 동일한 알고리즘을 사용하여 계산된다. 송신기에 의해 계산된 체크섬 값인 다음 바이트(예를 들어, 제9 바이트)는 반전되고 국지적으로(locally) 계산된 체크섬에 더해진다. 예를 들면, 만일 상기 결과가 모두 '1'이면, 체크섬은 통과된다. 아래의 표 1은 LIN 2.2A 규격으로부터 얻어진 예시를 도시한다. 표 1은, 계산값이 0x4A, 0x55, 0x93,0xE5인 4개의 데이터 바이트들에 대한 예시 값들을 사용한다.

*표 1*

[0049]

| Action | hex | Carry | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x4A | 0x4A | | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| +0x55 | 0x9F | | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| Add Carry | 0x9F | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| +0x93 | 0x132 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| Add Carry | 0x33 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| +0xE5 | 0x118 | | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| Add Carry | 0x19 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| Invert | 0xE6 | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| Receiver Verification | | | | | | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Check<br>Local + Received | 0x19<br>+0xE6 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

[0050] 일부 실시예들에서, 레거시(Legacy) 및 향상된(Enhanced), 2종류의 체크섬이 이용될 수 있다. 레거시 체크섬에서, 데이터 바이트들(D0 내지 D7)만이 체크섬을 계산하기 위해 사용된다. 향상된 체크섬에서, 데이터(D0 내지 D7)와 PID가 포함된다. 어느 체크섬이 계산에 사용될지는, 특정 비트(예를 들어, COEN 비트)를 사용하는 소프트웨어에 의해 제어될 수 있다.

[0051] **LIN 송신 소프트웨어 모델**

[0052] 다양한 실시예들에서, 자동화된 LIN 프로토콜을 구비한 UART는 송신 소프트웨어 모델을 포함할 수 있다. 송신 소프트웨어 모델은 특정 신호를 어서트함으로써 설정될 수 있다. 예를 들면, 핀 신호(예를 들어, P2H/L)가 송신할 바이트들의 소망 개수로 설정될 수 있다. 일부 실시예들에서, 마스터/슬레이브 LIN 모드에서, 마스터 프로세스는 송신될 PID를 위한 값을 (예를 들어, P1L[5:0]에) 로드할 수 있다. 슬레이브 프로세스는 송신될 최대 8바이트들을 송신 버스 FIFO에 기록할 수 있다. 일부 실시예들에서, 슬레이브-전용(Slave-Only) 모드에서는, PID 송신이 필요하지 않기 때문에 상기 값은 무시될 수 있다.

[0053] 일부 실시예들에서, 마스터 슬레이브 모드에서는, PID 값이 기록되면 송신 프로세스가 개시될 수 있다. 송신 프로세스는 브레이크, 이어서 SYNC 바이트, 이어서 (예를 들어, P1L[5:0]로부터) 로드된 값의 컨텐츠를 포함할 수 있다. 상기 컨텐츠의 직후에, PID 값들은 하드웨어에 의해 계산될 수 있고 송신될 수 있다. 그 이후에, UART는 송신 버스의 점유(occupancy)를 체크할 수 있다. 만일 버스가 비어있지 않으면, 이것은 슬레이브 프로세스가 바이트들을 송신하기를 원하는 U ART에 상주함을 의미한다. 버스의 바이트들은 핀 신호(예를 들어, P2H)에 저장된 송신 바이트들의 개수까지 송신된다. 만일 버스가 비었으면, 송신 출력(예를 들어, TXS)은, 버스가 기록되는 것 및 로드된 값의 컨텐츠가 기록되는 것 중 어느 것이라도 먼저 이루어지기까지, 하이가 유지될 수 있다. 만일 로드된 값(예를 들어, P1L)이 기록되면, 마스터가 새로운 프레임을 시작하고 있다고 가정되고, 체크섬/패리티 엔진은 리셋되고, 모든 BREAK-SYNC 송신 프로세스는 다시 시작된다.

[0054] 일부 실시예들에서, 슬레이브-전용 모드에서는, SYNC를 수신하면 UART는 바이트들이 송신되도록 준비한다(arms). 버스에 기록되면 송신 프로세스가 시작된다. 만일 SYNC가 수신되기 전에 송신 버스에 기록되면, 송신은 SYNC가 완료될 때까지 보류된다. 이용가능하다면, 다음 바이트들은, 버스가 비워지면 비활성 출력을 이용하여(with) 송신 버스(예를 들어, TXB[7:0]) FIFO로부터 송신된다. 이것은 UART가 핀 신호(예를 들어, P2H/L)로부터 바이트들의 개수를 송신할 때까지 반복된다. 일부 실시예들에서, 송신 워터마크 인터럽트 신호(예를 들어, int_tx)는 워터마크에 도달할 때마다 인터럽트를 트리거할 수 있다. 핀 신호(예를 들어, P2H/L 데이터 워드들)로부터의 바이트들의 개수의 끝에서, UART는 계산될 수 있고 체크섬 비트(예를 들어, 체크섬 0(비트 COEN=1) 및 체크섬 1(비트 COEN=1) 중 어느 하나)를 송신할 수 있다. 일부 실시예들에서, 마스터/슬레이브 모드에서는, 새로운 송신을 시작하기 위해, 로드된 값(예를 들어, P1L)의 컨텐츠는 마지막 데이터가 송신된 이후에 기록되어야만 한다. 로드된 값(예를 들어, P1L)의 기록은 새로운 패킷을 트리거하기 때문에, 슬레이브 송신의 중간에 로드된 값을 기록하면 무시될 것이다.

[0055] **2.14.2 LIN 수신(슬레이브) 전용 소프트웨어 모델**

[0056] 다양한 실시예들에서, 자동화된 LIN 프로토콜을 구비한 UART는 특정 슬레이브-전용 모드들을 위해 수신 소프트웨어 모델을 포함할 수 있다. 송신 소프트웨어 모델은 특정 신호를 어서트함으로써 설정될 수 있다. 예를 들면, 핀 신호(예를 들어, P3H/L)는 수신할 바이트들의 소망 개수로 설정될 수 있다. 일부 실시예들에서, UART는 BREAK가 라인에 나타나는 것을 기다릴 수 있다. 이것이 수행될 때, 샘플링 클록은 동기 프레임과 동기화된다. 수신 인터럽트 충돌 신호(예를 들어, RXBIF)는 플래그되고, 체크섬 및 패리티 계산기들은 리셋된다. 일부 실시예들에서, 수신된 제1 바이트는 PID이다. 워터마크/FIFO 설정에 관계없이 PID가 수신될 때에는, 수신 인터럽트(예를 들어, int_rx)는 항상 플래그된다. UART는 다른 값들(예를 들어, P0 및 P1)을 자동으로 계산하고 검증할 수 있다. 만일 이들이 일치하지 않으면(disaree), 충돌 인터럽트 신호(예를 들어, PERIF)는 플래그될 수 있다. 핀 신호(예를 들어, P3H/L)로부터의 바이트들의 개수가 수신될 때까지, 착신(incoming) 바이트들은 수신된다.

[0057] 일부 실시예들에서, 특정 비트(예를 들어, COEN)는 패킷의 마지막 바이트의 시작 비트가 수신되기 전에 정확한 값으로 설정되어야 한다. 수신 워터마크 인터럽트 신호(예를 들어, int_rx)는 워터마크에 도달할 때마다 인터럽트를 트리거할 것이다. (예를 들어, P3H/L로부터) 데이터 바이트들의 소망 개수가 수신되면, UART는 체크섬(예

를 들어 체크섬 0 또는 1)을 자동으로 계산할 수 있고 전부 1을 얻기 위해 수신된 체크섬과 XOR 연산할 수 있다. 국지적으로 계산된 체크섬은 메모리 부분(예를 들어, UxRXCHK 레지스터)에 저장될 수 있다. 수신 입력부로부터 수신된 체크섬은 마치 데이터인 것처럼 수신 버스에 저장될 수 있다.

[0058] 도 5는 본 개시의 특정 실시예들에 따른, LIN 마스터/슬레이브 모드의 예시적인 타이밍도를 도시한다. 도 6은 본 개시의 특정 실시예들에 따른, LIN 마스터/슬레이브 모드의 예시적인 타이밍도를 연속하여 도시한다. 도 7은 본 개시의 특정 실시예들에 따른, LIN 슬레이브 전용 모드의 예시적인 타이밍도를 도시한다. 도 8은 본 개시의 특정 실시예들에 따른, LIN 슬레이브 전용 모드의 예시적인 타이밍도를 연속하여 도시한다.

[0059] **비동기 어드레스 검출 모드**

[0060] 일부 실시예들에서, UART는 비동기 어드레스 검출 모드로 동작할 수 있다. 비동기 어드레스 검출 모드는 하나의 버스에 복수의 수신기가 연결된 모드에 사용될 수 있다. 송신기가 이 버스에 송신할 때, 모든 수신기가 어드레스지정되고 있는지 여부를 아는 것이 유용하다. 이것으로 인해, 각각의 수신 UART는 자체의 CPU로의 불필요한 인터럽트들을 제거할 수 있다.

[0061] 일부 실시예들에서, 각각의 트랜잭션은 송신기가 시작 비트(그에 앞선 BREAK를 갖거나 갖지 않음)를 송신함과 함께 시작한다. 이 시작 비트에 데이터 워드가 뒤따른다. 만일 데이터의 제9 비트가 0으로 설정되어 있으면, 모든 수신 UART들은 이것을 데이터로 감지할 것이다. 송신기가 제9 비트가 1로 설정된 바이트를 송신할 때에는, 모든 수신 UART들은 이것을 어드레스로 감지할 것이다. 이어서, UART들은 그것을, 제2 메모리 부분(예를 들어, 레지스터 P3)의 마스크와 AND연산된(ANDed) 메모리 부분(예를 들어, P2)에 프로그램된 어드레스와 매칭을 시도할 수 있다. 만일 매칭되면, 데이터는 수신 버퍼에 클로킹(clocking)된다. 송수신된 각각의 바이트는 USE_FIFO=0일 때 인터럽트 이벤트를 생성할 것이다. USE_FIFO=1일 때, 인터럽트 주파수는 워터마크 설정에 의존한다.

[0062] **어드레스 검출 TX 소프트웨어 모델**

[0063] 다양한 실시예들에서, UART는 어드레스 검출 송신 소프트웨어 모델을 포함할 수 있다. 일부 실시예들에서, 이 모델은 MODE 신호를 설정함으로써 설정될 수 있다. 예를 들면, MODE는, 어드레스 검출(0101)을 이용하여 비동기 9-비트 모드를 선택하도록 설정될 수 있다. 일부 실시예들에서, UART는 BREAK 송신을 유발할 수 있다. 예를 들면, UART는 SENDB 비트에 로직 1을 기록할 수 있다. SENDB 직후에 메모리 부분(예를 들어, P1 레지스터)에 기록하면, BREAK가 송신될 것이다. 일단 BREAK 송신이 완료되면, SENDB 비트는 자동-소거될 수 있다. 이것 뒤에는 송신될 메모리 부분(예를 들어, P1)의 8-비트 값이 이어질 수 있다. 제9 비트는 자동으로 1로 송신될 수 있다. 이것은 어드레스 송신이다. SENDB 직후에 송신 버스 레지스터에 기록하면, BREAK가 송신될 것이다. 일단 BREAK 송신이 완료되면, SENDB 비트는 자동 소거될 것이다. 이것 뒤에는 송신될 송신 버스 값의 8-비트 값과 자동으로 0으로 송신될 제9 비트가 뒤따른다. 이것은 데이터 송신이다. BREAK 직후에 무엇을 송신할지에 대한 결정은 메모리 부분(예를 들어, P1 레지스터)이 기록되는지 여부에 의존한다. 만일 메모리 부분(예를 들어, P1)과 송신 버스 둘 다가 기록되어 있으면, BREAK가 송신되고, 이어서 메모리 부분(예를 들어, P1)의 컨텐츠가 송신되고 그리고 송신 버스의 임의의 컨텐츠가 뒤따른다.

[0064] 만일 BREAK 송신을 원하지 않으면, SENDB 비트는 무시되거나 0으로 기록될 수 있다. 메모리 부분(예를 들어, P1 레지스터)에 기록하면 메모리 부분(예를 들어, P1)의 컨텐츠의 8-비트 값이 자동으로 1로 송신될 제9 비트와 함께 송신될 것이다. 이것은 어드레스 송신이다. 송신 버스 레지스터에 기록하면, 송신 버스의 8-비트 값이 자동으로 0으로 송신될 제9 비트와 함께 송신될 것이다. 이것은 데이터 송신이다. 먼저 송신해야 할 것에 대한 결정은 메모리 부분(예를 들어, P1 레지스터)이 기록되는지 여부에 의존한다. 만일 메모리 부분(예를 들어, P1)과 송신 버스 둘 다가 기록되면, 메모리 부분(예를 들어, P1)의 컨텐츠가 송신되고 그리고 송신 버스의 임의의 컨텐츠가 뒤따른다.

[0065] **어드레스 검출 RX 소프트웨어**

[0066] 다양한 실시예들에서, UART는 어드레스 검출 수신 소프트웨어 모델을 포함할 수 있다. 일부 실시예들에서, 이 모델은 MODE 신호를 설정함으로써 설정될 수 있다. 예를 들면, MODE는, 어드레스 검출(0101)을 이용하여 비동기 9-비트 모드를 선택하도록 설정될 수 있다. 일부 실시예들에서, UART는 매칭시킬 어드레스를 이용하여 제1 메모리 부분(예를 들어, P2)을 설정할 수 있다. 일부 실시예들에서, 제2 메모리 부분(예를 들어, P3)은 어드레스 마스크로서 설정될 수 있다. 예를 들면, 모든 어드레스를 수신하면 P3에 '0000 0000'을 기록한다. (수신된 어드레스 XOR P2) AND P3=0000 0000은 '어드레스 매치'를 의미한다.

[0067] 어드레스 매치가 발생할 때, 원래 수신된 어드레스는 수신된 FIFO에 로드된다. 이어서, 뒤따르는 모든 데이터(8-비트 바이트와 0으로 설정된 제9 비트)는 수신 FIFO에 로드된다. 이 프로세스는 제1 메모리 부분(예를 들어, P2)의 컨텐츠와 불일치하는 또하나의 어드레스(예를 들어, 8-비트 바이트와 1로 설정된 제9 비트)가 수신될 때까지 계속된다.

[0068] **비동기 맨체스터 모드**

[0069] 다양한 실시예들에서, 자동화된 프로토콜을 구비한 UART는 비동기 맨체스터 프로토콜을 이용할 수 있다. 이러한 실시예들에서, 특정 프로토콜은 사용자로 하여금 복수의 이용가능한 프로토콜 모드들 중 하나를 선택할 수 있게 하는 복수의 비트들에 의해 선택될 수 있다. 예를 들면, 특정의 환경설정은 "USE_PROT"를 논리 하이로 설정함으로써 맨체스터 프로토콜을 선택할 수 있다. 이 모드는, 모든 비트들이 하드웨어에 의해 맨체스터-인코딩되어 있다는 점을 제외하면, 범용 UART 연결 모드이다. DALI 모드에서 사용된 동일한 하드웨어가 여기에서 재사용될 수 있다.

[0070] 일부 실시예들에서, 사용자는 송신측(transmit direction)에서 송신 워드들을 송신 버스(예를 들어, TXB<7:0>)에 기록할 수 있다. 그리고나서 로우-하이 시작 비트와 이어서 버스의 컨텐츠가 뒤따라 송신될 수 있다. 수신측에서는, 각각의 워드가 수신되는대로, 그것을 수신 버스(예를 들어, RXB<7:0>)에 로드하고 착신 맨체스터로(incoming Manchester)부터 그것을 디코딩한다. 송수신된 각각의 바이트는 USE_FIFO=0일 때 인터럽트 이벤트를 생성할 것이다. USE_FIFO=1일 때, 인터럽트 주파수는 워터마크 설정에 의존한다.

[0071] 도 9는 본 개시의 특정 실시예들에 따른, 송신용 맨체스터 디코딩 방식의 예시를 도시한다. 도 10은 본 개시의 특정 실시예에 따른, 수신용 맨체스터 디코딩 방식의 예시를 도시한다. 도 11은 본 개시의 특정 실시예들에 따른, 수신용 맨체스터 디코딩 방식의 예시를 연속하여 도시한다. 도 12는 본 개시의 특정 실시예들에 따른, 수신용 맨체스터 디코딩 방식의 예시를 계속 연속하여 도시한다. 도 13은 본 개시의 특정 실시예에 따른, 수신용 맨체스터 디코딩 방식의 예시를 더욱 계속 연속하여 도시한다. 일부 실시예들에서, 수신 데이터용 클록 속도는 송신 데이터의 클록 속도와 다를 수 있고; 따라서 BRG에서 나오는 클록들을 카운트하기 위해 별도의 레지스터를 유지해야 할 수도 있다. 이러한 타이밍도들에서 상기 레지스터는 BRG-RX로 지칭될 수 있다. 이것은 내부 레지스터이므로, SFR 공간에서 판독 및 기록이 불가능하다.

[0072] 일부 실시예들에서, UART는 하드웨어를 사용하여 맨체스터 인코딩을 디코딩할 수 있다. 맨체스터 디코딩 방식의 하이에서 로우로의 천이들을 처리하기 위해 소프트웨어를 동작시킬 필요는 없다. 오히려, UART는 복구된 데이터 비트들을 단순히 수신한다. 이것은, 바이트들 간에 중단(break)할 필요 없이, 바이트들의 개수가 (연속하여(back-to-back)) 무제한인 맨체스터 송신들을 가능하게 한다.

[0073] **TX 소프트웨어 모델**

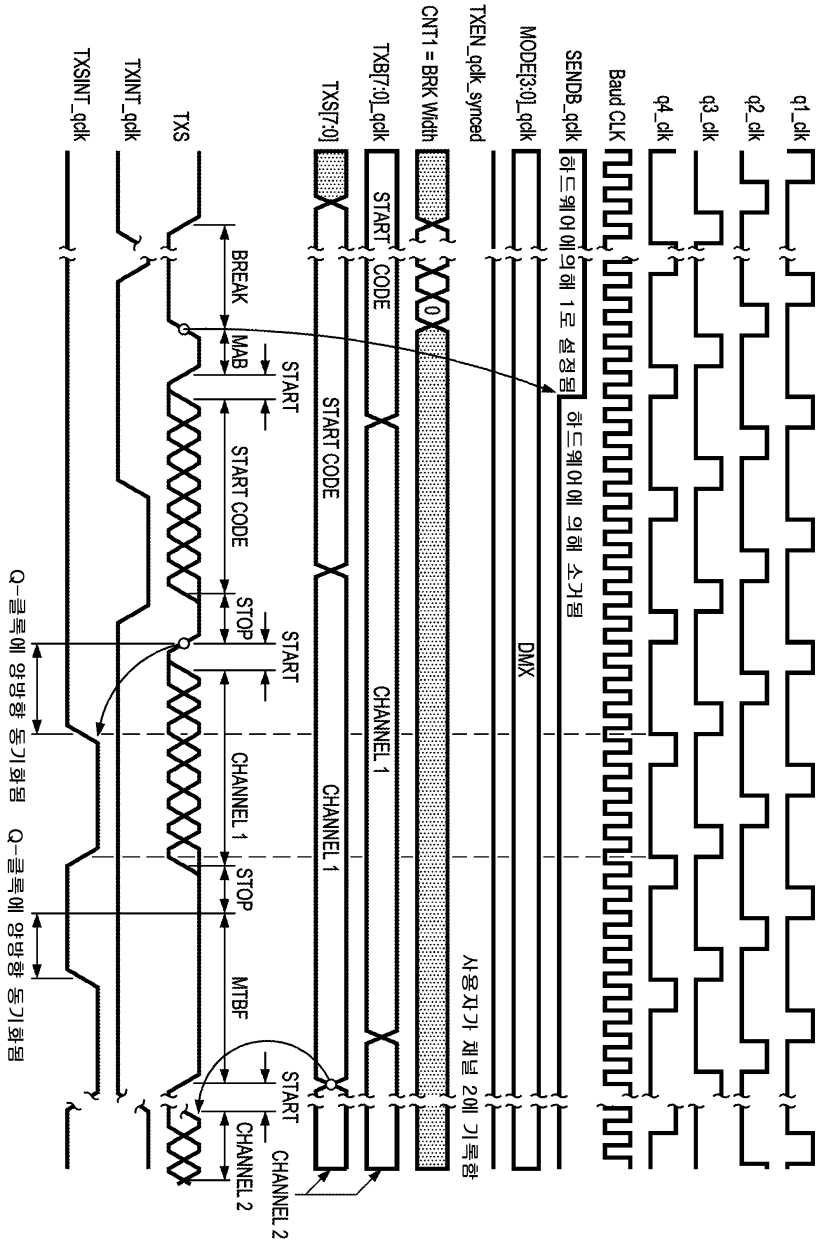
[0074] 다양한 실시예들에서, 자동화된 프로토콜을 구비한 UART는 송신 소프트웨어 모델을 이용할 수 있다. 이러한 실시예들에서, 모델은 UART가 송신될 워드들을 송신 버스에 기록함으로써 개시될 수 있다. 이것으로 인해, START가 출력될 것인데, 이것은 로우에서 하이로의 천이이고, 하나의 바이트의 데이터가 뒤따른다. 송신 워드들이 시프터가 비워지기 전에 송신 버스에 기록되는 한, 비트들은 그들 사이에 START 비트없이 연속하여 송신된다. 이것으로 인해, START 비트 이후에 바이트들의 무제한적인 개수가 연속으로 송신될 수 있다. 시프터가 비워지게 될 때, 송신 라인은 비활성화된다. 송신 버스가 비워진 이후에 새로운 워드가 송신 버스에 기록될 때, UART는 새로운 START 비트를 송신하고 이어서 송신 버스 워드를 송신하며, 사이클은 반복된다.

[0075] **RX 소프트웨어 모델**

[0076] 다양한 실시예들에서, 자동화된 프로토콜을 구비한 UART는 수신 소프트웨어 모델을 이용할 수 있다. 이러한 실시예들에서, 모델은 비활성화(1)와 이어지는 맨체스터 '1'(로우에서 하이)로 개시될 수 있다. 맨체스터 절반 비트 시간(Manchester half bit time)은 제1 로우 천이(the first low transition)로부터 얻어진다. 수신된 모든 데이터는 맨체스터로부터 디코딩되고 수신 FIFO에 로드된다.

## 도면

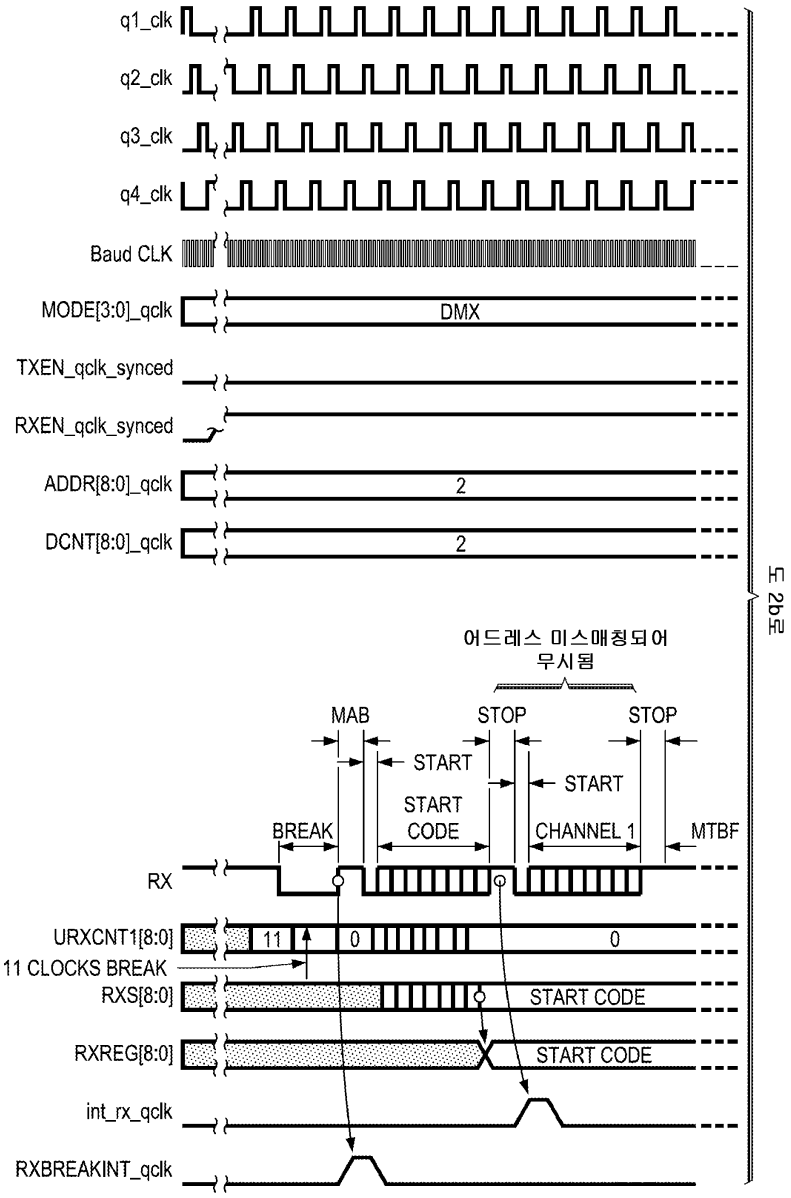
### 도면1



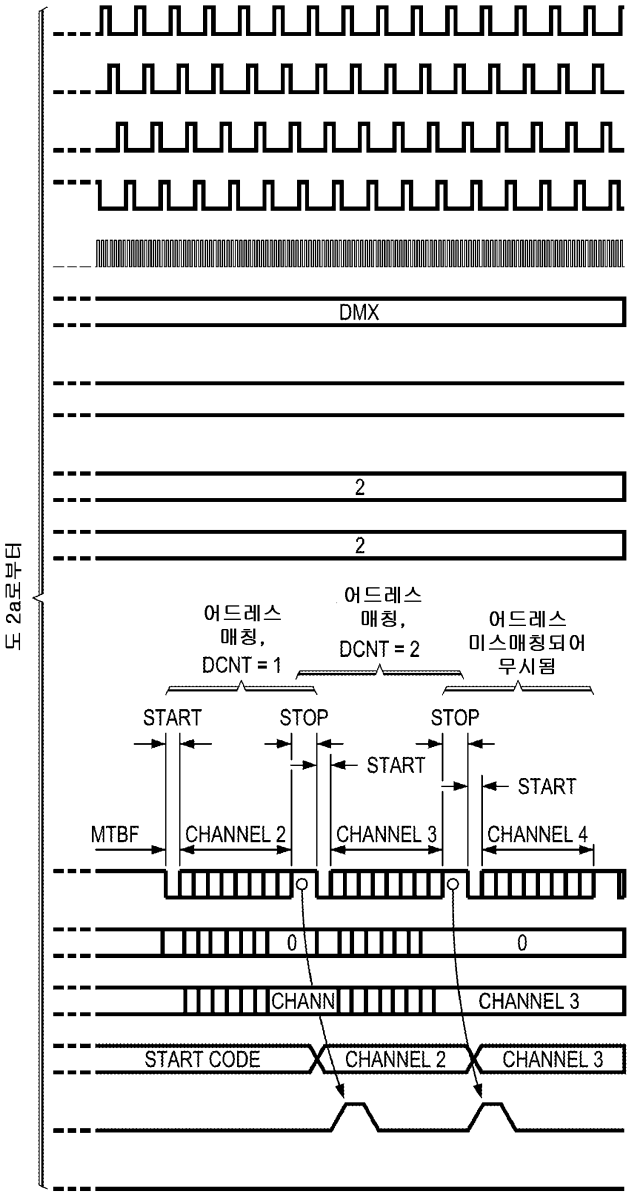

### 도면2

| 도 2a |
|---|
| 도 2b |

도면2a


q1_clk
q2_clk
q3_clk
q4_clk
Baud CLK
MODE[3:0]_qclk
DMX
TXEN_qclk_synced
RXEN_qclk_synced
ADDR[8:0]_qclk
2
DCNT[8:0]_qclk
2
도 2b로
어드레스 미스매칭되어
무시됨
MAB
STOP
STOP
START
START
START
CODE
BREAK
CHANNEL 1
MTBF
RX
URXCNT1[8:0]
11
0
0
11 CLOCKS BREAK
RXS[8:0]
START CODE
RXREG[8:0]
START CODE
int_rx_qclk
RXBREAKINT_qclk

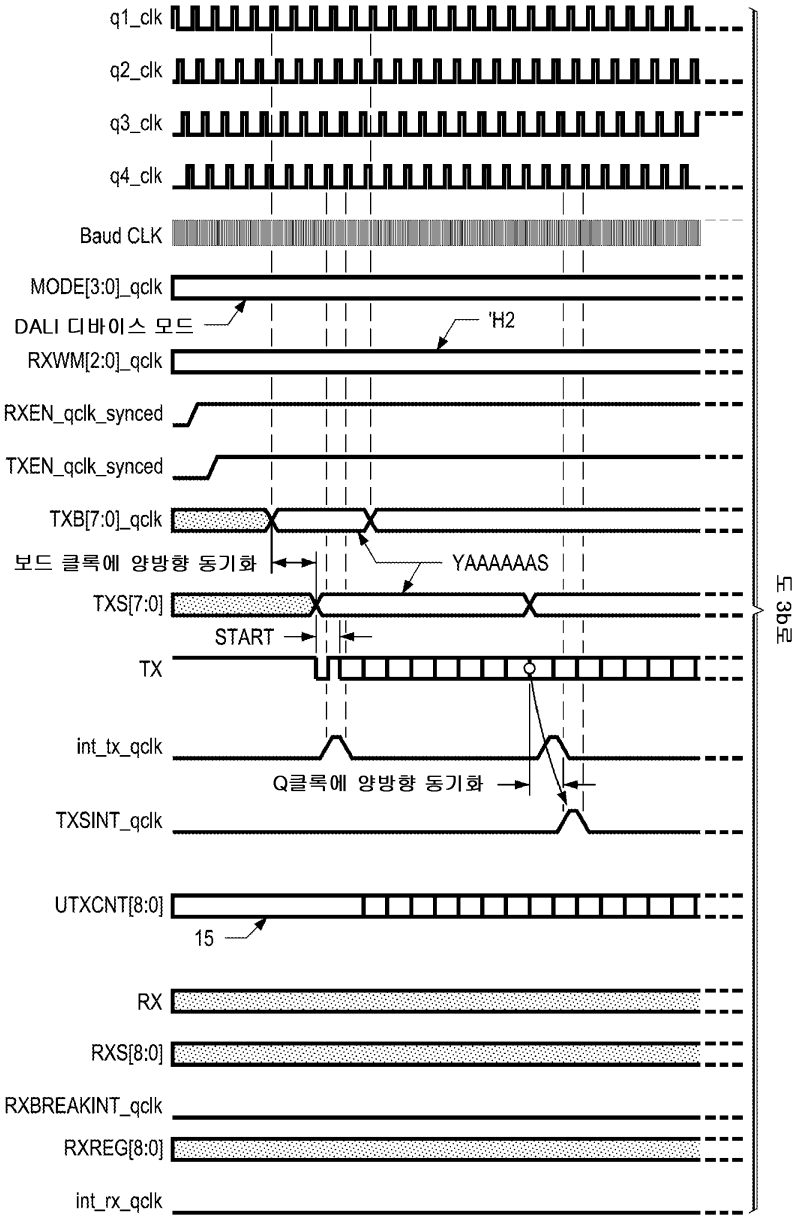
도면2b



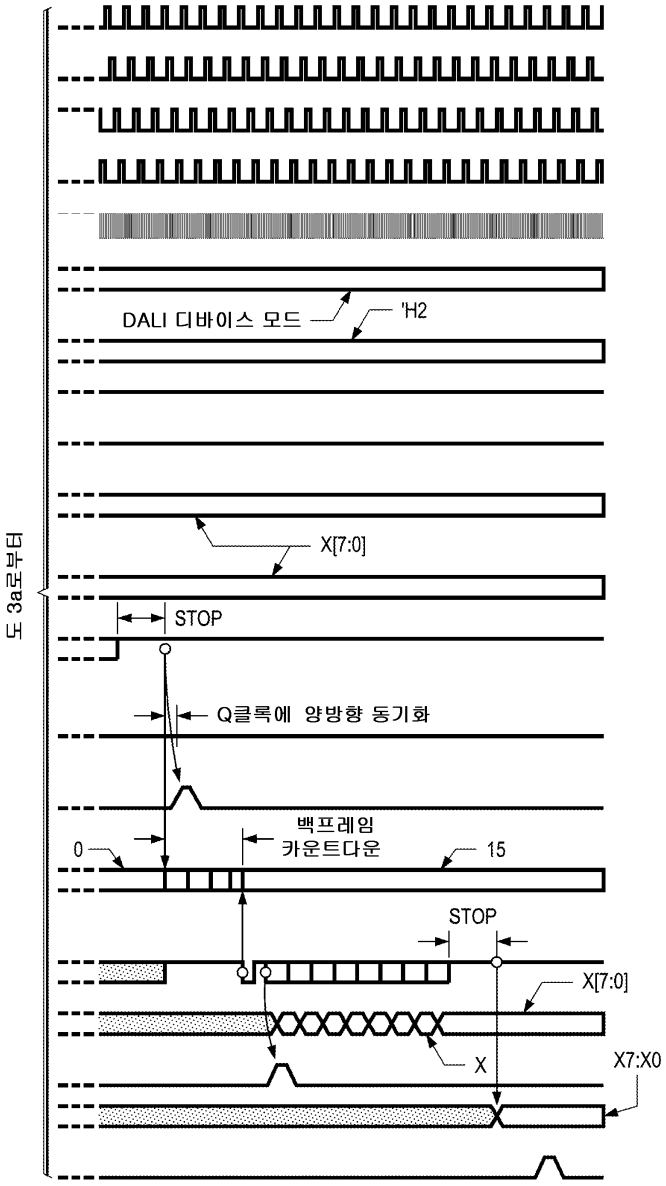

도면3

| |
|---|
| 도 3a |
| 도 3b |

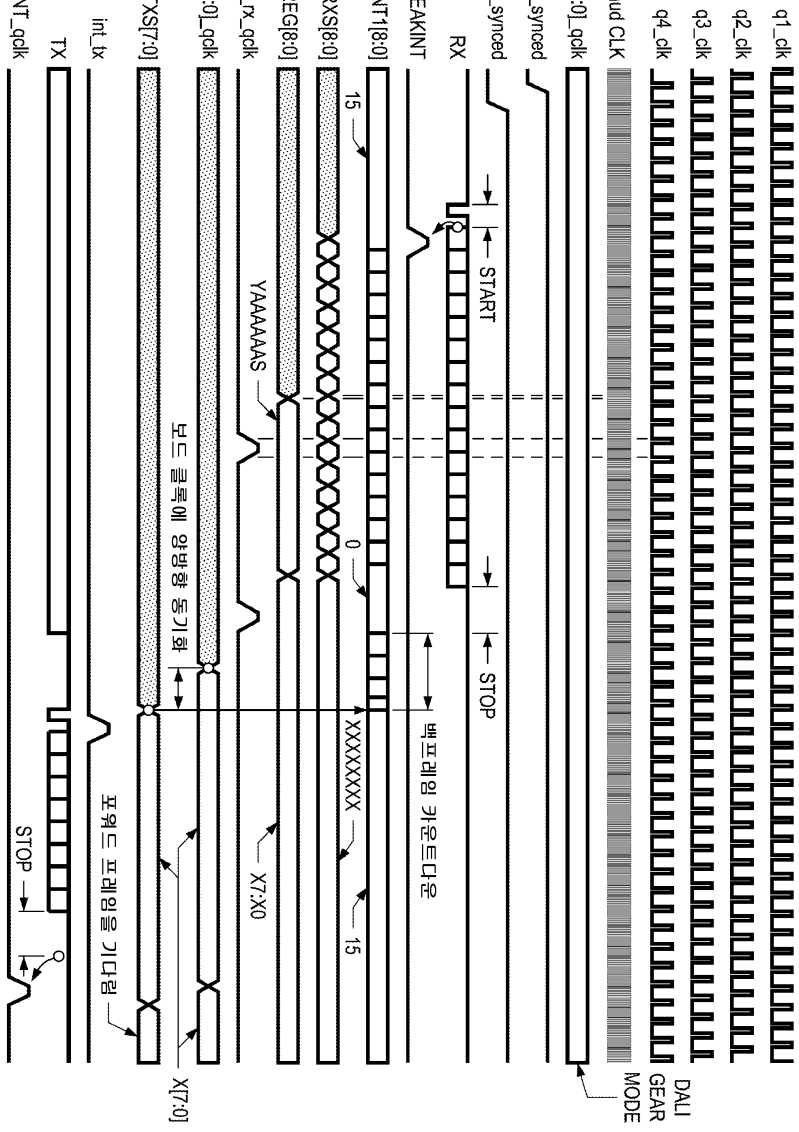
도면3a


q1_clk
q2_clk
q3_clk
q4_clk
Baud CLK
MODE[3:0]_qclk
DALI 디바이스 모드
'H2
RXWM[2:0]_qclk
RXEN_qclk_synced
TXEN_qclk_synced
TXB[7:0]_qclk
보드 클록에 양방향 동기화
YAAAAAAS
TXS[7:0]
START
TX
int_tx_qclk
Q클록에 양방향 동기화
TXSINT_qclk
UTXCNT[8:0]
15
RX
RXS[8:0]
RXBREAKINT_qclk
RXREG[8:0]
int_rx_qclk
도 3b로

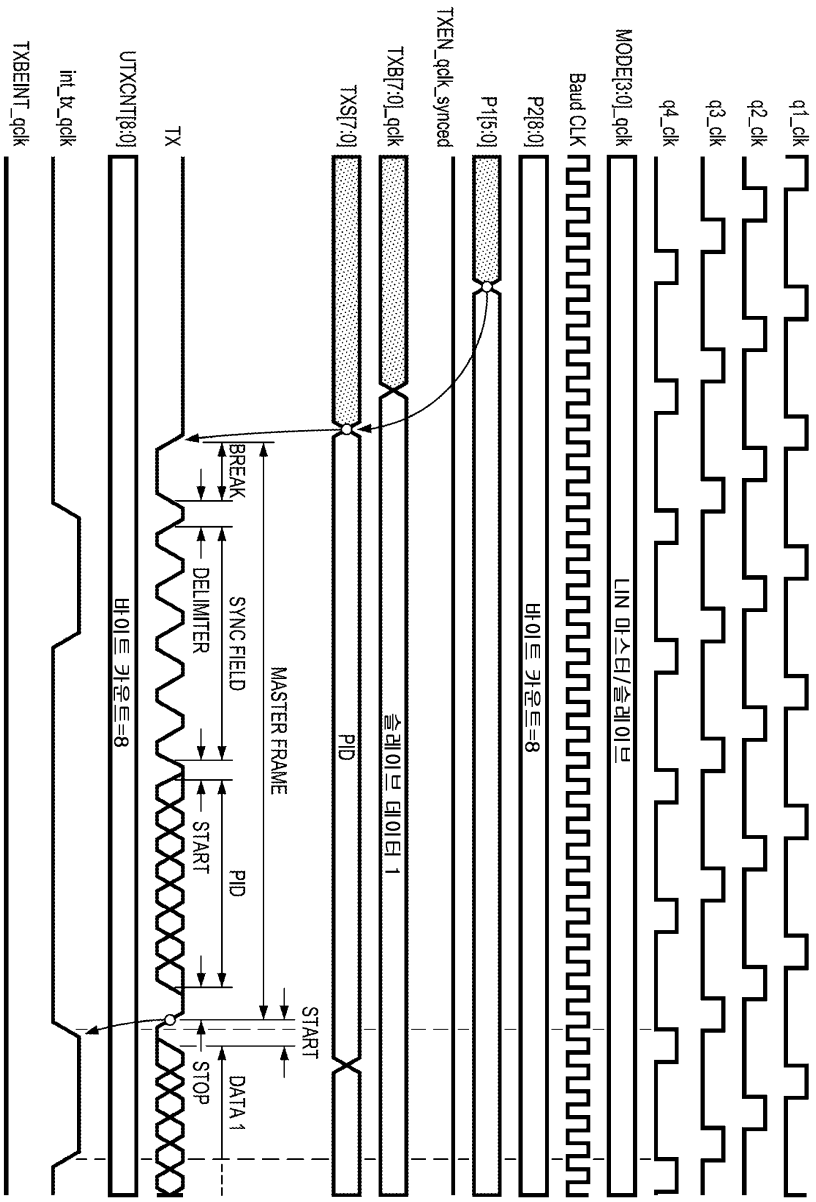
도면3b


도 3a로부터
DALI 디바이스 모드
'H2
X[7:0]
STOP
Q클록에 양방향 동기화
백프레임
카운트다운
0
15
STOP
X[7:0]
X
X7:X0

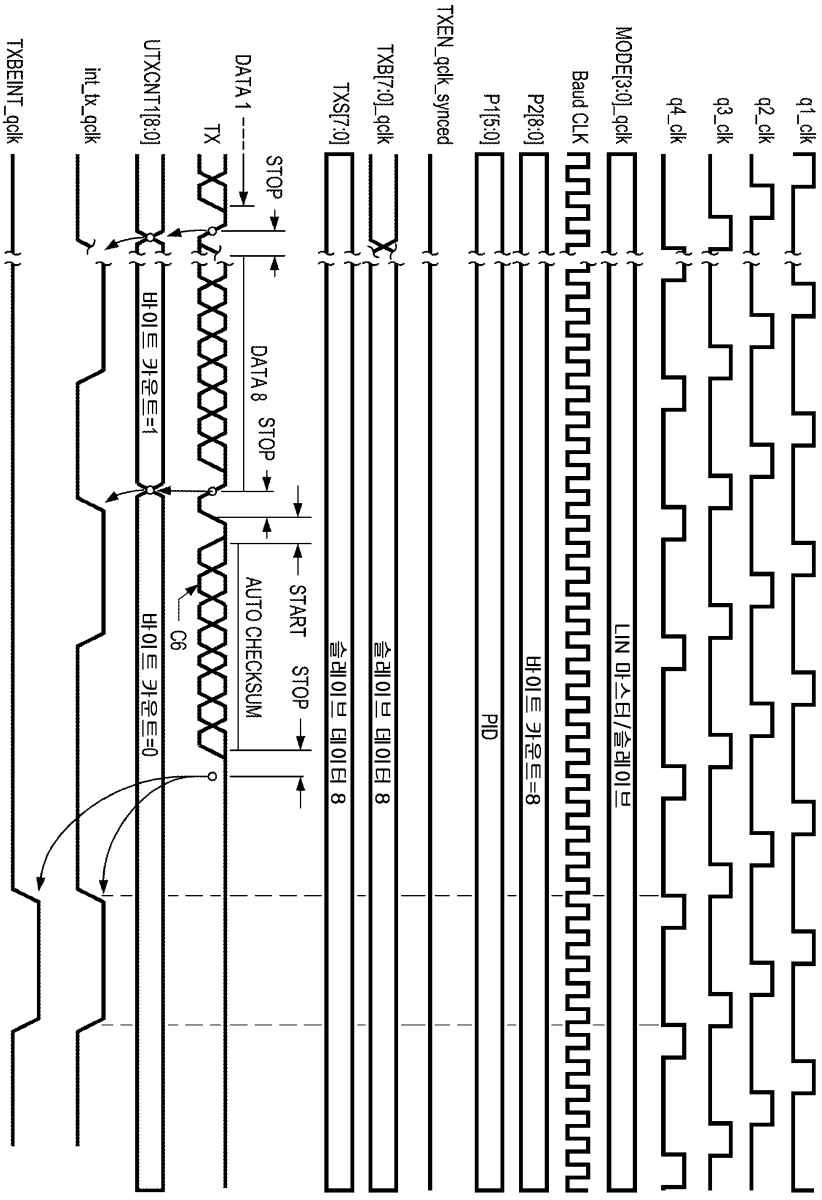
도면4


q1_clk
q2_clk
q3_clk
q4_clk
Baud CLK
MODE[3:0]_qclk
RXEN_qclk_synced
TXEN_qclk_synced
RX
RXBREAKINT
URXCNT1[8:0]
RXS[8:0]
RXREG[8:0]
int_rx_qclk
TXB[7:0]_qclk
TXS[7:0]
int_tx
TX
TXSINT_qclk
DALI GEAR MODE
START
STOP
15
0
XXXXXXXX
15
백프레임 카운트다운
YAAAAAAS
X7:X0
보드 클록에 양방향 동기화
X[7:0]
포워드 프레임을 기다림
STOP

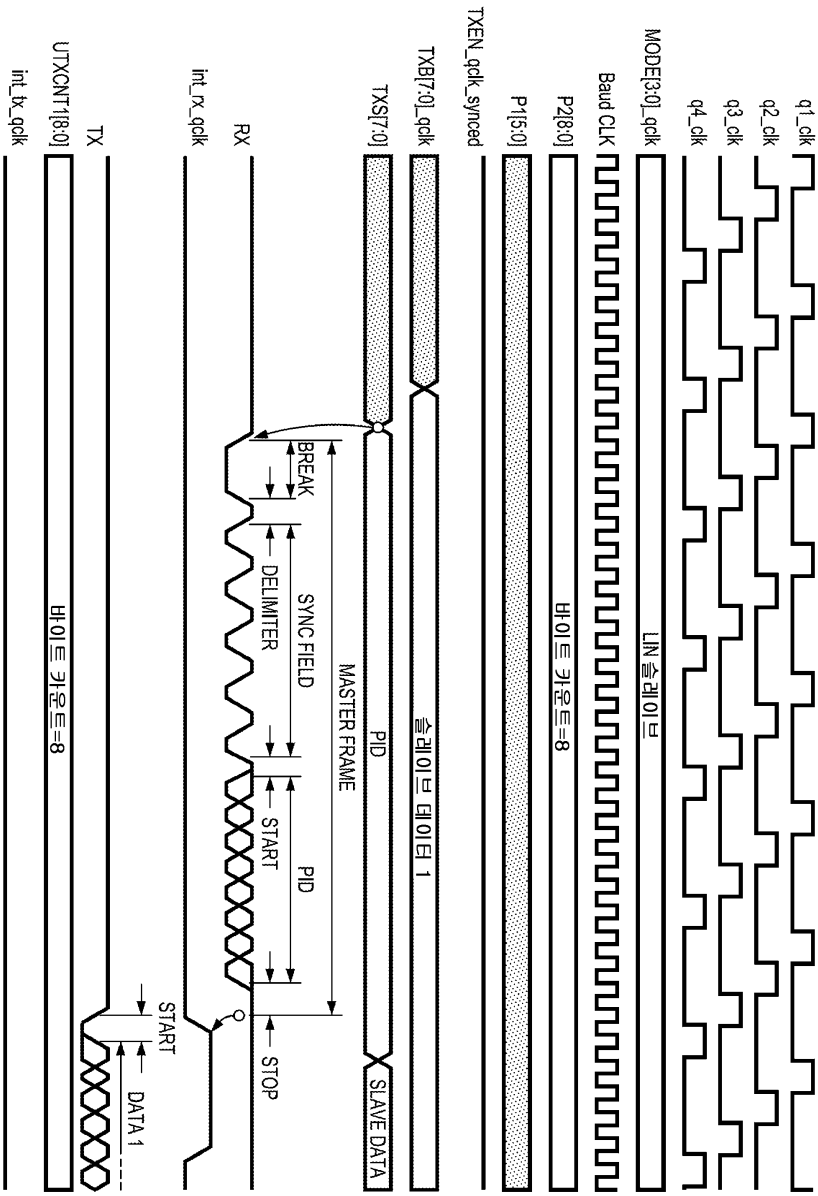
도면5


q1_clk
q2_clk
q3_clk
q4_clk
MODE[3:0]_qclk
LIN 마스터/슬레이브
Baud CLK
P2[8:0]
바이트 카운트=8
P1[5:0]
TXEN_qclk_synced
TXB[7:0]_qclk
슬레이브 데이터 1
TXS[7:0]
PID
TX
BREAK
DELIMITER
SYNC FIELD
START
PID
STOP
DATA 1
MASTER FRAME
START
UTXCNT[8:0]
바이트 카운트=8
int_tx_qclk
TXBEINT_qclk

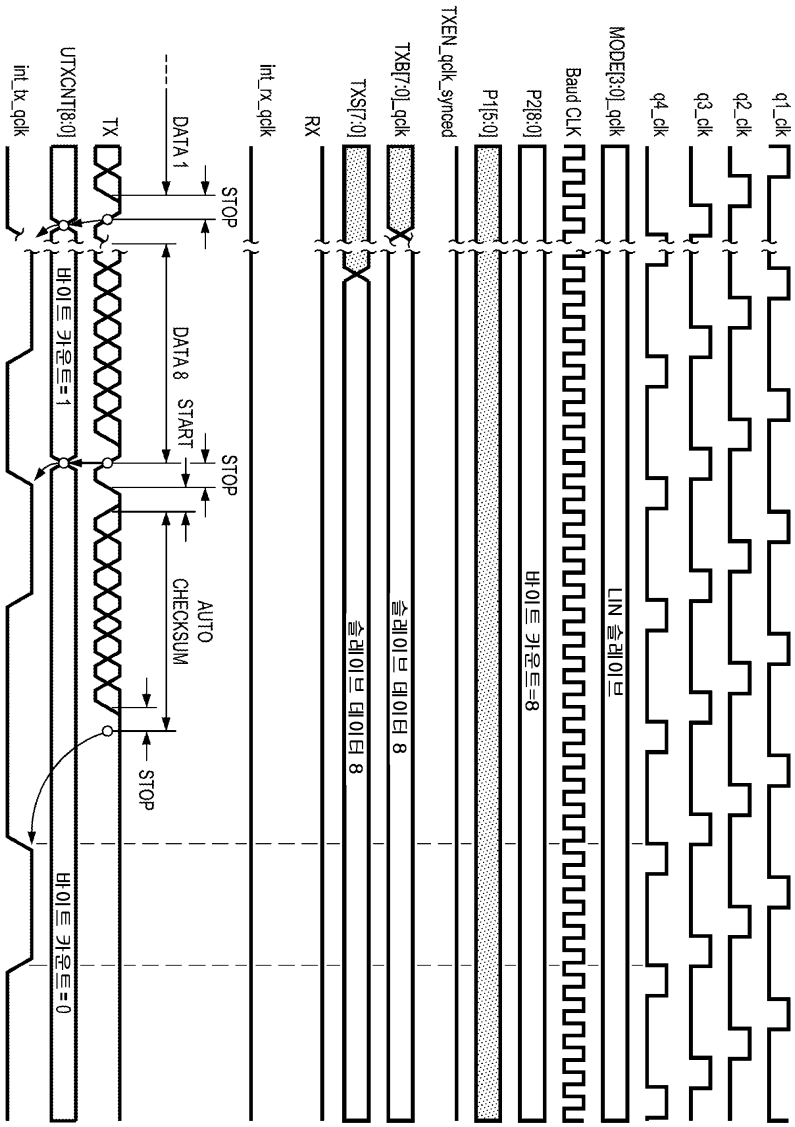
도면6


q1_clk
q2_clk
q3_clk
q4_clk
MODE[3:0]_qclk
Baud CLK
P2[8:0]
P1[5:0]
TXEN_qclk_synced
TXS[7:0]
TXB[7:0]_qclk
TX
UTXCNT1[8:0]
int_tx_qclk
TXBEINT_qclk
LIN 마스터/슬레이브
바이트 카운트=8
PID
슬레이브 데이터 8
슬레이브 데이터 8
DATA 1
STOP
DATA 8
STOP
START
STOP
AUTO CHECKSUM
C6
바이트 카운트=1
바이트 카운트=0

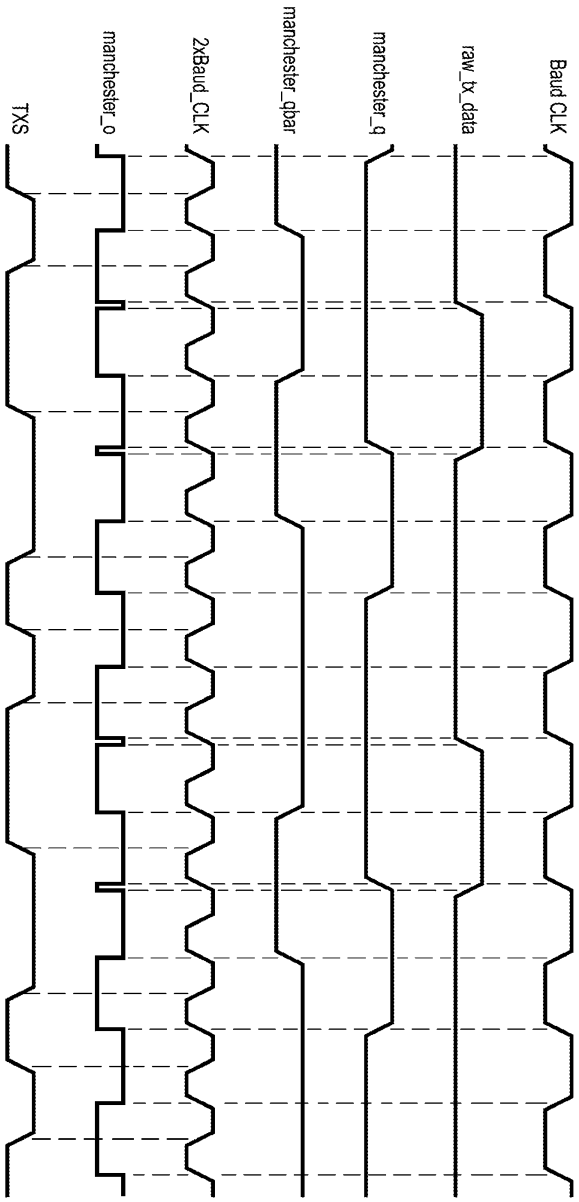
도면7


q1_clk
q2_clk
q3_clk
q4_clk
MODE[3:0]_qclk
LIN 슬레이브
Baud CLK
P2[8:0]
바이트 카운트=8
P1[5:0]
TXEN_qclk_synced
TXB[7:0]_qclk
슬레이브 데이터 1
TXS[7:0]
PID
SLAVE DATA
MASTER FRAME
BREAK
SYNC FIELD
PID
RX
DELIMITER
START
STOP
int_rx_qclk
START
DATA 1
TX
UTXCNT1[8:0]
바이트 카운트=8
int_tx_qclk

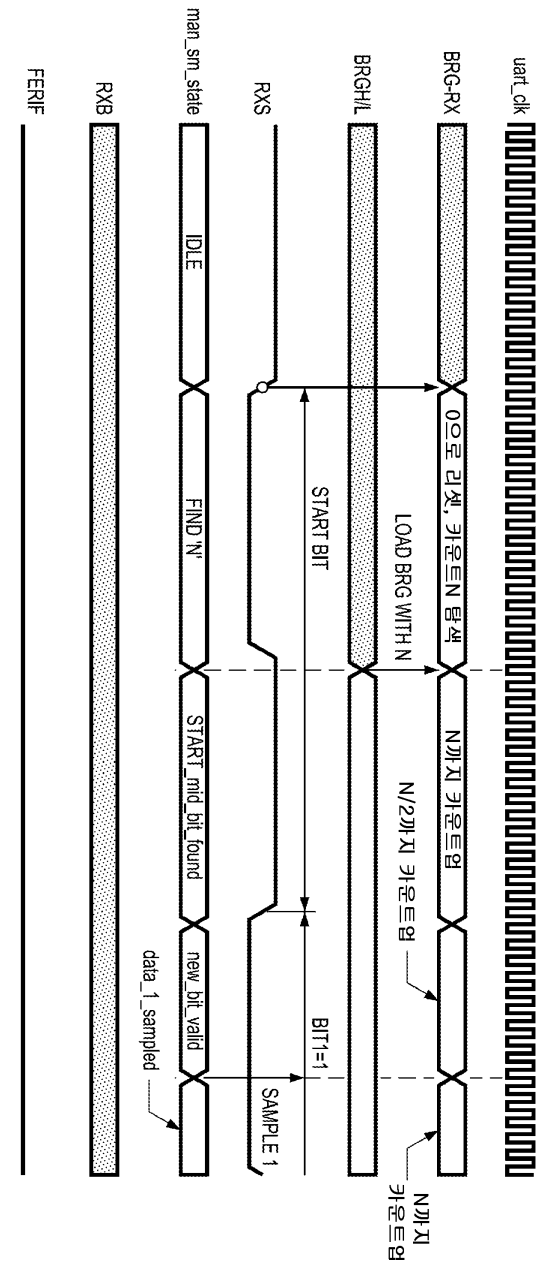
도면8


q1_clk
q2_clk
q3_clk
q4_clk
MODE[3:0]_qclk
Baud CLK
P2[8:0]
P1[5:0]
TXEN_qclk_synced
TXB[7:0]_qclk
TXS[7:0]
RX
int_rx_qclk
TX
UTXCNT[8:0]
int_tx_qclk
LIN 슬레이브
바이트 카운트=8
슬레이브 데이터 8
슬레이브 데이터 8
바이트 카운트=1
바이트 카운트=0
DATA 1
STOP
DATA 8
START
STOP
AUTO
CHECKSUM
STOP

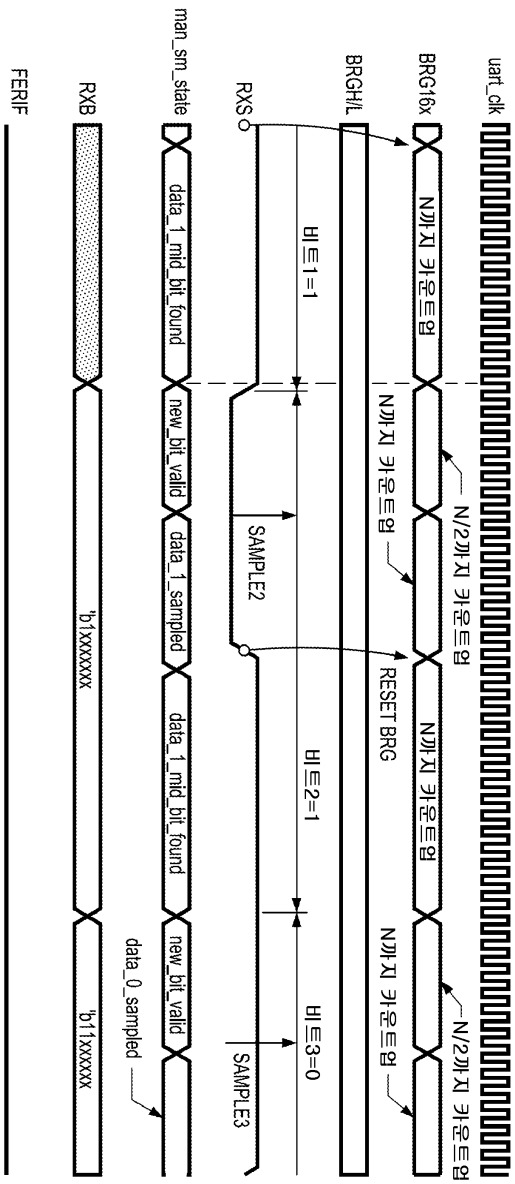
도면9


TXS
manchester_o
2xBaud_CLK
manchester_qbar
manchester_q
raw_tx_data
Baud CLK

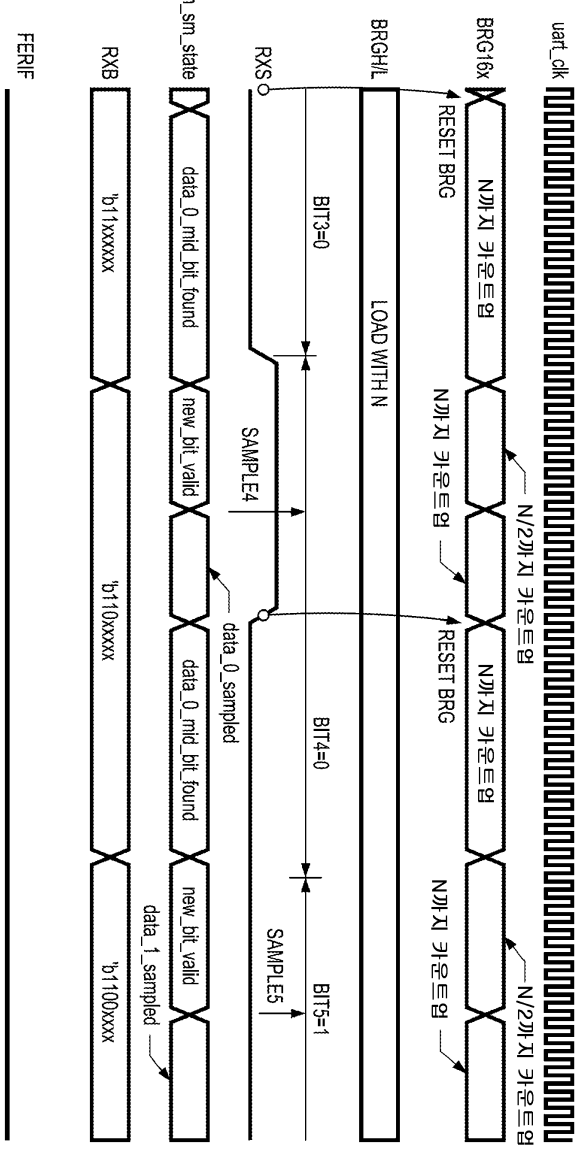
도면10


uart_clk
BRG-RX
BRGH/L
RXS
man_sm_state
RXB
FERIF
0으로 리셋, 카운트N 탐색
LOAD BRG WITH N
N까지 카운트업
N/2까지 카운트업
N까지 카운트업
START BIT
BIT1=1
SAMPLE 1
IDLE
FIND N'
START_mid_bit_found
new_bit_valid
data_1_sampled

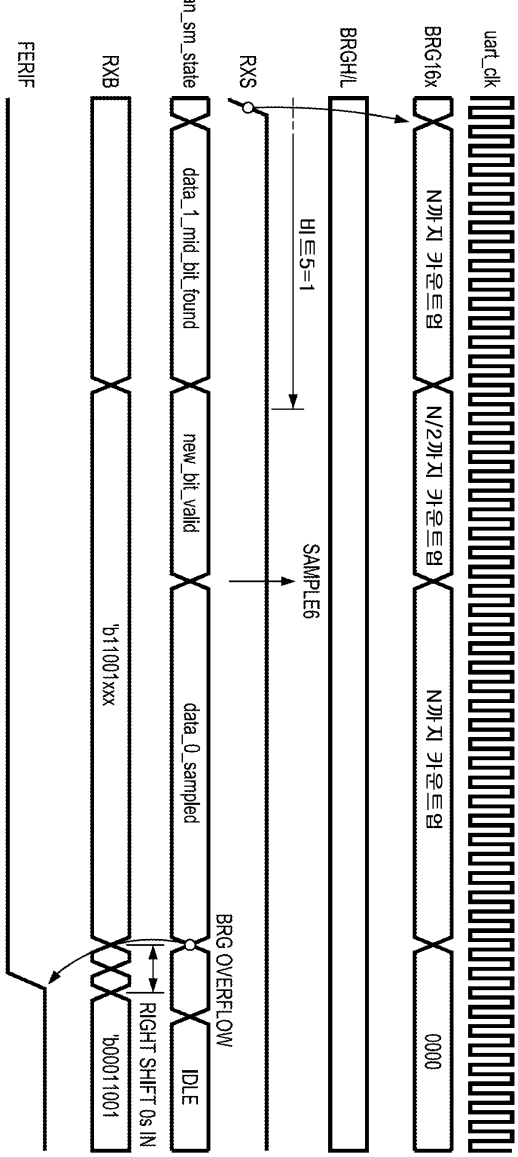
도면11


uart_clk
BRG16x
BRGH/L
RXS
man_sm_state
RXB
FERIF
N까지 카운트업
N/2까지 카운트업
RESET BRG
비트1=1
비트2=1
비트3=0
SAMPLE2
SAMPLE3
data_1_mid_bit_found
new_bit_valid
data_1_sampled
data_0_sampled
'b1xxxxxxx
'b11xxxxxx

도면12


uart_clk
BRG16x
RESET BRG
N까지 카운트업
N/2까지 카운트업
RXS
BRGH/L
LOAD WITH N
BIT3=0
SAMPLE4
BIT4=0
SAMPLE5
BIT5=1
man_sm_state
data_0_mid_bit_found
new_bit_valid
data_0_sampled
data_1_sampled
RXB
'b11xxxxxx
'b110xxxxx
'b1100xxxx
FERIF

도면13


uart_clk
BRG16x
BRGH/L
RXS
man_sm_state
RXB
FERIF
N까지 카운트업
N/2까지 카운트업
N까지 카운트업
0000
비트5=1
SAMPLE6
data_1_mid_bit_found
new_bit_valid
data_0_sampled
IDLE
BRG OVERFLOW
RIGHT SHIFT 0s IN
'b11001xxx
'b00011001