###### 4 Wire Train Bus (WTB)

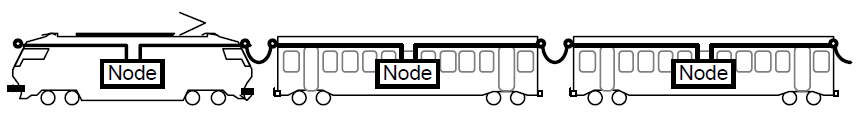
이 조항은 개방 열차에 적용한다. 다른 어플리케이션에서 사용자와 제조 업체 간의 계약 후에 사용할 수도 있다.

WTB에 관한 지침서는 A. 4절에 나와 있다

###### Gen[eral](#_bookmark64)

###### Contents of this clause

이 절은 철도 통신 네트워크의 구성 요소 중 하나인 WTB, 국제 UIC 열차의 경우와 같이 자주 결합되거나 분리되는 차량을 서로 연결하기 위해 주로 설계된 직렬 데이터 통신 버스를 지정한다.



**Figure 111 – Wire Train Bus**

This clause specifies:

1. 단일 회선과 이중 회선 구성의 물리적 매체;
2. 신호와 중복 처리;
3. 전송된 프레임과 텔레그램의 포맷과 타이밍;
4. 버스 트래픽의 조직;
5. 마스터쉽의 할당과 노드의 명명(이노규레이션);
6. 노드 또는 회선 장애 때 복구 절차;
7. 링크 계층 인터페이스와 계층 관리 인터페이스.

###### Structure of this clause

이 표준은 그림 112와 같이 참조 WTB 노드의 계층화된 구조를 따른다:

Subclause 4.1 General

* 규범적 요구 사항과 정의

Subclause 4.2 Physical Layer

* 노드와 노드 구조에 대한 케이블과 연결
* 신호 레벨과 신호 타이밍 사양

Subclause 4.3 Medium-dependent signalling

* 프레임 인코딩과 디코딩
* 물리적 중복 처리

Subclause 4.4 Frames and telegrams

* frame format
* telegram format
* telegram timing

Subclause 4.5 Link Layer Control

* addressing
* telegram types

Subclause 4.6 Medium Allocation

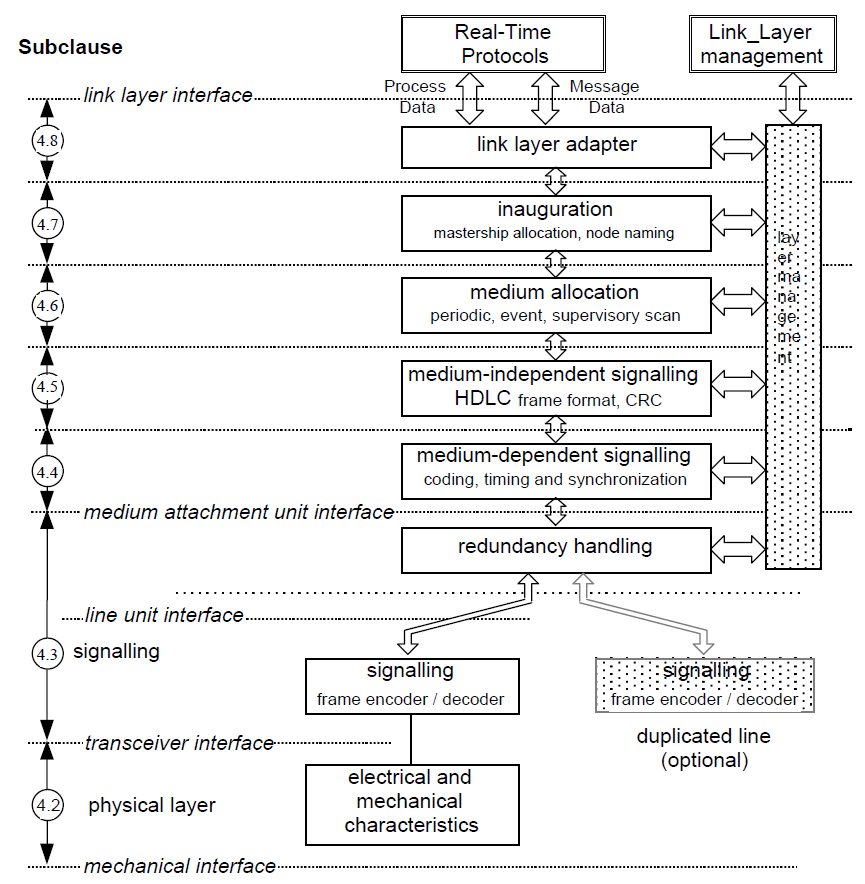
* periodic traffic
* sporadic traffic
* supervisory traffic

Subclause 4.7 Inauguration

* Master selection
* Node naming
* Topography distribution
* Mastership redundancy

Subclause 4.8 Link layer interface

* Process Data
* Message Data
* Layer Management Interface



**Figure 112 – Reference model of the WTB**

* 1. **Physical layer**

이 하위 절에서는 WTB의 물리적 매체로 1.0Mbit/s에서 작동하는 트위스트 페어 와이어 버스를 지정 한다.

이 표준은 다른 섹션, 노드와 커넥터가 신호 전송과 관련하여 가능한 한 균일한 전기 매체를 제공하는 것을 지정 한다.

###### Topology

* + - 1. **Bus sections**

WTB 버스는 다음 유형의 버스 섹션으로 서로 연결된 노드로 구성한다;

1. 차량을 따라 연결된 트렁크 케이블(연속된 차량에는 트렁크 케이블만 있음);
2. 다른 차량의 트렁크 케이블을 연결하는 점퍼 케이블;
3. 노드에 연결하기 위해 트렁크 케이블을 연장하는 연장 케이블.

###### Couplers

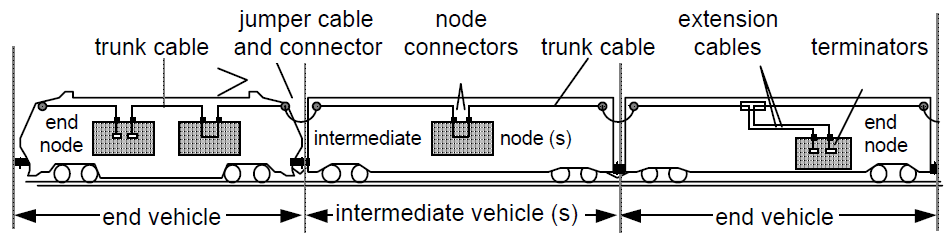
커넥터와 정크션 박스는 노드와 케이블 섹션을 조립하는데 사용할 수 있다. 각 차량은 버스의 일부와 다수의 노드를 운반한다.

###### Nodes

규칙적으로 작동할 때, 각 노드는 트렁크 케이블에 삽입되어 두 개의 버스 섹션에 연결하여야 한다.

1. 버스의 끝에 위치한 노드(종단 노드)는 연결된 두 개의 버스 섹션을 전기적으로 종단 처리해야 한다;
2. 버스 중간에 위치한 노드(중간 노드)는 연결된 두 개의 버스 섹션을 전기적으로 연결해야 한다.

한 노드에 연결된 두 개의 케이블 섹션은 Directtion\_1과 Direction\_2가 된다. 그림 113과 같이 열차에는 차량 당 하나 또는 여러 개의 노드가 있을 수 있다.



**Figure 113 – Train Composition (two Intermediate Nodes shown)**

###### Vehicle orientation

노드의 방향에 좌우 인식이 중요한 경우 다음 규칙을 준수해야 한다:

1. 차량의 한쪽 끝은 Extremity 1, 다른 쪽은 Extremity 2로 식별한다;
2. 노드의 Direction\_1은 Extremity 1에 연결하고, Direction\_2는 Extremity 2에 연결한다;
3. Direction\_1이 북쪽을 가리키면, 서쪽을 가리키는 차량의 측면은 side A로, 동쪽을 가리키는 측면은 side B로 명명한다;
4. 노드는 자신이 위치한 차량과 A, B에 대해 동일한 규칙을 사용한다.

주1: 한 노드의 Direction\_1은 두 노드가 같은 차량에 있는 경우를 제외하고는 다른 노드의 Direction\_1 또는 Direction\_2 중 하나를 가리킬 수 있다. 왜냐하면 서로에 대한 차량의 방향이 예측 가능하지 않기 때문이다.

주2: 차량의 Extremity 1은 주차 브레이크가 있는 차량의 반대편이다.

###### Vehicle specification (informal)

제조 업체는 완전한 열차가 아닌 개별 차량 또는 노드를 제공할 수 있기 때문에 버스, 차량과 노드의 사양은 별도로 처리한다.

이 표준은 전체 버스와 각 노드의 특성을 지정한다. 특정 용도의 경우, 차량 특성을 그로부터 추론한다.

차량 또는 노드의 수를 초과하지 않으면 열차의 적합성을 충족한다.

UIC 리플릿 557에 사용된 다음 계산은 UIC 리플릿 556에 명시한 대로 철도 차량에 적용한다. 대중 교통과 같은 다른 어플리케이션도 비슷한 계산을 사용할 수 있다.

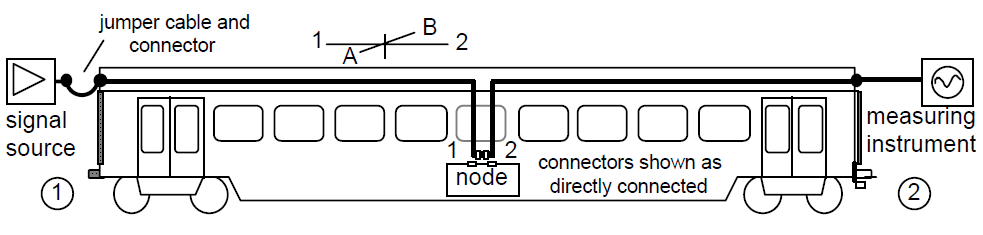
기준 열차 구성은 22대의 차량으로 구성되어 중계기를 사용하지 않고 860.0m의 케이블 길이를 제공한다. 일반적으로 차량 당 하나의 노드만 있지만 최대 10대의 차량이 두 번째 노드를 지원할 수 있으므로 총 32개의 노드를 지원한다.

4.2.5.2에 따르면 노드는 0.3dB(공칭 주파수에서)미만의 신호를 감쇠시키므로 노드로 인한 총 감쇠는 32 x 0.3dB = 9.6dB를 초과하지 않는다.

4.2.6.3에 따르면 수신기는 20.0dB의 동적 범위를 처리할 수 있으며 케이블, 커넥터와 기타 요소에 대한 전체 열차의 잔류 감쇠는 20.0 – 9.6 = 10.4dB를 초과할 수 없다.

따라서 차량에 할당된 최대 감쇠는 10.4dB / 22 = 0.5dB이다.

이 값은 노드가 제거되고, 연결이 단락된 상태에서 측정한다. 그림 114와 같이 측정 때 점퍼 케이블을 고려한다.



**Figure 114 – Vehicle measurement**

우회와 연장 케이블로 인해 차량 당 케이블 길이는 차량 길이의 약 150%이다. 26.0m의 차량 길이를 가정할 때, 매체는 860m(22 x 26.0 x 1.5 = 858.0m)를 초과하는 거리에 있어야 한다.

이러한 요구 사항을 충족시키려면 매체가 10.4dB /860.0m 또는 12.0dB/Km 미만의 감쇠를 나타내야 한다. 점퍼 케이블, 커넥터와 스플라이스는 더 높은 감쇠를 일으킬 수 있으므로 10.0dB/Km보다 감쇠가 적은 트렁크 케이블을 권장한다(4.2.2.4).

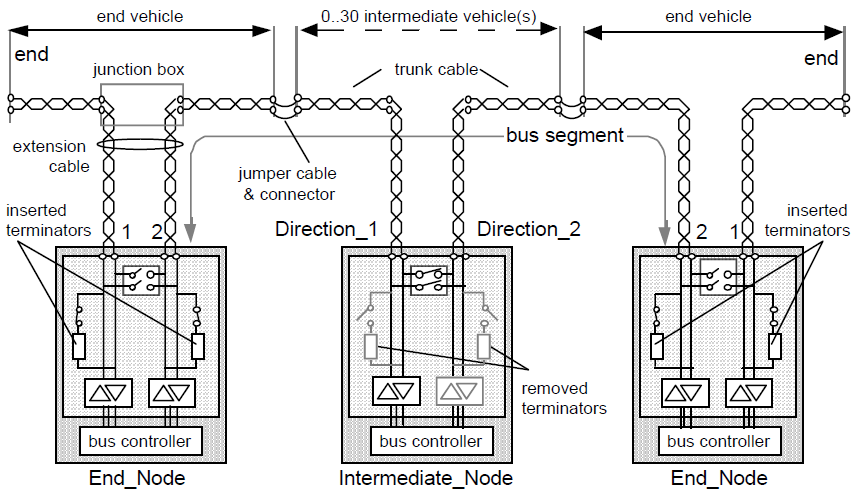
동일한 원리가 다른 왜곡 파라메터에도 적용 가능하다. 측정 방법은 4.2.5.2.1에서 설명한다.

차량에 여분의 Line\_A와 Line\_B를 장착하면 테스트는 각 라인에 개별적으로 적용한다.

###### Medium specifications

###### Topology

노드는 WTB 케이블에 삽입해야 하며, 각 노드는 그림 115처럼 두 개의 버스 섹션에 연결한다.



**Figure 115 – Connected nodes in regular operation**

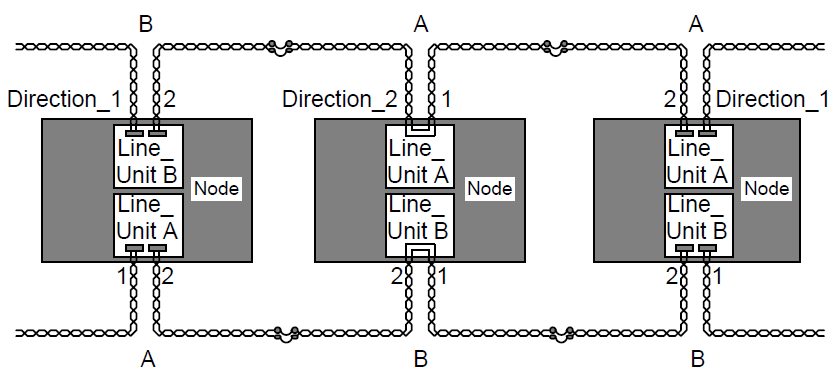
노드는 다음 중 하나를 수행할 수 있어야 한다:

1. 연결된 두 개의 버스 섹션 간에 전기 연속성을 설정하고, 중간 노드 또는
2. 터미네이트(임피던스 적응 네트워크)를 통해 연결된 버스 섹션을 전기적으로 종단 처리하여 종단 노드로 수행.

종단 노드는 양쪽 버스 섹션을 독립적으로 송수신할 수 있어야 하며 중간 노드는 송수신기 중 하나만 활성화해야 한다.

###### Duplicated medium (option)

이 표준은 그림 116과 같이 각 노드가 독립적인 회선 단위를 통해 두 개의 회선에 연결되는 중복 체계를 정의한다.



**Figure 116 – Double-line attachment**

이 옵션을 사용할 경우 다음 사양을 적용하여야 한다:

1. 회선은 Line\_A와 Line\_B로 식별하여야 한다;
2. 이 식별은 동일 차량 내의 모든 노드에 대해 일관성이 있어야 한다;
3. 다른 회선에 속하는 케이블은 뚜렷하게 표시하여야 한다;
4. Line\_A와 Line\_B는 Direction\_1과 Direction\_2에 대해 동일하게 구성하여야 한다.

주1: UIC 차량의 경우, 차량 사이에 단 하나의 선로만 연결할 수 없기 때문에 이중선 매체가 의무적이다.

주2: Line\_A는 차량의 side A와 연관되고 Line\_B는 side B와 관련된다.

주3: 차량의 방향을 예측할 수 없으므로 그림 116과 같이 한 차량의 Line\_A가 다른 차량의 Line\_B에 연결할 수 있다.

###### Bus Configuration rules

이 사양은 예상되는 최대 범위에서 작동하는 버스에 적용한다.

달리 명시하지 않는 한 모든 전기 값은 ± 4,0V (8,0Vpp)의 차동 진폭을 갖는 1.0MHz ± 0.01% 정현파 신호로 측정해야 한다.

###### Signalling speed

모든 버스 세그먼트는 맨체스터 인코딩으로 인해 1.0MHz의 신호 주파수에 해당하는 1.0Mbit/s ± 0.01%의 동일한 속도로 작동해야 한다(BT = 1.0µs, BR = 1.0MHz).

###### Delay due to nodes and cabling

T\_pd의 경우 버스 상의 종단 간 전송 지연은 두 노드 사이에서 60.0 µs를 초과하지 않아야 한다.

주1: 주어진 어플리케이션에 대한 전송 지연은 T\_pd = (L x 6.0 + R x T\_rd)로 평가될 수 있다.

L은 케이블 길이(트렁크, 확장과 점퍼 케이블)를 미터 단위로 나타낸 것이다;

6.0ns/m는 로드된 전송 라인의 전송 지연을 근사한다;

R은 중계기 수이다;

T\_rd는 중계기의 전파지연이다;

주2: WTB는 중계기 없이 860.0m를 연결할 수 있지만 일부 어플리케이션 프로그램에서는 중계기가 유용할 수 있다.

주3: 이 규격은 4.4.2.2와 4.3.2.2에 규정된 허용 지연을 나타낸다.

###### Attenuation due to nodes and cabling

동일 세그먼트에 위치한 두 노드 사이의 전체 전압 감쇠는 20.5dB를 초과해서는 안되며, 0.5BR에서 2.0BR 사이의 주파수에서 정현파 신호로 측정해야 한다.

주1: 감쇠는 노드 수와 전체 케이블 길이에 비례한다.

주2: 이 규격은 4.2.6.3에 명시된 수신기의 동적 범위와 일치한다.

###### Jitter due to nodes and cabling

최대 확장된 노드 수를 지원하는 터미네이트된 세그먼트는 이상적인 제로크로싱을 기준으로 ± 0.1BT의 에지 지터를 더하지 않아야 한다;

Test conditions:

* 회선은 22.0 ± 10%의 소스 임피던스를 통해 0.0V를 중심으로 4.0Vpp ± 10%의 차동 진폭의 소스로 구동한다;
* 구동 신호는 적어도 511비트의 반복 주기를 갖는 ‘0’과 ‘1’ 맨체스터 기호의 랜덤 신호이다.

주1: 섹션, 스텁, 커넥터 또는 부하 클러스터링 사이의 임피던스 불일치로 인한 간섭과 반사는 제로크로싱 타이밍에서 지터를 유발할 수 있다.

주2: 이 규격은 ISO/IEC 8802-3에서 가져와 4.2.6.3에 규정된 수신기의 허용 가능한 지터와 일치한다.

###### Skew between redundant lines (option)

Line\_A와 Line\_B 사이의 전송 지연의 최대 차이는 두 노드 사이에서 T\_skew = 30.0µs를 초과하지 않아야 한다.

주: 이 규격은 4.3.2.4.1에 규정된 수신기의 허용 가능한 스큐와 일치한다.

###### Cable specification

###### Mechanical

모든 케이블 섹션은 트윈 도체, 트위스트와 차폐, 재킷 케이블로 구성한다. 페어는 미터 당 12회 이상의 트위스트를 가져야 한다.

트렁크 케이블 와이어의 권장 단면적은 0.75mm2 (AWG 18)이다. 점퍼 케이블 와이어의 권장 단면적은 1.34mm2 (AWG 16)이다.

4.2.3.4와 같이 Sub-D 커넥터를 통한 간접 부착을 사용하는 경우, 연장 케이블의 각 전선의 단면은 0.56mm2 (AWG 20)이하이어야 한다.

###### Marking

트위스트 페어의 개별 전선은 X와 Y, 차폐는 S로 표시하여야 한다. 케이블의 개별 전선은 뚜렷하게 표시하여야 하며 모든 연결과 접속 점에서 표시를 유지하여야 한다.

###### Characteristic impedance

모든 버스 섹션은 0.5BR과 2.0BR 사이의 주파수에서 정현파 신호로 측정한 Zw = 120.0Ω(±10%) 의 미분 특성 임피던스를 나타낸다

###### Cable attenuation

케이블은 1.0BR에서 정현파 신호를 10.0dB/Km 미만으로, 2.0BR에서 14.0dB/Km 미만으로 감쇠시켜야 한다

###### Distributed capacitance

케이블의 차동(전선 대 전선) 분배 캐패시턴스는 1.0BR에서 65pF/m를 초과하지 않아야 한다.

###### Capacitive unbalance to shield

차폐에 대한 용량성 불균형은 1.0BR에서 1.5pF/m를 초과하지 않아야 한다.

###### Crosstalk rejection

두 쌍의 전선을 동일한 연장 케이블에 장착하는 경우, 한 쌍에서 다른 한 쌍으로의 신호 거부는 0.5BR에서 2.0BR 범위에서 55.0dB보다 커야 한다.

###### Shield quality

케이블의 전달 임피던스는 20.0MHz에서 IEC60096-1에 규정한 대로 20.0mΩ/m 보다 작아야 한다.

케이블의 차당 전달 임피던스는 IEC60096-1에 규정한 대로 측정한 20.0mΩ/m 보다 작아야 한다.

주: IEC 600OTE IEC60096-1은 주로 동축 케이블을 다루지만, 단락된 트위스트 페어를 내부 도체로 간주하고 차폐를 외부 도체로 간주하여 트위스트 페어로 사용할 수 있다.

###### Connector quality

주: 이 요구 사항은 차량 간의 커넥터에는 적용하지 않는다.

모든 케이블 연결은 전선과 차폐의 연속성을 제공해야 하며, 10.0mΩ보다작은저항을가져야한다

20.0MHz에서 측정된 커넥터의 전달 임피던스는 IEC60096-1에 규정한 대로 측정된 한 핀과 차폐 사이 20.0m보다 작아야 하고, 두 핀 사이 각각 2.0mΩ 미만이어야 한다.

###### Shielding concept

다양한 어플리케이션을 충족시키기 위해 두 가지 차폐 개념을 지정한다:

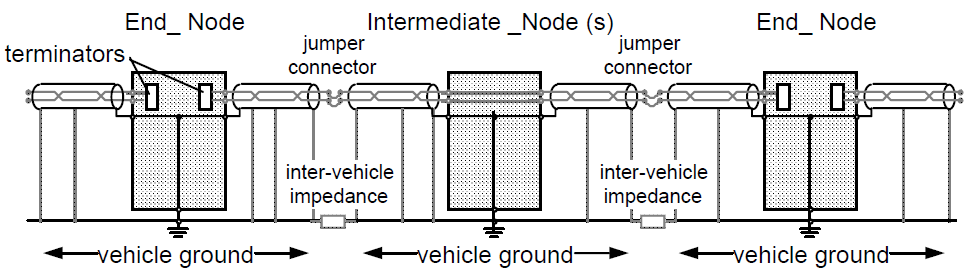
* 접지된 차폐 개념(선호되는 방법)과
* 플로팅 차폐 개념.

###### Grounded shield concept

접지 차폐 개념을 적용할 때:

* 차폐는 각 노드의 노드 접지에 직접 연결하여야 한다;
* 점퍼 케이블은 그림 117과 같이 차량 간의 차폐 연속성을 확보해서는 안 된다.

주: 차폐물은 차량 끝 단, 캐비닛 경계면 등과 같이 가능하면 짧은 경로에서 유도 전류를 루프백시키고 차폐물을 통해 EMC 장애를 일으키지 않도록 가능한 한 접지에 연결해야 한다. 이는 트랙션 장비의 큰 누설 전류를 방지하기 위해 차체의 우수한 전도성을 요구한다.



**Figure 117 – Grounded shield concept**

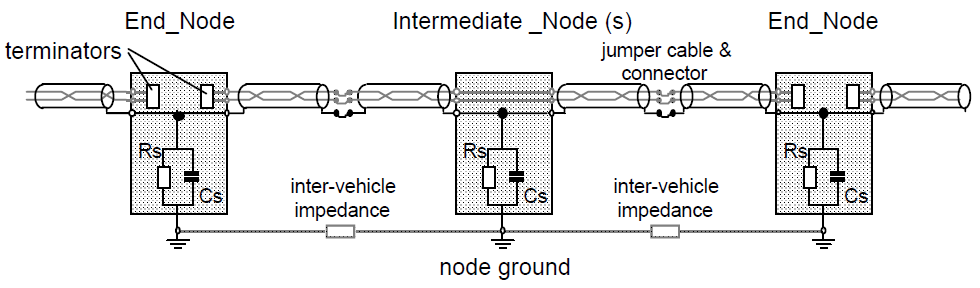
주: 접지 차폐의 개념은 UIC 리플릿 558에 명시되어 있다.

###### Floating shield concept

플로팅 차폐 개념을 적용할 때:

* 차폐물은 노드에 연결되지 않은 경우 지면과 격리하여야 한다;

차폐는 각 노드에서 RC 회로에 의해 노드 접지에 연결하여야 하며, 그림 118과 같이 저항 Rs = 47.0kΩ ±5%와 캐페시터 Cs = 100.0nF ±10% / 750.0 V가 병렬로 연결한다:



**Figure 118 – Floating shield concept**

###### Terminator

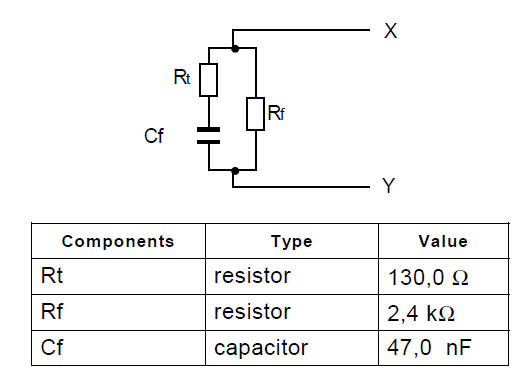
종단 노드는 터미네이터로 연결된 두 버스 세그먼트를 전기적으로 종단시켜야 한다.

터미네이터는 Zw ±5%의 값을 가지며 0.5BR에서 2.0BR의 주파수 범위에서 0.087 라디안 미만의 위상 각을 갖는 비극성 저항이어야 한다.

터미네이터는 케이블 차폐로부터 격리하여야 한다.

터미네이터는 dc에 2.4 kΩ의 저항을 제공해야한다. X와 Y사이에 인가된 전류는 적어도 지속되는 1.0W의 전력을 소모할 수 있다(심지어 프리팅을 필요로 하지 않는 어플리케이션에서도).

예: 권장하는 회로가 그림 119에 있다.



**Figure 119 – Terminator**

###### Medium attachment

다음 부속 절은 노드를 첨부하는 두 가지 방법을 지정한다:

* 직접 연결된 노드는 커넥터없이 트렁크 케이블에 직접 연결한다;
* 간접적으로 부착된 노드는 커넥터를 통해 트렁크 케이블에 연결한다.

###### Node connection points identification

노드는 노드에 부착된 두 개의 버스 섹션을 ‘Direction\_1’과 ‘Direction\_2’로 식별해야 한다.

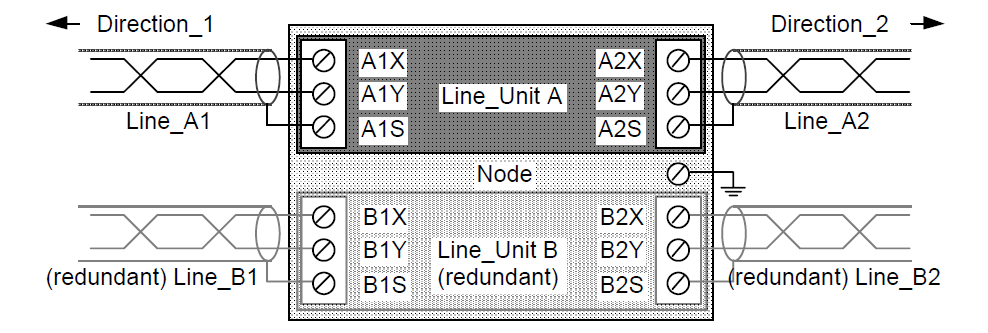
노드는 그것에 붙어있는 두 개의 회선을 ‘Line\_A’와 ‘Line\_B’로 식별해야 한다. 하나의 회선만 사용되는 경우 Line\_A이어야 한다.

케이블 연결 장치는 회선 장치를 가리키며 다음과 같이 명명한다:

1. A1X, A1Y and A1S for Line\_A1 (Line\_A in Direction\_1) and
2. A2X, A2Y and A2S for Line\_A2 (Line\_A in Direction\_2), respectively
3. B1X, B1Y and B1S for Line\_B1 (Line\_B in Direction\_1) and
4. B2X, B2Y and B2S for Line\_B2 (Line\_B in Direction\_2).

###### Direct node attachment

직접 부착된 노드는 케이블에 연결하여야 하며 그림 120과 같이 전기적 요건과 기계적 요건을 충족시키는 나사 또는 기타 고정 장치로 부착해야 한다:

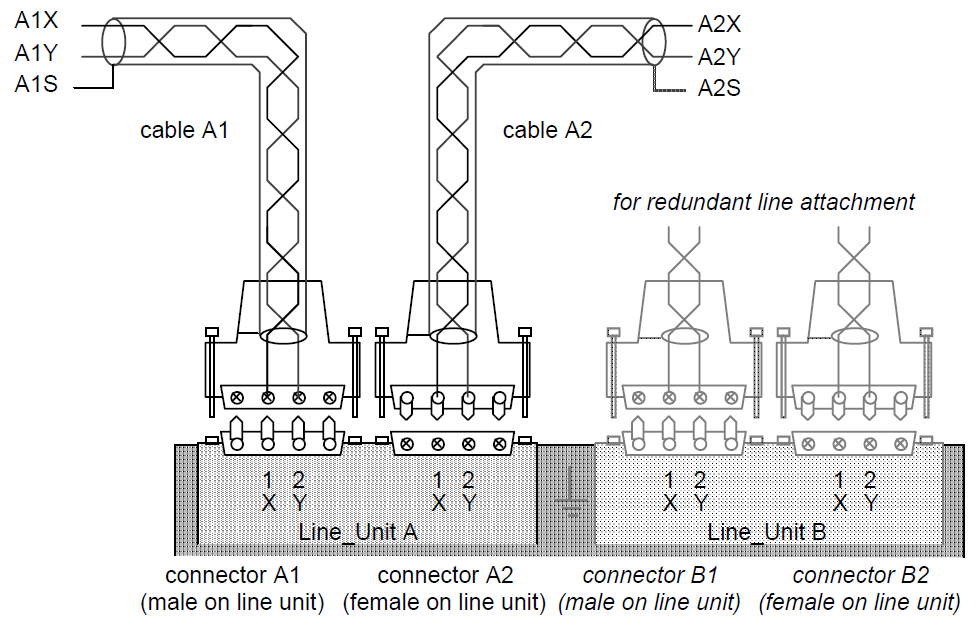


**Figure 120 – Direct node attachment (optional double-line)**

케이블과 차폐 연속성을 제공하기 위해 노드를 제거하고 두 방향의 케이블을 함께 연결할 수 있어야 한다.

###### Indirect node attachment

간접적으로 부착된 노드는 그림 121과 같이 단일 선로 연결 장치에 두 개의 커넥터를 사용하거나 이중 선로 연결 장치에 네 개의 커넥터를 사용해야 한다.



**Figure 121 – Indirect attachment**

* + - 1. **Connector (optional)**

상호 호환성이 요구되는 곳에 간접적으로 부착된 노드는 다음과 같이 케이블에 부착하여야 한다:

1. 커넥터는 초소형 D 커넥터(IEC 60807)이어야 한다;
2. 커넥터는 다음과 같이 차폐된 전도성 케이스를 가져야 한다:
   * 접지 차폐 개념의 경우 이 케이싱은 케이블 차폐에 연결되고 고정되었을 때 콘센트와 전기적 접촉을 해야 한다;
   * 플로팅 차폐 개념의 경우 케이싱은 차폐에서 격리할 수 있다;
3. 커넥터에는 미터 단위 나사가 있어야 한다;
4. 커넥터는 다음과 같은 극성과 배열을 가져야 한다:
   * Direction\_1은 Line Unit의 male 커넥터와 female 케이블을 사용한다;
   * Direction\_2는 female 커넥터와 male 케이블을 사용한다;
   * 같은 선의 커넥터가 수직으로 배치된 경우, Direction\_1은 상부 커넥터, Direction\_2는 하부 커넥터, Line\_A는 상부 쌍이다;
   * 같은 선의 커넥터가 수평으로 배열된 경우, Direction\_1은 왼쪽 커넥터이고, Direction\_2는 오른쪽 커넥터이며, 노드쪽으로 보면 Line\_A가 상위 쌍이 된다;
5. 두 방향의 케이블 커넥터를 연결하고 고정시켜 케이블과 차폐 연속성을 제공할 수 있어야 한다;
6. 그림 122와 같이 커넥터 (male 또는 female)는 표 60에 지정된 핀 할당을 가져야 한다.

**Table 60 – WTB connector pin assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | X positive wire | 6 | reserved |
| 2 | Y negative wire | 7 | reserved |
| 3 | reserved for shield | 8 | reserved |
| 4 | reserved | 9 | reserved |
| 5 | reserved |  | |



1X

reserved 6

2Y

reserved

3 reserved

reserved 8

4 reserved

reserved 9

5 reserved

**Figure 122 – WTB connector, front view**

주: 플로팅 차폐 개념은 소켓 케이스와 소켓을 전기적으로 분리하는 것이 바람직하다.

###### Node specifications

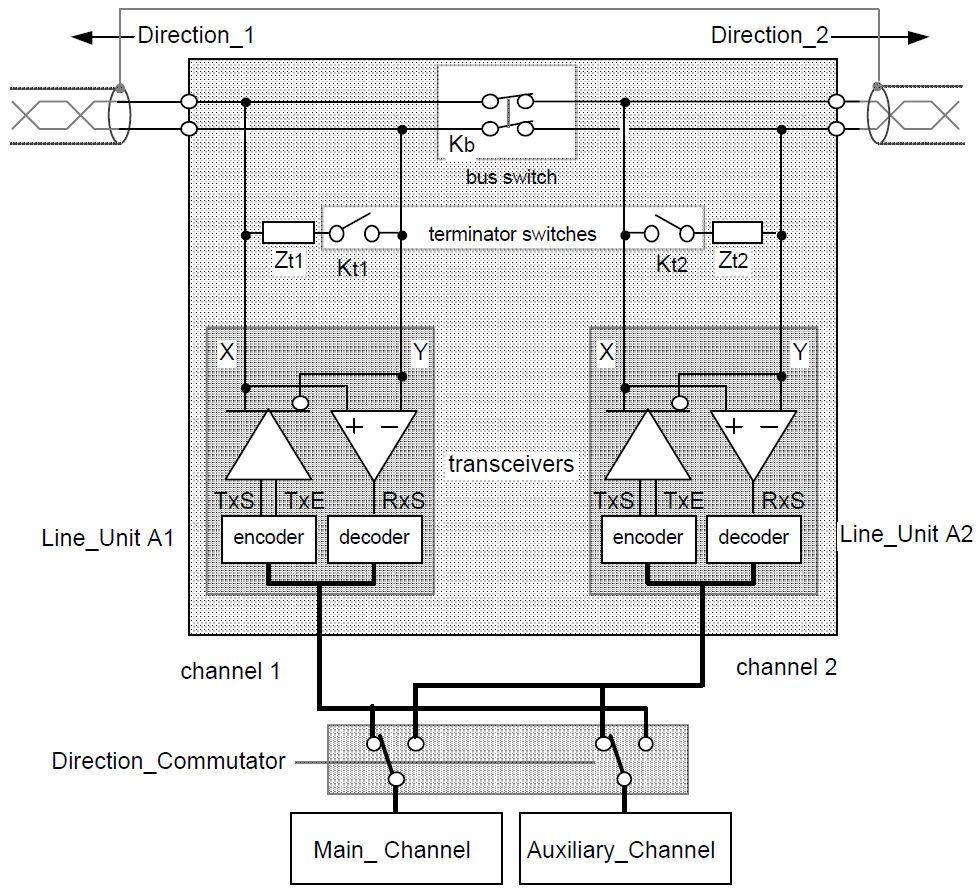
###### Node elements

노드는 매체 접속 장치(Medium Attachment Unit, MAU)를 통해 매체에 부착하여야 한다.

단일 회선 연결을 위한 MAU는 다음을 포함하여야 한다:

1. a Line Unit;
2. a Direction Commutator;
3. a Main Channel and an Auxiliary Channel.

예: 중간 설정 스위치가 있는 MAU가 그림 123에 나와있다.



**Figure 123 – Example of MAU Structure**

###### Line Unit elements

회선 단위는

1. 두 방향을 연결 또는 연결 해제하는 버스 스위치 (Kb);
2. 종단 설정의 노드에 종단기(Zt1, Zt2)를 삽입하거나 중간 설정의 노드에서 종단기를 제거하는 각 방향에 대한 종단기 스위치(Kt1, Kt2);
3. 각 방향에 하나씩 두 개의 송수신기회로. 각 송신기는 바이너리 신호 TxS(signal)와 TxE(enable)로 제어한다. 수신기의 출력은 RxS(디지털 또는 아날로그 신호)이다. 송수신기는 적절한 수단, 예를 들어 트랜스로서 회선으로부터 전기적으로 절연한다;
4. 두 개의 맨체스터 인코더/디코더, 트랜시버마다 하나씩, 각각의 송신기 또는 수신기를 통합할 수 있다. 맨체스터 인코더/디코더의 출력과 입력은 모뎀 인터페이스로 지정한다;
5. 절연 스위치에 연결된 과전압/단락에 대한 보호회로는 그림 123에 나타나있지 않다.

주1: 이중화 매체에 부착된 노드는 두 개의 선로 유닛을 갖는다.

주 2: 스위치 Kb와 Kt1, Kt2는 그러한 기계식 계전기가 사용되는 경우 동일한 기계 계전기의 접점일 수 있다.

###### Main Channel and Auxiliary Channel

주 채널과 보조 채널은 HDLC 프레임과 제어 신호를 회선 유닛과 송수신할 수 있어야 한다.

주: 보조 채널은 추가 노드를 감지하고 그 주소를 수신하기 위한 프레임만 송수신하기 때문에 주 채널에 대한 동작을 단순화할 수 있다.

###### Direction Commutator

방향 정류기는 주 채널을 Direction\_1에 연결하고 보조 채널을 Direction\_2에 연결한다, 또는 그 반대의 경우도 마찬가지이다.

주: 방향 정류기는 반드시 물리적 요소일 필요는 없으며 논리 또는 소프트웨어로 구현할 수 있다.

###### Node and switch settings

이 절에서는 중간 설정 또는 단말 설정의 두 가지 설정에서 노드 특성에 대해 설명한다.

###### Intermediate Setting

노드는 중간 설정에서:

1. 두 선로 구간 사이의 연속성을 확립한다(Kb 닫힘);
2. 터미네이터 Zt1과 Zt2를 모두 제거한다(Kt1과 Kt2가 열림);
3. 주 채널을 트랜시버 1 또는 트랜시버 2의 선로에 연결한다;
4. 보조 채널과 사용하지 않는 트랜시버를 셧다운한다.

주: 중간 설정은 비 작동(비 활성화 또는 전원 꺼짐) 노드와 버스 끝에 있지 않은 노드가 수행한다.

###### End Setting

노드는 종단 설정에서:

1. 두 부분을 분리한다 (Kb는 열림);
2. 두 터미네이터를 연결한다(Kt1과 Kt2가 닫힘);
3. 보조 채널을 한 방향으로 연결하고 주 채널을 다른 방향으로 연결한다.

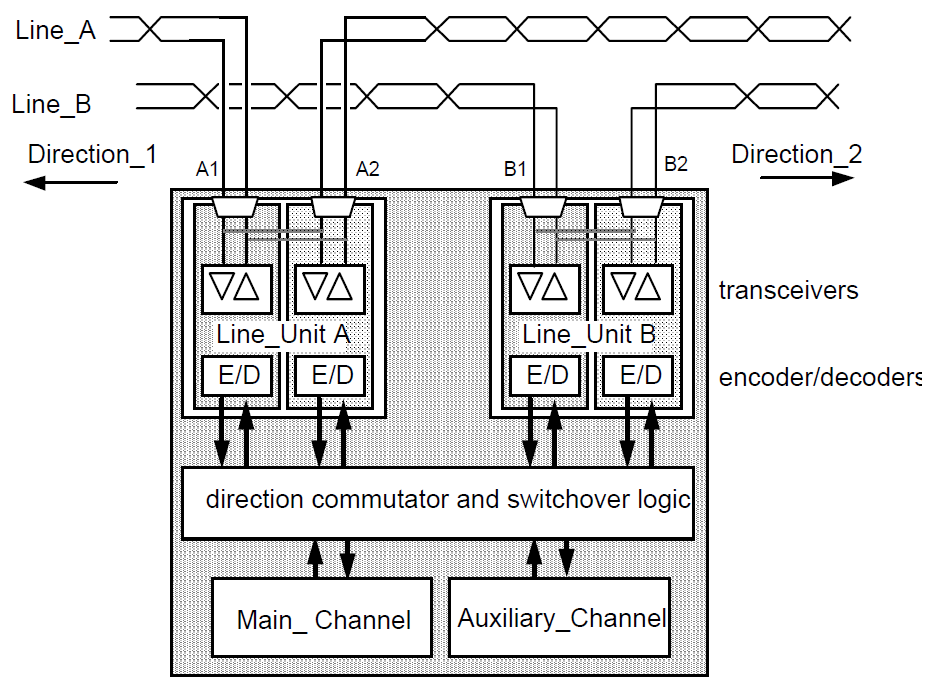
주: 종단 설정은 이름 없는 노드, 버스 끝의 노드(특히 슬레이브가 없는 마스터), 또는 슬립 모드의 노드가 수행한다.

###### Duplicated Line Units (option)

이 옵션을 사용할 경우 다음 사양을 적용하여야 한다:

* 이중 부착을 위해 설계된 MAU는 Line\_A와 Line\_B에 별도의 Line Unit로 부착하여야 한다;
* 노드 설정(단말 혹은 중간)은 두 선로 단위 모두에 동일하게 적용한다;
* 다른 회선을 계속 사용하면서 한 회선을 제거할 수 있어야 한다.

예: 이중 회선 작동을 위한 중간 부착 장치가 그림 124에 나와있다. 전환 논리로 신호가 Line\_A 또는 Line\_B 중 하나로부터 수신할 수 있다.

****

**Figure 124 – Node with redundant Line Units**

* + 1. **Line Unit specifications**

라인 유닛 A만 언급되었지만 이중화된 매체가 사용되는 경우 라인 유닛 B에도 적용한다.

###### Galvanic separation

노드 케이싱과 A1X, A1Y, A2X 또는 A2Y 사이의 절연 전압과 절연 저항은 IEC 60571에 명시된 값을 초과해야 한다.

주: 이 값들은 커렌트 에디션 0.50kV r.m.s 와 1.0MΩ이다.

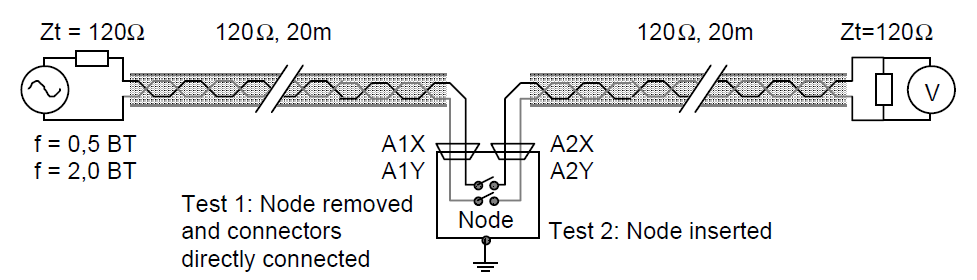
###### Insertion losses of a Line Unit

###### Attenuation measurement

삽입 손실 측정을 위해 제네레이터의 정현파 신호(내부 임피던스 = Zt)를 20.0m의 케이블을 통해 A1X과 A1Y에 인가하고 전압계(임피던스 Zt와 병렬로 연결된)로 측정한다(그림 125참조). 다른 지점은 A2X and A2Y 지점에 부착된 20.0m 케이블 중 하나 또는 그 반대의 경우도 있다.

감쇠는 두 차동 전압의 dB로 표현된 비율로 정의한다:

1. 노드를 제거하고 케이블 커넥터를 결합할 때 제 1전압은 4.0Vpp로 설정하고;
2. 노드를 연결할 때 측정하는 제 2전압(시험에 따라 단말 설정 또는 중간 설정에서).



**Figure 125 – Attenuation measurement**

###### Node In End Setting

종단 설정의 라인 유닛(Kb 열림, Kt1과 Kt2 닫힘)은 A1X와 A1Y 사이 또는 A2X와 A2Y 사이에 적용된 1.0BR 신호에 나타나야 하며, 임피던스는 4.2.2.6에 명시된 종단기에 해당한다.

종단 설정의 회선 장치는 A1X와 A1Y 사이에 적용하고 A2X와 A2Y 사이에서 측정한 신호 또는 그 반대로 측정한 신호를 55.0dB이상 감쇠시켜야 한다.

###### Node In Intermediate Setting

중간 설정의 선로 단위(Kb 닫힘, Kt1과 Kt2 열림) 중 하나:

1. 수신기가 정상 동작 상태이고 송신기가 고 임피던스인 경우, 또는
2. 수신기나 송신기에 전원이 공급되지 않으면 정현파 신호를 감쇠시켜야 한다:

* 0.5BR과 1.0BR 사이에 0.3dB 미만; 그리고
* 0.4dB 이하에서 2.0BR까지.

노드는 A1X와 A1Y 사이에 인가되거나 A2X와 A2Y 사이에 인가된 48.0V의 포지티브 또는 네거티브 DC 전압에 대해 최소 1MΩ 의 저항을 제공하여야 한다.

###### Switches specifications

버스 선로에 연결된 모든 스위치 (Kb, Kt, 등.):

1. 최소한 500.0V r.m.s의 격리를 제시해야 한다. 열린 환경에서;
2. 닫힌 상태에서 초기 접촉 저항이 0.050Ω 미만이어야 한다;
3. 107 사이클 후에 닫힌 설정에서 접촉 저항이 0.100Ω 미만으로 규정하여야 한다;
4. 바운스 시간을 포함하여 10.0ms 미만으로 하나의 설정에서 다른 설정으로 전환해야 한다.

주: 릴레이는 솔리드 스테이트 또는 메커니즘 타입일 수 있다.

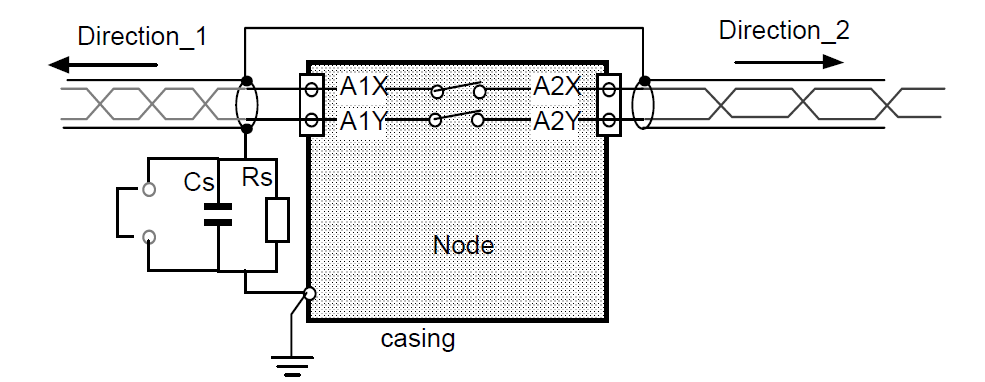
###### Shield connection to a Line Unit

각 노드에서 Direction\_1과 Direction\_2의 케이블 차폐는 접촉 저항이 0.010Ω 미만인 콘센트를 통해 서로 연결하여야 한다.

회선 유닛은 차폐를 노드 케이스에 연결하는 수단을 제공해야 한다:

1. 4.2.2.5.1(접지 차폐)에 명시한 대로 낮은 임피던스를 통해 직접 연결한다;
2. 4.2.2.5.2(플로팅 차폐)에 규정된 RC 네트워크를 통해.

예: 노드에서의 차폐 부착 원리가 그림 126에 나와있다.



**Figure 126 – Shield grounding in the Line Unit**

###### Fritting (option)

릴레이와 커넥터의 접촉 산화를 극복하기 위해 노드는 어느 방향으로든 선로 X와 Y사이에 연속 전압을 인가하는 프리팅을 사용할 수 있다.

프리팅 옵션이 사용되면 이 절의 명세를 사용한다.

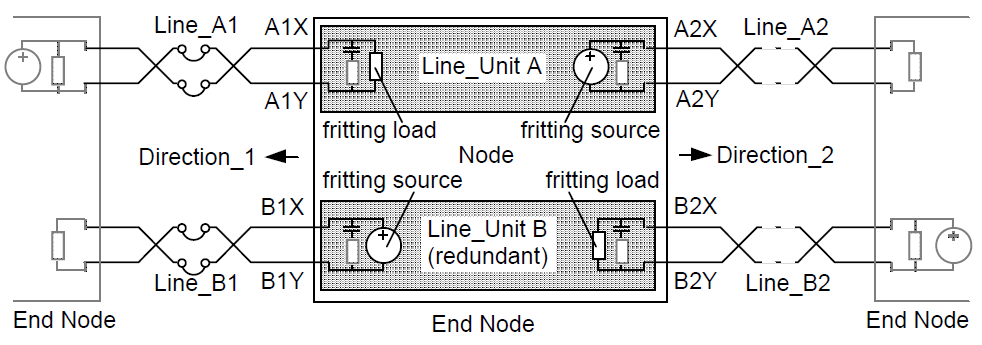
주1: 비 중복 매체에 대해서는 프리팅 사용을 명시하지 않는다.

주2: 프리팅을 사용하지 않는 노드와 그것을 사용하는 노드는 동일한 버스 상에 혼재될 수 있다.

###### Fritting source and load

프리팅을 지원하는 노드는 각 방향에 대해 프리팅 전압 소스와 프리팅 전압 부하를 제공해야 한다.

여분의 물리적 매체의 경우, 노드는 각 방향에 하나씩 두 개의 독립적인 프리팅 전압 소스를 제공해야 하고 그림 127과 같이 다른 노드의 프리팅 전압에 대한 부하를 제공해야 한다.



**Figure 127 – Fritting source and load**

프리팅 소스의 양극은 각각 A2X, B1X에 연결하여야 한다. 프리팅 소스의 네거티브는 각각 A2Y, B1Y에 연결하여야 한다.

프리팅 소스는 48.0V (+20%, -10%)의 직류 전압을 연결 지점(A2X/A2Y, B1X/B1Y)에 공급해야 한다.

해당 지점은 4.2.2.6에서 규정한 터미네이터를 연결하거나 개방된 채로 있어야 한다.

프리팅 소스의 리플은 0.5 ~ 2.0BR의 범위에서 0.100Vpp 미만이어야 한다. 프리팅 소스가 전달 하는 전류는 80.0mA/DC를 초과하지 않아야 한다. 프리팅 소스는 IEC60571을 준수하도록 출력 절연을 위한 입력을 가져야 한다.

프리팅 소스는 예를 들어 0.10H의 인덕터 또는 노드의 삽입 손실을 충족시키는 임의의 다른 배치에 의해 회선으로부터 분리하여야 한다.

소스의 스위치 온 상수 시간은 0.5ms ~ 5.0ms 범위 내에 있어야 한다.

소스의 스위치 오프 시간은 0.5ms ~ 5.0ms 범위 내에 있어야 한다.

동일한 노드에서 두 개의 프리팅 전압원 사이의 감쇠는 0.5BR과 2.0BR사이의 범위에서 50.0dB보다 커야 한다.

###### Applying fritting

종단 노드는 활성 보조 채널을 통해 프리팅 소스를 스위치 켜야 한다. 이름 없는 노드는 두 방향 모두 활성 보조 채널로, 슬립 모드의 노드는 활성 보조 채널을 갖지 않는다.

주1: 노드는 예를 들어 소비를 줄이기 위해 그 프리팅 소스를 펄스 할 수 있다.

주2: 전자파 간섭 레벨을 준수하는 한, 주 채널을 통한 프리팅 소스의 스위칭이 허용된다.

###### Transceiver specifications

이중 버스에 연결된 MAU에는 A1, A2, B1과 B2라는 네 개의 트랜시버가 있다. 단일 구성에서 송수신기들(A1과 A2)만 사용한다. 다음 사양은 해당 제품에 적용한다.

###### Conventions

달리 명시하지 않는 한 다음 기본 측정 조건을 유지한다:

1. 트랜시버의 특성은 케이블 섹션이 노드에 부착된 X와 Y지점에서 측정한다;
2. 모든 전압은 X와 Y 사이의 차동 전압 (Ux-Uy)으로 측정한다;
3. 송신기를 측정할 때 수신기의회로는 정상적인 수신 상태에 있다. 수신기를 측정할 때 송신기의 회로는 하이 임피던스 상태에 있다;
4. 모든 저항 값은 ±1%이며 모든 캐패시터 값은 ±10%이다.

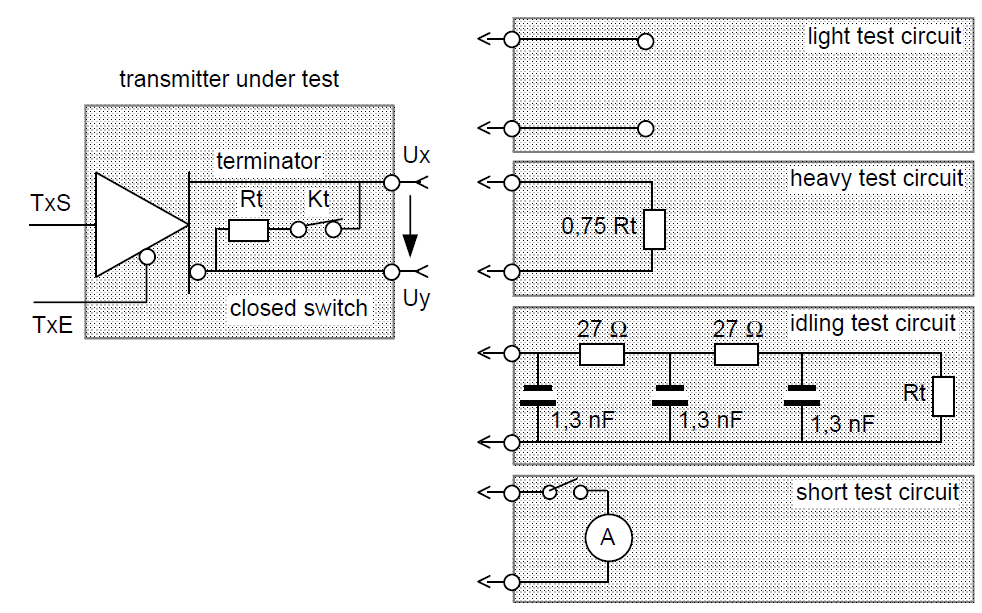
###### Transmitter

###### Test circuit for transmitter

케이블과 노드를 사용하여 트랜스미터 부하를 근사하기 위해 네 개의 테스트 회로를 지정한다:

1. 광 테스트 회로는 오픈 라인을 시뮬레이션한다(단말 설정의 노드에서 처럼). 총 저항 부하의 값은 종단 저항 값과 동일하다;
2. 완전한 테스트 회로는 완전히 로드된 버스를 시뮬레이트한다. 전체 저항 부하 값은 종단 저항 값이 0.42와 같다;
3. 공회전 시험 회로는 저항 부하가 없는 860.0m의 케이블을 모의 시험한다. 캐패시터는 각각 1.3nF ±10%의 값을 가지며 저항은 각각 27.0Ω ±1%의 값을 가진다.
4. 간단한 시험 회로는 회선 고장을 시뮬레이트한다. 전류 측정 회로만 구성한다.

이 회로는 그림 128에 나와있다.



**Figure 128 – Transmitter test circuits**

단말 설정의 노드(Kb 열림, Kt 닫힘)로 측정한다.

주: 회선 장치의 터미네이터는 시험 회로 사양에서 고려한다.

###### Transmitter output signal

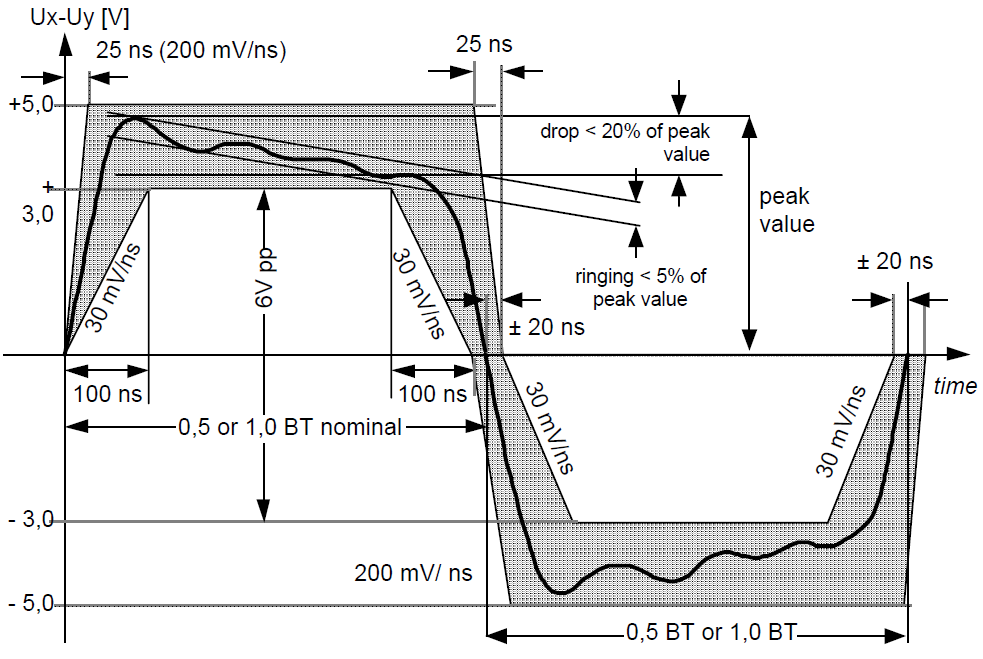
주: 사용된 데이터 인코딩 때문에 송신기는 프리앰블과 종료 딜리미터 사이에 길이가 1비트 또는 길이가 0.5비트인 펄스를 생성한다.

이 사양은 0.5비트와 1비트 펄스(포지티브 또는 네거티브) 모두에 적용한다. 송신기는 차동 드라이버이어야 한다.

출력 신호는 노드의 연결 지점에서의 차동 전압(Ux-Uy)이다.

4.2.6.2.1에 정의된 완전한 시험 회로와 광 시험 회로 중 하나에 연결된 경우 송신기는 그림 129와 같이 다음 사양을 준수해야 한다:

1. 출력 신호는 선택적으로 포지티브와 네거티브가 되어야 한다;
2. 출력 신호의 진폭은 완전한 시험 회로에서 적어도 [±](#_bookmark47)3.0V이고 광 시험 회로에서는 최대 ±7.0 V이다;
3. 피크 진폭은 출력 신호의 최대 진폭으로 정의한다. 신호는 이 피크 진폭으로부터 20%이상 떨어져서 다음에 예상되는 제로 – 천이로부터 0.100s가 떨어지지 않아야 한다. 이 시간 동안 평균 전압 강하에 대한 진폭의 울림은 피크 값의 5%를 초과해서는 안 된다;
4. 출력 신호의 슬루율은 언제든지 0.20V/ns보다 작아야 하고 제로크로싱의 100.0ns내에서 0.03V/ns보다 커야 한다;
5. 고정 진폭에 대한 최대 진폭의 비로 정의되는 출력 신호의 오버 슈트는 고정 진폭의 10%를 초과해서는 안 된다;
6. 이상 신호와 실제 제로크로싱 간의 시간차로 정의되는 출력 신호의 에지 왜곡은 1비트 시간의 ±2%를 초과하지 않아야 한다.



**Figure 129 – Pulse wave form at transmitter**

주: 트랜스와 직렬로 연결된 프리팅 캐패시터 때문에 전압 강하가 발생할 것이다.

###### Transmitter noise

송신하지 않는 송신기에 의해 발생된 잡음은 1.0KHz ~ 4.0BR의 주파수 범위에서 5.0mV/ r.m.s 값을 초과하지 않아야 한다.

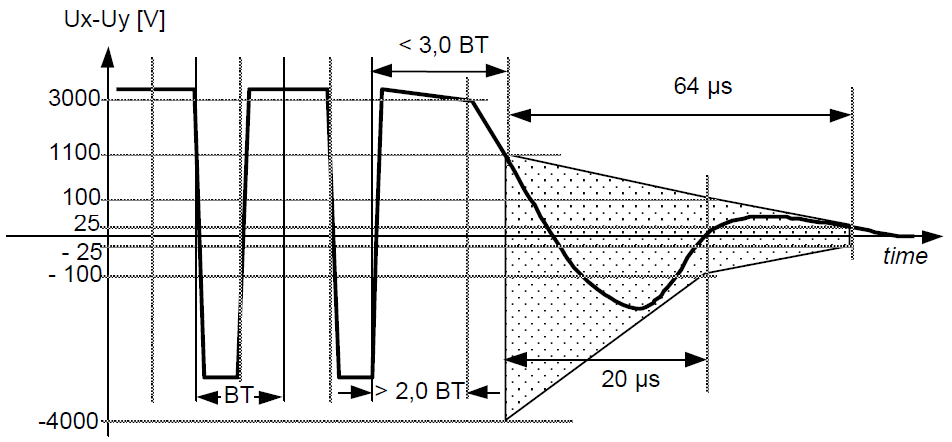
###### Transmitter end of frame

송신기에 의해 생성된 프레임의 끝은 다음 조건 하에서 시험하여야 한다:

1. 송신기는 가능한 가장 긴 프레임을 송신한다;
2. Frame\_Data 비트는 ‘1’과 ‘0’ 심볼의 의사 랜덤 시퀀스이다;
3. 프레임은 4.3.1.4에 명시된 종료 딜리미터로 닫힌다;
4. 송신기는 4.2.6.2.1의 공회전 시험 회로를 구성한다.
5. 트렌스미터가 비 활성화되기 전에 평균 차동 진폭이 4.5V보다 크다

이 조건에서 출력 신호는 그림 130과 같이 다음 제한 범위 내에 있어야 한다:

* 1. 마지막으로 네거티브에 포지티브로 천이한 후 100.0ns, 2.0BT ±100ns일 때 출력 신호는 0.300V이상을 유지해야 한다;
  2. 마지막 네거티브에서 포지티브로 천이한 후 3.0BT 이내에 출력 신호는 1.100V 이하로 내려야 한다;
  3. 20.0µs 이내에 출력 신호가 처음 1.100V에 도달할 때 시작하여 출력 신호 진폭은 0.100V를 초과하지 않아야 한다;
  4. 64.0µs 내에서 출력 신호가 처음 1.100V에 도달할 때 시작하여 출력 신호 진폭은 0.025V를 초과하지 않아야 한다.



**Figure 130 – Signal and idling at transmitter**

주: 트랜스미터 아이들링 후 위상 흔들림은 각 비트 셀의 신호 밸런싱을 통해 최소화할 수 있다. 4.3.1.4에 명시된 바와 같이 종료 딜리미터의 균형을 조정하여 추가로 줄일 수 있다.

###### Transmitter fault tolerance

활성화 여부에 관계없이 트랜스미터는 열 안정성이 도달할 때까지 연결 지점에서 간단한 테스트회로(4.2.6.1)의 적용을 허용해야 하며 간단한 테스트 회로가 제거된 후에 정상 작동을 재개해야 한다.

단락 전류는 1.0A를 초과하지 않아야 한다.

주: 적합성 시험을 위해 열 안정성은 1시간 후에 도달한 것으로 간주된다.

###### Transmitter anti-jabber

각 송신기는 전송 지속 시간이 가능한 가장 긴 프레임(프리엠블, 종료 딜리미터와 비트 스터핑 포함)의 지속 시간 +20%와 같은 T\_jabber값을 초과하는 경우 송신기를 버스 라인에서 분리하는 독립 회로를 포함해야 한다.

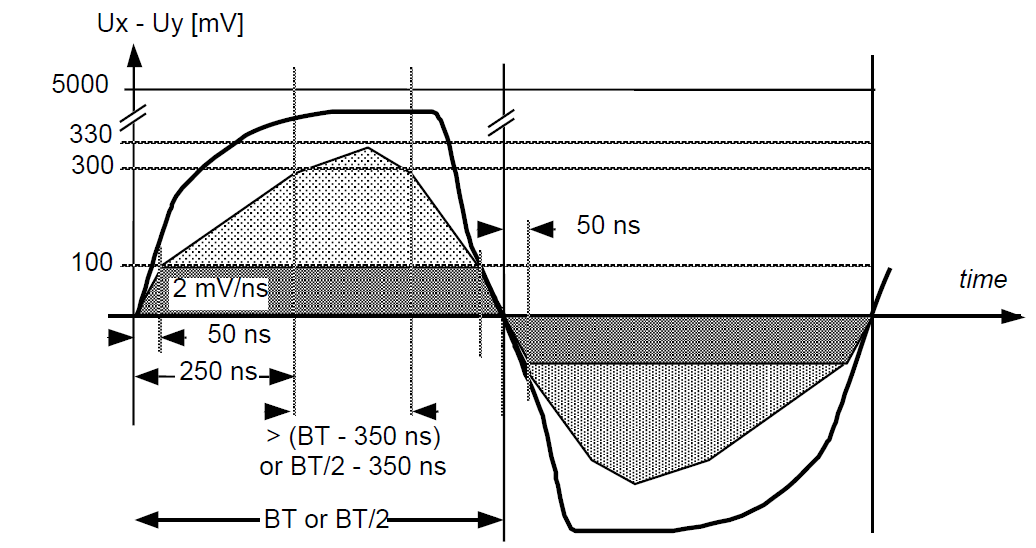
###### Receiver specifications

* + - * 1. **Receiver test signal (guideline)**

수신기의 특성은 연결 지점에 테스트 신호를 적용하고 노드에 직접 액세스하거나 마스터 프레임에 대한 응답과 같은 노드 내부 응답을 통해 수신된 프레임을 관찰하여 테스트한다.

수신기를 테스트하려면 그림 131과 같이 다음과 같은 모양의 테스트 신호를 연결 지점에 적용한다:

1. 시험 신호의 진폭이 1.00V보다 작으면 시험 신호의 기울기가 2.0mV/ns를 초과한다;
2. 감도를 시험하기 위해 시험 신호는 선행 제로크로싱의 100.0ns후에 시작하여 적어도 (0.5BT-350.0ns) 지속되는 동안 0.300V이상으로 유지한다, 각각(1,0 BT – 0,350 s). 피크 진폭은 0.330V와 5.00V 사이에서 변한다;
3. 둔감도를 시험하기 위해 시험 신호 진폭은 규정된 값을 초과하지 않아야 한다;
4. 에지 왜곡을 시험하기 위해 제로 전압의 교차는 규정된 값만큼 변한다.



**Figure 131 – Receiver signal envelope**

프레임 오류가 누락되었거나 유효하지 않은 프레임(4.3.1.5.3 참조), 잘못된 프레임 크기 또는 잘못된 Frame\_Data비트 (FCS 오류)로 감지되었다.

###### Receiver polarity

TxS의 하이 레벨은 포지티브의 차동 전압(Ux–Uy)에 해당하며 수신기의 RxS 신호의 하이 레벨과 일치해야 한다.

TxS의 로우 레벨은 네거티브의 차동 전압(Ux – Uy)에 해당하며 수신기의 RxS 신호의 로우 레벨과 일치해야 한다.

회선이 유휴 상태이면 RxS의 상태는 정의하지 않는다.

###### Receiver sensitivity

1초당 1000프레임의 속도로 64개의 랜덤 데이터 비트를 포함하는 프레임을 수신하는 수신기는 4.2.6.3.1에서 정의된 시험 신호의 진폭이 3 x 3 x 10+6프레임에서 3개 이하의 프레임 오류를 검출해야 한다. 최소 값과 최대 값 사이의 차이가 있다.

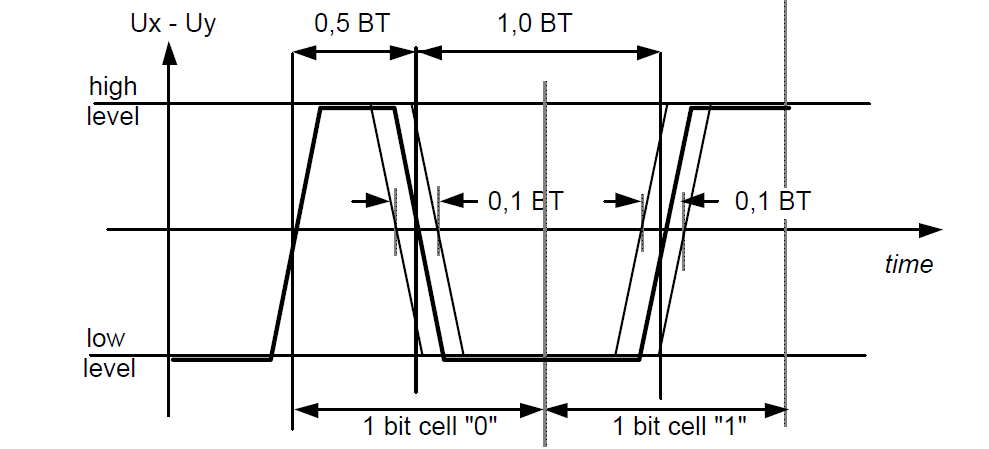
주: 그림 131에 따르면 수신기는 약 23.6dB인 0.500V ~ 0.330V의 전압 범위에서 작동할 수 있다. 4.2.2.3.3에서 중간 감쇠가 20.0dB보다 작게 지정되었다는 것을 고려하면 약 4.0dB의 잡음 여유를 남긴다.

###### Receiver insensitivity

시험 신호(4.3.1.5.3)가 0.100V보다 작으면 수신기는 유효한 프레임(4.3.1.5.3 참조)을 디코딩하지 않아야 한다.

###### Receiver edge distortion

1초당 1000 프레임의 속도로 64개의 랜덤 데이터 비트를 포함하는 프레임을 수신하는 수신기는 시험 신호 에지가 그림 132와 같이 ±10% 이내의 제로 전압 라인을 무작위로 교차할 때 3x3x10+6 프레임에서 3개 이하의 프레임 오류를 검출해야 한다.



**Figure 132 – Receiver edge distortion**

* + - * 1. **Receiver noise rejection**

초당 1000 프레임의 속도로, 그리고 신호 진폭이 0.700V(1.400V peak-to-peak)인 64개의 랜덤 데이터 비트를 포함하는 프레임을 수신하는 수신기는 3x10+6 작동 때:

* 진폭이 4.000V.r.m.s인 케이싱과 두 데이터 선로 사이에 적용된 공통 모드 정현파 신호가 있는 경우 65.0Hz ~ 1.5MHz의 주파수 사이에서; 또는
* 진폭 140V r.m.s에서 1.0kHz ~ 4.0MHz의 대역폭에 걸쳐 분포하는 부가적인 준 백색 가우스 잡음(X와 Y 사이에 적용됨)이 있는 상태.

###### Medium-dependent signalling

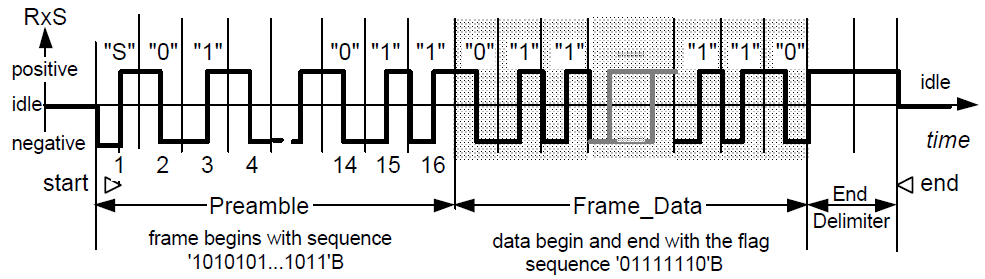
###### Frame encoding and decoding

###### Conventions

인코딩과 디코딩은 전송되거나 수신된 신호가 바이어스된 레벨없이 바이너리라고 가정한다. 회선이 유휴 상태이면 수신 레벨을 정의하지 않는다.

RxS는 회선으로부터 수신된 이상적인(아날로그) 수신 신호를 나타낸다.

프레임은 그림 133과 같이 프리앰블로 시작하여 종료 딜리미터로 끝나는 포지티브와 네거티브의 순서로 전송되어 아이들 상태로 돌아간다.

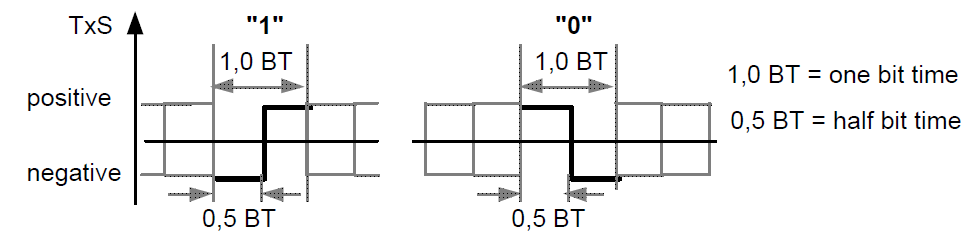


**Figure 133 – Idealised frame on the line (16 bit Preamble shown)**

###### Bit encoding

프리앰블과 Frame\_Data 비트는 그림 134와 같이 맨체스터 코드로 인코딩한다:

* ‘1’비트는 셀의 중간에서 포지티브 레벨로 가는 비트 셀의 첫 번째 절반 동안 네거티브의 레벨로 인코딩 한다;
* ‘0’비트는 셀의 중간에서 네거티브 레벨로 가는 비트 셀의 전반부 동안 포지티브의 레벨로 인코딩 한다.

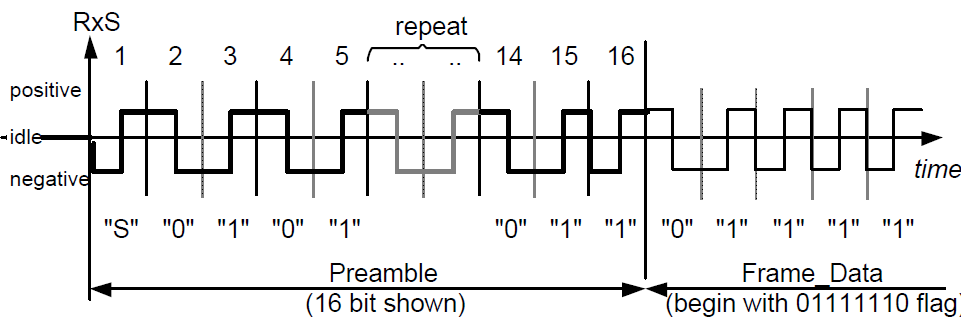


**Figure 134 – Bit encoding**

에지 거리는 프레임 종료 딜리미터까지 전체 비트 시간 (‘0’ - ‘1’ / ‘1’ - ‘0’) 또는 절반 비트 시간 (‘0’ / ‘1’)이다.

###### Preamble encoding

프레임은 그림 135와 같이 ‘1’비트로 전송된 시작 비트 S로 시작하여 (‘0’,’1’)비트 쌍과 ‘1’비트 쌍으로 구성된 비트 시퀀스로 구성된 프리앰블로 시작한다.



**Figure 135 – Preamble**

시작 비트와 닫는 ‘1’ 사이에는 최소 7쌍과 최대 15쌍(‘0’,’1’)이 있다.

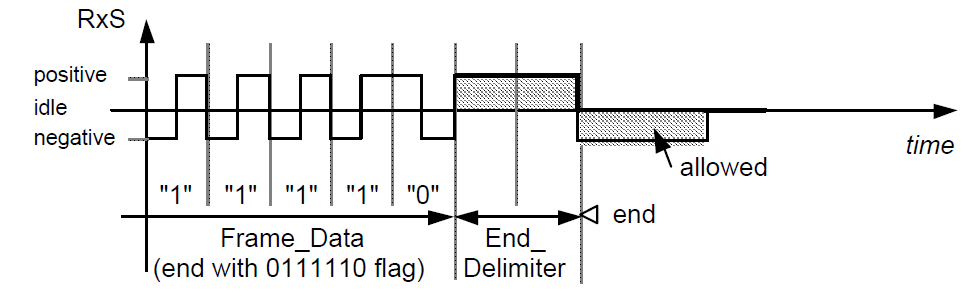
디코더는 프리앰블을 디코딩하여 Rxs의 극성을 검사할 수 있지만 X와 Y 가 실수로 바뀌어도 자동으로 신호를 반전시켜서는 안된다.

주: 다음에서 16비트의 프리앰블만을 고려할 것이다.

###### End Delimiter

프레임은 2.0 BT의 지속 시간 동안 포지티브의 레벨에서 선로를 유지하는 종료 딜리미터에 의해 닫는다.

그림 136과 같이 불균형을 보상하기 위해 2.0 BT의 네거티브 레벨을 포지티브 레벨 이후에 첨부할 수 있다.



**Figure 136 – End Delimiter**

주1: 네거티브 값이 없는 종료 딜리미터는 불균형을 유발하여 회선의 레벨을 불 안정시킨다(4.2.6.2.4 참조). 보정 펄스는 생성하지 않는 사용 회로를 사용할 수 있도록 하기 위해 필수 사항은 아니지만 권장 사항이다.

주2: HDLC 플래그(4.4.1 참조) 때문에 프레임의 마지막 비트는 ‘0’이다.

###### Signal quality supervision

다음 사양에서는 디코더가 신호 품질 감독과 중복성 전환을 위해 Carrier\_Sense(CS)와 Signal\_Quality\_Error(SQE)라는 두 개의 신호를 생성한다고 가정한다.

###### Carrier\_Sense

디코더는 4.3.1.3에 따라 프리앰블의 마지막 수신 비트를 감지한 후 0.5BT 내에서 CS를 인가한다.

디코더는 종료 딜리미터를 감지한 후 0.5 BT 내의 CS를 취소하거나 ‘0’ 또는 ‘1’이 아닌 비트들, 또는 종료 딜리미터를 검출하지 않는다.

###### Signal\_Quality\_Error

디코더는 4.3.1.3에 따라 프림앰블의 마지막 수신 비트를 검출한 후 0.5 BT 내에서 SQE를 취소한다.

디코더는 CS가 인가되는 동안 ‘0’이나 ‘1’이 아닌 비트들, 또는 종료 딜리미터가 아닌 비트를 검출하면 0.5 BT 내에서 SQE를 인가한다.

###### Valid frame

프레임은 프리앰블, ‘0’과 ‘1’비트의 숫자와 종료 딜리미터로 구성되는 경우 유효한 것으로 정의한다.

예: 유효한 프레임과 해당 CS, SQE 신호는 그림 137에 나와있다.



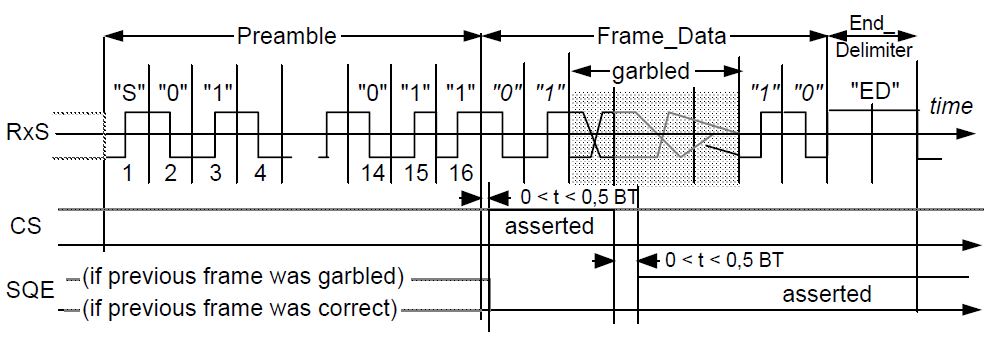
**Figure 137 – Valid frame, RxS, CS and SQE signals**

리던던시 제어를 위해 유효 프레임은 프리앰블과 적어도 8개의 데이터 비트 순서로 구성된다.

###### Not valid frame

CS가 인가되는 동안 SQE가 0.5BT보다 긴 시간 동안 인가되면 프레임은 유효하지 않은 것으로 정의한다.

예: 깨진 프레임을 수신했을 때 신호 타이밍을 그림 138에 표시한다.



**Figure 138 – Garbled frame, RxS, CS, SQE signals**

SQE가 활성화되면 디코더는 다음 프리앰블이 수신될 때까지 모든 데이터를 무시한다.

###### Duplicated line handling (option)

이 사양은 선택적인 중복 계획을 정의한다. 이 옵션을 사용하는 경우 다음 사양은 Direction\_1과 Direction\_2에서 Line\_A와 Line\_B까지 모두에 적용한다.

###### Principle

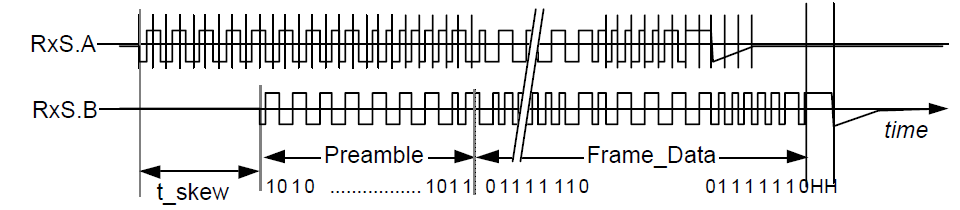
노드는 Line\_A와 Line\_B를 통해 동일한 데이터를 동시에 전송하고 노드는 신뢰할 수 있는 회선이라고 하는 한 회선의 데이터를 받아들이는 반면 Observed Line이라고 하는 다른 라인은 모니터링 한다.

각 노드는 자신의 물리 계층에 의해 생성된 신호 또는 링크 계층의 요청에 따라 다른 노드와 독립적으로 신뢰할 수 있는 회선과 관찰된 회선을 선택한다.

매체와 독립적으로 유지하기 위해 신뢰할 수 있는 회선의 선택은 회선 단위 인터페이스가 정의하는 회선 단위로 생성되는 신호에 의존한다.

###### Skew

Line\_A와 Line\_B의 신호는 서로 다른 지연을 겪기 때문에 그림 139와 같이 송신기, 수신기 또는 회선상의 임의의 위치에서 스큐(타이밍 차이)가 다르다.



**Figure 139 – Redundant Lines (as seen at a receiver)**

###### Redundant transmission

여분의 회선 유닛을 가진 MAU는 Line\_A와 Line\_B (Line\_A1과 Line\_B1, 또는 Line\_A2와 Line\_B2)를 통해 동일한 신호를 전송한다.

동일한 방향으로 Line\_A와 Line\_B 사이의 라인 유닛의 출력에서 측정된 신호간의 타이밍 차이는 T\_skew\_t = 1.0µs를 초과하지 않는다.

###### Redundant reception

###### Skew at reception

수신기는 수신기에서 최대 기울기 (T\_skew\_r)가 32.0s인 것을 허용한다.

###### Line\_Disturbance

Line\_A에 대해서는 DA1과 DA2, Line\_B에는 DB1과 DB2라고 하는 노드에 연결된 각 버스 섹션에 대해 ‘Line\_Disturbance’ 신호가 있다.

회선의 ‘Line\_Disturbance’ 신호를 다음과 같은 경우에 인가한다:

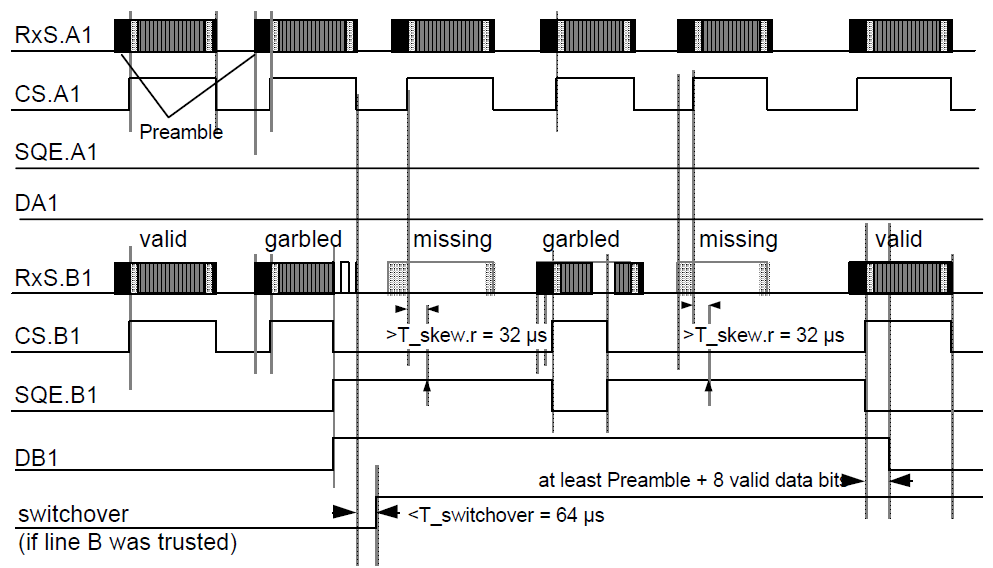
* 디코더는 해당 회선에서 ‘Signal\_Quality\_Error’를 인가한다; 또는
* 디코더는 중복 회선의 회선 유닛이 ‘Carrier\_Sense’(프레임 누락)를 인가한 후에 T\_skew\_r내에서 ‘Carrier\_Sense’를 생성하지않는다.

‘Line\_Disturbance’ 신호는 취소한다:

* 디코더가 4.3.1.5.3에서 정의한 대로 유효한 프레임을 수신하는 경우.

비 중복 모드에서 사용하지 않는 회선은 영구적으로 방해받는 것으로 간주한다.

예: Line\_A1의 모드 프레임은 유효하지만, Line\_B1이 방해받는 경우를 그림 140에 표시한다.



**Figure 140 – Line\_Disturbance signals**

###### Switchover

(단말)노드의 Direction\_1과 Direction\_2는 서로 다른 회선을 신뢰할 수 있다 (예: Line\_A1과 Line\_B2).

전환 유닛은 T\_switchover = 64.0µs 내에서 신뢰할 수 있는 회선과 관측된 회선을 교환한다.

1. 관찰 회선의 Line\_Disturbance가 인가되지 않는 동안 신뢰하는 회선의 Line\_Disturbance가 인가되면;
2. 링크 계층이 요청할 경우(특히 크기, FCS 또는 프로토콜 오류가 발생하는 경우).

###### MAU report

MAU는 링크 관리부에 다음을 보고한다:

1. 어느 회선을 통해 프레임을 수신했는가;
2. 회선과 방향 중 어느 하나에서의 Line\_Disturbance의 각 인가(assertion);
3. 노드 보고서에 대한 네 개의 Line\_Disturbance 신호 (DA1, DA2, DB1, DB2) 의 상태(4.7.2.2 참조).

주: 회선이 완전히 비 활성화 된 경우(또는 연결되지 않은 경우), Line\_Disturbance는 오류 발생 후 한번 토글한다. 그러나 회선이 인터럽트되면 Line\_Disturbance는 인터럽트 위치에 따라 거의 모든 프레임 이후에 전환할 수 있다.

###### 4.3.3 Line Unit interface

회선 유닛 인터페이스는 회선 유닛으로 들어오고 나가는 신호를 정의한다.

이 인터페이스는 노드 내부에 남아 있을 수 있지만 테스트를 위해 사용할 수 있도록 하는 것이 좋다. 이 인터페이스는 적합성 테스트에서 다루지 않는다.

다음의 명세는 인터페이싱을 용이하게 하고 이 표준의 다른 하위 절에서 사용되는 신호를 정의한다.

노출하게되면, 회선 유닛 인터페이스는 ITU-T 권고안 V.24 에 정의한 대로 모뎀 신호, 각 트랜시버에 대해 별도로 표 61에 명시한 대로 추가 제어 신호로 구성한다.

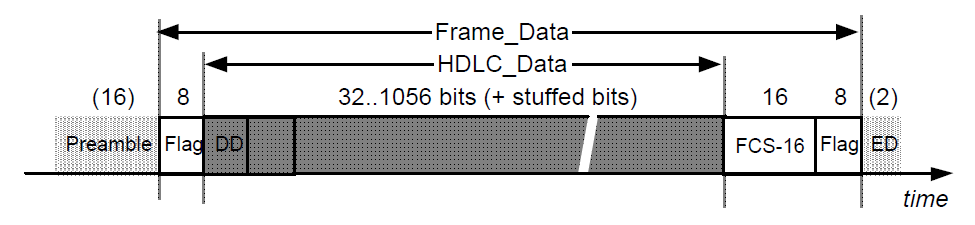
**Table 61 – Signals of the Line Unit Interface**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Designation** | **Clause of**  **ITU-T Rec. V.24** | **Direction** | **Meaning** |
| GND | Signal Ground | 102 | - | common return |
| TxD | Transmitter Data | 103 | to Line Unit | Frame\_Data, with no Preamble or End Delimiter, supplied as NRZ signal, clocked by TxC. |
| RxD | Receiver Data | 104 | from Line Unit | NRZ sequence, excluding Preamble  and End Delimiter, clocked by RxC. |
| RTS | Request To Send | 105 | to Line Unit | commands to send the Preamble; clearing this signal commands the generation of the End Delimiter. |
| CTS | Clear To Send | 106 | from Line Unit | signals that the Preamble has been transmitted and requests data to follow. |
| TxC | Transmitter Clock | 114 | from Line Unit | generated by the transceiver to clock in the TxD data to send. |
| RxC | Receiver Clock | 115 | from Line Unit | generated by the decoder to clock out the received RxD data. |
| SQE | Signal\_Quality\_Error | Not V.24 | from Line Unit | as specified in 4.3.1.5.2 |
| CS | Carrier\_Sense | Not V.24 | from Line Unit | as specified in 4.3.1.5.1 |
| Kx | Switch control signals | Not V24 | to Line Unit | define at least the two basic switch settings: End Setting and  Intermediate Setting, for both lines.  In addition, switch control signals may isolate a transceiver from the line or connect it to the line. |

###### Frames and telegrams

###### Frame\_Data format

Frame\_Data 형식은 그림 141과 같이 ISO/IEC 13239에 정의한 HDLC 형식을 따른다.



**Figure 141 – HDLC Frame structure**

프레임은 ISO/IEC 13239에 명시한 바와 같이 하나의 ‘0’비트 다음에 여섯 개의 인접한 ‘1’비트와 하나의 ‘0’비트로 구성된 단일 플래그로 시작된다.

시작 플래그는 적어도 32비트와 최대 1056비트(ISO/IEC 13239에 의거하여 스터핑 비트를 포함하지 않음)로 구성된 HDLC 데이터가 따라온다.

HDLC 데이터는 (ISO/IEC 13239에 대한 제한으로서)정수의 옥텟을 가진다. 첫 번째 옥텟은 4.5.1에 명시한 8비트 목적지 장치이다. 두 번째 옥텟은 ISO/IEC 13239의 제어 필드로 해석되어서는 안 된다.

HDLC 데이터는 ISO/IEC 13239가 규정한 대로 구축된 16비트 프레임 체크 시퀀스(FCS)인 오류 검출 코드가 따라온다.

프레임은 시작 플래그와 동일한 단일 마침 플래그에 의해 끝난다. 마침 플래그는 다음 프레임의 시작 플래그로 사용될 수 없다.

송신기는 ISO/IEC 13239의 ‘유휴’ 또는 ‘취소’시퀀스를 전송해서는 안된다.

프레임은 일련의 옥텟으로 간주된다. ISO/IEC 13239에서 규정한 대로 각 옥텟의 최 하위 비트를 먼저 전송한다.

주1: 비트 순서 지정 규칙은 옥텟 내의 비트 전송 순서를 유일하게 유지한다. 두 개의 FCS 옥텟은 ISO/IEC 13239의 예외로 인해 가장 중요한 비트를 먼저 전송한다.

주2: HDLC의 비트 채우기는 플래그 시퀀스가 플래그 사이의 데이터에 나타나지 않도록 한다. 송신기는 데이터에서 5개의 연속 ‘1’의 각 그룹 뒤에 ‘0’을 끼운다. 수신자는 ‘1’그룹을 따라 오는 ‘0’을 제거한다. 채워진 비트는 링크 계층에서 보이지 않지만 프레임 크기를 최대 20%까지 늘릴 수 있다.

###### Telegram timing

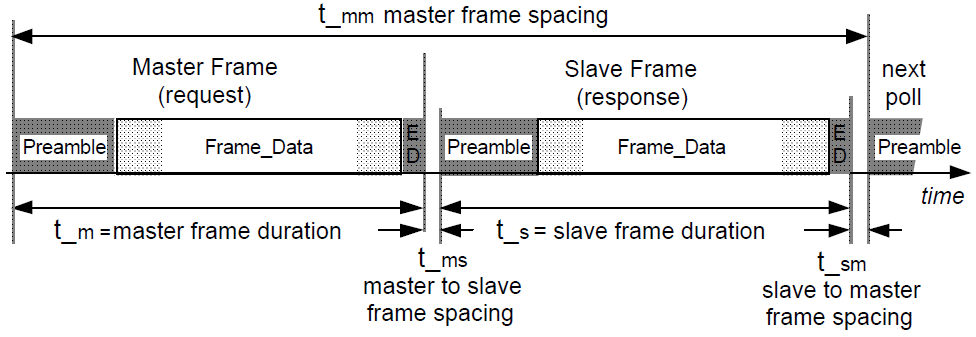
###### Conventions

버스 구간은 마스터라고 하는 하나의 노드가 제어하며, 이는 자신의 의도대로 버스에서 전송할 수 있다. 다른 노드는 슬레이브이며 마스터가 요청한 경우에만 전송할 수 있다.

종단 노드는 보조 채널의 마스터로 간주한다. 노드 내의 마스터 기능과 슬레이브 기능은 서로 다르다. 버스 트래픽은 마스터가 전송한 마스터 프레임과 그에 따라 슬레이브가 지정된 시간 내에 응답한 슬레이브 프레임으로 구성한, 텔레그램이라고 하는 쌍으로된 프레임으로 구성한다.

프레임 간 간격은 종료 딜리미터의 마지막 전환에서부터 프리앰블 시작 비트의 중간 전이까지 측정한다.

예: 텔레그램의 타이밍은 그림 142와 같다.



**Figure 142 – Telegram timing**

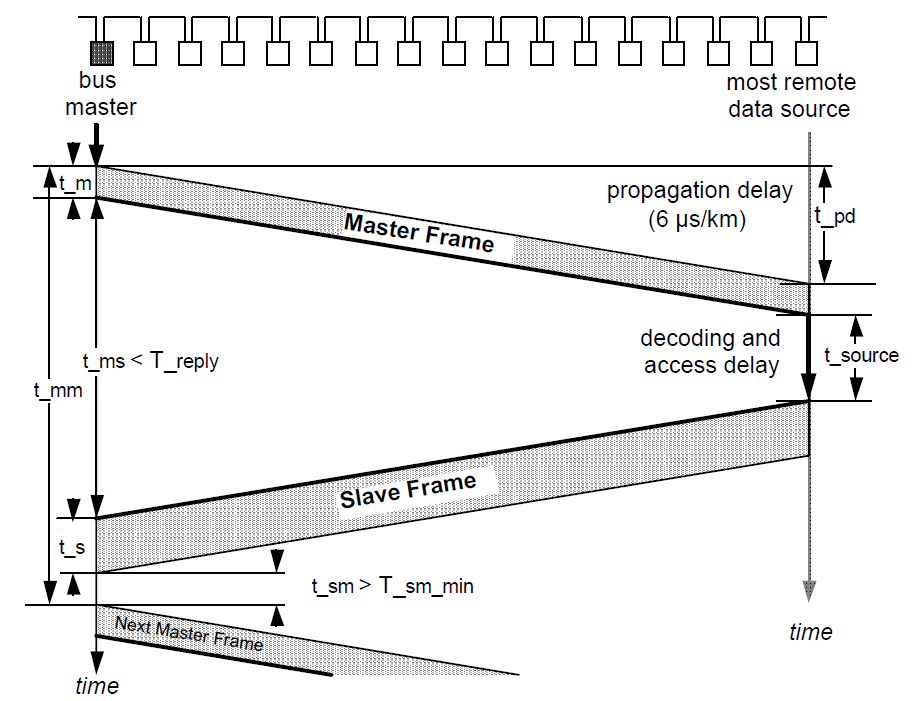
###### Computation of the Reply Delay

응답 지연 T\_reply는 주어진 버스에 대해 마스터에서 측정한 최대 지연 시간과 마스터 프레임에서 응답으로 전송하는 슬레이브 프레임의 시작 시간 사이에 나타날 수 있는 최대 지연 시간이다.

응답 지연은 전송 지연, 디코딩과 액세스 지연의 합으로 구성한다.

T\_reply는 슬레이브 프레임을 수신하지 못한 경우 다음 마스터 프레임을 보내기 전에 마스터가 기다려야 하는 시간을 알려주는 구성 파라메터이다.

예: 마스터가 버스의 한 쪽 끝에 있다고 가정하고 노드가 17개(16개 섹션)인 구성을 그림 144에 표시한다.



**Figure 143 – Example of Interframe spacing**

주어진 어플리케이션에 대한 최악의 경우 응답 릴레이 T\_reply는 다음과 같이 계산된다:

T\_reply [µs] = 2 x T\_pd + T\_source\_max

* T\_source\_max는 마스터 프레임의 디코딩을 해석하고 소스에서 응답한다(4.4.2.4.2 참조);
* T\_pd는 주어진 어플리케이션에서 종단 노드 사이에서 프레임의 최악의 전송 지연이다(4.2.2.3 참조).

프레임의 타임아웃은 다음과 같이 계산된다:

|  |  |  |
| --- | --- | --- |
| **Bit times** | **Main channel** | **Auxiliary channel** |
| application data bits header  CRC  total bits | 1024 bits  32 bits  16 bits  1072 bits | 16 bits  32 bits  16 bits  64 bits |
| bit stuffing (worst case x 1,2)  2 x flag  end delimiter  preamble maximum size  Total bits | 1286 bits  16 bits  2 bits  32 bits  1336 bits | 77 bits  16 bits  2 bits  32 bits  127 bits |
| duration at 1,0 Mbit/s  2 x propagation delay response  time of slave | 1336,0 µs  120,0 µs  300,0 µs | 127,0 µs  120,0 µs  800,0 µs |
| Total | 1756,0 µs | 1047,0 µs |

###### Collision

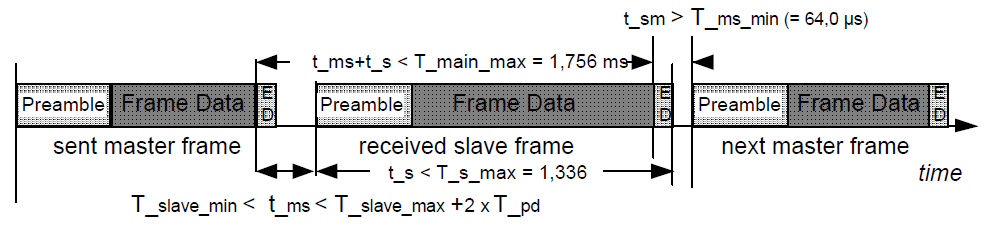
충돌은 여러 송신기가 동시에 작동할 때 발생한다. 이 상황은 동시에 전송하는 다른 버스 세그먼트의 종단 노드 사이에서만 정상적으로 발생한다. 충돌은 무 응답 또는 유효하지 않은 프레임과 구별되지 않는다.

###### Transmitted frame spacing

###### Master side

주 채널에서 마스터는 전송된 마스터 프레임이 끝난 후 타임아웃(T\_main\_max = 1.756ms) 동안 마스터 프레임에 대한 응답으로 슬레이브 프레임을 수신 할 것으로 기대하고 그림 144와 같이 수신된 슬레이브 프레임이 끝나거나 타임아웃(1.756ms + 0.064ms = 1.820ms)이 경과한 후 다음 마스터 프레임(T\_sm\_min = 0.064ms)를 전송할 수 있다:

보조 채널에서 종단 노드는 시간(T\_aux\_max = 1.047ms) 동안 마스터 프레임에 대한 응답으로 슬레이브 프레임을 수신할 것을 기대하고 수신된 검출 응답을 종료하거나 타임아웃(1.047ms + 0.064ms = 1.111ms)이 경과한 후 다음 마스터 프레임(T\_ms\_min = 0.064ms)을 송신할 수 있다.

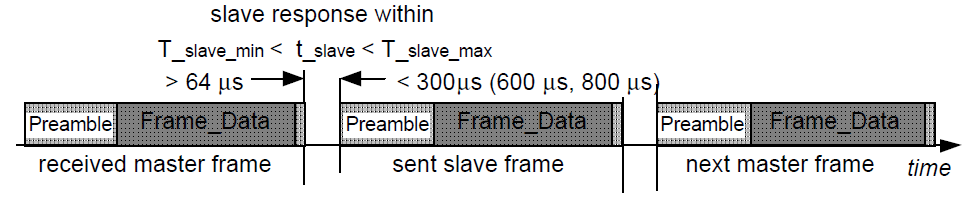


**Figure 144 – Frame spacing measured at the master side**

###### Slave side

어드레스를 가진 슬레이브는 그림 145와 같이 프레임 전송을 시작한다:

* 마스터 프레임의 끝을 받은 후 T\_source\_min = 64.0s 보다 빠르지 않아야 하며;
* 마스터 프레임의 끝을 받은 후 T\_source\_max = 0.300ms 이상이어야 한다. 이 시간은 이노규레이션 중 0.600ms로 증가하고 정상 작동 때에는 0.800ms로 증가한다.



**Figure 145 – Frame spacing at the slave**

###### Link Layer Control

###### Addressing

링크 계층은 소스와 목적지 Device\_Address 모두에 대해 8비트 식별 자를 사용한다.

노드의 주 채널은 이노규레이션 절차에 의해 할당된 1 (‘00000001’B)에서 63 (‘00111111’B) 범위에서 장치 주소를 지정한다.

마스터의 주 채널은 ‘마스터’주소 1 (‘00000001’B)을 수신한다.

장치 주소 0 (‘00000000’B)은 ‘자체’주소라야 하며 전송할 수 없다.

장치 주소 64 ~ 126과 128 ~ 254는 향후를 위해 예약되어 있다.

주소 255(‘11111111’B)는 모든 노드가 수신하는 브로드캐스트 주소이다. 이름 없는 노드는 두 채널 모두 127(‘01111111’B)에 응답한다. 노드의 보조 채널은 ‘unnamed’ 주소에 응답한다.

‘노드’ 주소는 이름이 있는 슬레이브 또는 마스터 주소이다.

###### Frame structure

###### Elements of the HDLC Frame

HDLC 데이터는 시작 플래그와 검사 시퀀스 사이에 포함된 데이터로 구성되며 그림 146과 같이 stuffed 비트는 제외된다:



**Figure 146 – HDLC Data format**

HDLC 데이터는 그림 147과 같은 필드로 구성한다:

HDLC\_Data::= RECORD

{

destination\_device UNSIGNED8 -- 목적지 주소 또는 ‘브로드캐스트 주소;

응답 프레임에서는 기본적으로

‘마스터’주소이다.

link\_control Link\_Control -- 8비트 링크 제어

source\_device UNSIGNED8 -- 프레임을 전송하는 소스 노드의 8비트

주소, 요청 프레임에서는 기본적으로

‘마스터’주소이다.

link\_data\_size UNSIGNED8 -- 크기 필드,다음에 나오는 링크 데이터의

8비트 크기 옥텟 수, Link Data 필드가

무효이면 0이 된다.

link\_data Link\_Data -- WORD8[link\_data\_size] 집합.

0과 1024 비트 사이의 데이터;

4.5.3.1 참조

}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device (DD) | | | | | | | |
|  | link\_control (LC) | | | | | | | |
|  | source\_device (SD) | | | | | | | |
|  | link\_data\_size (SZ) | | | | | | | |
| link\_data | | | | | | | | |
|  | between 0 and 1024 bits | | | | | | | |

주: DD, LC, SD, SZ는 그림에 나와있다.

**Figure 147 – Format of HDLC Data**

###### Link Control Field

링크 제어 필드는 다음과 같이 구분한다:

1. 요청 (마스터 프레임)
2. 응답 (슬레이브 프레임)

링크 제어 필드는 세 가지 유형의 텔레그램을 구별한다:

1. 배포된 프로세스 데이터 베이스 갱신하는데 사용되는 프로세스 데이터 텔레그램;
2. 메시지 전송에 사용되는 메시지 데이터 텔레그램;
3. 버스 관리와 이노규레이션에 사용되는 관리 데이터 텔레그램.

링크 제어 필드는 폴링된 노드가 4비트를 통해 산발적인 전송 요구, 상태 변경 또는 이노규레이션 조건을 제어할 수 있게 한다:

1. ‘A\_bit’: (Attention) 메시지 데이터 전송 필요;
2. ‘C\_bit’: (Change) 노드 상태 변경;
3. ‘I\_bit’: (Inhibit) 이노규레이션이 허용되지 않음;
4. ‘RI\_bit’: (Remote Inhibit) 원격 이노규레이션이 허용되지 않음.

링크 제어는 표 62에 규정한 대로 8비트로 부호화한다.

**Table 62 – Link Control encoding**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Encoding** | | | | | | | |
|  | **Frame Type** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Process Data and**  **Message Data** | Process\_Data\_Request/  Process\_Data\_Response | M | 0 | A | C | I | 0 | 0 | 0 |
| Message\_Data\_Request/  Message\_Data\_Response | M | 0 | A | C | 0 | 1 | 1 | 1 |
| **Supervisory Data** | Detect\_Request/  Detect\_Response | M | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Status\_Request/  Status\_Response | M | 1 | 0 | RI | 0 | 0 | 0 | 1 |
| SetInt\_Request/  SetInt\_Response | M | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| SetEnd\_Request/  SetEnd\_Response | M | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| Unname\_Request | M | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Naming\_Request/  Naming\_Response | M | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| Topography\_Request/  Topography\_Response | M | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| Presence\_Request/  Presence\_Response | M | 1 | 0 | RI | I | 1 | 1 | 1 |

표 62에 명시되지 않은 비트 조합은 예약되어 있으며 무시한다.

링크 제어 유형은 텍스트 형식으로 다음과 같이 지정된다:

Link\_Control::= RECORD

{

mq ENUM1 -- most significant

{

SR (0), -- ‘0’ in a slave response

MQ (1) -- ‘1’ in a master request

}

sup ENUM1

{

PM (0), -- process data or message

SP (1) -- supervisory data

},

ONE\_OF [sup]

[PM] RECORD

{

a\_bit BOOLEAN1, -- ‘1’ if A\_bit set in

Process\_Data\_Response or

Message\_Data\_Response

c\_bit BOOLEAN1, -- ‘1’ if C\_bit set in

Process\_Data\_Response or

Message\_Data\_Response

i\_bit BOOLEAN1, -- ‘1’ if I\_bit set in

Process\_Data\_Request or

Process\_Data\_Response

pom ENUM3

{

PROCESS\_DATA (0), -- Process\_Data\_Request or

Process\_Data\_Response

MESSAGE\_DATA (7) -- Message\_Data\_Request or

Message\_Data\_Response

}

},

[SP] RECORD -- Supervisory\_Data\_Request or

Supervisory\_Data\_Response

{

res0 WORD1 (0), -- reserved, = 0

rem\_inh BOOLEAN1 -- ‘1’ if ‘RI\_bit’ set in

Status\_Response and

Presence\_Response

i\_bit BOOLEAN1, -- ‘1’ if ‘I\_bit’ set in

Presence\_Request or

Presence\_Response

supervisory\_type ENUM3 -- distinguishes supervisory data

{

DETECT (0), -- Detect\_Request/Detect\_Response

STATUS (1), -- Status\_Request/Status\_Response

SETINT (2), -- SetInt\_Request/SetInt\_Response

SETEND (3), -- SetEnd\_Request/SetEnd\_Response

UNNAME (4), -- Unname\_Request

NAMING (5), -- Naming\_Request/Naming\_Response

TOPOGRAPHY (6), -- Topography\_Request or

Topography\_Response

PRESENCE (7) -- Presence\_Request or Presence\_Response

}

}

}

}

###### Handling of ‘Attention’, ‘Change’ and ‘Inhibit’

Process\_Data\_Responses와 Message\_Data\_Responses는 비동기 이벤트를 알리기 위해 다음 비트를 ‘1’로 설정한다:

1. A\_bit (Attention)는 메시지 데이터를 위한 전송 대기열이 전송을 위한 프레임을 포함하는 한 세트한다;
2. the C\_bit (Change)는 Node\_Status의 변경을 알리도록 세트되고 노드가 상태 요청을 받으면 리셋된다;
3. the I\_bit (Inhibit)는 이노규레이션을 금지하기 위해 다음과 같은 방법으로 세트한다:

* 노드 상의 어플리케이션이 이노규레이션을 금지하는 한, 노드는 모든 프로세스 데이터와 관리 데이터에 I\_bit를 세트한다 (Message\_Data\_Response는 이 비트를 세트하지 않는다);
* 마스터는 자신이 명명한 모든 노드에서 수신한 프로세스 데이터 응답의 I\_bit의 OR 조합을 전송한 모든 Presence\_Request의 I\_bit로 복사한다;
* 종단 노드는 Presence\_Request에서 수신한 I\_bit를 Detect Response의 I\_bit와 Presence\_Responses의 I\_bit에 삽입한다.

1. RI\_bit는 이노규레이션을 금지하기 위해 다음과 같은 방법으로 세트한다:

* 종단 노드는 원격 구성의 감지 응답에서 읽은 I\_bit를 Presence\_Response들과 Status Response들의 RI\_bit에 적용한다.

###### Size, FCS and protocol errors

다음은 자신의 주소 또는 브로드캐스트 주소를 통해 수신된 프레임에만 적용한다:

수신기는 프레임을 무시하고 무시된 프레임이 방해 받는 것으로 간주되는 회선을 고려한다:

1. 프레임 체크 시퀀스가 잘못된 경우;
2. 길이가 ‘link\_data\_size’필드에서 표현된 길이와 일치하지 않는 경우;;
3. 마스터 프레임과 동일한 유형(프로세스 데이터, 메시지 데이터, 관리 데이터)이 아닌 슬레이브 프레임인 경우).

보조 채널은 프레임을 무시하고 프로토콜 오류를 보고한다:

요청 감지, 응답 감지 또는 명명 요청 중 하나가 아닌 경우 보고 한다.

주 채널은 감지 요청 또는 감지 응답인 경우 프레임을 무시한다.

마스터는 마스터 프레임에 대한 응답으로 오는 두 번째 슬레이브 프레임을 무시한다(장애없이 충돌).

###### Telegram formats and protocols

###### Link Data field

링크 계층 제어는 프로세스 데이터, 메시지 데이터, 관리 데이터를 구분한다. 주소 지정 제한을 고려하기 위해 HDLC 데이터의 정의는 링크 헤더를 포함한다:

HDLC\_Data::= RECORD

{

ONE\_OF [link\_control.sup] -- depends on link\_control

{

[PM] ONE\_OF [link\_control.pom] -- Process Data or Message Data

{

[PROCESS\_DATA]

ONE\_OF [link\_control.mq] -- Request or Response

{

[MQ] Process\_Data\_Request,

[SR] Process\_Data\_Response

}

[MESSAGE\_DATA]

ONE\_OF [link\_control.mq] -- Request or Response

{

[MQ] Message\_Data\_Request,

[SR] Message\_Data\_Response

}

},

[SP] Supervisory Data

}

}

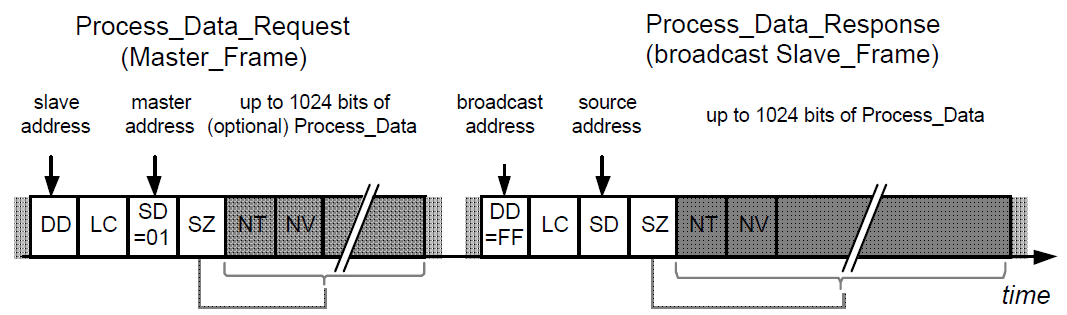
링크 데이터의 처음 4 옥텟은 링크 헤더를 형성하며 모든 프레임에서 동일한 형식과 의미를 가진다.

###### Process Data

###### Action

마스터는 다른 노드(또는 그 자신. 옵션)로부터 프로세스 데이터의 전송을 요청한다. 프로세스 데이터 요청을 전송 함으로서, 처리된 노드가 프로세스 데이터 응답으로 응답하고, 다른 모든 노드로 브로드캐스트한다.

프로세스 데이터 텔레그램은 그림 148과 같이 프로세스 데이터 요청 다음에 프로세스 데이터 응답으로 구성된다.



**Figure 148 – Process Data telegram**

###### Process Data Request

프로세스 데이터 요청은 그림 149와 같이 다음 형식을 가진다:

Process\_Data\_Request::= RECORD

{

destination\_device UNSIGNED8 -- node address or

‘master’ address for self-

poll

link\_control Link\_Control -- Process\_Data\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- = 0 or

(option (0 < link\_data\_size

128)

ARRAY [link\_data\_size] OF WORD8 -- Process Data (option)

} -- contents defined by the

application

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
|  | link\_control | | | | | | | |
|  | source\_device | | | | | | | |
|  | link\_data\_size | | | | | | | |
| ARRAY [link\_data\_size] OF | | | | | | | | |
|  | WORD8 | | | | | | | |

**Figure 149 – Format of Process Data Request**

###### Process Data Response

프로세스 데이터 응답은 그림 150과 같이 다음 형식을 가진다:

Process\_Data\_Response::= RECORD

{

destination\_device UNSIGNED8 -- destination\_device =

broadcast

link\_control Link\_Control -- Process\_Data\_Response

source\_device UNSIGNED8 -- node or ‘master’ address

link\_data\_size UNSIGNED8 -- (0  link\_data\_size 128)

ARRAY [link\_data\_size] OF WORD8 -- contents defined by the

application;

it is recommended that the two first octets be the ‘Node\_Key’ and that the application checks that it corresponds to the Node\_Key of that node it received in the Topography.

}

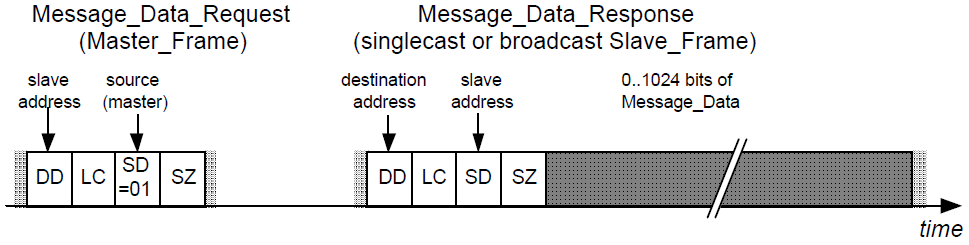
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
|  | link\_control | | | | | | | |
|  | source\_device | | | | | | | |
|  | link\_data\_size | | | | | | | |
| ARRAY [link\_data\_size] OF | | | | | | | | |
|  | WORD8 | | | | | | | |

**Figure 150 – Format of Process Data Response**

###### Message Data

###### Action

마스터는 다른 노드(또는 그 자신)가 Message\_Data\_Response를 가지고 메시지 데이터를 전송하도록 요구한다. 그림 151에서와 같이 어드레싱된 노드는 Message\_Data\_Response로 단일 노드 또는 브로드 캐스트로 응답한다.



**Figure 151 – Message Data telegram**

주: 메시지 데이터 구조는 2절에 명시하고 있다.

###### Message Data Request

메시지 데이터 요청은 그림 152와 같이 다음 형식을 가진다:

Message\_Data\_Request::= RECORD

{

destination\_device UNSIGNED8 -- node address or

‘master’ address for self-

poll

link\_control (= MESSAGE\_DATA) -- Message\_Data\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- =0

} -- void body

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |

**Figure 152 – Format of Message Data Request**

###### Message Data Response

메시지 데이터 응답은 그림 153과 같이 다음 형식을 가진다:

Message\_Data\_Response::= RECORD

{

destination\_device UNSIGNED8 -- node address or

broadcast address

link\_control Link\_Control -- Message\_Data\_Response

source\_device UNSIGNED8 -- node address

link\_data\_size UNSIGNED8 -- (0  link\_data\_size 128)

ARRAY [link\_data\_size] OF WORD8 -- Message Data contents defined

by Clause 2

}

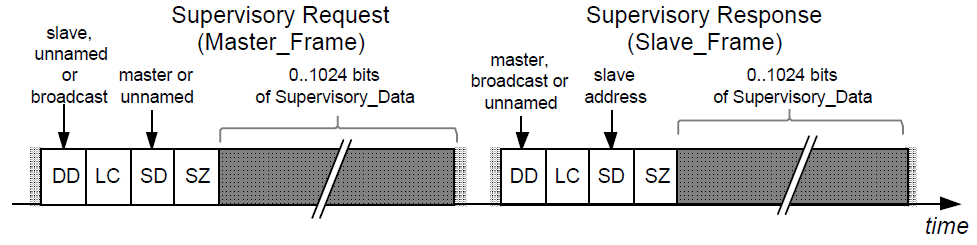
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
|  | link\_control | | | | | | | |
|  | source\_device | | | | | | | |
|  | link\_data\_size | | | | | | | |
| ARRAY [link\_data\_size] OF | | | | | | | | |
|  | WORD8 | | | | | | | |

**Figure 153 – Format of Message Data Response**

###### Supervisory Data

###### Action

마스터는 노드로부터 관리 데이터를 요구하거나 관리 데이터 요구와 함께 관리 데이터를 노드에 보낸다. 그림 154에서와 같이 지정된 데이터는 관리 데이터 응답으로 응답한다.



**Figure 154 – Supervisory telegram**

보조 채널을 통해 종단 노드가 마스터의 역할을 수행할 수 있다.

###### Supervisory telegram formats

관리 프레임은 다음 형식을 가진다:

Supervisory\_Data::= ONE\_OF [link\_control.mq]

{

[MQ] Supervisory\_Data\_Request,

[SR] Supervisory\_Data\_Response

}

Supervisory\_Data\_Request::= ONE\_OF [link\_control.supervisory\_type]

{

[DETECT] Detect\_Request,

[PRESENCE] Presence\_Request,

[STATUS] Status\_Request,

[NAMING] Naming\_Request,

[SETINT] SetInt\_Request,

[SETEND] SetEnd\_Request,

[TOPOGRAPHY] Topography\_Request,

[UNNAME] Unname\_Request,

}

Supervisory\_Data\_Response::= ONE\_OF [link\_control.supervisory\_type]

{

[DETECT] Detect\_Response,

[PRESENCE] Presence\_Response,

[STATUS] Status\_Response,

[NAMING] Naming\_Response,

[SETINT] SetInt\_Response,

[SETEND] SetEnd\_Response,

[TOPOGRAPHY] Topography\_Response

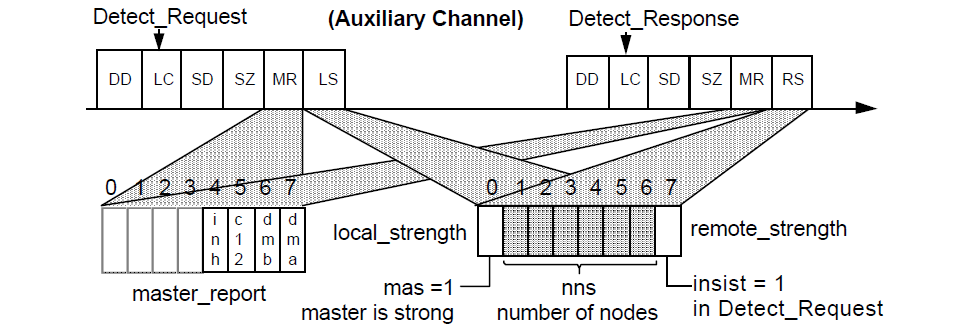
}

###### Detection telegram

###### Action

종단 노드는 감지 요청을 통해 다른 노드로 자신의 존재를 알리며 다른 노드(존재할 수도 있고 응답할 수 있는 경우)는 감지 응답으로 응답한다.

감지 텔레그램은 그림 155와 같다.



**Figure 155 – Detection telegram**

###### Detect\_Request

Detect\_Request은 그림 156과 같은 형식을 가진다:

Detect\_Request::= RECORD

{

destination\_device UNSIGNED8 -- ‘unnamed’ address

link\_control Link\_Control -- Detect\_Request

source\_device UNSIGNED8 -- ‘unnamed’ address

link\_data\_size UNSIGNED8 -- =2

master\_report Master\_Report -- see 4.7.2.5

local\_strength Composition\_Strength -- copy of LocStr of

requesting node( 4.7.2.4)

-- ‘ins’ is set to ‘1’.

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| master\_report | | | | | | | |
| mas | Nns | | | | | | ins |

**Figure 156 – Format of Detect Request**

###### Detect Response

감지 응답은 그림 157과 같은 형식을 가진다:

Detect\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘broadcast’ address

link\_control Link\_Control -- Detect\_Response

source\_device UNSIGNED8 -- ‘unnamed’ address

link\_data\_size UNSIGNED8 -- =2

master\_report Master\_Report, -- same as in Detect\_Request

(for the other composition) remote\_strength Composition\_Strength -- RemStr of responding node

remote node sets ‘ins’ if

its composition insists

( 4.7.2.4)

}

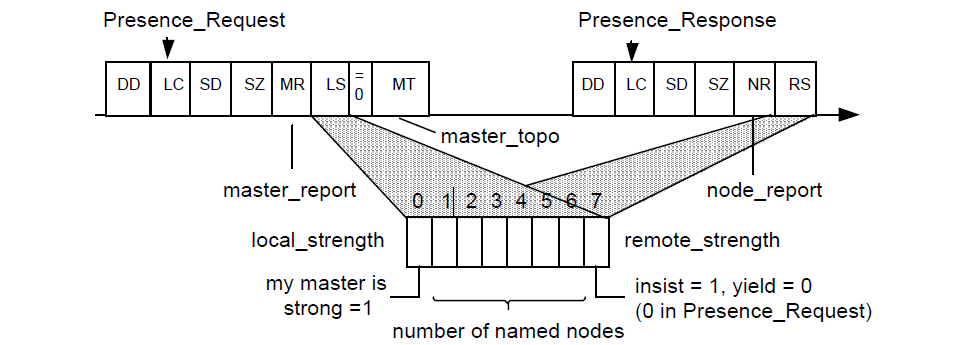
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| master\_report | | | | | | | |
| mas | nns | | | | | | ins |

**Figure 157 – Format of Detect Response**

###### Presence telegram

###### Action

마스터는 그림 158과 같이 종단 노드가 프레즌스 응답으로 응답할 요청을 통해 종단 노드의 프레즌스과 다른 구성의 프레즌스 가능성을 알리도록 종단 노드에 요청한다.



**Figure 158 – Presence telegram**

###### Presence Request

프레즌스 요청은 다음과 같은 형식이다(그림 159):

Presence\_Request::= RECORD

{

destination\_device UNSIGNED8 -- address of End Node

link\_control Link\_Control -- Presence\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- = 4

master\_report Master\_Report -- see 4.7.2.5

local\_strength Composition\_Strength -- copy of LocStr of master,

‘ins’ is ‘0’.

reserved1 WORD4 (=0) -- reserved, = 0

master\_topo Master\_Topo -- see 4.7.2.7.

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| master\_report | | | | | | | |
| mas | Nns | | | | | | ins |
| reserved1 | | | |  | | | |
| master\_topo | | | | | | | |

**Figure 159 – Format of Presence Request**

###### Presence Response

프레즌스 응답은 그림 160과 같이 다음 형식을 가진다:

Presence\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘broadcast’ address

link\_control Link\_Control -- Presence\_Response

source\_device UNSIGNED8 -- address of End Node

link\_data\_size UNSIGNED8 -- =2

node\_report Node\_Report -- see [4.7.2.2](#_bookmark68)

remote\_strength Composition\_Strength -- copy of RemStr of End

Node

‘ins’ = ‘1’ indicates

that the other

composition insists.

}

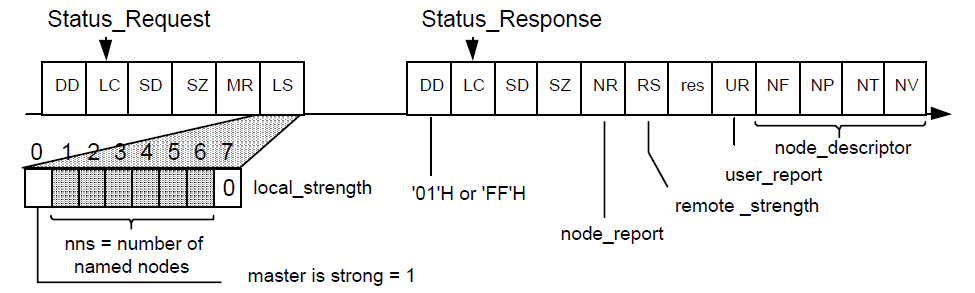
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| node\_report | | | | | | | |
| mas | nns | | | | | | ins |

**Figure 160 – Format of Presence Response**

###### Status telegram

###### Action

마스터는 그림 161과 같이 상태 응답으로 응답하는 슬레이브에게 상태 요청으로 노드의 상태를 요청한다.



**Figure 161 – Status telegram**

###### Status Request

상태 요청은 그림 162와 같은 형식이다:

Status\_Request::= RECORD

{

destination\_device UNSIGNED8 -- node address

link\_control Link\_Control -- Status\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- = 2

master\_report Master\_Report -- see 4.7.2.5

local\_strength Composition\_Strength -- LocStr of master, ‘ins’

is 0

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| master\_report | | | | | | | |
| mas | nns | | | | | | ins |

**Figure 162 – Format of Status Request**

###### Status Response

상태 응답은 그림 163과 같은 형식이다:

Status\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘master’ or ‘broadcast’

address

link\_control Link\_Control -- Status\_Response

source\_device UNSIGNED8 -- node address

link\_data\_size UNSIGNED8 -- = 8

node\_report Node\_Report -- see 4.7.2.2

remote\_strength Composition\_Strength -- remote composition

strength in end node,

0 in intermediate node.

reserved1 WORD8 (=0) -- reserved, =0

user\_report User\_Report -- see 4.7.2.3

node\_descriptor Node\_Descriptor -- see 4.7.2.1

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| node\_report | | | | | | | |
| mas | nns | | | | | | ins |
| reserved1 | | | | | | | |
| user\_report | | | | | | | |
| node\_frame\_size | | | | | | | |
| reserved1 | | | | | node\_period | | |
| node\_type | | | | | | | |
| node\_version | | | | | | | |

**Figure 163 – Format of Status Response**

###### Set to Intermediate telegram

###### Action

마스터는 SetInt 요청을 보내어 슬레이브가 중간 설정에서 스위치를 요구한다. SetInt Request는 그림 164와 같이 SetInt Response로 응답한다.



**Figure 164 – Set-to-Intermediate telegram**

###### SetInt Request

SetInt 요청은 그림 165와 같은 형식이다:

SetInt\_Request::= RECORD

{

destination\_device UNSIGNED8 -- ‘master’ address

link\_control Link\_Control -- SetInt\_Response

source\_device UNSIGNED8 -- node address

link\_data\_size UNSIGNED8 -- =0

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |

**Figure 165 – Format of SetInt Request**

###### 4.5.3.8.3 SetInt Response

SetInt Response는 그림 166과 같은 형식이다:

SetInt\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘master’ address

link\_control Link\_Control -- SetInt\_Response

source\_device UNSIGNED8 -- node address

link\_data\_size UNSIGNED8 -- =0

}

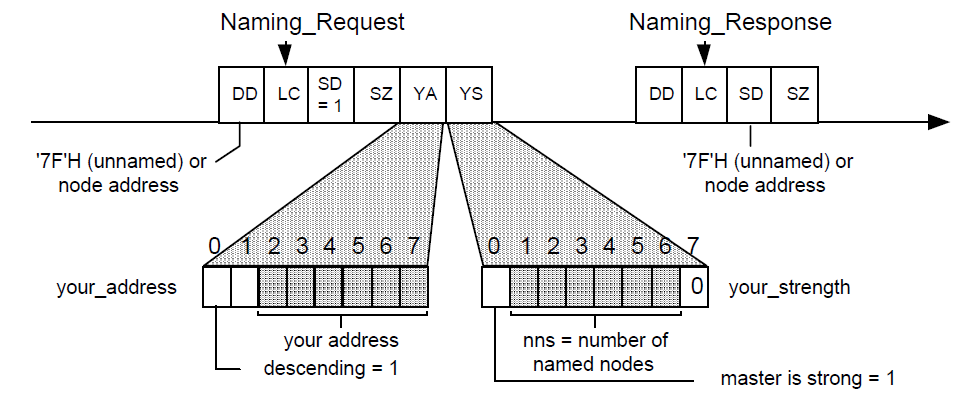
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |

**Figure 166 – Format of SetInt Response**

###### Naming telegram

###### Action

마스터는 할당된 주소와 강도를 네이밍 요청으로 슬레이브에게 전달하며 슬레이브는 그림 167과 같이 네이밍 응답으로 이를 확인한다.



**Figure 167 – Naming telegram**

###### Naming Request

네이밍 요청은 그림 168과 같은 형식이다:

Naming\_Request::= RECORD

{

destination\_device UNSIGNED8 -- node address or ‘unnamed’

address

link\_control Link\_Control -- Naming\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- = 2

dir1 BOOLEAN1 -- ‘1’ if named in ascending

direction

rsv1 WORD1 -- reserved, =0

your\_address UNSIGNED6 -- as assigned by master

your\_strength Composition\_Strength -- as assigned by master

‘ins’ = 0

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| dir1 | rsv1 | your\_address | | | | | |
| mas | nns | | | | | | ins |

Figure 168 – Format of Naming Request

###### Naming Response

네이밍 응답은 그림 169와 같은 형식이다:

Naming\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘master’ address or ‘unnamed’

address

link\_control Link\_Control -- Naming\_Request

source\_device UNSIGNED8 -- ‘node’ address or ‘unnamed’

address (4.5.3.9)

link\_data\_size UNSIGNED8 -- =0

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |

**Figure 169 – Format of Naming Response**

###### Unname telegram

###### Action

마스터는 그림 170과 같이 모든 슬레이브에게 Unname Request를 브로드캐스트하여 자신의 주소를 지우도록 요청한다, 슬레이브 응답은 없어야 한다(이름 없는 응답은 없다).



**Figure 170 – Unnaming telegram**

###### Unname Request

Unname Request은 그림 171과 같은 형식이다:

Unname\_Request::= RECORD

{

destination\_device UNSIGNED8 -- ‘broadcast’ address

link\_control Link\_Control -- Unname\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- =0

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |

**Figure 171 – Format of Unname Request**

###### Set to End telegram

###### Action

마스터는 그림 172와 같이 슬레이브에게 단말 설정으로 전환하고 슬레이브가 SetEnd Response로 응답하도록 SetEnd Request를 보내어 새로윤 조성 강도를 채택하도록 요청한다.



**Figure 172 – Set to End telegram**

###### SetEnd Request

SetEnd 요청은 그림 173과 같은 형식이다:

SetEnd\_Request::= RECORD

{

destination\_device UNSIGNED8 -- node address

link\_control Link\_Control -- SetEnd\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- = 2

reserved1 WORD8 (=0) -- = 0

local\_strength Composition\_Strength -- LocStr as seen by master

(4.7.2.4)

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| reserved1 | | | | | | | |
| mas | nns | | | | | | ins |

**Figure 173 – Format of SetEnd Request**

###### SetEnd Response

SetEnd 응답은 그림 174와 같은 형식이다:

SetEnd\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘master’ address

link\_control Link\_Control -- SetEnd\_Response

source\_device UNSIGNED8 -- node address

link\_data\_size UNSIGNED8 -- = 0

}

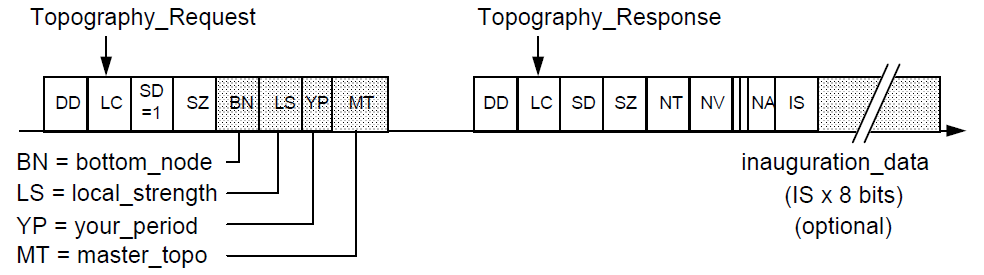
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |

**Figure 174 – Format of SetEnd Response**

###### Topography telegram

###### Action

마스터는 토포그라피에 요청 의해 토포그라피를 슬레이브에게 전달한다. 슬레이브는 그림 175와 같이 토포그라피 응답으로 이를 인지한다:



**Figure 175 – Topography telegram**

###### Topography Request

토포그라피 요청은 그림 176과 같은 형식이다:

Topography\_Request::= RECORD

{

destination\_device UNSIGNED8 -- node address (may be master)

link\_control Link\_Control -- Topography\_Request

source\_device UNSIGNED8 -- ‘master’ address

link\_data\_size UNSIGNED8 -- = 4

bottom\_node UNSIGNED8 -- address of End Node in

Direction\_1 from the master local\_strength Composition\_Strength -- Composition strength seen

by master

(local\_strength.ins = 0). your\_period UNSIGNED4 -- assigned Individual Period

(4.7.2.1 and 4.6.2)

master\_topo Master\_Topo -- see 4.7.2.7

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| bottom\_node | | | | | | | |
| mas | nns | | | | | | ins |
| your\_period | | | |  | | | |
| master\_topo | | | | | | | |

**Figure 176 – Format of Topography Request**

###### Topography Response

토포그라피 응답은 그림 177과 같은 형식이다:

Topography\_Response::= RECORD

{

destination\_device UNSIGNED8 -- ‘broadcast’ address

link\_control Link\_Control -- Topography\_Response

source\_device UNSIGNED8 -- [source](#_bookmark66) node address (may be

master)

link\_data\_size UNSIGNED8 -- 0 • link\_data\_size • 128

node\_type Node\_Type -- first part of Node\_Key

node\_version Node\_Version -- second part of Node\_Key

sam BOOLEAN1 -- ‘1’ if same direction as

master

rsv1 WORD1 -- reserved, =0

node\_address UNSIGNED6 -- node address given by

inauguration

inaug\_data\_size UNSIGNED8 -- 0 • inauguration data size 

124 octets)

inauguration\_data ARRAY [inaug\_data\_size] OF WORD8

application defined

inauguration data

}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| destination\_device | | | | | | | |
| link\_control | | | | | | | |
| source\_device | | | | | | | |
| link\_data\_size | | | | | | | |
| node\_type | | | | | | | |
| node\_version | | | | | | | |
| sam | rsv1 | node\_address | | | | | |
| inaug\_data\_size | | | | | | | |
| ARRAY [inaug\_data\_size] OF | | | | | | | | |
|  | WORD8 | | | | | | | |

**Figure 177 – Format of Topography Response**

주: ‘inaug\_data\_size’와 ‘link\_data\_size’의 중복은 의도적이며 타당성 검사에 사용한다.

###### Medium allocation

* + 1. **Organisation**

다음 사양은 하나의 마스터가 있고 버스가 어플리케이션 프로그램 데이터를 전송할 수 있는 일반 동작에 적용한다. 이노규레이션이 끝나고 조성 변경이 발생하면 정규 운영으로 들어간다.

주: 여러 노드 중 마스터의 선택은 4.7에서 정의하고있다. 마스터의 선택은 여기에 설명된 중간 할당의 일부가 아니다.

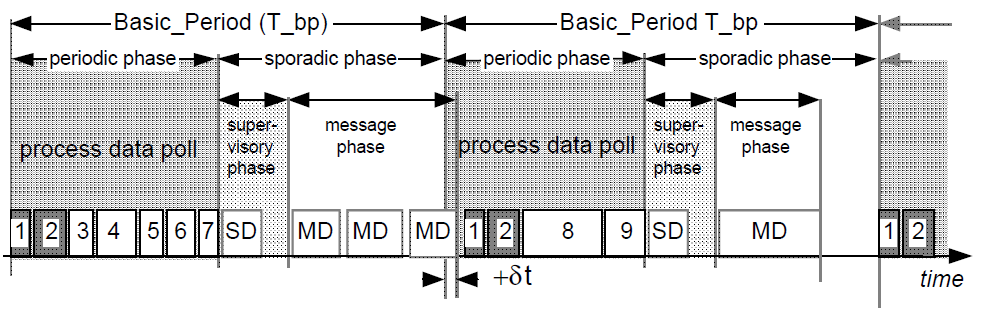
###### Basic Period

* + - * 1. **Structure of the Basic Period**

마스터는 버스 활동을 ‘ ‘Basic Period’라고 하는 고정된 기간으로 나눈다

기본 주기는 그림 178과 같이 두 단계로 나눈다:

1. 주기적인 데이터 전송에 사용되는 주기적 위상
2. 전송에 사용되는 산발적 위상:
   * 관리 데이터; 와/그리고
   * 메시지 데이터.



**Figure 178 – Structure of the Basic Period**

* + - * 1. **Duration of the Basic Period**

기본 주기의 지속 시간은 T\_bp = 25.0ms ±1.0ms이다.

마스터는 기본 주기가 시작된 후 주기적 단계가 완료되지 않는 한 메시지 데이터 요청 또는 관리 데이터 요청을 요구할 수 없다.

주: 이것은 주기가 시작되는 시점이 폴링을 포함하여 가능한 가장 긴 메시지 데이터 또는 관리 데이터 프레임을 전송하는데 걸리는 시간(약 2.0ms)만큼 지연될 수 있다는 것을 의미한다. 다음 기본 주기는 다른 산발적인 데이터가 없을 경우 예정된 시간에 시작될 것으로 예상된다.

###### Periodic Phase

* + - 1. **Individual Period**

동일한 노드의 두 개의 후속 폴링 사이의 간격을 개별 기간 T\_ip라고 한다.

개별 기간은 T\_ip = (2n x T\_bp)가 되도록 기본 주기 T\_bp의 배수이다.

주: 개별 기간은 각 노드의 어플리케이션이 정의한다. 각 노드는 이노규레이션 중에 노드 설명자(4.7.2.1 참조)에서 원하는 개별 기간(node\_period)와 슬레이브 프레임 크기(node\_frame\_size)를 알린다.

버스의 노드 중 가장 긴 개별 기간은 매크로 기간을 정의한다.

###### Periodic List

주기 목록은 매크로 기간의 각 기본 주기에 폴링하는 모든 노드의 목록이다. 그것은 또한 각 기본 주기에 산발적인 단계를 위해 남은 시간을 정의한다.

마스터는 각 노드가 원하는 개별 기간(node\_period)과 이노규레이션 중 각 노드에서 수신한 프로세스 데이터 크기(node\_frame\_size)를 기반으로 주기 목록을 구성한다.

마스터는 기본 주기 동안 전송을 균등하게 분산시켜 기본 주기의 40%를 산발적 주기로 남겨둔다.

주기 기간이 기본 주기의 60%를 초과하는 경우 주기 기간이 기본 기간의 60% 미만이 될 때 까지 가장 긴 기간을 가진 노드의 개별 기간은 두 배가 된다.(???)

이 조치가 충분하지 않은 경우 두 번째 노드 기간이 가장 긴 노드의 기간이 두배가 되며, 최단 기간의 노드 기간이 두 배가 될 때까지 계속된다(필요한 경우).

각 노드에 대해 선택된 개별 기간은 토포그라피 요청에서 각 노드에 your\_period로 전달한다.

###### Polling of the End Nodes

마스터는 프레즌스 요청 프레임(Presence Request)을 송신하여 기본 노드에서 하나의 종단 노드를 폴링하고, 다음 주기에서 다른 종단 노드를 폴링하고, 종단 노드는 프레즌스 응답 프레임(Presence Response)을 브로드캐스트함으로서 응답한다.

주: 프레즌스 신호는 모든 노드가 버스와 마스터의 무결성을 감독하여 다른 구성의 존재를 감지할 수 있다.

###### Error conditions and treatment

마스터는 주기 목록에서 응답을 중지하는 노드를 제거해서는 안되지만 노드가 다음 이노규레이션 때까지 또는 노드가 버스에 다시 접속할 때까지 계속 폴링한다.

주1: 실패한 노드는 새로운 이노규레이션을 통해서만 구성에서 제거될 수 있다.

노드는 연속된 3회의 전송에 응답하지 않는 프로세스 데이터에 가입한 노드의 소멸을 어플리케이션에 알리며 복구되면 다시 접속을 알린다.

주2: 프로세스 데이터에 대한 누락 시간 감독은 누락된 노드의 관리을 제공한다.

예상 트래픽(예: 종단 노드 누락, 누락된 마스터, 폴링 없음)을 관찰하지 않는 정상 작동중인 노드의 동작은 4.7.4.9.3에서 정의한 것과 같다.

###### Sporadic phase

* + - 1. **Event announcement**

노드는 자신의 Process\_Data\_Response또는 Message\_Data\_Response에 ‘A\_bit’또는 ‘C\_bit’를 설정하여 산발적 전송을 요청한다.

몇몇 노드가 주기적 주기 동안 산발적 전송을 알리면 마스터는 이러한 요청을 라운드 로빈 방식으로 처리해야 하므로 같은 노드가 다시 서비스되기 전에 다른 모든 요청을 제공한다.

모든 관리 데이터 요청(‘C\_bit’)은 메시지 데이터 요청(‘A\_bit’)이 발생하기 전에 처리한다.

###### Message List

마스터는 이전 프로세스 데이터 또는 메시지 데이터 프레임 중 하나에 ‘A\_bit’를 세트하는 노드의 주소를 메시지 목록에 삽입한다.

마스터는 메시지 데이터 요청에 의한 변경을 알리는 노드를 폴링하고 노드에서 메시지 데이터를 폴링할 때 메시지 데이터 응답이 항상 A\_bit 세트를 가진 경우를 제외하고 메시지 목록에서 노드를 제거한다.

###### Supervisory List

마스터는 이전 프로세스 데이터 또는 메시지 데이터 프레임 중 하나에 ‘C\_bit’를 세트한 노드의 주소를 관리 목록에 삽입한다.

마스터는 상태 요청에 의한 변경을 알리는 노드를 폴링하고 상태 응답을 받으면 관리 목록에서 그 주소를 제거한다.

주: 이 목록은 일반적으로 무효이다. 버스 길이가 연장된 경우 종단 노드의 주소와 해당 설명자를 변경하거나 슬립 요청 변경을 알리는 노드의 주소를 포함한다(슬립 또는 슬립 취소).

###### Background scanning (option)

마스터는 메시지 리스트 또는 관리 리스트에 포함되지 않은 메시지 데이터 또는 스테이터스 노드를 폴링할 수 있다.

주: 프로세스 데이터를 보내지 않지만 메시지 데이터를 전송할 수 있는 노드는 백그라운드 검색에 의해 폴링된다.

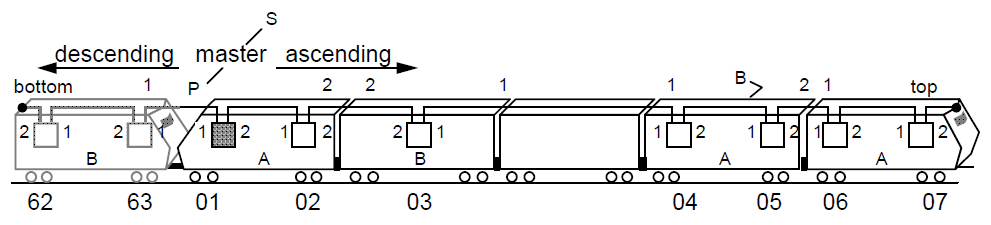
###### Inauguration

###### General

###### Address Allocation

이노규레이션 절차는 그림 179와 같이 각 노드에 주소를 할당한다:

* 마스터에서 Direction\_1의 노드는 63부터 시작하여 내림차순으로 순차적으로 번호가 정해지며 마지막 노드는 맨 아래 노드이다;
* 마스터에서 Direction\_2의 노드는 02부터 시작하여 오름차순으로 순차적으로 번호가 정해지며 마지막 노드는 최상위 노드이다.



**Figure 179 – Node position numbering**

###### Node ranking

어플리케이션은 다음 중 하나의 노드를 순위 지정하여 다른 마스터쉽 전략을 구현할 수 있다:

* strong node, or
* weak node or
* slave node.

###### Strong node

강한 노드가 어플리케이션에 의해 마스터가 되도록 승격된다. 마스터쉽을 행사하는 강한 노드를 강한 마스터라고 한다.

강한 노드는 어플리케이션에 의해 약한 노드로 강등될 수 있다. 마스터가 강하면, 모든 노드로 강등 신호를 보내고 강한 노드가 정해질 때까지 버스를 약한 마스터로 유지한다.

주1: 일반적으로 어플리케이션은 하나의 노드만 구성의 강한 노드로 정한다. 버스 상에 하나 이상의 강한 노드가 존재하면 이노규레이션 절차에 따라 강한 노드가 있는 것과 같이 독립적인 구간으로 버스가 분할된다. 구간당 하나의 마스터만 있을 수 있기때문이다.

주2: 강한 노드는 선행 차량과 같이 특정 적용 기능에 대한 지배를 허용한다. 이러한 결합은 Process\_Data\_Request프레임 (4.5.3.1)에 프로세스 데이터를 포함시키기 위해 필요하다.

###### Weak node

약한 노드는 어플리케이션이 마스터가 되도록 허용한 노드이다. 마스터쉽을 가진 약한 노드를 약한 마스터라고 한다.

일반적으로 어플리케이션은 여러 노드 또는 모든 노드를 구성에서 약한 노드로 지정한다.

강한 노드가 없는 구성에서 모든 노드를 동등하게 처리하는 경합 해결은 약한 노드중 하나만 약한 마스터가 되고 다른 모든 노드는 슬레이브가 되도록 한다.

약한 마스터가 강한 노드 또는 더 많은 슬레이브를 제어하는 다른 약한 마스터의 존재를 감지하면 강등되어 다른 마스터의 슬레이브가 된다.

어플리케이션에 의해 강한 노드로 승격한 약한 노드는 강한 마스터가 되고 이미 마스터가 아닌 경우 버스를 시작한다.

주: 약한 노드는 명시적 어플리케이션 명령없이 버스를 작동할 수 있다. 약한 노드는 이노규레이션을 수행하고 또 다른 (약한)노드를 마스터로 할당함으로서 마스터의 실패를 극복할 수 있다.

###### Slave node

슬레이브 노드는 어플리케이션이 언제든지 마스터가 될 수 있도록 허용하지 않는 노드이다.

주: 슬레이브 노드는 복구에 참여하지 않을 수 있다. 이 모드는 테스트 목적이나 강한 노드와 함께 사용할 때 유용하다.

###### Descriptors

다음 데이터 구조는 관리 데이터 프레임과 버스 관리 메시지에서 사용하며, 전송 표기법에서 정의한다. 링크 계층 인터페이스의 경우, 대응하는 ‘C’-데이터 유형은 4.8.4에 명시하고 있다.

###### Node Descriptor

각 노드는 그 특성을 식별하기 위해 노드 디스크립터를 구현한다.

노드 디스크립터는 그림 180과 같이 다음 데이터 구조로 표현한다:

Node\_Descriptor::= RECORD

{

node\_frame\_size UNSIGNED8, -- ‘link\_data\_size’ of the

Process\_Data\_Response of

this node

reserved1 WORD5 (=0) -- reserved, =0

node\_period UNSIGNED3 -- Individual Period requested

by node, expressed as 2n

multiple of the Basic Period

T\_bp,

n = (1 .. 128):

0: 1 T\_bp (25,0 ms)

1: 2 T\_bp (50,0 ms)

2: 4 T\_bp (100,0 ms)

3: 8 T\_bp (200,0 ms)

4: 16 T\_bp (400,0 ms)

5: 32 T\_bp (800,0 ms)

6: 64 T\_bp (1,6 s)

7: 128 T\_bp (3,2 s)

node\_type UNSIGNED8 -- first part of Node\_Key

node\_version UNSIGNED8 -- second part of Node\_Key

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| node\_frame\_size | | | | | | | |
| reserved1 | | | | | node\_period | | |
| node\_type | | | | | | | |
| node\_version | | | | | | | |

**Figure 180 – Format of Node Descriptor**

주1: 노드 디스크립터는 특정 어플리케이션을 위해 제조자와 사용자간에 합의한다.

주2: 노드 디스크립터는 네이밍 응답, 상태 응답과 토포그라피 응답, 그리고 내부 변수로 사용한다.

주3: UIC Node Descriptorsms UIC 556 리플릿에서 명시하고 있다.

주4: 어플리케이션은 위조(4.8.2.2)로부터 더 큰 보호를 보장하기 위해 각 프로세스 데이터 프레임의 첫 2바이트로 노드 키를 삽입할 수 있다.

###### Node Report

각 노드는 회선 장애와 구성 변경을 보고하기 위해 노드 보고서를 구현한다.

노드 보고는 그림 181과 같이 다음 구조로 표현한다.

Node\_Report::= BITSET8

{

da1 (0) -- Line\_A1 disturbed, copies DA1

of 4.3.2.4

da2 (1) -- Line\_A2 disturbed, copies DA2

of 4.3.2.4

db1 (2) -- Line\_B1 disturbed, copies DB1

of 4.3.2.4

db2 (3) -- Line\_B2 disturbed, copies DB2

of 4.3.2.4

int (4) -- set if node in Intermediate

Setting,

reset if in End Setting.

dsc (5) -- set if the Node Descriptor

has been changed,

reset if the node receives a

new Topography.

slp (6) -- set when a node wants to

enter the low-power mode,

reset when sleep request is

negated.

sam (7) -- set if node has the same

orientation as master

reset if it has the oppo site

direction.

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| da1 | da2 | db1 | db2 | int | dsc | slp | sam |

**Figure 181 – Format of Node Report**

###### User Report

각 노드는 어플리케이션 특정 장애를 보고하기 위해 어플리케이션 특정 옥텟을 다른 어플리케이션에 전송하기 위한 사용자 보고서를 구현한다.

사용자 보고서는 그림 182와 같이 한 옥텟으로 표현한다:

User\_Report::= WORD8 -- application-defined

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| ur7 | ur6 | ur5 | ur4 | ur3 | ur2 | ur1 | ur0 |

**Figure 182 – Format of User Report**

주: 어플리케이션은 링크 계층 서비스를 통해 사용자 보고서의 개별 비트를 세트할 수 있다.

###### Composition strength

각 노드는 구성에 노드가 몇 개인지, 마스터가 강하거나 약한지 여부를 나타내는 Local Composition Strength variable(LocStr) 변수를 구현한다.

각 노드는 Direction\_1 또는 Direction\_2에서 감지한 원격 조성의 강도를 나타내기위해 각 보조 채널의 RemStr (1)과 RemStr (2)에 대한 원격 조성 강도 변수를 구현한다.

주: 이 변수는 해당 채널이 활성화된 경우에만 사용된다. 중간 노드는 이들을 필요로 하지 않는다.

조성 강도는 그림 183과 같이 다음 구조로 표시한다.

Composition\_Strength::= RECORD

{

mas BOOLEAN1, -- set if the master of this

composition is strong

reset if the master is weak.

nns UNSIGNED6, -- number of named nodes in the

composition

ins BOOLEAN1, -- set if the composition

insists in a conflict,

reset if the composition

yields

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| mas | Nns | | | | | | ins |

**Figure 183 – Format of Composition Strength**

강도 비교를 단순화하기 위해 로컬 또는 원격 조성 강도는 다음 값, 부호없는 옥텟으로 계산한다:

RemStr or LocStr = (mas x 128) + 2 x nns + ins;

EXAMPLES

RemStr = 0 no further nodes detected;

RemStr = 1 unnamed node found;

RemStr = 2 yielding lone weak master;

RemStr = 3 insisting lone weak master;

RemStr = 4 yielding weak master with one node;

RemStr = 13 insisting weak master with six nodes;

RemStr > 128 composition named by a strong master.

###### Master\_Report

각 노드는 중복성의 장애를 보고하고 장애의 방향을 식별할 수 있도록 Master\_Report를 구현한다.

Master\_Report는 그림 184와 같이 다음과 같은 구조로 표현한다.

Master\_Report::= BITSET8

{

rsv1 (=0) -- reserved, set to 0

rsv2 (=0) -- reserved, set to 0

rsv3 (=0) -- reserved, set to 0

rsv4 (=0) -- reserved, set to 0

inh -- set if any node of this

composition inhibits inauguration

c12 -- set if this frame is sent in

Direction\_2 relative to this node

dmb -- set if Line\_B of Main Channel

disturbed (copies the DB1 or DB2 bit)

dma -- set if Line\_A of Main Channel

disturbed (copies the DA1 or DA2 bit)

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| rsv1 | rsv2 | rsv3 | rsv4 | inh | c12 | dmb | dma |

**Figure 184 – Master\_Report**

###### Topo Counter

각 노드는 노드에 의해 수신된 각각의 연속 토포크라피에 대해 1씩 증가하는 모듈로64 카운터 값인 Topo Counter를 구현한다. 이 카운터는 0 값을 취하지 않고 63 에서 1 로 직접 카운터한다.

Topo Counter는 그림 185와 같이 다음과 같은 구조로 표현한다.

Topo\_Counter::= UNSIGNED6 -- 0 = not used

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| x | x | topo\_counter | | | | | |

**Figure 185 – Format of Topo Counter**

주: Topo Counter는 이노규레이션이 진행되는 동안 열차 버스를 통해 교환되는 메시지의 일관성을 보장하기 위해 실시간 프로토콜이 사용한다.

###### Master Topo

각 노드는 Master Topo를 구현하는데, 마스터에 의해 분산된 각각의 연속 토포그라피에 대해 1씩 중가하는 12비트 카운터이다.

Master Topo는 그림 186과 같이 다음과 같은 구조로 표현한다.

Master\_Topo::= UNSIGNED12

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| master\_topo | | | | | | | | | | | |

**Figure 186 – Format of Master Topo**

노드가 처음으로 마스터가 될 때, 12비트 마스터 토포는 무작위로 초기화된다.

마스터는 토포그라피를 분배할 때마다 마스터 토포를 1씩 증가시킨다.

주: 이 카운터는 일시적으로 연결이 끊어진 노드가 올바른 구성을 다시 통합하는지 여부를 확인할 수 있게 한다.

###### Inauguration\_Counter

각 노드는 이름 없는 노드의 상태를 통과하여 이노규레이션 절차를 거친 횟수를 등록하기 위해 16비트 Inauguration\_Counter를 구현한다.

주: 이 카운터는 진단에 사용된다.

###### Detection of other compositions (informal)

###### Detection protocol

단말 노드:

* 오픈 엔드에 연결된 추가 노드의 존재를 감지한다; 그리고
* 추가 노드에 자신의 현재 상태를 보고한다.

그 결과 종단 노드는 로컬 조성 강도(LocStr)가 오픈 엔드(또는 단독 마스터의 경우 열린 엔드들)를 포함하는 검색 요청을 보낸다.

이 탐지 요청은 처음에 이 조성이 서로 틀릴 경우 자신의 ‘insist’비트를 설정하여 논쟁을 주장한다.

모든 경우에 종단 노드(이름이 있든 없든)는 프레임 간 간격 제한 시간(4.4.2.4)내에 탐지 응답(또는 다른 탐지 요청)에 의해 수신된 탐지 요청에 응답한다.

수신 종단 노드는 원격 조성의 강도(RemStr)와 자체 강도(LocStr)를 비교한다.

* 다른 조성이 자체 조성보다 약한 경우에는 ‘insist’ 비트가 세트된다;
* 다른 조성이 자신의 것보다 강하거나 같으면 ‘insist’ 비트와 수율을 리셋한다;
* 두 조성에 모두 강한 마스터가 이름을 붙이면 ‘insist’ 비트가 세트된다.

Presence 요청을 수신한 노드는 Presence 응답에 표시된 강도를 채택한다.

종단 노드는 이후의 각 프레즌스 응답에서 보고한다:

1. 다른 노드의 존재;
2. 원격 조성의 강도;
3. 동일한 강점의 경우에 양보하거나 주장할 지역 결정.

###### Collision avoidance rules

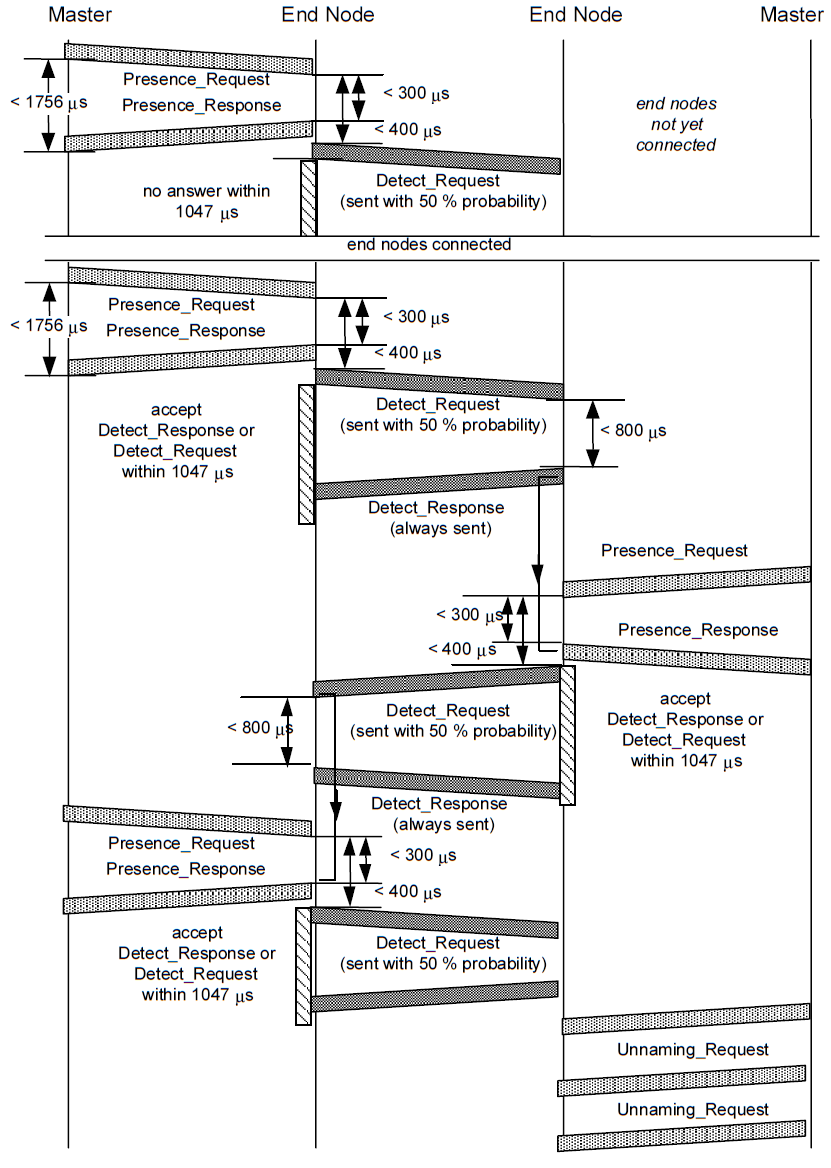
두 개의 개별 조성의 종단 노드가 감지 요청을 비 동기적으로 전송하기 때문에 노드는 응답 감지가 예상될 때까지 깨진 프레임이나 다른 감지 요청을 수신할 수 있다. 다음 다섯 가지 규칙은 경합을 줄인다:

1. 아직 정상 동작 상태에 있지 않은 종단 노드는 자신의 주 채널을 통해 이름 요청 또는 상태 요청을 수신한 후 최신 보조 타이머를 통해 탐지 요청을 보낸다;
2. 정상 동작 중인 종단 노드는 반복된 충돌을 피하기 위해 50%의 확률로 자신의 주 채널을 통해 존재 요청을 수신한 후 최신 400.0µs에 보조 채널을 통해 탐지 요구를 전송한다;
3. 단독 마스터는 적어도 29.0ms 마다 한 번만, 최대 21.0ms 마다 한 번씩 보조채널을 통해 탐지 요청을 보내며, 반복되는 충돌을 피하도록 25.0ms 4.0ms의 범위 내에서 전송을 임의로 한다;
4. T\_detecting\_response = 1.047ms 동안 수신된 검출 응답 또는 검출 요구가 아닌 프레임을 무시하고 그 시간 이후에 수신된 검출 요구 또는 주소 부여 요구가 아닌 프레임을 무시한다;
5. 마스터는 종단 노드가 따라올 수 있는 속도보다 높은 속도로 탐지 요청을 전송하지 않는다. 이는 상태 요청, 프레즌스 요청과 주소 부여 요청에 대한 프로토콜에 의해 보장된다.

주1: 또한 종단 노드인 마스터는 자신에 대한 프레즌스 요청을 보낸다.

주2: 또한 종단 노드인 마스터는 자신에게 상태 요청을 보낸다.

예: 그림 187은 일반적인 감지 프로세스의 타이밍 다이어그램을 보여준다.

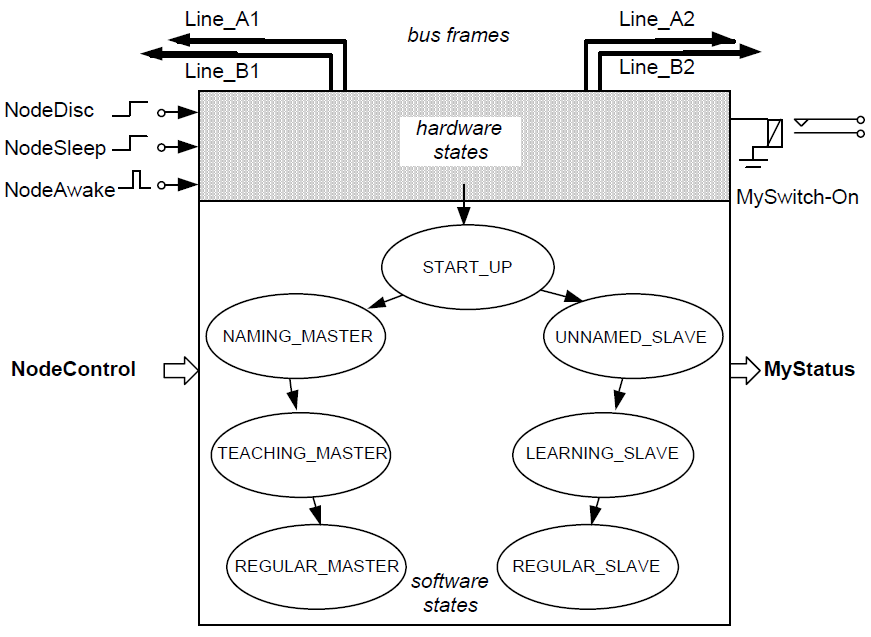


**Figure 187 – Timing Diagram of detection protocol**

###### State diagrams of the inauguration

###### Node structure

노드는 그림 188에 나와있는 주 상태 중 하나에 있다. 이 주 상태는 다음 하위 절에 정의된 작은 상태로 세분된다.



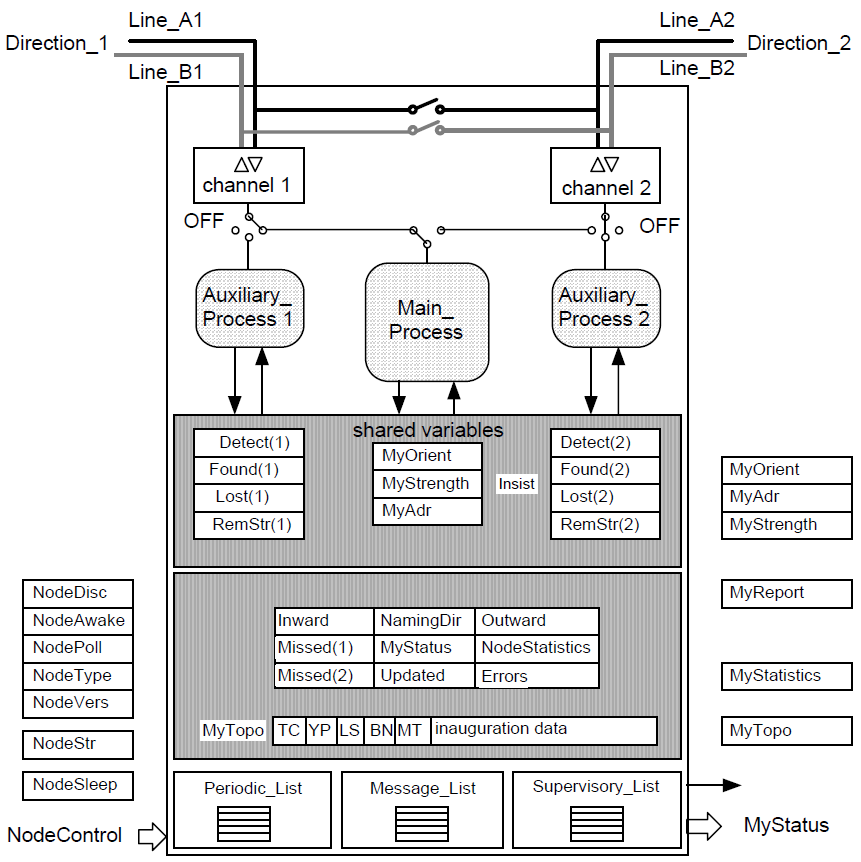
**Figure 188 – Major node states and application settings**

전력 소비를 줄이기 위해 노드가 저 전력, 슬립 상태에 있을 수 있다. 노드가 슬립 모드에서 완전히 깨어나지 않기 때문에 일부 상태와 그 상태는 MAU 하드웨어에서 구현한다. 이들은 하드웨어 상태로 간주하며 해당 제어와 상태 변수는 하드웨어 요소로 상징한다. 다른 상태는 소프트웨어 상태로 간주한다.

###### Inauguration process structure

개념적으로, 그리고 이 사양의 목적을 위해 노드 활동은 그림 189와 같이 세 가지 프로세스로 나뉜다:

* 주소를 가진 모든 노드에서 활성화되는 기본 프로세스. 주소를 가진 슬레이브에서 이 프로세스는 마스터의 방향으로 연결한다. 마스터에서 이 프로세스는 마스터가 지정한 첫 번째 슬레이브에 연결한다;
* 두 개의 동일한 보조 프로세스. 각 프로세스는 한 방향으로 연결한다. 그것들은 종단 노드에서 버스의 열린 쪽 방향으로만 활성화한다.



**Figure 189 – Node processes (End Setting)**

이 명세서의 목적을 위해, 주 프로세스와 보조 프로세스는 주기적 프로세스인 것으로 가정한다. 즉 이들은 소정의 간격(디폴트로 각각의 기본 주기)으로 실행하고, 주기를 끝내기 전에 스스로 중지한다. 하나의 주 프로세스와 보조 프로세스는 다른 노드의 프로세스와 병렬로 실행한다.

프로세스는 서로(그리고 스스로)를 제어한다:

1. 버스를 통해 프레임을 전송함으로서;
2. 타임아웃을 시작하여;
3. 공통(폴링된) 변수에 의해.

프로세스 내의 전환을 트리거한다:

1. 제어 변수에 의해;
2. 버스를 통해 수신된 프레임들에 의해;
3. 경과된 타임아웃에 의해.

상태 전이의 조건은 동일한 프로세스 또는 다른 프로세스의 공통(폴링된) 변수이다.

노드의 상태는 ‘MyStatus’ 구조로 표시한다.

###### Specification language

이노규레이션 절차는 SDL/GR 언어 [ITU-T Z.100]에서 명시하고 있다.

다이어그램을 단순화하기 위해 다음 규칙을 적용한다:

1. SDL에 대한 제한으로서, 이벤트의 큐잉이 없다. 즉 주어진 상태의 노드는 프로세스가 그 상태에 있는 동안 발생하는 이벤트만을 고려한다;
2. 다이어그램을 단순화하기 위해 ‘IF/ELSE’ 문을 포함한다. Boolean은 YES/NO 또는 ‘1’/‘0’ 또는 ‘TRUE’/‘FALSE’를 무의미하게 취한다;
3. 문장과 매크로의 이름은 모두 대문자로 표시한다. 매크로가 하나의 상태만 가질 때 상태는 매크로와 같은 이름을 가지지만 매크로에 대한 진입 점은 하나 뿐이다;
4. 각 상태에 대해, 상태와 동일한 이름을 갖는 타이머(예: NAMED\_SLAVE 상태에 대한 T\_named\_slave)가 관련될 수 있으며, 상태가 입력될 때마다 재 시작된다;
5. 다음 상태가 지정되지 않으면 동작은 시작된 상태로 돌아간다.

###### Node variables

###### NodeControl

노드는 표 63에 지정된 NodeControl 데이터 구조에 의해 제어한다.

**Table 63 – NodeControl data structure**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Meaning** |
| NodeAwake | BOOLEAN1 | NodeSleep과 NodeInhibit가 모두 취소인 경우 노드를 슬립 상태에서 벗어나게 하여 이노규레이션 모드로 참여시킨다. |
| NodeDisc | BOOLEAN1 | 중간 설정에서 노드를 패시브 상태로 전환한다. 파워-업 회로에 의해 취소된다; 일반적으로 전원 공급이 불충분할 때(예: 공칭 값의 70% 미만) 발생한다; 노드가 회복 불가능할 때 이 신호가 인가될 수도 있다. |
| NodeSetUp | RECORD | 명령은 노드에 구성 데이터를 제공하며 특히 다음을 제공한다:  NodeDescriptor와 NodeStrength (slave/weak/strong) |
| NodeInhibit | BOOLEAN1 | 장애로부터 복구할 때를 제외하고 노드가 이노규레이션을 못하도록 한다 |
| NodeSleep | BOOLEAN1 | 종료 설정에서 노드를 저 전력, 슬립 상태로 설정하지만 활동을 감지하면 노드가 꺠어나는 것을 막지 않는다. 일반적으로 배터리 충전이 45분 이상 중단된 경우와 같이 버스를 차단해야 한다고 판단하는 경우 |

###### MyStatus

노드는 표 64에 명시된 MyStatus 데이터 구조를 통해 어플리케이션에 상태를 제공한다.

**Table 64 – MyStatus data structure**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Meaning** |
| MyTopo | RECORD | 최근 토포그라피에서 수신한 토포그라피의 현재 버전; 이 변수는 토포그라피의 사본이다(모든 노드가 동일한지는 수신할 때까지 일관성이 없을 수 있음). |
| MyOrient | ANTIVALENT2 | 노드 위치가 마스터와 같은 방향인지 반대 방향인지를 나타낸다. ‘00’B: 오류, ‘01’B: 같은, ‘10’B: 반대, ‘11’B 정의되지 않음. |
| MyDir | BOOLEAN1 | 주소 지정 요구로 수신한 노드가 마스터와 같은 방향을 가지는 경우 TRUE. |
| MyAdr | UNSIGNED6 | 주소 지정 요구 (your\_address)를 받거나 이름 없는 노드의 ‘unnamed’주소, 또는 마스터에 ‘마스터’주소를 받는 것 |
| MyReport | NodeReport | 노드 상태 응답 프레임(4.5.3.7)에서 노드가 마스터에 보낼 형식의 보고서. |
| MyStrength | Composition\_Strength | 보조 채널을 통한 주소 지정 요청 또는 주 채널을 통한 SetEnd 요청, 상태 요청 또는 토포크라피 요청에 의해 수신된 해당 노드의 로컬 강도. |
| MySwitchOn | BOOLEAN1 | 노드의 전원이 켜지면 ‘1’, 저 전력으로 전환하면 ‘0’. |
| MyStatistics | RECORD | 송수신 프레임과 오류 카운터 통계s |

###### Shared variables

주 프로세스와 보조 프로세스는 공유 변수에 대한 정보를 교환한다.

이들 변수는 각각의 프로세스에 의해 폴링하는 것으로 가정한다. 즉 주 프로세스와 보조 프로세스 사이에 비 동기 신호가 없다.

여러 프로세스에서 변수에 동시에 액세스할 수 있으므로 변수를 LOCK과 UNLOCK으로 잠그면 변수를 일관되게 읽고 수정할 수 있다(예: 이름 없는 노드가 양방향에서 동시에 이름이 지정되지 않도록).

인덱싱된 변수(1,2)는 각 Auxiliary\_Process에 고유하다. MyStatus 데이터 구조의 모든 변수는 공유 변수로 간주한다. 다른 공유 변수는 표 65에서 지정한다.

**Table 65 – Shared Variables of a node**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Meaning** |
| Detect (1)  Detect (2) | BOOLEAN1  BOOLEAN1 | Direction\_1에서 감지가 활성화한 경우 ‘1’  Direction\_2에서 감지가 활성화한 경우 ‘0’ |
| Found (1) Found(2) | BOOLEAN1 BOOLEAN1 | 인가:   * 다른 노드를 찾은 Auxiliary\_Process에 의해, 또는 * 0과 다른 RemStr이 있는 Presence Response에 의해   취소:   * 노드가 그 방향으로 주어질 때; * 노드가 중간 또는 * 노드가 이름 없음이 되었을 때 |
| Lost(1)  Lost(2) | INTEGER8  INTEGER8 | 이미 탐지되었지만 이름이 지정되지 않은 노드의 손실을 탐지 하는 카운터 |
| Insist | BOOLEAN1 | 종단 노드가 양 방향을 추구하면 ‘1’,  어느 방향이 더 강한 조성을 감지하면 ‘0’ |
| LocStr | Composition\_Strength | 노드가 속한 조성의 강도 |
| RemStr(1) RemStr(2) | Composition\_Strength Composition\_Strength | 자신의 Auxiliary\_Process에 의해 감지되거나 감시 프레임의 remote\_strength 필드에서 수신된 다른 조성의 강도 |

###### Main Process Variables

표 66에 나열된 변수는 주 프로세스에서 사용한다. 다른 로컬 변수는 해당 매크로 또는 프로시저의 SDL 다이어그램에 나타난다.

**Table 66 – Variables of Main Process**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Meaning** |
| Topo | RECORD | 토포그라피, 주소 지정된 노드 당 하나의 항목 포함 |
| TopoCounter | Topo\_Counter | see 4.7.2.6 |
| MasterTopo | Master\_Topo | see 4.7.2.7 |
| Inward | ANTIVALENT2 | 마스터 방향 (주소 지정된 슬레이브를 위해) |
| Outward | ANTIVALENT2 | 마스터와 반대 방향 (주소 지정된 슬레이브를 위해) |
| NamingDir | ANTIVALENT2 | 마스터의 이름 지정 (0 = 없음, 1 = 아래, 2 = 위) |
| Missed(1)  Missed(2) | UNSIGNED8  UNSIGNED8 | 각 방향에서 종단 노드의 손실을 감지한다. |
| Errors | UNSIGNED8 | 오류 카운터는 바람직하지 않은 결과로 증가하고 그 같은 발견의 유리한 결과로 인해 0으로 재 설정된다. 따라서 각 종류의 발견에 대해 암묵적으로 개인화되었다. |
| C\_bit | BOOLEAN1 | 이 비트가 세트되어 있으면 모든 Process\_Data\_Responses 또는 메시지 데이터 응답이 C\_bit 세트를 전달한다 |

마스터 작동은 표 67에 지정된 목록에 따라 다르다.

**Table 67 – Lists of Main Process**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Meaning** |
| Periodic\_List [n] | RECORD | List of the addresses to be polled for Process Data in a  particular Basic Period, n being the Basic Period position in the Macro Period |
| Message\_List | RECORD | List of nodes to be polled for Message Data, ordered in the order as which the transmission requests have been  detected (signalled by the A\_bit) |
| Supervisory\_List | RECORD | List of nodes to be polled for Supervisory Data, ordered in the order as which the transmission requests have been detected (signalled by the C\_bit) |

* + - 1. **Auxiliary\_Process**
         1. **Process**

그림 190과 같이 Auxiliary\_Process 조성에 속하지 않는 노드가 있는지 감지한다. 종단 설정에서의 노드만 실행한다. 고립된 노드에는 두 개의 활성 Auxiliary\_Processes가 있다. 프레임 교환은 보조 채널을 통해 이루어진다.

Auxiliary\_Process는 두 가지 상태로 구성한다.

노드가 주소 지정되고 양보하지 않는 경우에만 입력되는 DETECTING RESPONSE 상태와 Auxiliary\_Process가 활성 상태로 유지되는 DETECTING RESPONSE상태가 있다.

###### State ‘DETECTING\_RESPONSE’

양보하지 않는 주소 지정 노드는 탐지 요청을 전송하고 DETECTING\_RESPONSE 상태의 응답 또는 탐지 요청을 기대한다.

그것은 DETECTING\_RESPONSE 상태를 기대할 것이다.

1. 타임아웃 T\_detecting\_response
   * DETECTING\_REQUESTS로 이동하라;
2. 감지 응답 또는 감지 요청:
   * 원격 조성의 존재 여부와 강도를 기록한다( (Found (this\_dir) = ‘1’; Lost = 0),
   * 각각의 힘을 비교하고 다른 구성이 강하면 ‘insist’ = ‘0’, 그리고
   * DETECTING\_REQUESTS로 이동한다;
3. 다른 유형:
   * 무시하고 DETECTING\_REQUESTS로 이동하라.

주1: Detect Request는 슬레이브가 보낸 유일한 마스터 프레임이다.

주2: 어떤 경우에는 노드가 무조건 종료될 수 있다.

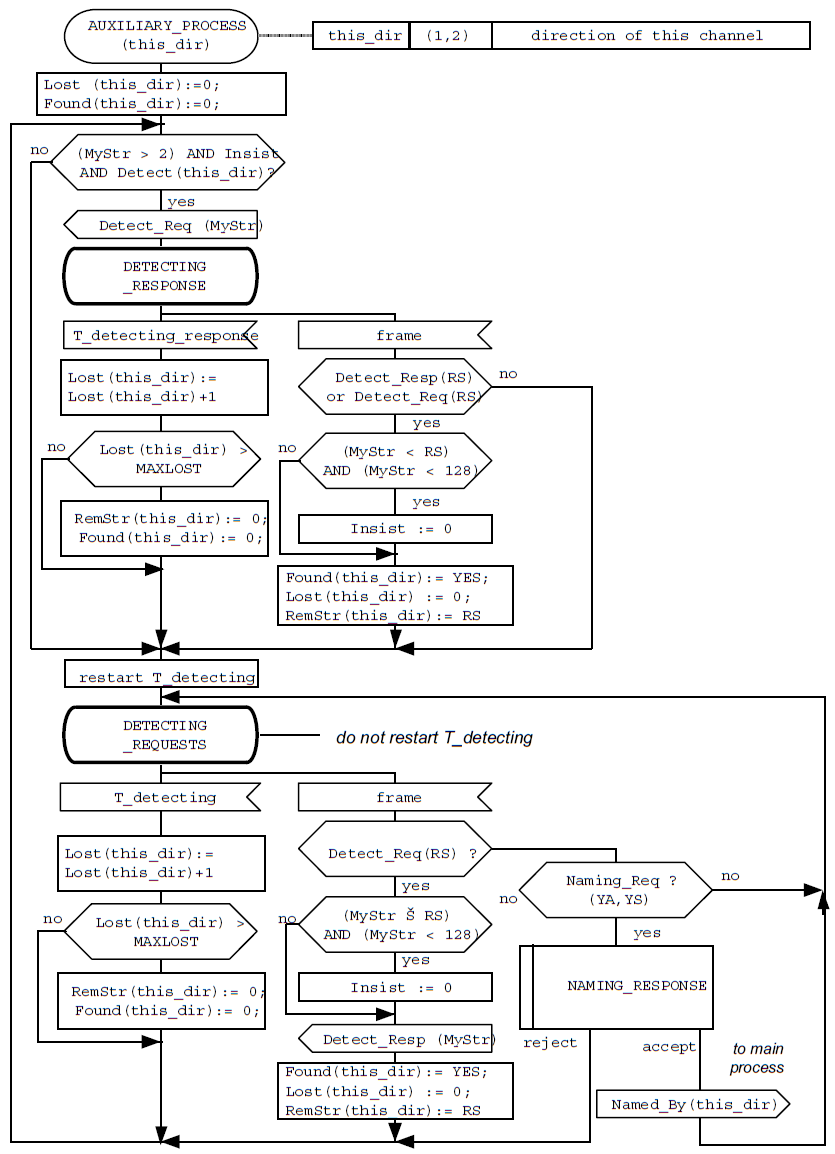
###### State ‘DETECTING\_REQUESTS’

Auxiliary\_Process는 DETECTING\_REQUESTS 상태에서 기대할 것이다:

1. 타임아웃 T\_detecting:
   * 노드가 T\_detecting 동안 프레임을 수신하지 않은 경우, 노드는 이 방향의 ‘Lost’ 카운터를 증가시키고 ‘DETECTING\_RESPONSE’ 상태로 이동한다;
   * 노드가 MAXLOST 연속 탐지 기간 동안 프레임을 수신하지 않은 경우 노드는 변수 ‘Found’를 리셋하고 ‘RemStr(this\_dir)’을 0으로 세트하며 ‘DETECTING\_RESPONSE’ 상태로 이동한다;
2. Detect Request:
   * 다른 노드 (Found(this\_dir) = ‘1’, Lost = 0)의 존재를 등록한다;
   * 각 강점을 비교하고 요청자가 강하거나 동등한 강점을 가진 경우 ‘Insist’를 ‘0’으로 설정하여 강하지 않은 경우를 산출하고; 그리고
   * 감지 요청을 보낸다;
3. 주소 지정 요청:
   * 이 전에 발견된 더 강한 조성의 감지 요청 (‘Found(1)’ , respectively ‘Found(2)’ =‘0’)을 받지 못한 경우, 이 프레임이 방해받는 것으로 수신되고 다른회선이 이미 방해를 받더라도 다른 회선을 신뢰하다면 회선을 정의한다;(???)
   * 그렇지 않으면 NAMING\_RESPONSE 매크로를 실행해야 하고 성공하면 Auxiliary\_Process 를 종료한다.
4. other type of frame:
   * 이 프레임이 방해받는 것으로 수신되고 다른회선이 이미 방해를 받더라도 다른 회선을 신뢰한다면 회선을 정의하고 T\_detecting의 리셋없이 DETECTING\_REQUEST로 이동한다.

주1: 감지 요청과 주소 지정 요청은 노드가 보조 채널을 통해 기대하는 유일한 유형의 프레임이다. 노드가 이전에 발견 요청을 수신하지 않고 주소 지정 요청을 수신하면 이것은 프로토콜 오류로 해석한다.

주2: 감지 요청은 충돌에 의해 왜곡되었을 수 있으므로 반복되는 충돌을 피하기 위해 T\_detecting 시간은 중앙 값으로 무작위 화한다.



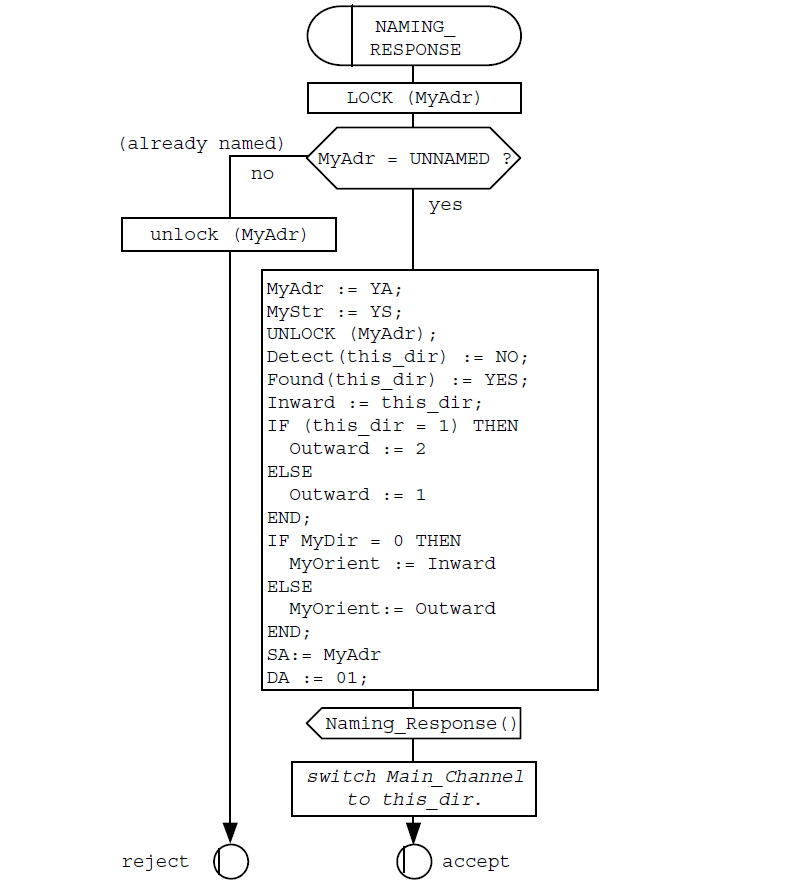
**Figure 190 – AUXILIARY\_PROCESS states**

* + - * 1. **Macro ‘NAMING\_RESPONSE’**

그림 191에서 볼 수 있듯이 ‘NAMING\_RESPONSE’ 매크로는 노드에 주소를 할당한다:

* 노드가 이미 주소를 가진 경우 주소 지정 요청을 무시한다. 이 상황은 노드가 다른 채널을 통해 주소 지정된 경우에만 발생해야 한다;
* 노드의 주소가 지정되지 않은 경우 노드는 주소 지정 응답으로 응답한다 그 방향으로 감지를 멈추고 채널을 주 채널에 할당한다.

주: 노드 이름이 없음을 확인하면 다른 Auxiliary\_Process도 노드에 액세스할 수 있으므로 MyAdr이라는 독점적인 읽기 동작이 필요하다.

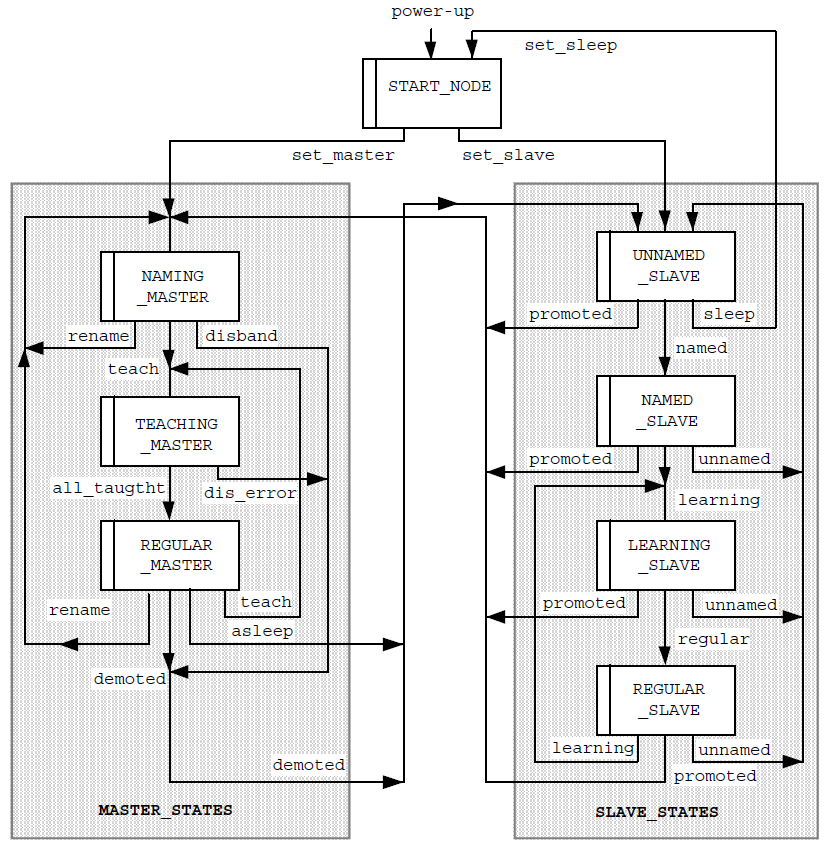


**Figure 191 – NAMING\_RESPONSE macro**

###### Major states of the Main Process

그림 192에서 나타나듯 주요 프로세스는 주요 상태로 구분한다.

이러한 상태는 매크로(한번만 표시됨) 또는 프로시저(여러 번 나타날 수 있음)로 표시하는 여러 상태로 세분되며 다음 절에서 설명한다.



**Figure 192 – States of MAIN PROCESS**

The START\_NODE state is described in Table 68.

**Table 68 – ‘START\_NODE’**

**‘START\_NODE’**

노드는 장애, 리셋 또는 전원 공급 후 ‘START\_NODE’상태로 전환한다.

노드가 수동이거나 버스 또는 어플리케이션 신호가 최대 전력으로 복원되기를 기다리는 저 전력 상태에서 전원이 차단된 상태에서 시작한다.

그런 다음 NodeSetUp 명령을 통해 어플리케이션이 노드를 구성한다. 노드가 강하게 구성되어 있으면 NAMING\_MASTER 상태로 되고 약한 노드 또는 종속 노드로 구성되면 UNNAMED\_SLAVE 상태가 된다.

노드가 마스터로 작동하는 상태는 표 69에 나와있다.

**Table 69 – ‘MASTER STATES’**

|  |  |
| --- | --- |
| **‘NAMING\_MASTER’** | 단독 마스터로 노드를 초기화한다. 두 감지 채널이 활성화된 상태에서 종단 설정의 노드를 설정한다. 다른 노드를 검색하기 위해 양 방향으로 감지 프레임을 전송하기 시작한다.  마스터가 노드를 감지하면 주소를 지정하고 Node Status 목록을 갱신한다. 어느 방향으로도 이름 없는 노드가 발견되지 않으면 마스터는 이름 지정을 닫고 ‘TEACHING\_MASTER’ 상태로 간다..  약한 마스터가 더 강한 구성을 발견하면 강등되며 ‘UNNAMED\_SLAVE’ 상태로 돌아간다. 이전에 지정된 노드의 이름을 해제해야 한다 (disband).  마스터가 복구할 수 없는 오류를 발견하면 다시 시작하여 프로시저를 재 가동한다. 법적인 상황일 수 있으므로 이 상태에는 시간 제한이 없다. (lone vehicle). |
| **‘TEACHING\_MASTER’** | 마스터는 각 노드에 하나씩 설명자와 이노규레이션 데이터를 다른 모든 노드에 브로드캐스트하도록 요청하여 토포그라피를 배포한다.  배포가 성공하면 마스터는 ‘REGULAR\_MASTER’ 상태가 되고  배포가 실패하면 ‘NAMING\_MASTER’ 상태로 들어간다. |
| **‘REGULAR\_MASTER’** | 마스터는 정기적으로 노드를 폴링한다.  마스터는 다음과 같은 예외적 상황을 처리한다:   * 어떤 노드의 설명 또는 상태 변경; * 마스터 강도의 변화.   마스터는 아래에서 ‘REGULAR\_MASTER’ 상태를 벗어난다:   * 버스 생략 (종단 노드가 더 이상 감지되지 않음); * 버스 연장 (추가 노드의 주소 지정) |

노드가 슬레이브로 작동하는 상태(SLAVE states)는 표 70에 나열되어 있다.

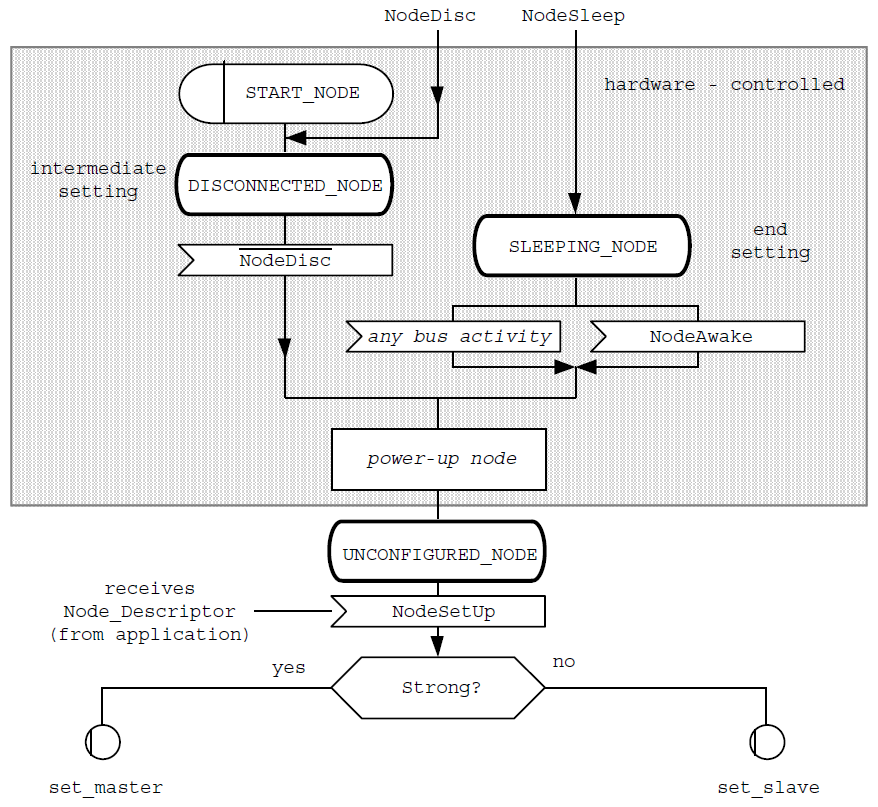
**Table 70 – ‘SLAVE STATES’**

|  |  |
| --- | --- |
| **‘UNNAMED\_SLAVE’** | ‘UNNAMED\_SLAVE’는 Auxiliary\_Processes를 모두 수신(전송하지 않음)하는 종단 설정에 있다. 마스터로부터 주소 지정 프레임을 받거나 어플리케이션이 강하거나 약한 노드로 승격할 때 이 상태를 벗어난다. |
| **‘NAMED\_SLAVE’** | 주소 지정 요청은 노드가 주 채널을 마스터쪽으로 전환하도록 한다. 다른 방향의 Auxiliary\_Process는 활성 상태로 유지된다. 마스터는 개방 노드의 추가 노드를 확인하기 위해 정기적으로 상태 요청을 보낸다.  이름 없는 노드를 추가로 발견하면 노드를 중간 설정을 전환하여  Auxiliary\_Process를 끈다.  토폴로지 요청 또는 응답을 수신하면 노드는 이 상태를 떠나며 이는 주소 지정 단계의 끝을 알린다. |
| **‘LEARNING\_SLAVE’** | 주소 지정된 슬레이브는 모든 다른 노드에서 이노규레이션 데이터를 수신하고 자신의 이노규레이션 데이터를 다른 모든 노드에 브로드캐스트한다. 이노규레이션 카운터를 1씩 증가시킨다.  노드는 프레즌스 요청을 수신할 때 이 상태를 떠나며 마스터 신호는 규칙적인 동작을 입력하거나 정규 동작을 나타내는 모든 프레임을 입력한다. |
| **‘REGULAR\_SLAVE’** | 이 상태에서 노드는 전체 토폴로지로 갱신된 것으로 간주된다.  노드가 종단 노드라면 활성 Auxiliary\_Process는 버스 연장을 점검하여 프레즌스 응답을 통해 마스터에 보고한다.  모든 노드는 두 노드의 존재를 관리한다. 세 개의 프레즌스 응답에 해한 종단 노드를 차례로 감지하지 못하면 노드는 ‘UNNAMED\_SLAVE’ 상태로 돌아간다.  갱신된 경우 노드는 프로세스 데이터에 대해 정기적으로 폴링되고 다른 노드의 Process\_Data를 수신할 것이다.  Topography\_Response 또는 Topography\_Request를 수신하면  LEARNING\_SLAVE 상태로 돌아가며 마스터는 주소 변경 없이 조성 변경을 알린다. |

모든 슬레이브 상태에서, 슬레이브는 강한 마스터 상태로 승격되어 ‘NAMING\_MASTER’ 상태로 직접 전달될 수 있다.

###### Macro ‘START\_NODE’

그림 193에서 보듯이 이 매크로는 노드가 마스터 또는 슬레이브가 되기 전에 통과하는 하드웨어 제어와 소프트웨어 제어 상태를 정의한다.



**Figure 193 – Macro ‘START\_NODE’**

###### State ‘DISCONNECTED\_NODE’

어플리케이션이 NodeDisc를 인가할 때 노드가 전원을 끊거나 노드가 심각한 손상을 입었음을 알리면 노드는 무조건 이 상태가 된다.

이것이 전원이 공급되지 않는 노드의 초기 상태이다. 상태 ‘DISCONNECTED\_NODE’에 있는 노드는 중간 설정에 있다.

어플리케이션이 노드에 전원을 공급하고 NodeDisc 신호를 리셋하면 노드는 이 상태를 유지한다. 그런 다음 ‘UNCONFIGURED\_NODE’ 상태가 된다.

###### State ‘SLEEPING\_NODE’

노드는 NodeSleep 명령이 인가되고 NodeDisc 신호가 취소되면 해당 상태로 들어간다.

노드는 저 전력 상태에 있는 종단 설정 상태에 있다.

노드는 다음과 같은 상태를 유지한다:

1. 어플리케이션이 ‘NodeAwake’ 명령을 인가할 때, 또는
2. 버스 활동을 감지할 때.

노드가 슬립 상태가 될 때, 노드를 완전히 켜고 ‘UNCONFIGURED\_NODE’ 상태로 가기 위해 ‘MySwitchOn’을 인가한다. MySwitchOn 신호는 ‘DISCONNECTED\_NODE’ 와 ‘SLEEPING\_NODE’를 제외한 모든상태에 대해 하드웨어로서 설정한다.

주1: 이 상태로 들어가면 버스를 중단한다. 버스에서 활동이 있으면 노드가 다시 깨어난다. 따라서 어플리케이션은 다른 노드가 노드를 다시 깨우지않도록 ‘NodeSleep’ 신호를 충분히 길게 유지해야 한다.

주2: 완전한 전원 손실이 노드를 중간 설정으로 가져오므로 버스 스위치에 활동적이 된다.

주3: ‘Any bus activity’은 SEQ 신호가 없는 Carrier\_Sense(4.3.1.5 참조) 또는 올바른 형식의 프레임이 어느 채널을 통해 수신되었는지를 나타내는 다른 모든 수단으로 정의한다.

###### State ‘UNCONFIGURED\_NODE’

노드는 종단 설정 상태이며 노드 설명자와 이노규레이션 데이터를 기다린다.

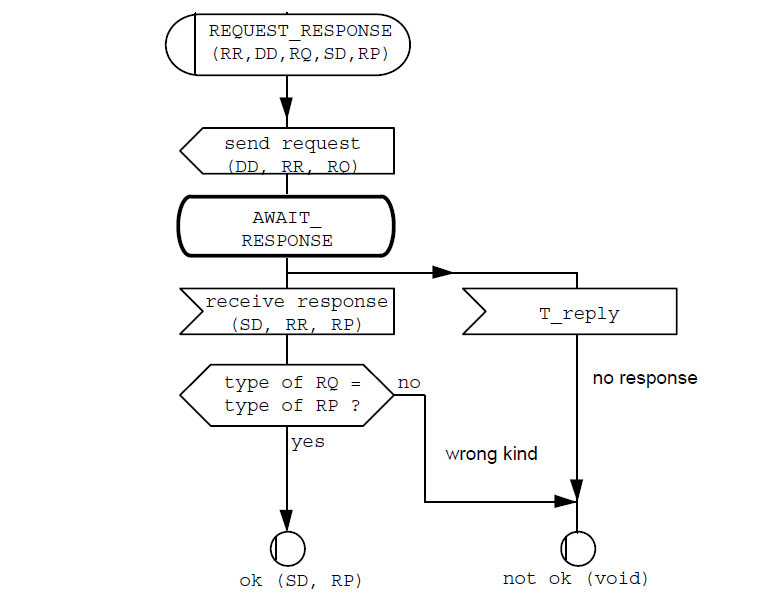
노드는 유효한 노드 설명자가 있는 NodeSetUp 명령을 수신한 후 UNCONFIGURED\_NODE 상태를 벗어난다,

그것은 강한 노드라면 NAMING\_MASTER 매크로로, 약한 노드 또는 슬레이브 노드라면 UNNAMED\_SLAVE 매크로로 갈 것이다.

###### Master States

###### Procedure REQUEST\_RESPONSE

그림 194와 같이 프로시저는 요청을 보내고 해당 응답을 기다린다..



**Figure 194 – Procedure REQUEST\_RESPONSE**

The parameters are:

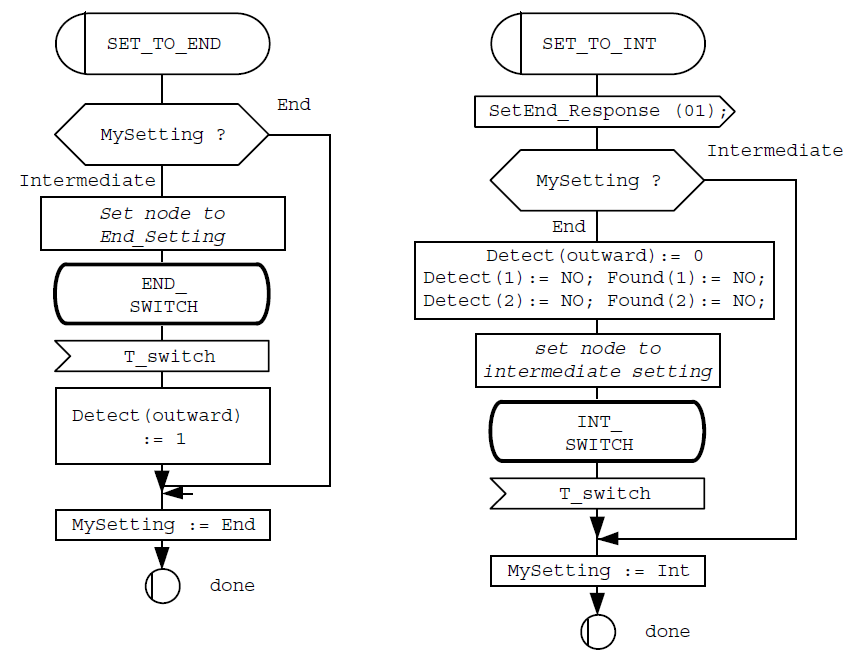
1. RR: Request/Response type, one of: {Presence, Status, Naming, Topography};
2. DD: Destination\_Device;
3. SD: Source\_Device;
4. RP: Response parameters (depends on RR, according to the frame definition);
5. RQ: Request parameters (depends on RR, according to the frame definition).

This procedure exits:

* with the response of the slave, or
* after the T\_reply time-out elapses.

###### Procedures ‘SET\_TO\_END’ and ‘SET\_TO\_INT’

그림 195에서 보듯이 이 두 절차는 노드를 중간 설정으로, 각각 종단 설정으로 설정한다(아직 해당 상태에 있지 않은 경우).



**Figure 195 – Procedures ‘SET\_TO\_INT’ and ‘SET\_TO\_END’**

Detect(1) 또는 Detect(2) 변수는 각각 ‘AUXILIARY\_PROCESS’의 시작과 정지를 제어한다.

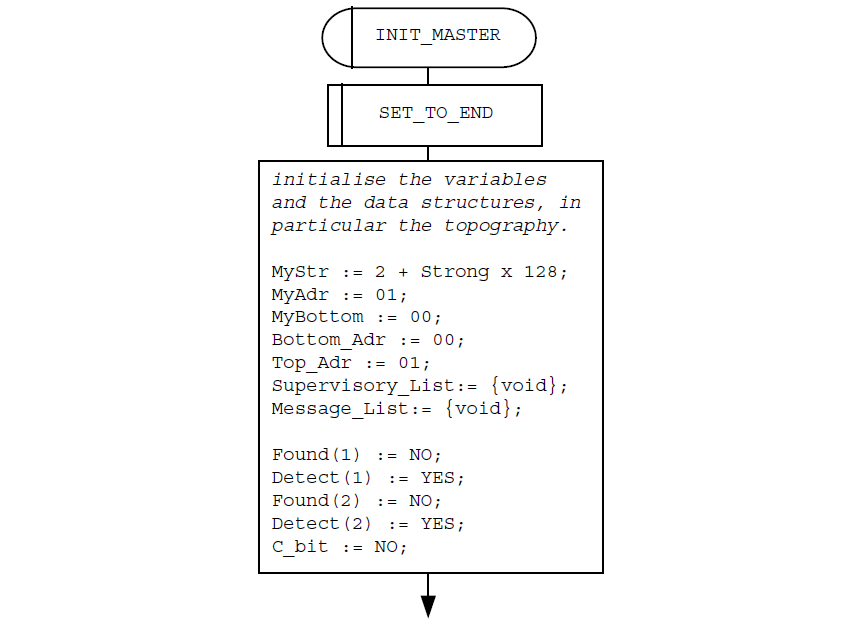
종료할 노드가 마스터 자체이거나 노드가 주소가 지정되지 않은 경우 Detect(1)과 Detect(2)가 둘 다 ‘1’이 된다.

###### Macro ‘INIT\_MASTER’

그림 196에서 보듯이, 이 매크로는 강한 마스터에 대한 슬레이브 승격의 결과로 또는 노드가 없는 슬레이브가 버스 활동 부족으로 마스터가 된 결과로서 노드를 마스터로 구성하도록 한다.

노드는 먼저 종단 설정에서 자신을 초기화하고(아직 설정하지 않은 경우) 주소와 강도를 설정하며, 그리고 void 토폴로지를 설정하고 두 감지 채널을 모두 활성화하여 감지 요청을 전송한다. 이 구성에서 마스터는 폴링하는 것처럼 작동한다.

설정이 완료되면 마스터는 찾을 수 있는 다른 이름의 주소를 지정할 준비를 한다.



**Figure 196 – Macro ‘INIT\_MASTER’**

###### Macro ‘NAMING\_MASTER’

그림 197에서, 이 매크로는 마스터가 다른 노드의 주소를 지정하는 상태를 포함한다. 마스터는 초기화하는 INIT\_MASTER 매크로를 실행하여 이 상태로 들어간다.

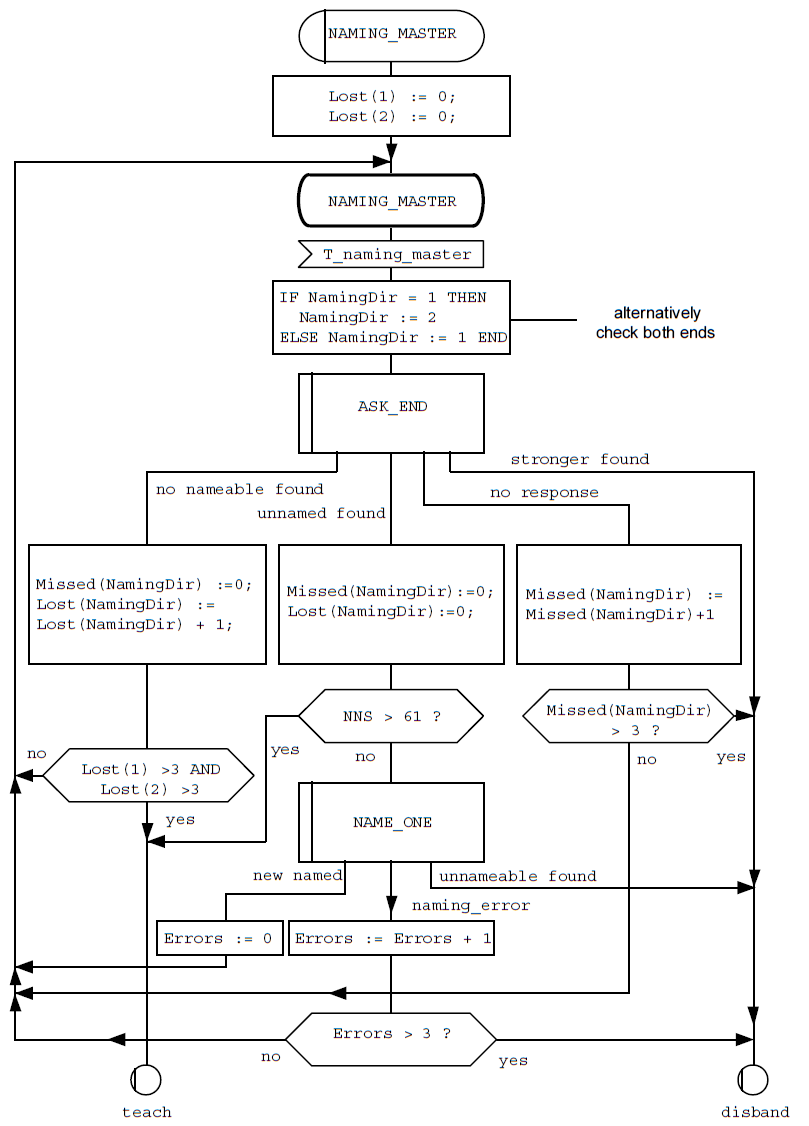
마스터는 ‘AWAIT\_PERIOD’ 상태에서 T\_naming\_master 주기의 시작을 기다린다. T\_naming\_master 지연이 이미 경과한 경우 마스터는 즉시 진행한다.

마스터는 ASK\_END 매크로를 실행한다. ASK\_END 매크로는 다른 노드가 조성을 발견했는지 묻고, 해당 매크로의 결과에 따라 동작한다:

1. 감지된 조성이 강하면 마스터는 UNNAMING\_MASTER 매크로로 이동하여 조성을 해제한다;
2. 감지된 조성이 현재 조성보다 약하거나 동등한 강도를 갖는다면, 마스터는 다른 구성이 해체될 때 까지 기다린다(이것은 정의되지 않은 시간이 걸린다);
3. 종단 노드가 응답하지 않으면 마스터는 오류를 등록하고 나중에 다시 시도한다. 동일한 방향으로 3 개의 연속된 오류가 발생하면 UNNAMING\_MASTER 매크로를 실행하여 해체한다;
4. 종단 노드가 이름 없는 노드를 보고하는 경우, 마스터는 노드를 새로운 종단 노드로 지정한다;
5. 동일한 방향으로 3 회의 연속된 폴에 대해 주소 지정된 노드가 발견되지 않으면 TEACHING\_MASTER 상태로 진행한다.

주1: 마스터는 T\_naming\_master 주기당 하나의 노드를 주소 지정한다.

주2: NAMING\_MASTER 상태에서 이노규레이션은 항상 허용한다.



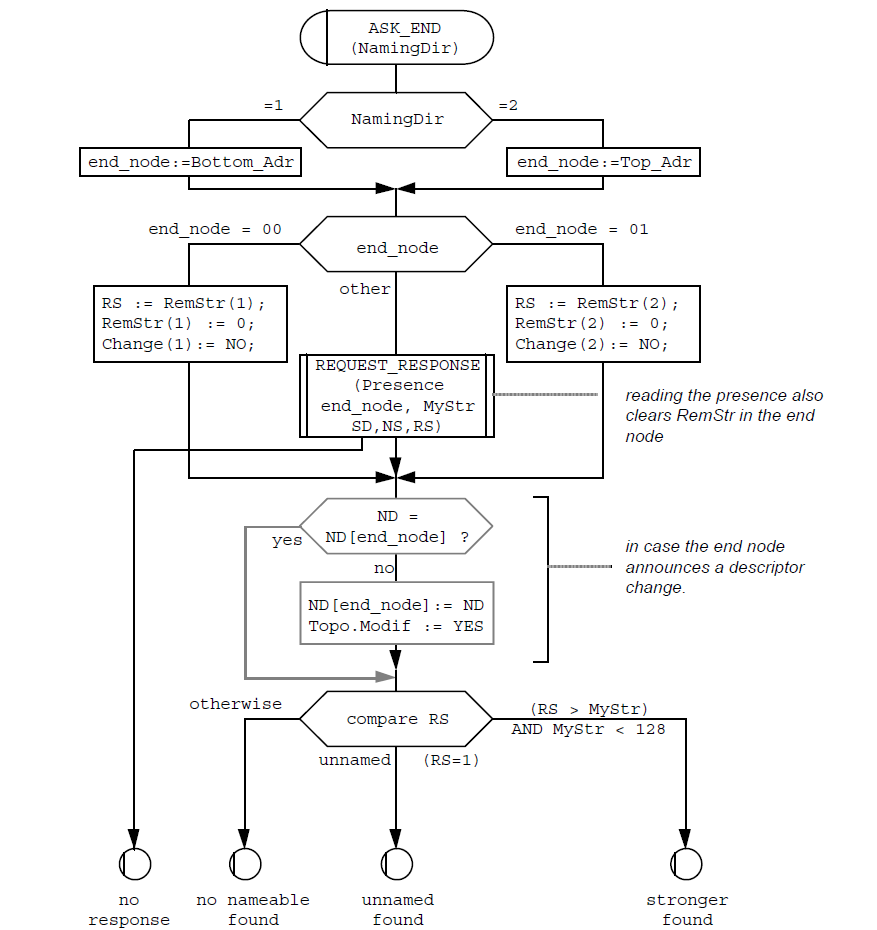
**Figure 197 – Macro ‘NAMING\_MASTER’**

###### Macro ‘ASK\_END’

그림 198에서 보듯이 이 매크로는 ‘NamingDir’ 방향으로 종단 노드의 존재를 검사하고 가능한 원격 구성을 보고하도록 요청한다. 이것은 마스터가 그 자체로 종단 노드인 경우를 암시적으로 포함한다.

감지된 원격 강도에 따라 마스터는 세 가지 경우를 구분한다:

1. 주소 지정 가능한 노드가 없다 (노드가 없거나 이미 지정된 노드);
2. 이름 없는 노드가 발견되었다(원격 노드는 주소 지정을 수락한다);
3. 강한 구성 발견(원격 노드가 더 강한 조성에 속함).



**Figure 198 – Macro ASK\_END**

###### Procedure ‘NAME\_ONE’

그림 199에서와 같이, 종단 노드가 주소가 없는 노드의 존재를 알릴 때 ‘NAMING\_MASTER’ 매크로 프로시저를 호출한다.

이 프로시저에는 주소 지정의 방향인 파라메터가 있다.

마스터가 자체로서 종단 노드이고 이미 다른 노드에 노드를 명명한 경우 버스를 통해 명령을 보내지 않고 중간 설정으로 이동한다.

그렇지 않은 경우, 마스터는 종단 설정 상태를 유지하고 종단 노드에 이름 지정 방향으로 SetInt 요청을 보낸다.

SetInt 응답을 수신하지 못하면 마스터는 T\_switch를 기다렸다가 SetInt 요청을 반복한다.

세 번의 시험 후에 SetInt 응답을 받지 못하면 마스터는 ‘naming error’로 이 절차를 종료한다.

그렇지 않으면 SetInt 응답을 수신한 후, 마스터는 T\_switch를 기다린 다음 노드의 주소를 ‘your\_address’로, 조성 강도를 ‘your\_strength’로 다음과 같은 프로토콜을 사용하여 주소 지정되지 않은 노드로 지정 요청을 보낸다;

1. 마스터는 Destination\_Device = ‘unnamed’인 첫 번째 시험에 대해 주소 지정 요청을 전송한다;
2. 첫 번째 시도에 대해 응답을 수신하지 못한 경우 마스터는 T\_aux\_main 시간을 대기하고 Destination\_Device가 할당된 노드 주소인 주소 지정 요청을 다시 전송한다;
3. 두 번째 시도에 대한 응답을 받지 못하면 마스터는 Destination\_Device 가 ‘unnamed’ 주소인 주소 지정 요청을 다시 전송한다;
4. 세 번째 시도에 대한 응답을 받지 못하면 마스터는 T\_aux\_main 시간을 대기하고 Destination\_Device가 할당된 노드 주소인 주소 지정 요청을 다시 전송한다;
5. T\_reply내에서 네 번째 시도에 대한 응답을 받지 못한 경우, 마스터는 Destination\_Device가 ‘unnamed’ 주소인 주소 지정 요청을 다시 전송한다;
6. 다섯 번째 시도에 대한 응답을 받지 못하면 마스터는 T\_aux\_main 시간을 대기하고 Destination\_Device 가 할당된 주소인 주소 지정 요청을 다시 전송한다;
7. 여섯 번째 시도까지 T\_reply내에서 응답을 받지 못하면 마스터는 SetEnd 요청을 전송하여 이전 종단 노드를 종단 설정으로 복원한 다음 NAME\_ONE 프로시저를 ‘unnameable\_found’ 상태로 두기전에 스위치가 다시 열리도록 T\_switch를 대기한다.

그렇지 않고 주소 지정이 성공하면 T\_aux\_main을 대기하고 새로 지정된 노드에 상태 요청을 보낸다.

마스터가 신뢰할 수 있는 회선에서만 상태 응답을 수신하고 관리 회선이 방해을 받으면 마스터는 상태 요청을 반복하지 전에 1.2ms 기다린 다음(상태 응답을 기다린다) 검사하기 위해 상태 요청을 세번 반복한다. 회선 품질을 검사하고 결과를 다음 Master\_Report에 전달한다.

3 회의 상태 요청 전송에 대한 응답을 받지 못하면 마스터는 ‘unnameable\_found’ 상태로 프로시저를 종료한다. 그렇지 않고, 상태 응답을 수신하면 토포그라피, 종단 노드의 주소와 조성 강도를 실현하고 ‘new\_named’를 종료한다.

주1: 이름 없는 노드는 주소 지정 요청을 수신하고 그 보조 채널을 통해 응답한다

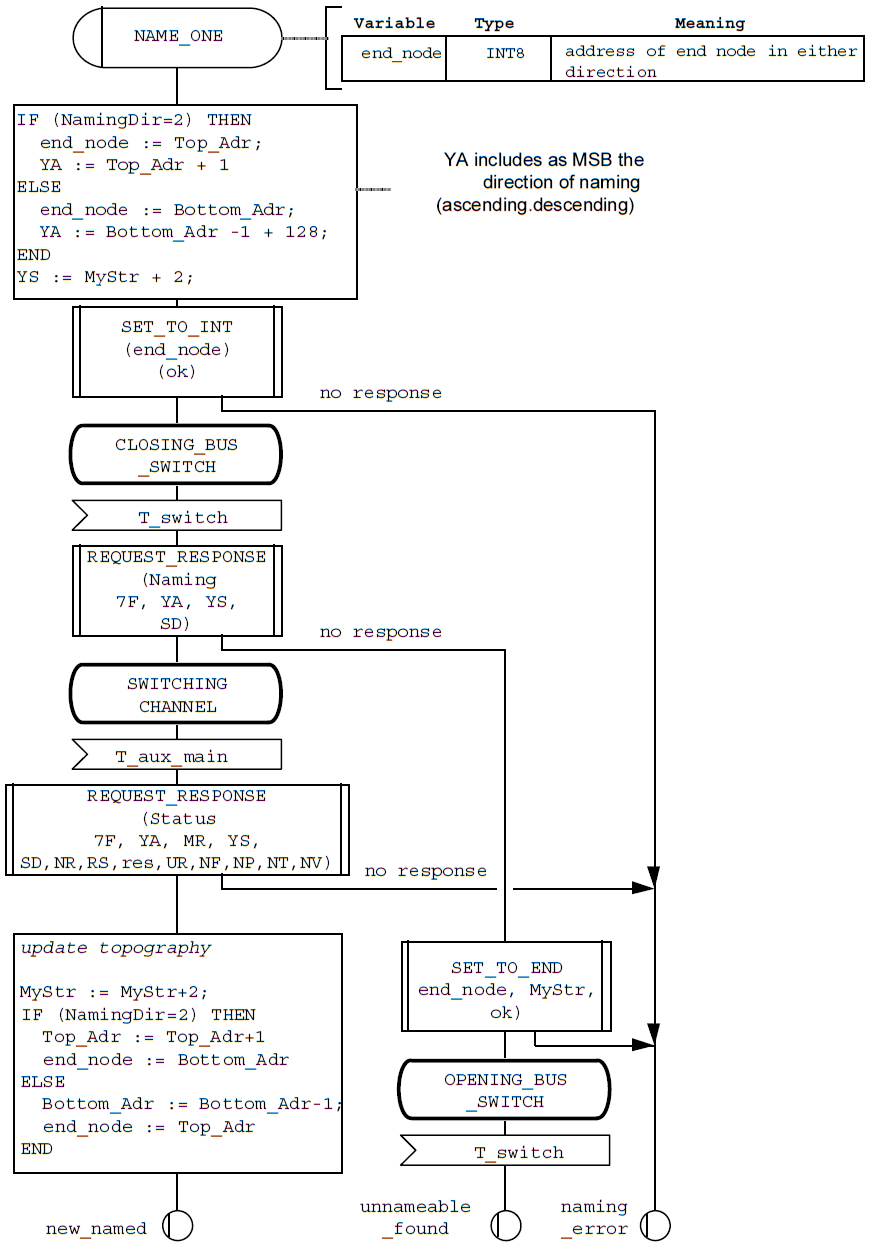
주2: T\_aux\_main 지연은 보조 채널에서 주 채널로 전환할 수 있게 한다. 이 시간 동안 트래픽이 혼란스럽다.

주3: 새로 주소 지정된 노드는 주소 지정 응답으로 응답하여 주소 지정을 수락하거나 응답하지 않음으로서 주소 지정을 거부한다.

주4: 노드의 주소가 지정되었지만 응답이 손실된 경우 Destination\_Device가 ‘unnamed’인 시도는 실패한다. 반대로 주소 지정이 되지않으면 할당된 노드 주소인 Destination\_Device 시도가 실패한다.

주5: 주소 지정된 노드는 노드 설명자와 감지된 추가 노드의 원격 강도가 있는 상태 응답을 반환한다.

주6: 장애가 발생한 경우 상태 요청을 3 회 반복하면 마스터는 프레임 손실과 자체 - 종단 노드 사이의 하나의 중복 회선 손실을 구별할 수 있다.



**Figure 199 – Procedure NAME\_ONE**

###### Macro ‘TEACHING\_MASTER’

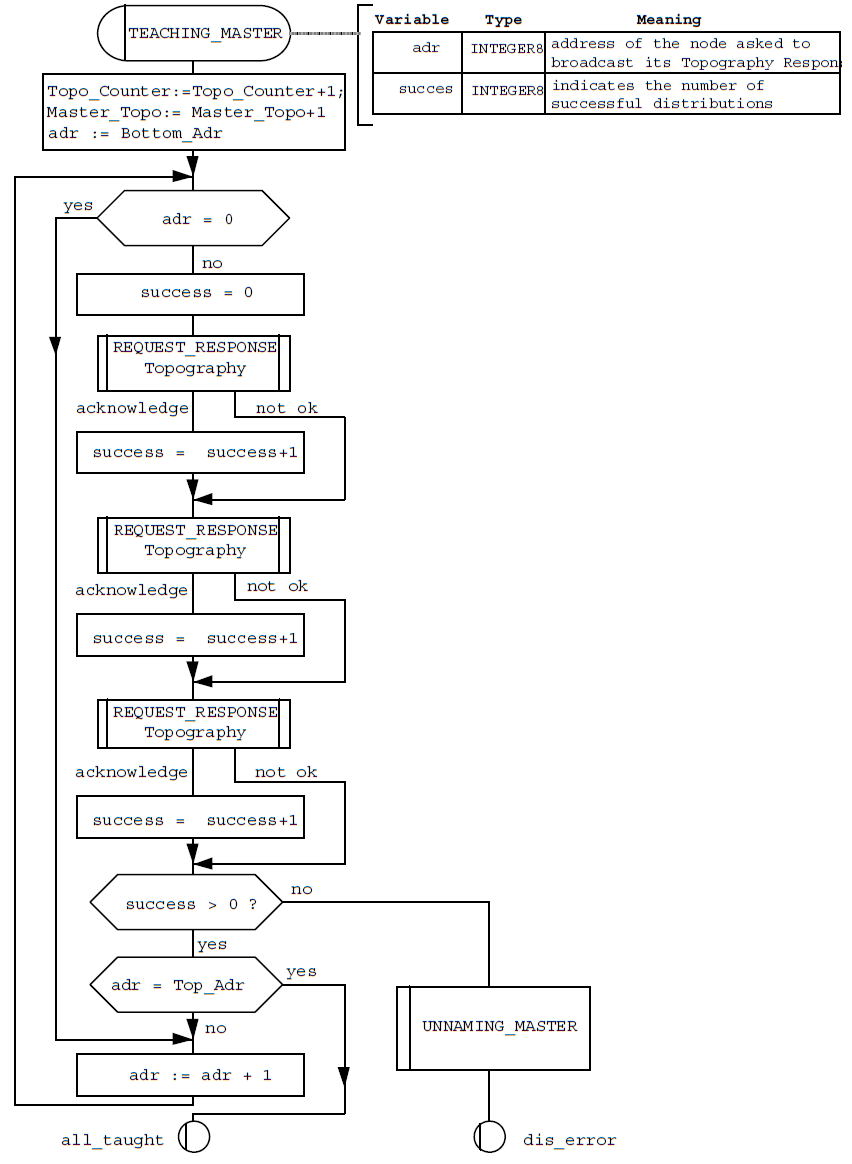
그림 200에서와 같이 마스터는 주소 지정 노드가 더 이상 발견되지 않거나 노드 설명자의 변경을 알리는 노드를 감지하면 토포크라피 정보를 모든 지정된 노드에 배포한다.

이 효과를 위해 TopoCounter와 MasterTopo를 증가시킨다.

노드는 아래 쪽 노드부터 시작하여 자신을 포함하여 최상위 노드로 끝나는 모든 주소 지정된 노드에 순서대로 토포그라피 요청을 세번 보낸다(자체 폴링).

노드는 다음과 같은 상태를 유지한다:

* 토포그라피 요청을 세 번 보낸 후에 응답을 받지 못한 경우, UNNAMING\_MASTER 매크로로 이동한다;
* 모든 노드에서 토포크라피 응답을 수신한 경우 REGULAR\_MASTER 매크로로 이동한다.

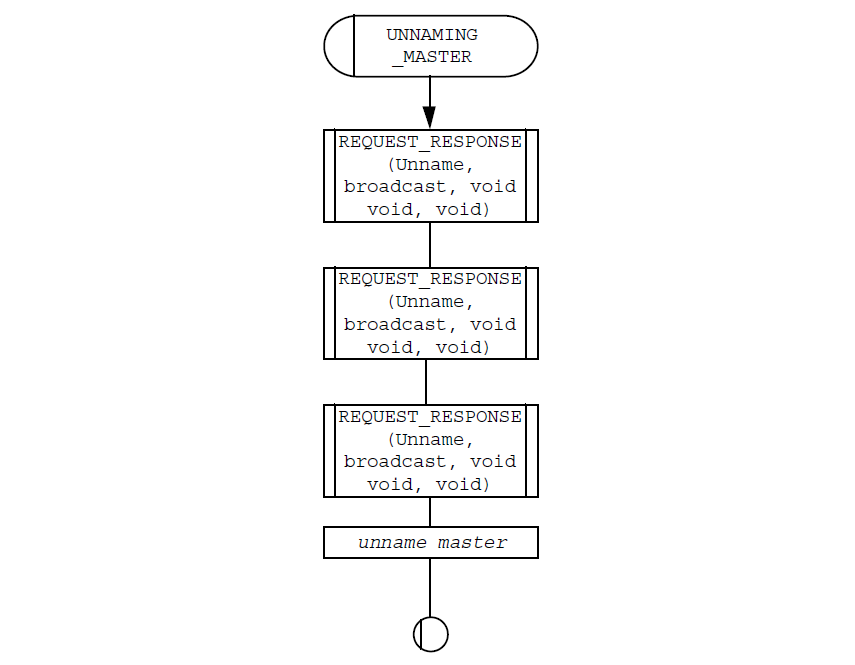


**Figure 200 – Macro TEACHING\_MASTER**

###### Macro ‘UNNAMING\_MASTER’

그림 201과 같이 마스터는 모든 노드의 주소를 지정 해제하여 조성을 해제한다. 이 효과를 위해 3 개의 Unname 요청을 1.0ms 이내에 순차적으로 브로드캐스트할 것이고, 강한 노드라면 NAMING\_MASTER 매크로로 가고 아니면 UNNAMED\_SLAVE 매크로로 간다.

주: 지연은 모든 노드가 Unname 요청을 수신하는데 필요하며, 슬레이브는 종단 설정으로 가기전에 동일한 지연을 관찰한다.



**Figure 201 – Macro ‘UNNAMING\_MASTER’**

###### Macro ‘REGULAR\_MASTER’

그림 202와 같이 마스터는 정상적으로 작동한다.

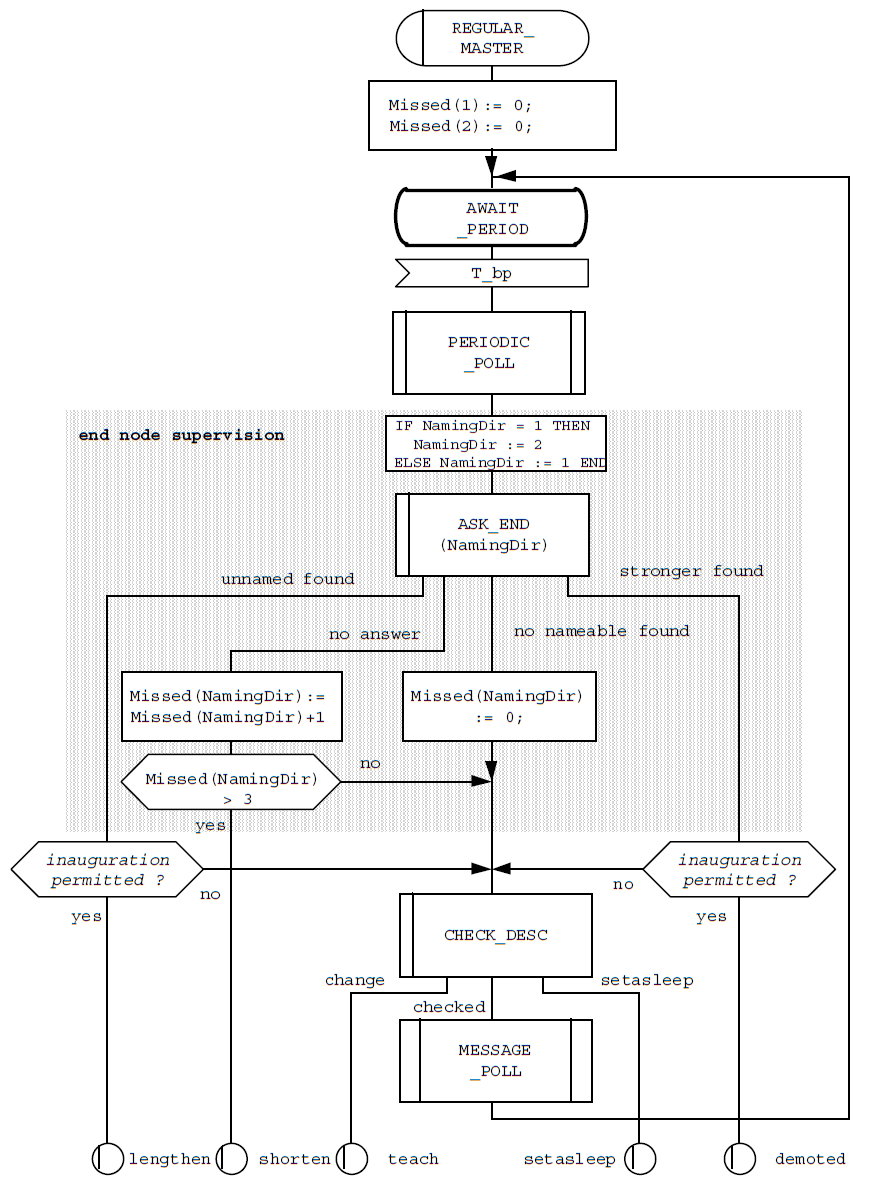
‘REGULAR\_MASTER’ 매크로는 세 개의 블록으로 나뉜다:

1. ‘PERIODIC\_POLL’: 마스터는 자신의 Periodic\_List에서 노드를 폴링하여 프로세스 테이터를 찾는다. 이 단계는 4.7.4.8.11에 자세히 나와있다;
2. ‘SUPERVISORY\_POLL’:
   * 마스터는 기본 주가마다 한 노드에 Presence\_Request를 보낸다. 같은 방향으로 4 개의 연속된 Presence\_Request사이의 간격은 6.5 x T\_bp보다 작다. 이를 위한 편리한 방법은 Presence\_Request를 기본 주기의 고정된 비율(예: 각 Basic\_Period의 시작)로 보내고,
   * 마스터는 ‘C’ 비트를 발생시킨 노드의 상태를 확인한다. 이 점검은 Periodic\_Phase 이후에 이루어진다.
3. ‘MESSAGE\_POLL’: 마스터는 4.7.4.8.12에 명시한 대로 Periodic\_Phase가 시작되기 전에 남은 시간 동안 메시지 데이터에 대한 노드를 폴링한다.

마스터는 ‘REGULAR\_MASTER’ 매크로를 벗어난다:

1. 3 회의 연속 폴 동안 종단 노드를 감지하지 못하면 그 조성을 해체한 다음 ‘NAMING\_MASTER’로 이동한다;
2. 다른 이름의 조성이 있음을 감지하고 이노규레이션이 활성화 된 경우 UNNAMING\_MASTER를 실행하여 해당 구성을 해체한 다음 ‘NAMING\_MASTER’ 매크로로 이동한다;
3. 조성 변경을 감지하고 ‘TEACHING\_MASTER’ 매크로로 이동하면;
4. UNNAMED\_SLAVE로 가기 전에 이노규레이션이 설정된 상태에서 자신보다 강한 조성을 발견하고 조성을 해제하면;
5. 어플리케이션 명령으로 슬립 상태이거나 연결이 끊어진 다음 ‘UNNAMED\_SLAVE’ 매크로로 이동한 경우

주: ‘UNNAMED\_SLAVE’는 다른 모든 노드도 이 상태에 있을 때 슬립 모드로 전환한다.

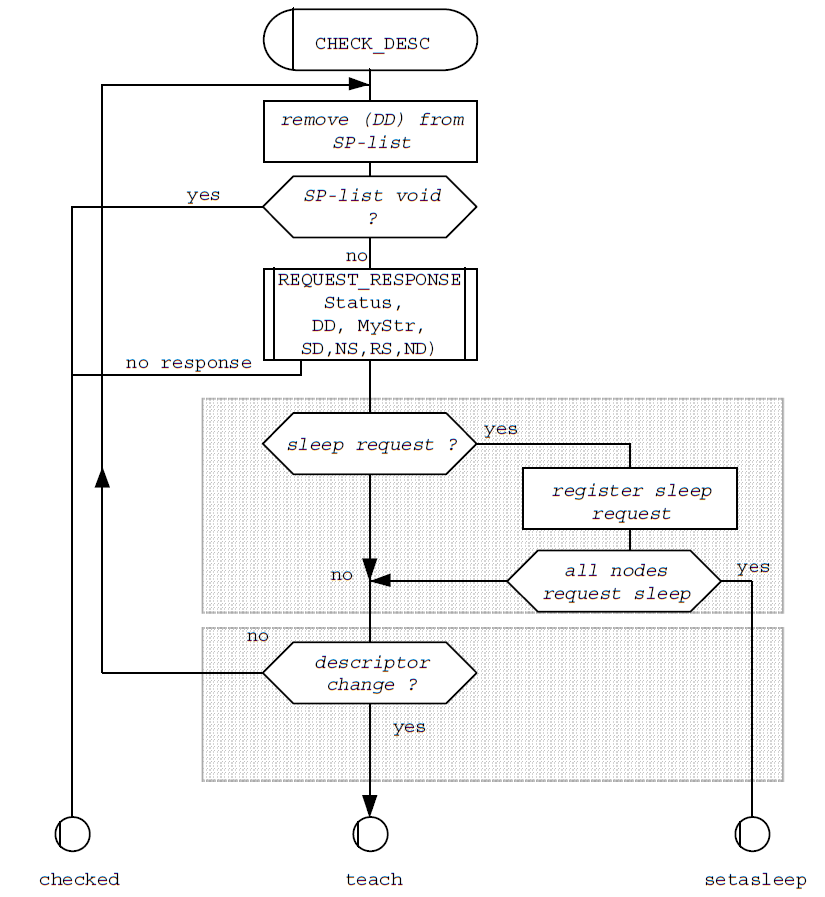


**Figure 202 – Macro ‘REGULAR\_MASTER’**

###### Macro ‘CHECK\_DESC’

그림 203에서, 이 매크로는 설명자를 변경하거나 슬립 모드로 전환하도록 요청한 Intermediate\_Node를 확인한다.

종단 노드의 경우 종단 노드가 추가적으로 버스 길이를 연장할 수 있기 때문에 별도로 처리한다.



**Figure 203 – Macro CHECK\_DESC**

###### Macro ‘PERIODIC\_POLL’

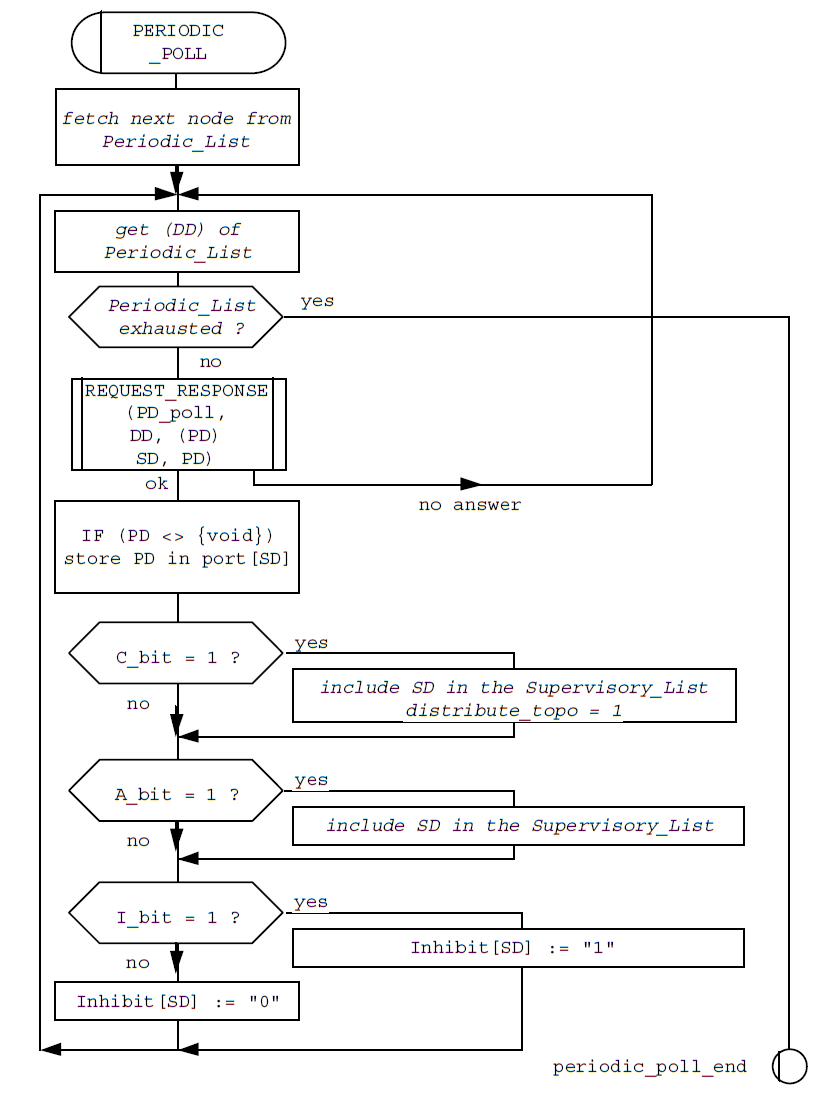
그림 204와 같이 마스터는 Periodic\_List에 따라 노드를 스캔하고 이 주기에 폴링할 주소를 나열한다.

프로세스 데이터에 대한 노드를 폴링할 때 마스터는 다음을 수행한다:

1. 폴링된 노드의 존재와 프로세스 데이터를 기록한다;
2. 노드 설명자를 변경한 노드로부터 조성 변경(C\_bit)을 기록한다;
3. A\_bit를 인가함으로서 메시지 데이터 전송에 필요로 하는 노드를 기록한다;
4. ‘I\_bit’에 의해 이노규레이션을 금지하는 노드를 기록한다.

마스터는 Process\_Data\_Request를 수신하지 않으면 동일한 기본 주기에서 Process\_Data\_Response를 반복하지 않는다.

마스터가 자신을 폴링하면 Process\_Data\_Response를 보내기 전에 Process\_Data\_Request를 자신에게 보낸다.



**Figure 204 – Macro PERIODIC\_POLL**

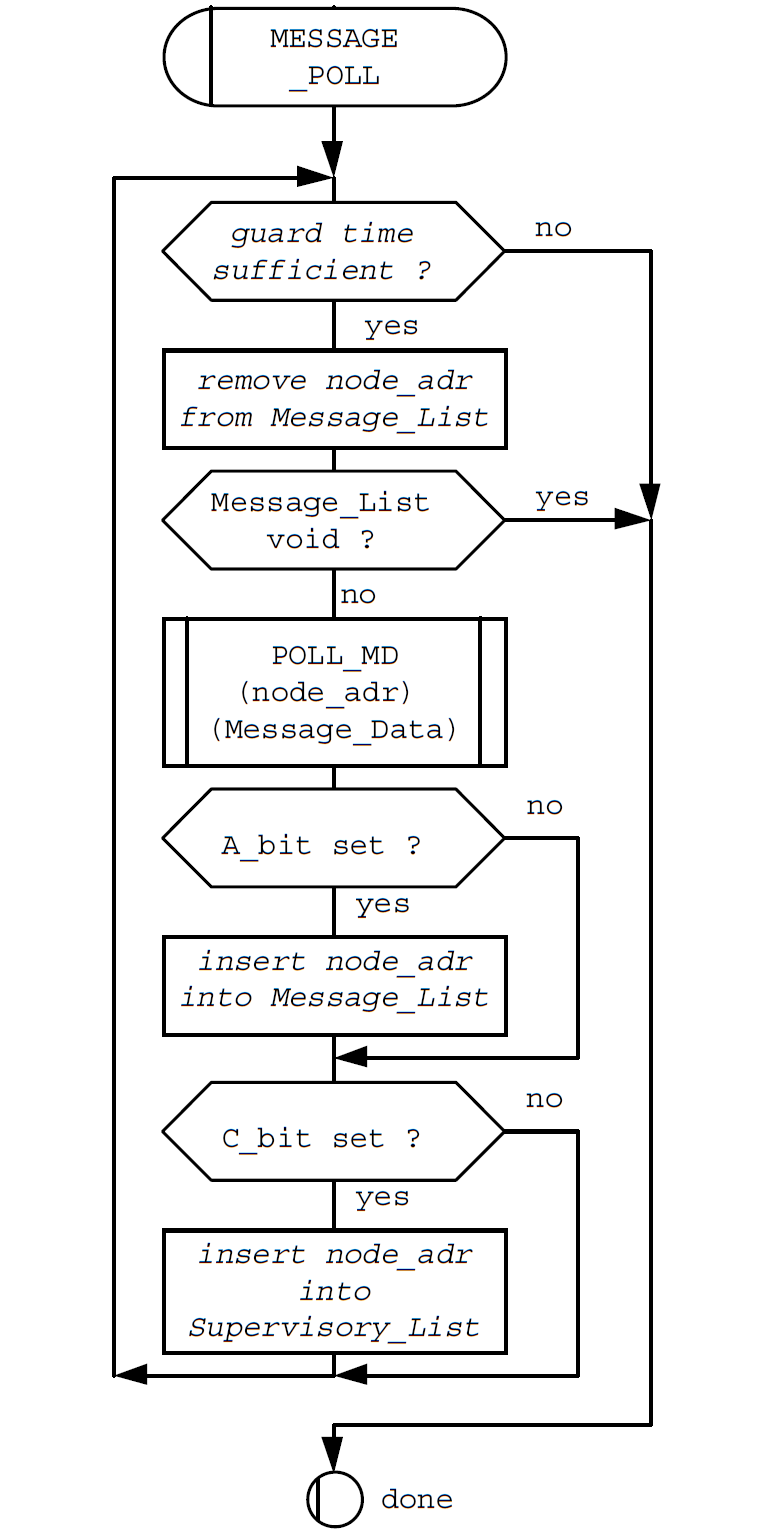
###### Macro ‘MESSAGE\_POLL’

그림 205와 같이 마스터는 다음 Periodic\_Phase 이전에 충분한 시간이 남아있으면 메시지 데이터를 전송한다.

마스터가 메시지 데이터 응답을 받지 못하더라도 동일한 기본 주기에서 메시지 데이터 요청을 반복해서는 안된다.

마스터가 자신을 폴링하면 메시지 데이터 응답을 보내기 전에 스스로에게 메시지 데이터 요청을 보낸다.

노드는 다음 주기가 시작되기 전에 완전한 프레임을 보낼 시간이 더 이상 없다면 그 상태를 벗어나서 다음 주기 ‘AWAIT\_PERIOD’로 돌아간다.



**Figure 205 – Macro MESSAGE\_POLL**

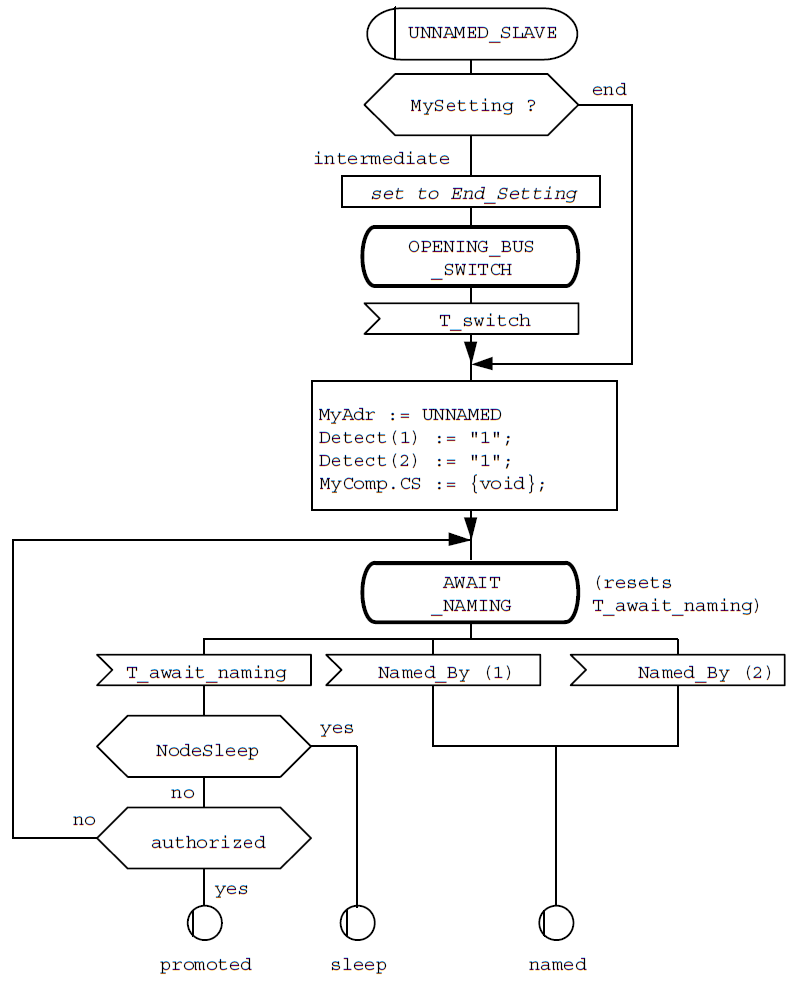
###### Slave States

###### Macro ‘UNNAMED\_SLAVE’

그림 206에서와 같이, 한 노드는 보조 채널을 모두 듣고 AWAIT\_NAMING 상태에서 주소 지정을 기다리며 종단 설정에 있다.

AWAIT\_NAMING 상태로 들어갈 때 노드는 T\_await\_naming 타이머를 리셋하고 다음을 기다린다:

1. T\_await\_naming 타이머
   * 어플리케이션에서 NodeSleep 명령을 받으면 중간 설정으로 전환하고 저 전력 상태인 NODE\_SLEEP으로 이동한다.
   * 약한 노드로 구성된 경우 NAMING\_MASTER로 이동하거나, 또는
   * 그렇지 않으면 UNNAMED\_SLAVE로 돌아간다.
2. 보조 채널의 ‘NamedBy’신호,
   * 주 채널을 이름이 지정된 방향으로 잠근다.,
   * NAMED\_SLAVE로 이동한다.



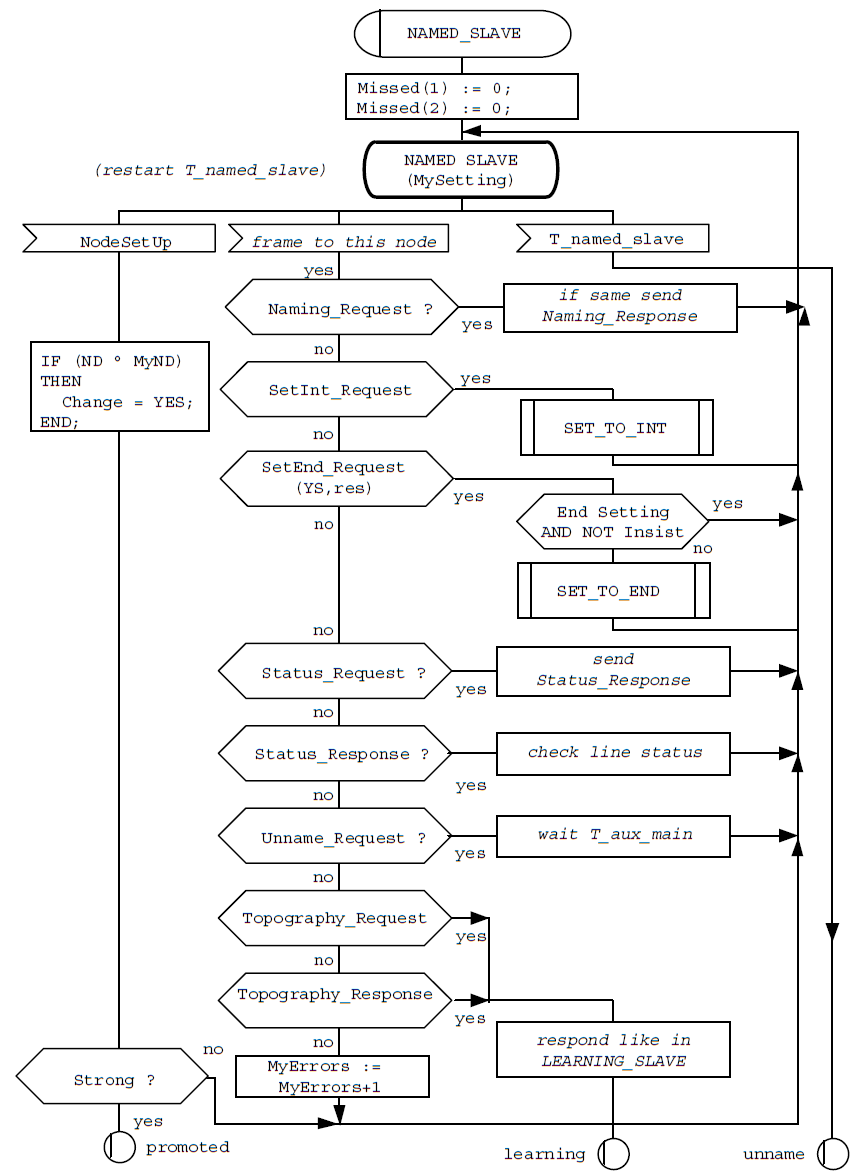
**Figure 206 – States ‘UNNAMED\_SLAVE’**

###### Macro ‘NAMED\_SLAVE’

그림 207에서 보듯이 이 상태의 노드는 주소가 지정되었으며 마스터의 요청에 응답하여 이 상태 내에서, 종단 설정에서 중간 설정으로 또는 그 반대로 변경할 수 있다.

NAMED\_SLAVE 상태로 들어갈 때 노드는 T\_named\_slave 타이머를 리셋하고 다음을 기다린다:

1. 노드 설명자의 변경:
   * 강한 노드로 승격된 경우 NAMING\_MASTER 매크로로 이동하거나
   * 아니면 ‘C\_bit’를 인가하여 NAMED\_SLAVE로 돌아간다;
2. T\_named\_slave 타임아웃:
   * UNNAMED\_SLAVE로 간다;
3. Naming Request:
   * 마스터에게 Naming Response를 보낸다;
4. SetInt Request:
   * SetInt 응답을 마스터에게 보내고, 아직 설정에 있지 않다면 중간 설정으로 간다;
5. SetEnd Request:
   * SetEnd 응답을 마스터로 전송하고, 종단 설정으로 이동한다;
6. Status Request:
   * Status 응답 브로드 캐스트;
7. Status Response:
   * 이중화 회선을 확인하기 위해 상태를 기록한다;
8. Unname Request:
   * T\_aux\_main을 대기(모든 노드가 세 개의 Unname 요청 중 하나를 수신하도록 허용),
   * T\_await\_naming 타이머의 임계 값을 가장 높은 값으로 설정한다. 그리고
   * ‘UNNAMED\_SLAVE’상태로 돌아간다.
9. Topography Request, Topography Response:
   * LEARNING\_SLAVE에서와 같이 응답하고 LEARNING\_SLAVE로 이동한다;
10. none of the above:
    * 오류 카운터를 증가시킨다.

****

**Figure 207 – States ‘NAMED\_SLAVE’**

주1: 주소 지정 단계에서 버스 관리는 상태 응답을 사용하여 종단 노드를 무시하고 수행한다.

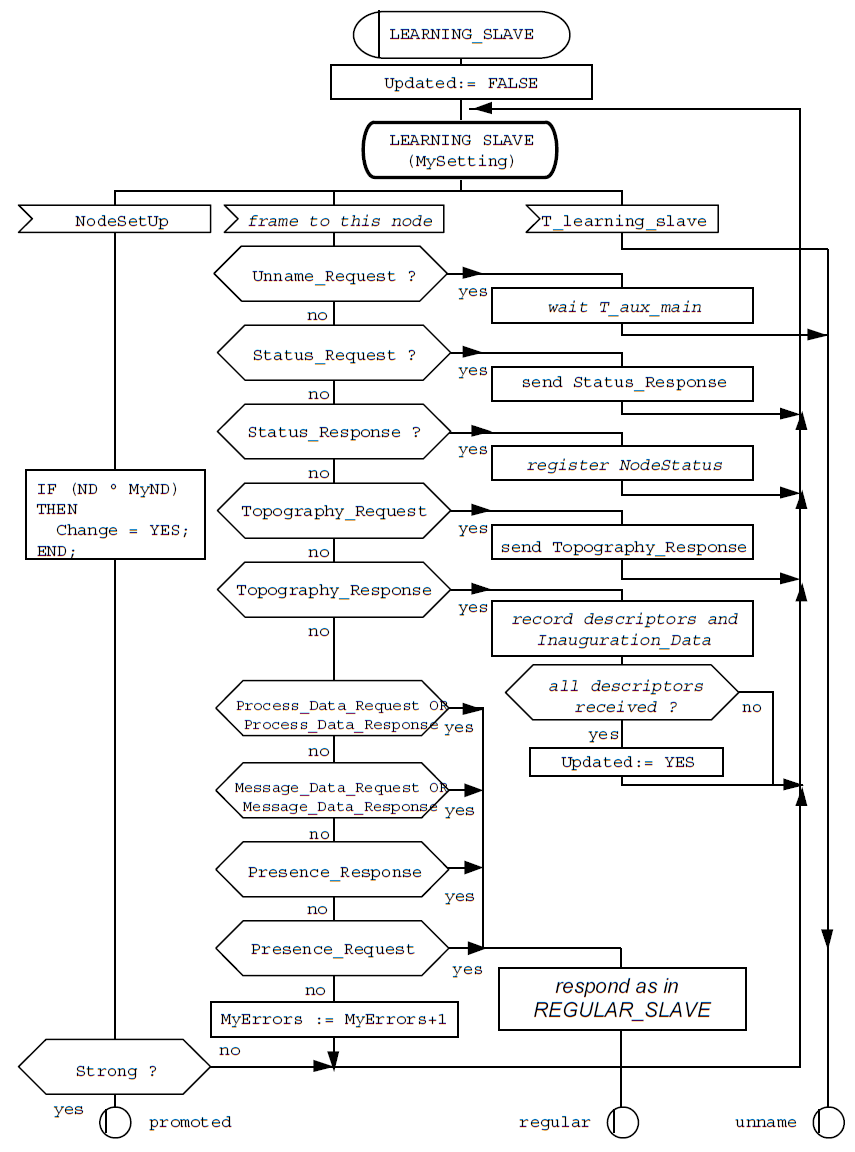
주2: 노드는 이 상태에서 메시지 데이터 요청, 메시지 데이터 응답, Process\_Data\_Requests, Process\_Data\_Responses, 프레즌스 요청과 프레즌스 응답을 무시한다.

###### Macro ‘LEARNING\_SLAVE’

그림 208과 같이 노드는 버스를 관리하면서 다른 모든 노드에서 토포그라피 정보를 수신한다. 노드는 설정이나 주소를 변경하지 않는다.

LEARNIG\_SLAVE 상태로 들어갈 때 노드는 T\_learning\_slave 타이머를 리셋하고 다음을 기다린다:

1. 노드 설명자의 변경:
   * 강한 노드로 승격된 경우 NAMING\_MASTER 상태로 이동하거나;
   * 아니면 ‘C\_bit’를 인가한다;
2. T\_learning\_slave 타임아웃:
   * T\_await\_naming 타이머 값을 변경하지 않고 UNNAMED\_SLAVE로 이동한다;
3. Unname Request:
   * T\_aux\_main 대기(모든 노드가 세 개의 Unname 요청 중 하나를 수신하도록 허용),
   * T\_await\_naming 타이머의 임계 값을 가장 높은 값인 T\_await\_max로 설정하고,
   * ‘UNNAMED\_SLAVE’ 상태로 이동한다;
4. Status Request:
   * 마스터에게 상태 응답을 보낸다;
5. Status Response:
   * 상태를 등록한다;
6. Topography Request:
   * 토포크라피 응답을 브로드캐스트한다;
7. Topography Response:
   * 토포그라피를 갱신하고, 완전한 토포그라피를 소유하고 있는지 확인하며, 모든 Topography\_Requests는 동일한 Master\_Topo와 함께 수신되었으며 설정된 경우 갱신됨, Updated = TRUE로 설정;
8. Presence Request, Presence Response, Process\_Data\_Request, Process\_Data\_Response, Message\_Data\_Request, Message\_Data\_Response:
   * REGULAR\_SLAVE에서와 같이 응답하고 REGULAR\_SLAVE로 이동한다;
9. none of the above:
   * 오류 카운터를 증가시킨다.

****

**Figure 208 – Macro ‘LEARNING\_SLAVE’**

주: learning 주기에서 버스 관리는 토포크라피 응답으로 종단 노드를 무시하고 수행한다.

###### Macro ‘REGULAR\_SLAVE’

그림 209에서와 같이 이것은 프로세스 데이터와 메시지 데이터를 송수신하고 해당 응답의 표시 비트를 설정하여 이벤트를 발생하는 노드의 일반 작동 상태이다. 노드는 두 타이머를 통해 종단 노드의 활동을 관리한다.

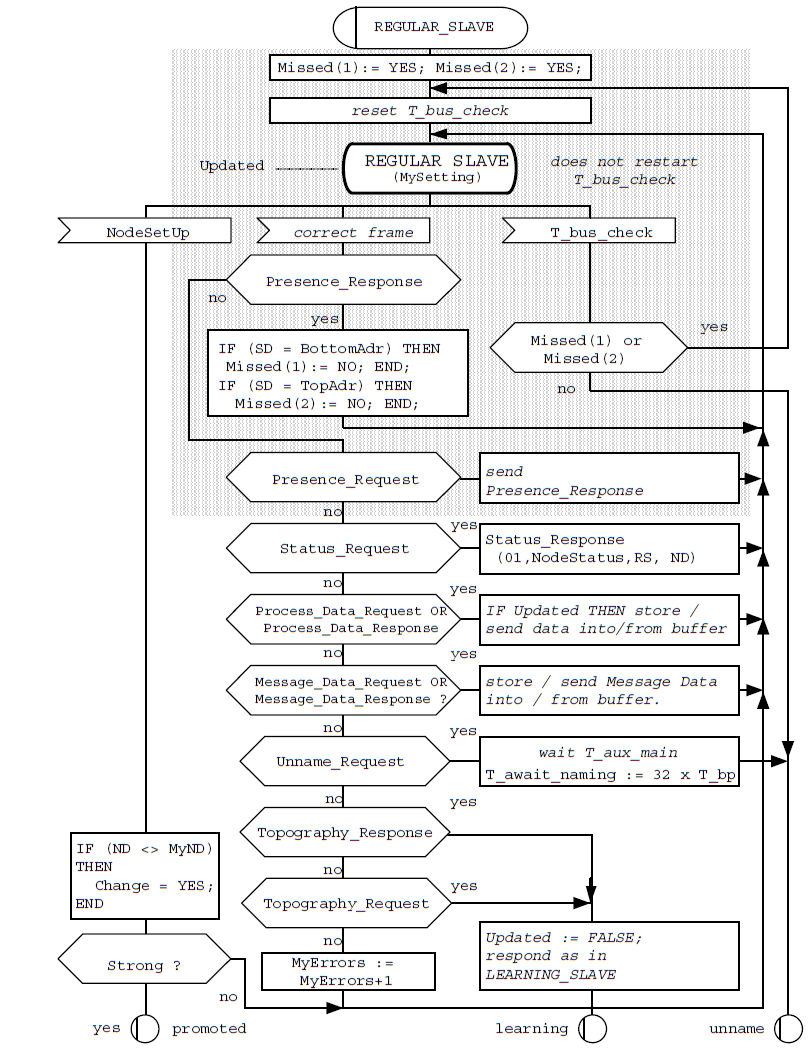
REGULAR\_SLAVE 상태에서 노드는 다음을 기대한다:

1. 노드 설명자의 변경:
   * 강한 노드로 승격된 경우 NAMING\_MASTER 상태로 이동하거나,
   * 아니면 C-bit를 인가한다;
2. 관리하는 종단 노드 각각에 대한 T\_bus\_check 타임아웃:
   * T\_await\_naming의 시작 값을 그대로 유지하면서 UNNAMED\_SLAVE로 이동한다;
3. Presence Request:
   * 프레즌스 응답을 브로드캐스트한다. (노드는 종단 노드가 아닌 경우 프레즌스 요청을 무시한다);
4. Presence Response:
   * 해당 T\_bus\_check 타이머를 다시 시작한다;
5. Status Request:
   * 마스터에게 상태 응답을 보낸다;
6. Unname Request:
   * T\_aux\_main 대기 (모든 노드는 세 개의 Unname 요청 중 하나를 수신하도록 허용),
   * T\_await\_naming 타이머의 임계 값을 가장 높은 값으로 설정하고,
   * UNNAMED\_SLAVE로 간다;
7. 프로세스 데이터가 없는 Process\_Data\_Request:
   * 노드가 갱신된 경우, Process\_Data\_Response를 보내고 소스 포트에서 데이터를 읽은 다음,
   * 아니면 void Process\_Data\_Response를 보낸다;
8. (선택적으로) 프로세스 데이터가 포함된 Process\_Data\_Request:
   * Updated가 TRUE인 경우 이 데이터를 직접 마스터 데이터 전용 싱크 포트에 기록하고 Process\_Data\_Response를 보내고 소스 포트에서 데이터를 읽은 다음,
   * 아니면 무시하고 void Process\_Data\_Response를 보낸다;
9. Process\_Data\_Response:
   * Updated가 TRUE인 경우, 이 데이터를 소스 주소에 해당하는 싱크 포트에 기록하고,
   * 아니면 무시한다;
10. Message Data Request:
    * 송신 대기열이 무효인 경우 마스터에게 void Message Data Response (link\_data\_size = 0)를 보내거나, 아니면
    * 전송 대기열에서 추출된 메시지 데이터 패킷과 함께 메시지 데이터 응답을 보낸다;
11. Message Data Response:
    * 수신 메시지 데이터를 Receive\_Queue에 저장한다, 아니면 무시한다(4.8.3 참조);
12. Topography Request or Topography\_Response:
    * Updated를 FALSE로 하고, LEARNING\_SLAVE에서와 같이 응답하고 그 상태로 간다;
13. none of the above:
    * 오류 카운터를 증가시키고 LEARNING\_SLAVE로 돌아간다.

주1: T\_bus\_check 타이머는 단일 타이머와 카운터로 구현할 수 있다.

주2: 이 기능은 T\_bus\_check가 실행하기 때문에 REGULAR\_SLAVE와 연관된 타이머가 없고 이 상태로 들어갈 때 마다 다시 시작한다.

주3: 종단 노드의 관리는 각 종단 노드에 대해 개별적으로 T\_bus\_check 타이머를 사용하여 수행할 수 있다.



**Figure 209 – Macro ‘REGULAR\_SLAVE’**

###### Time-outs

권장되는 시간 제한 값 (±20%)은 표 71에 나열되어 있다.

**Table 71 – Time constant values**

|  |  |  |
| --- | --- | --- |
| **Time constant name** | **Value** | **Usage** |
| T\_await\_naming | 1) T\_await\_min = 1,0 ms + T\_switch | 조성을 재 명명하는 마스터 |
| 2) ((63 – MyAdr) + 0,5) x T\_bp | 마스터의 direction 1에 있는 노드 |
| 3) (MyAdr-1) x T\_bp | 마스터의 direction 2에 있는 노드 |
| 4) T\_await\_max = 32 x T\_bp | 초기화되거나 명시적으로 이름 없는 노드 |
| T\_aux\_main | 1,0 ms | 보조 채널에서 주 채널로 또는 그 반대로 스위치 지연 |
| T\_await\_response | 1,756 ms | 마스터가 슬레이브 프레임을 기다리는 시간, 이 시간은 HDLC 컨트롤러가 프레임의 끝만을 신호로 보낼 수 있기 때문에 가능한 가장 긴 프레임을 차지한다. |
| T\_bp | 25,0 ms | Basic Period (정규 동작) |
| T\_naming\_master | 15,0 ms | naming period (이노규레이션) |
| T\_named\_slave | 15,0 ms | 주소 지정 주기 중 마스터 관리 |
| T\_learning\_slave | 15,0 ms | learning 주기 중 마시터 관리 |
| T\_bus\_check (1)  T\_bus\_check(2) | 6,5 x T\_bp | 정상 작동 중 종단 노드 관리 |
| T\_detecting | 2 x T\_naming\_master | 이노규레이션 중 Detect\_Request 사이의간격(있는 경우) |
| 2 x T\_bp | 정상 동작 중 프레즌스 여부 요청/ 감지 요청(있는 경우) 사이의 간격 |
| T\_detecting\_response | 1,047 ms | 종단 노드가 응답 감지를 기다리는 시간 |
| T\_new\_inaug | n x T\_bp | 2 회의 연속 이노규레이션 사이의 최소 시간. n은 어플리케이션에 의해 설정된다 |
| MAXLOST | 50 | T\_detecting x MAXLOST는 다른 노드가 더 이상 존재하지 않는다고 종단 노드가 가정한 후의 시간이다 |
| T\_switch | 10,0 ms | 릴레이를 닫거나 열 때의 기본 지연 |

###### Link layer interface

###### Link layer layering

링크 계층 인터페이스는 그림 210과 같이 세 가지 서비스를 제공한다:

1. 변수 서비스에 의해 사용되는 Link Process\_Data\_Interface(LPI)는 2절(실시간 프로토콜)에서 명시하고있다, WTB에 특정한 파라메터만 이 표준에 명시하고있다;
2. 메시지 서비스에 의해 사용되는 Link Message\_Data\_Interface(LMI)는 2절(실시간 프로토콜)에서 명시하고 있다, WTB에 특정한 파라메터만 이 표준에 명시하고있다;
3. 링크 계층의 구성과 버스의 관리를 허용하는 Link Supervision\_Interface(LSI)는 WTB에 고유하며 이 표준에서 규정한다.

****

**Figure 210 – Link layer layering**

###### Link Process\_Data\_Interface

###### General

링크 계층과 상위 계층 간의 프로세스 데이터 인터페이스는 Traffic\_Store라고 하는 공유 메모리이며 버스와 어플리케이션에서 동시에 액세스할 수 있다.

Traffic\_Store는 프로세스 데이터의 전송 또는 수신을 위해 준비된 정확히 하나의 프레임을 포함하는 여러 포트로 구성한다.

각 포트는 Traffic\_Store 식별자와 12 비트 포트 주소로 노드 내에서 식별한다. 포트의 구현은 표준화의 일부가 아니다.

제 2항 ‘Real-Time Protocols’은 포트 접근을 규정한다.

###### WTB specific

WTB의 Traffic\_Store는 각각 최대 1024 비트를 갖는 64개의 포트를 보유해야한다.

In all cases:

* 각 노드는 프로세스 데이터를 브로드캐스팅할 하나의 소스 포트를 가져야하며 최상위 6비트는 ‘000000’B이고 포트 주소의 최하위 6비트는 이 노드의 주소이다;
* 각 노드는 버스 상의 서로 가능한 노드로부터 프로세스 데이터를 수신하기 위한 싱크 포트를 가져야하며, 최상위 6비트는 ‘000000’B이고 포트 주소의 최하위 6비트는 소스 노드의 주소이다.

마스터가 Process\_Data\_Request에 프로세스 데이터를 포함하는 어플리케이션에서 폴링된 슬레이브에만 적용한다:

* 마스터는 프로세스 데이터를 각 가능한 슬레이브로 전송할 하나의 소스 포트를 가져야하며, 최상위 비트는 ‘000010’B이고 포트 주소의 최하위 6비트는 대상 노드 주소이다;
* 각 슬레이브는 마스터 프로세스 데이터를 수신할 하나의 싱크 포트를 구현해야 하며 최상위 비트는 ‘000010’B이고 포트 주소의 최하위 6비트는 노드 주소이다.

노드는 수신된 토포그라피에 포함되지 않았거나 토포그라피에서 수신된 서술자를 해석할 수 없는 노드로부터의 프로세스 데이터를 받아들이지 않는다.

노드는 Node\_Type을 알고 있는 다른 노드로부터 프로세스 데이터를 받아들일 수 있으며, 받아들인 Node\_Version이 알고 있는 Node\_Version과 다르지만, 두 Node\_Version중 낮은 값에 따라 데이터를 디코딩한다.

노드는 새로운 Node\_Key를 포함하는 토포그라피를 수신하지 않고 Process\_Data\_Response의 형식을 변경할 수 없다.

추가 보호로서 프레임 유형 (Node\_Type + Node\_Version = Node\_Key)을 식별하기 위해 프로세스 데이터의 처음 두 옥텟을 예약하는 것이 좋다.

###### Link Message\_Data\_Interface

###### General

2절 ‘Real-Time Protocols’에 명시된 Link Message\_Data\_Interface(LMI)는 수신된 메시지 데이터 프레임을 검색하기 위한 메시지 데이터 프레임과 서비스를 전송하는 서비스를 제공한다. 덧붙여 전송 확인과 수신 표시 서비스를 지원한다.

Link Message\_Data\_Interface는 모든 상위 프로토콜이 구축되는 기본 서비스를 제공한다:

1. 네트워크 계층은 네트워크와 디렉토리 기능을 통한 라우팅을 제공한다;
2. 전송 프로토콜은 메시지의 반 이중 종단 간 제어를 제공한다;
3. 세션 계층은 메시지를 결합하여 원격 프로시저 호출을 제공한다;
4. presentation 계층은 데이터 표현을 통합한다;
5. 어플리케이션 인터페이스는 클라이언트와 서버 인터페이스를 제공한다.

###### Packet size

void 패킷의 ‘link\_data\_size’필드는 0 이다. ‘link\_data\_size’ 필드는 128보다 작다.

###### Protocol\_Type

실시간 프로토콜에 대한 Protocol\_Type은 ‘00xxx111’B(메시지 데이터 응답)로 설정된 link\_control 필드가 표시한다.

###### Message transport protocol

WTB 노드는 자신의 Connect\_Request에 124 옥텟보다 큰 패킷 크기를 알리지 않는다.

Connect\_Response에 응답할 때 노드는 자신의 Connect\_Response에 124옥텟 또는 제안된 패킷 크기 중 작은 값과 동일한 크기의 패킷을 지정한다.

###### Link management interface

###### General

링크 관리 인터페이스는 WTB에만 해당한다.

구성을 위한 일반 서비스를 제공하고 링크 계층과 이벤트 보고 검사를 한다..

다음의 항은 특정 구현을 의미하지는 않는다. 같은 의미를 제공하는 인터페이스는 모두 허용한다.

다음 인터페이스 절차의 파라메터 형식은 규정하고 있지 않다. 그러나 열차 네트워크 관리 표준(5절)은 파라메터 형식으로 권장되는 메시지 형식을 제안한다.

###### Interface procedures

이 인터페이스의 프로시저 앞에는 ls\_t(link supervision, WTB)가 붙는다.

###### Type LS\_T\_RESULT

The procedure returns are of the type LS\_T\_RESULT:

|  |  |  |
| --- | --- | --- |
| **Constant** | **Code** | **Meaning** |
| L\_OK | 0 | success |
| L\_BUSY | 1 | try again later |
| L\_CALLING\_SEQUENCE | 2 | wrong command sequence |
| L\_MISSING\_UDF | 3 | user-defined function unknown |
| L\_CONFIGURATION\_INVALID | 4 | Topography or node list invalid |

###### Constants LS\_T\_STATE

다음 상수는 노드가 현재 주요 상태임을 나타낸다:

|  |  |  |
| --- | --- | --- |
| **Constant** | **Code** | **Meaning** |
| LS\_INITIALIZED | 0 | node in state |
| LS\_CONFIGURED | 1 | UNCONFIGURED node in state CONFIGURED |
| LS\_READY\_TO\_NAME | 2 | node in state NAMING\_MASTER |
| LS\_READY\_TO\_BE\_NAMED | 3 | node in state UNNAMED\_SLAVE |
| LS\_INHIBITED | 4 | node Inauguration inhibited |
| LS\_REGULAR\_STRONG | 5 | node in state REGULAR\_MASTER (strong) or TEACHING\_MASTER |
| LS\_REGULAR\_SLAVE | 6 | node in state REGULAR\_SLAVE or TEACHING\_MASTER |
| LS\_REGULAR\_WEAK | 7 | node in state REGULAR\_MASTER (weak) or  TEACHING\_MASTER |

주: 노드가 슬립 상태임을 나타낼 수 없음.

###### Reporting

###### Procedure ls\_t\_Report

|  |  |  |
| --- | --- | --- |
| **Action** | 사용자에게 링크 계층의 변경 사항을 보고한다.  이 프로시저는 링크 계층에서 호출되며 이전에 가입해야 한다 (ls\_t\_Configure 참조) | |
| **Syntax** | Typedef LS\_T\_RESULT (\* ls\_t\_Report) (ls\_report) | |
| **Input** | ls\_report | one of the LR\_REPORT report codes. |

###### Constants LR\_REPORT

The value of the report codes shall be as follows:

|  |  |  |
| --- | --- | --- |
| **Constant** | **Code** | **Meaning** |
| LR\_CONFIGURED | 16 | link layer is configured |
| LR\_STRONG | 17 | node is now operational master |
| LR\_SLAVE | 18 | node is now operational slave |
| LR\_PROMOTED | 19 | node changed from weak to strong master |
| LR\_NAMING\_SUCCESSFUL | 20 | master indicates end of inauguration |
| LR\_NAMED | 21 | node is a named slave |
| LR\_WEAK | 22 | master changed to weak master |
| LR\_REMOVED | 23 | node removed from configuration |
| LR\_DEMOTED | 24 | weak master detected a strong master |
| LR\_DISCONNEXION | 25 | node disconnected |
| LR\_INHIBITED | 26 | inauguration inhibited |
| LR\_INCLUDED | 27 | included in composition |
| LR\_LENGTHENING | 28 | master detected train lengthening |
| LR\_DISRUPTION | 29 | node detected loss of End Node |
| LR\_MASTER\_CONFLICT | 30 | strong master detected another strong master |
| LR\_NAMING\_FAILED | 31 | failure while naming |
| LR\_NEW\_TOPOGRAPHY | 32 | reception of a new Topography |
| LR\_NODE\_STATUS | 33 | naming started |
| LR\_POLL\_LIST\_OVF | 34 | partially operational |
| LR\_ALLOWED | 35 | missing node |

###### Initialisation service

###### Procedure ls\_t\_Init

|  |  |
| --- | --- |
| **Action** | 링크 계층을 초기화하고 변수를 미리 정의된 값으로 설정한다.  호출 후 링크 계층은 명령을 수신할 준비가 되어있어야 한다. 이 절차는 하드웨어 리셋 후 한번만 호출된다.  이 프로시저는 실행 의존적이다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_Init (void); |

###### Reset service

* + - * 1. **Procedure ls\_t\_Reset**

|  |  |
| --- | --- |
| **Action** | 링크 계층을 미리 정의된 값으로 리셋한다.  호출 후 링크 계층은 유휴 상태가 되어 명령을 수신 할 준비가 된다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_Reset (void); |

###### Configuration service

###### Procedure ls\_t\_Configure

|  |  |  |
| --- | --- | --- |
| **Action** | 링크 계층을 구성한다.  호출 후, 노드는 통신을 시작할 준비가 되어있어야 한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_Configure (  Type\_Configuration \* p\_configuration  ); | |
| **Input** | p\_configuration | Pointer to the following configuration data structure. |

###### Type\_NodeKey

Type\_NodeKey의 자료 구조는 다음 요소를 포함한다:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| node\_type | UNSIGNED8 | type of the node as indicated by the application |
| node\_version | UNSIGNED8 | version of the node as indicated by the application |

주: Type\_NodeKey는 구조 Node\_Key 에 대응하는 C-Type이다(4.7.2.1 참조).

###### Type\_NodeDescriptor

Type\_NodeDescriptor의 데이터 구조는 다음 요소를 포함한다:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| node\_frame\_size | UNSIGNED8 | size of the Process Data frame in octets. |
| node\_period | UNSIGNED8 | value of the Node\_Period in 2n multiples of the Basic Period (node\_period is the value of n). |
| node\_key | Type\_NodeKey | see this data type. |

주: Type\_NodeDescriptor는 Node\_Descriptor구조에 해당하는 C-Type이다(see 4.7.2.1).

###### Type\_Configuration

데이터 구조 설정은 다음 요소를 포함한다:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| transmission\_rate | UNSIGNED16 | transmission rate in kbit/s, default: 1000 kbit/s |
| basic\_period | UNSIGNED16 | Basic Period in milliseconds, default: 25,0 ms |
| fritting\_disabled | UNSIGNED16 | = 1 if fritting disabled, default: 0 |
| node\_descriptor | Type\_NodeDescriptor | see 4.8.4.6.3 |
| poll\_md\_when\_idle | UNSIGNED8 | =1 if background scanning enabled, default: 0 (see 4.6.3.4). |
| sink\_port\_count | UNSIGNED16 | Maximum number of sink ports, default: 22 |
| source\_port\_count | UNSIGNED16 | Maximum number of source ports, default: 1 |
| port\_size | UNSIGNED8 | Maximum length of a Port in octets, default: 128 |
| p\_traffic\_store | WORD32 | Pointer to the traffic store, default: NULL |
| ls\_t\_report | WORD32 | Call back function for reports, default: NULL |
| max\_number\_nodes | UNSIGNED8 | Maximum number of nodes whose inauguration data to be stored, default: 0 |
| inaug\_data\_max\_size | UNSIGNED8 (124) | Maximum number of octets of application defined inauguration data to send, default: 0 |
| s\_inaug\_data\_size | UNSIGNED8 (124) | Actual number of octets of application defined inauguration data to send, default: 0 |
| p\_inaug\_data\_list | WORD32 | Pointer to data area where to copy inauguration data from, default: NULL |

###### Type\_Inauguration\_Data

Type\_Inauguration\_Data 데이터 구조는 다음 요소를 포함한다:

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | | **Type** | **Meaning** |
| inaug\_data\_max\_size | | UNSIGNED8 (124) | maximum number of octets of application defined inauguration data stored, default: 0 |
| nr\_descriptors | | UNSIGNED8 | number of nodes whose inauguration data is stored, default: 0 = invalid |
| node\_descriptions | | ARRAY [nr\_descriptors] OF | list of the application defined inauguration data for each WTB node, consisting of: |
|  | node\_type | WORD8 | first part of Node\_Key |
| node\_version | WORD8 | second part of Node\_Key |
| sam | BOOLEAN1 | ‘1’ if same orientation as master |
| rsv1 | WORD1 (=0) | reserved, =0 |
| node\_address | UNSIGNED6 | address of node from which inauguration data was received. |
| inauguration\_data\_siz e | UNSIGNED8 | size of Inauguration\_Data ( 124 octets) |
| inauguration\_data | ARRAY[inaug\_data\_len  ] OF WORD8 | application defined inauguration data |

‘node\_descriptions’는 이노규레이션이 시작되기 전에 초기화되어야 한다. 이노규레이션이 끝나면 ‘node\_descriptions’ 테이블에는 nr\_descriptors’행이 각 노드마다 하나씩 포함된다.

###### Set Slave service

###### Procedure ls\_t\_SetSlave

|  |  |
| --- | --- |
| **Action** | 노드가 마스터가 되지 않도록 한다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_SetSlave (void); |

###### Set Weak service

###### Procedure ls\_t\_SetWeak

|  |  |
| --- | --- |
| **Action** | 노드가 약한 마스터가 되도록 한다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_SetWeak (void); |

###### Set Strong service

###### Procedure ls\_t\_SetStrong

|  |  |
| --- | --- |
| **Action** | 강한 마스터가 되도록 노드에게 명령한다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_SetStrong (void); |

NOTE This command causes an inauguration.

###### StartNaming service

###### Procedure ls\_t\_StartNaming

|  |  |
| --- | --- |
| **Action** | 노드가 이노규레이션을 시작하도록 명령한다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_StartNaming (void); |

###### Remove service

###### Procedure ls\_t\_Remove

|  |  |
| --- | --- |
| **Action** | 구성에서 자신을 제거하고 수동 상태로 이동하도록 노드에 명령한다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_Remove (void); |

###### Inhibit service

###### Procedure ls\_t\_Inhibit

|  |  |
| --- | --- |
| **Action** | 추가 노드가 감지되는 경우 버스 연장을 방지한다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_Inhibit (void); |

###### Allow service

###### Procedure ls\_t\_Allow

|  |  |
| --- | --- |
| **Action** | 추가 노드가 감지되면 버스 길이를 늘릴 수 있다.. |
| **Syntax** | LS\_T\_RESULT ls\_t\_Allow (void); |

###### SetSleep service

###### Procedure ls\_t\_SetSleep

|  |  |
| --- | --- |
| **Action** | 노드에서 슬립 상태가 발생하였다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_SetSleep (void); |

###### CancelSleep service

###### Procedure ls\_t\_CancelSleep

|  |  |
| --- | --- |
| **Action** | 노드가 슬립 상태에서 벗어났다. |
| **Syntax** | LS\_T\_RESULT ls\_t\_CancelSleep (void); |

###### GetStatus service

###### Procedure ls\_t\_GetStatus

|  |  |  |
| --- | --- | --- |
| **Action** | 링크 계층의 물리적 상태를 검색한다. | |
| **Syntax** |  |  |
|  | LS\_T\_RESULT | ls\_t\_GetStatus |
|  |  | ( |
|  | Type\_WTBStatus\* | p\_status |
|  |  | ); |
| **Input** | p\_status | WTB\_Status 데이터 구조를 넣을 위치에 대한 포인터. |

###### Type\_Node\_Status

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| node\_report  user\_report | BITSET8  BITSET8 | ‘C’ declaration corresponding to Node\_Report (4.7.2.2.)  ‘C’ declaration corresponding to User\_Report (4.7.2.3) |

###### Type\_WTBStatus

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| wtb\_hardware\_id | UNSIGNED8 | 하드웨어 식별 |
| wtb\_software\_id | UNSIGNED8 | 링크 계층 소프트웨어 버전 식별 |
| hardware\_state | ENUM8 | 0: LS\_OK 정상 동작  1: LS\_FAIL, 하드웨어 오류 |
| link\_layer\_state | LS\_T\_STATE | see type definition |
| net\_inhibit | ENUM8 | 1: 어떤 노드는 이노규레이션을 금지한다 |
| node\_address | UNSIGNED8 | 이노규레이션에 의해 할당된 노드 주소 |
| node\_orient | UNSIGNED8 | 마스터를 기주느로 한 노드 방향:  0: L\_UNKNOWN  1: L\_SAME  2: L\_INVERSE |
| node\_strength | UNSIGNED8 | 노드의 강도  0: L\_UNDEFINED  1 L\_SLAVE  2: L\_STRONG  3: L\_WEAK |
| node\_descriptor | Type\_NodeDescriptor | see type definition |
| node\_status | Type\_Node\_Status | see type definition |

###### Get WTB nodes service

###### Type\_NodeList

Type\_NodeList의 자료 구조는 다음 요소를 포함한다:

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | | **Type** | **Meaning** |
| nr\_nodes | | UNSIGNED8 | 조성에서 노드 수 |
| bottom\_node | | UNSIGNED8 | 마스터로부터 Direction\_1에 있는 종단 노드의 주소.  이 옥텟의 최상위 두 비트는 0이다. |
| top\_node | | UNSIGNED8 | 마스터로부터 Direction\_2에 있는 종단 노드의 주소.  이 옥텟의 최상위 두 비트는 0이다. |
| node\_status\_list | | ARRAY  [MAX\_NODES] OF | 아래 노드에서 시작하여 노드가 있는 순서대로 노드 상태 목록을 구성하고 다음으로 구성되는 최 상위 노드로 끝난다: |
|  | node\_status | Type\_Node\_Status | see type definition |

###### Procedure ls\_t\_GetWTBNodes

|  |  |  |
| --- | --- | --- |
| **Action** | 토포그라피에서 모든 노드의 노드 보고서와 사용자 보고서 목록을 읽는다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_GetWTBNodes  (  Type\_NodeList \* p\_nodes  ); | |
| **Input** | p\_nodes | pointer to location where to put the list of nodes. |

###### Get Topography service

###### Procedure ls\_t\_GetTopography

|  |  |  |
| --- | --- | --- |
| **Action** | 어플리케이션이 일반 동작을 시작하기 전에 배포된 토포그라피를 읽을 수 있도록 허용한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_GetTopography  (  Type\_Topography \* p\_topography  ); | |
| **Input** | p\_topography | pointer to the place where to put the Topography. |
| **Result** |  | returns L\_CONFIGURATION\_INVALID if the Topography is not valid. |

###### Type\_Topography

토포 그라피 데이터 구조는 다음 요소를 포함해야 한다:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| node\_address | UNSIGNED8 | 이 옥텟의 최상위 두 비트는 0이다. 이 옥텟의 하위 6비트는 이스테이션이 연결된 노드의 주소이다. |
| node\_orient | UNSIGNED8 | 마스터에 대한 상대적인 노드의 방향:  0: L\_UNKNOWN  1: L\_SAME  2: L\_INVERSE |
| topo\_counter | UNSIGNED8 | 최하위 6비트는 노드의 Topo카운터의 6비트를 복사하며 최상위 2 비트는 0이다. |
| individual\_period | UNSIGNED8 | 마스터에 의해 노드에 할당된 주기는 기본 주기의 2의 거듭 제곱(ms)이다. |
| is\_strong | UNSIGNED8 | 1: 버스는 강한 마스터에 의해 제어된다  0: 버스는 약한 마스터에 의해 제어된다 |
| number\_of\_nodes | UNSIGNED8 | 이노규레이션 결과에 따른 노드 수 |
| bottom\_address | UNSIGNED8 | 마스터에서 Direction\_1에 있는 종단 노드의 주소, 이 옥텟의 최상위 두 비트는 0이다. |
| top\_address | UNSIGNED8 | 마스터에서 Direction\_2에 있는 종단 노드의 주소, 이 옥텟의 최상위 두 비트는 0이다. |
| inauguration\_data | Type\_Inauguration\_Data | see type definition |

###### Change Node Descriptor service

###### Procedure ls\_t\_ChgNodeDesc

|  |  |  |
| --- | --- | --- |
| **Action** | 링크 계층에 새로운 서술자를 제공한다. 정상 동작 중에 이 절차를 호출하면 트래픽이 중단되고 새로운 토포그라피가 발생한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_ChgNodeDesc (  Type\_NodeDescriptor \* node\_descriptor  ); | |
| **Input** | node\_descriptor | pointer to a Node\_Descriptor data structure |

###### Change User Report service

###### Procedure ls\_t\_ChgUserReport

|  |  |  |
| --- | --- | --- |
| **Action** | 어플리케이션이 사용자 보고서를 수정할 수 있게 한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_ChgUserReport (  UNSIGNED8 set\_mask  UNSIGNED8 clear\_mask  ); | |
| **Input** | set\_mask | set the bits set in the mask to 1 in User\_Report. |
|  | clear\_mask | clears the bits set in the mask to 1 in User\_Report |

###### Change Inauguration\_Data service

###### Procedure ls\_t\_ChgInauguration\_Data

|  |  |  |
| --- | --- | --- |
| **Action** | 어플리케이션이 이 노드의 이노규레이션 데이터를 수정할 수 있게 한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_ChgInauguration\_Data (  UNSIGNED8 inaug\_data\_size  void\* p\_inauguration  ); | |
| **Input** | inaug\_data\_size | size in octets of the inauguration data (≤ 124) |
|  | p\_inauguration | user-defined inauguration data |

###### Get Statistics service

###### Procedure ls\_t\_GetStatistics

|  |  |  |
| --- | --- | --- |
| **Action** | 사용과 오류에 대한 통계 정보를 제공한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_GetStatistics (  Type\_LLStatisticData \* p\_statistic\_data  ); | |
| **Input** | p\_statistic\_data | pointer to the statistic data structure (see 4.8.4.22.3) |

###### Type\_LineStatus

Type\_LineStatus의 자료 구조는 다음 요소를 포함한다:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| transmitted\_count | UNSIGNED32 | Number of frames transmitted by this node |
| received\_count | UNSIGNED32 | Number of frames received with no errors by this node |
| errors\_count | UNSIGNED16 | Number of erroneous frames received |
| timeouts\_count | UNSIGNED16 | Number of elapsed time-outs when response expected |

주: 이 카운터는 최대 값에 도달할 때 랩-어라운드되며 초기 값은 지정하지 않는다.

###### Type\_LLStatisticData

Type\_LLStatisticData의 자료 구조는 다음 요소를 포함한다:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Meaning** |
| basic\_period\_count | UNSIGNED32 | incremented for each Basic Period |
| inauguration\_count | UNSIGNED16 | incremented for each new inauguration |
| topography\_count | UNSIGNED16 | incremented for each new topography |
| transmitted\_md\_count | UNSIGNED32 | incremented for each sent Message Data Response |
| received\_md\_count | UNSIGNED32 | incremented for each received Message Data Response |
| line\_status\_a1 | Type\_LineStatus | See type |
| line\_status\_a2 | Type\_LineStatus | See type |
| line\_status\_b1 | Type\_LineStatus | See type |
| line\_status\_b2 | Type\_LineStatus | See type |
| line\_switch\_count | UNSIGNED32 | incremented for each switchover to the redundant line. |

주: 이 카운터는 최대 값에 도달할 때 랩-어라운드되며 초기 값은 지정하지 않는다.

###### Get Inauguration\_Data service

|  |  |  |
| --- | --- | --- |
| **Action** | 이노규레이션 데이터를 반환한다. | |
| **Syntax** | LS\_T\_RESULT ls\_t\_GetInaugData (  void \* \* p\_inaug\_data\_list  ); | |
| **Output** | p\_inaug\_data\_list | pointer to inauguration\_data of all named nodes |