NGBF Standard

차세대방송표준포럼표준(국문표준)

NGBF-STD-019 제정일: 2017년 1월 06일

부가정보 전송을 위한 음향데이터 포맷

Acoustic Data Format for transmitting Additional Information

ngB 차세대방송표준포럼
Next-Generation Broadcast Standards Forum

표준초안 검토 위원회	**디지털라디오분과위원회**				
표준안 심의 위원회	**운영위원회**				
	성명	**소 속**	**직위**	**위원회 및 직위**	**표준번호**
표준(과제) 제안	정영호	ETRI	책임	-	
표준 초안 작성자	백승권	ETRI	선임	-	NGBF-STD-0xx
	정영호	ETRI	책임	-	
	서상원	ETRI	연구원	-	

본 표준 발간 이전에 접수된 지식재산권 확약서 정보는 본 표준의 '부록(지식재산권 확약서 정보)'에 명시하고 있으며, 이후 접수된 지식재산권 확약서는 차세대방송표준포럼 웹사이트에서 확인할 수 있습니다.
본 표준과 관련하여 접수된 확약서 외의 지식재산권이 존재할 수 있습니다.

발행인 : 차세대방송표준포럼 의장
발행처 : 차세대방송표준포럼
06130, 서울특별시 강남구 테헤란로 7길 22 신관 1108호
Tel : 02-568-3556, Fax : 02-568-3557
발행일 : 2017.01.xx

서 문

1 표준의 목적

본 표준은 음향채널 환경에서 오디오 신호를 이용하여 부가정보를 사용자 단말에 전송하기 위한 음향데이터 포맷을 정의한다.

2 주요 내용 요약

본 표준에서는 오디오 신호에 삽입되어 음향채널을 통해 전송되는 부가정보의 비트스트림 구조 및 관련 신택스를 규정한다. 비트스트림 구조는 동기화를 위한 프리앰블, 헤더 및 페이로드 데이터로 구성되며, 부가정보 유형에 따라 상이한 구조를 갖는다.

3 인용 표준과의 비교

3.1 인용 표준과의 관련성

해당사항 없음

3.2 인용 표준과 본 표준의 비교표

해당사항 없음

Preface

1 Purpose

The standard defines an acoustic data format for transmitting additional information to user device via audio signal on acoustic channel environment.

2 Summary

The standard specifies the bitstream structure of the additional information, which is inserted in the audio signal and transmitted through the acoustic channel, and the associated syntax. The bitstream structure is composed of preamble, header and payload data for synchronization, and has a different structure depending on the type of additional information.

3 Relationship to Reference Standards

3.1 Relevance to the Reference Standard

Not applicable.

3.2 Comparison of the Reference and the Quoted Standard

Not applicable.

목 차

부가정보 전송을 위한 음향데이터 포맷

Acoustic Data Format for transmitting Additional Information

1 적용 범위

본 표준은 음향채널 환경에서의 부가정보 전송을 위해 오디오 신호에 삽입되는 부가정보의 비트스트림 구조 및 관련 신택스를 정의한다. 부가정보의 비트스트림은 음향데이터 전송 기술을 이용하여 삽입되며, 본 표준에서는 특정 음향데이터 전송 기술에 대해 규정하지 않는다.

2 인용 표준

해당사항 없음

3 용어 정의

3.1 음향채널

음향채널은 스피커와 마이크 간의 물리적 공간을 의미한다. 스피커를 통해 오디오 신호가 재생되고 마이크로 수음되기까지 오디오 신호가 거치게 되는 채널 공간으로 이를 음향채널이라 정의한다. 음향채널에서 빈번이 발생하는 왜곡은 주변 잡음에 의한 잡음 왜곡과 반사음으로 인한 잔향 왜곡 등이 있다.

3.2 부가정보

본 표준에서 정의하는 부가정보는 텍스트, 테이블 인덱스, 타임코드 데이터 등의 유형으로 구분되며, 비트스트림으로 변환되어 오디오 신호에 삽입되기 이전의 정보를 의미한다.

3.3 음향데이터

부가정보를 오디오 신호에 삽입하기 위해 비트스트림으로 변환한 형태를 음향데이터라고 정의한다. 이는 음향채널을 통해 부가정보를 전송할 목적으로 생성한 비트스트림 데이터이며, 앞서 정의한 부가정보와 구분된다.

3.4 음향데이터 전송 기술

안정적인 부가정보 전송을 위해 음향채널 왜곡에 강인하도록 음향데이터를 오디오 신호에 삽입하고 이를 추출하는 기술이다.

4 약어

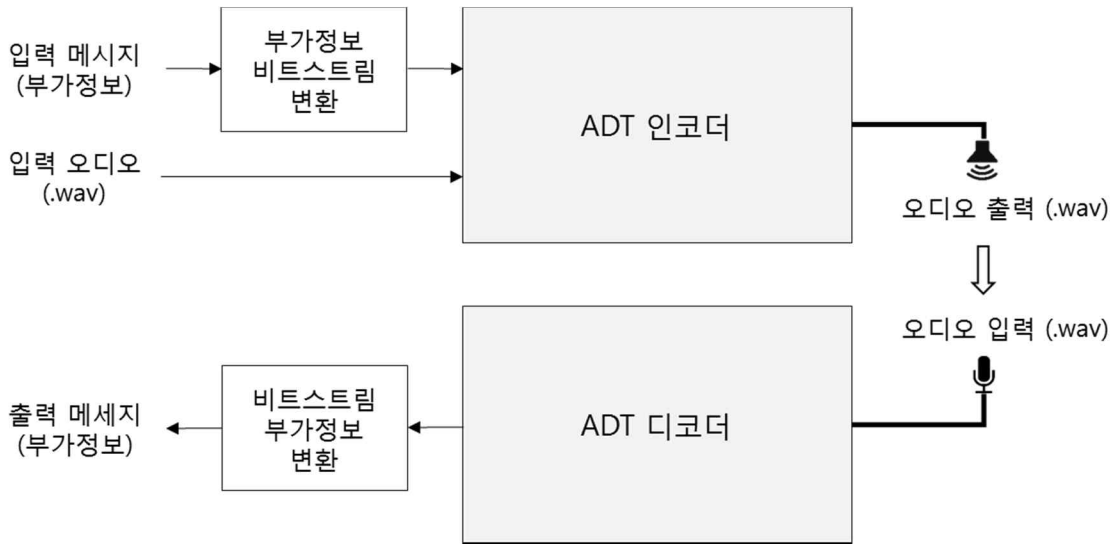
ADT	Acoustic Data Transmission
BER	Bit Error Rate
CRC	Cyclic Redundancy Check

5 음향데이터 포맷

5.1 음향데이터 전송 기술 개요

음향데이터 전송(ADT) 기술의 구성도는 (그림 5-1)과 같다. ADT 기술은 부가정보를 오디오 신호에 삽입하기 위한 인코더 기술과 부가정보를 추출하기 위한 디코더 기술로 구성된다. ADT 인코더는 삽입하고자 하는 부가정보를 비트스트림 형태인 음향데이터로 변환하여 오디오 신호에 삽입한다. 이때 삽입된 부가정보로 인한 오디오 음질 열화는 최소화되어야 하며, 음향채널 왜곡에 강인한 특성을 갖도록 해야 한다.

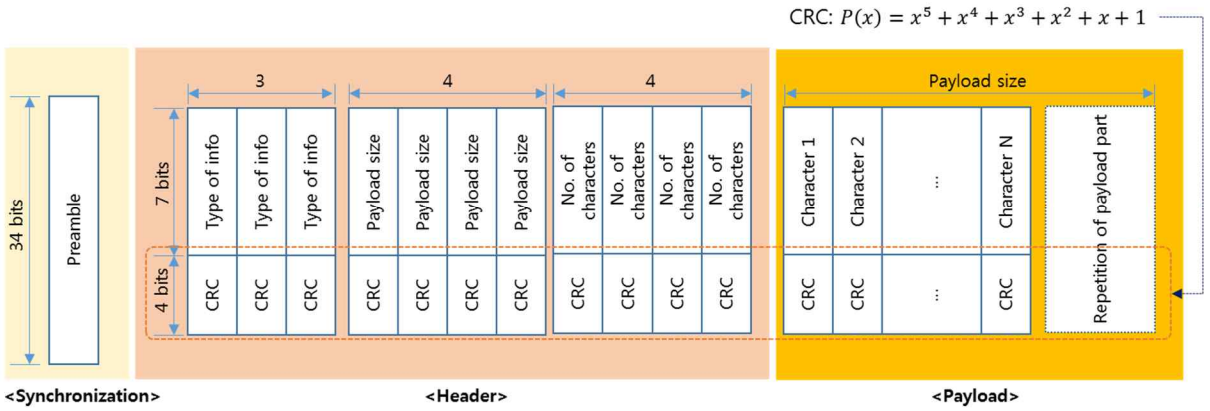
ADT 디코더는 음향데이터가 삽입된 오디오 신호를 입력 받아 비트스트림 형태의 음향데이터를 추출하고 이를 부가정보로 변환하여 최종 출력한다. 이에 앞서 처리되어야 하는 것은 입력된 오디오 신호 및 비트스트림 데이터에 대한 동기화이다. 본 표준에서는 특정 ADT 기술에 대한 규격을 규정하지 않으므로, 비트스트림 동기화를 위한 처리 부분만을 포함한다. 즉, 본 표준에서는 전송된 음향데이터의 비트스트림 정보가 어떻게 해석되어야 하는지를 정의하고 있으며, 이는 (그림 5-1)에서의 '부가정보 비트스트림 변환'에 해당한다.



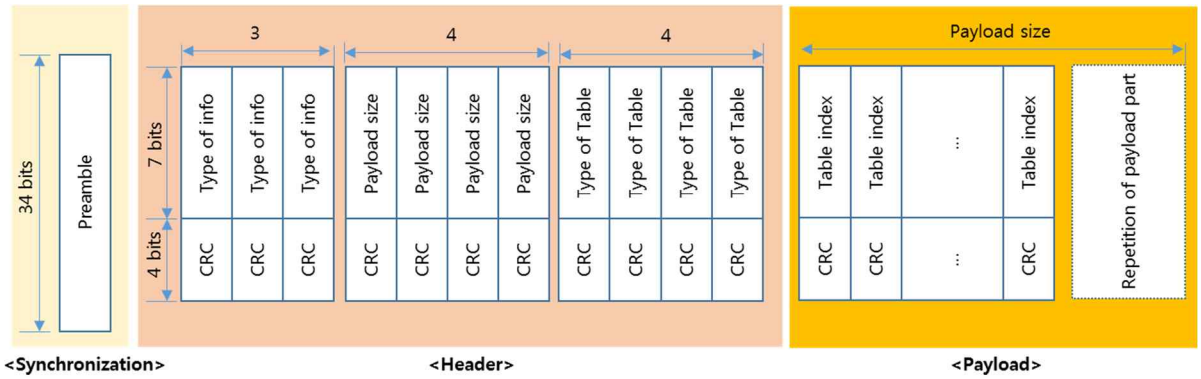
(그림 5-1) ADT 기술 구성도

5.2 음향데이터 비트스트림 구조

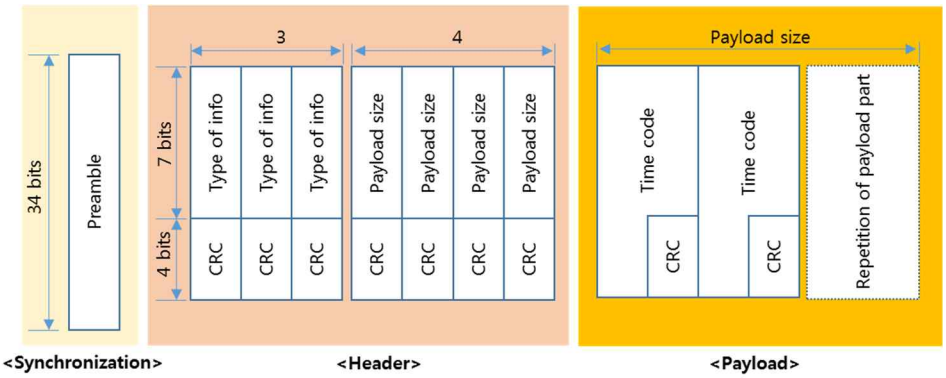
음향데이터는 비트스트림 형태로 ADT 디코더에서 추출되며, 부가정보 유형에 따라 그 구조를 달리한다. 이는 추출된 비트스트림 구조로부터 전송된 음향데이터 포맷을 알 수 있기 때문이며, 이를 도식화하면 (그림 5-2) ~ (그림 5-4)와 같다.



(그림 5-2) 텍스트 타입 음향데이터 비트스트림 구조



(그림 5-3) 테이블 인덱스 타입 음향데이터 비트스트림 구조



(그림 5-4) 타임코드 타입 음향데이터 비트스트림 구조

먼저 공통적으로 비트스트림 동기화를 위한 프리앰블 데이터가 비트스트림 내에 가장 앞단에 위치한다. 그 뒤로 음향데이터의 헤더 및 페이로드 데이터가 뒤따른다. 프리앰블 데이터는 송수신단에서 약속된 코드 정보로써, 자기 상관도가 높은 랜덤 이진 코드를 사용한다. 음향데이터의 코드워드 길이는 7 bit로 정의하며, 각각의 코드워드마다 4 bit의 CRC 코드를 정의한다. CRC 코드 생성식은 다음과 같다.

$$\mathrm{CRC} = \mathrm{x}^5 + \mathrm{x}^4 + \mathrm{x}^3 + \mathrm{x}^2 + \mathrm{x}^1 + 1$$

부가정보 유형에 따른 음향데이터 포맷은 5.2.1 ~ 5.2.3절에서 자세히 규정한다.

5.2.1 텍스트 타입 음향데이터 포맷

텍스트 타입으로 전송되는 부가정보는 자유도가 가장 높으며, ASCII 코드로 표현될 수 있는 모든 조합의 텍스트 정보를 전송할 수 있다.

헤더 정보는 (그림 5-2)에서와 같이 'Type of Info', 'Payload Size', 'No. of Characters'의 3개 필드로 구성되며, 'Type of Info' 필드는 부가정보 유형을 정의한다. 'Payload Size' 필드는 텍스트 데이터가 포함된 비트스트림 영역의 크기를 나타내며, 이를 설정하기 위해 'No. of Characters' 필드를 통해 전송되는 텍스트의 길이 정보가 필요하다. 따라서 'Payload Size'는 'No. of Characters'의 배수가 되는 코드워드 크기로 설정된다. 이때 하나의 코드워드는 7 bit로 구성되며, 이는 ASCII 코드 길이와 같다. 7 bit에 대한 CRC 코드 길이는 4 bit이며, 이를 이용해 각 코드워드(ASCII 코드)에 대한 오류 유무를 판단한다. 마찬가지로, 'No. of Characters'도 7 bit이므로 하나의 페이로드에서 전송 가능한 문자의 종류는 최대 127개 이다.

헤더 정보를 구성하는 개별 필드 데이터는 반복 전송되는데, 이는 음향채널을 통한 전송 시 발생하는 비트 오류를 감안하여 설계된 것으로 안전한 헤더 정보 전송을 위한 것이다.

5.2.2 테이블 인덱스 타입 음향데이터 포맷

테이블 인덱스 타입은 전송하고자 하는 부가정보가 특정 테이블의 인덱스 값인 경우에 적용된다. 헤더 정보는 (그림 5-3)에서와 같이 'Type of Info' 필드는 부가정보 타입을, 'Payload Size' 필드는 전송하는 테이블 인덱스 데이터의 크기를 정의한다.

특정 테이블 인덱스는 서비스 제공자가 정의할 수 있으며, 다양한 부가서비스 제공을 위해 사전에 정의된 테이블 정보를 사용자 단말에서 접근할 수 있다. 특정 테이블에 대한 구분은 'Type of Table' 필드에서 규정한다.

테이블 인덱스를 이용한 서비스 예를 살펴보면, 콘텐츠 관련 웹 접근 URL 정보를 사전에 테이블로 구축하고, 이와 같은 테이블 정보가 복수로 필요할 경우 'Type of Table' 필드를 이용하여 정의한다. 페이로드 정보에서 식별된 테이블 인덱스를 이용하여 사전에 정의된 테이블에서의 URL 정보를 찾아 부가서비스를 제공한다.

'Table index'는 7 bit로 구성되며, 총 128 개의 테이블 인덱스를 전송할 수 있다. 또한 테이블의 종류도 'Type of Table'에 대하여 7 bit를 할당함으로써 최대 128개의 상이한 테이블을 구축하여 활용할 수 있다.

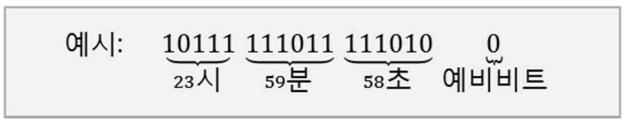
5.2.3 타임코드 타입 음향데이터 포맷

타임코드 타입은 전송하고자 하는 부가정보가 해당 콘텐츠의 시간정보를 나타내는 경우에 적용된다. 헤더 정보는 (그림 5-4)에서와 같이 'Type of Info'와 'Payload size' 필드를 포함하고 있으며, 본 표준에서 정의하는 타임코드 구조는 (그림 5-5)와 같다.

시간 (5 bits)	
bits	Value
00000	0
00001	1
00010	2
10111	23
...	reserved
11111	

분 (6 bits)	
bits	Value
000000	0
000001	1
000010	2
111011	59
...	reserved
111111	

초 (6 bits)	
bits	Value
000000	0
000001	1
000010	2
111011	59
...	reserved
111111	



(그림 5-5) 타임코드 구조

시/분/초 정보 표현을 위해 각각 5/6/6 bit로 배정된 17 bit의 타임코드에, 예비 비트 1 bit를 포함한 총 18 bit의 코드워드로 구성한다. 마찬가지로 4 bit CRC 코드를 활용하여 18 bit에 대한 오류 유무를 검증한다.

5.3 음향데이터 신택스

부가정보 유형별 비트스트림 구조에 대한 신택스 정의는 <표 5-1> ~ <표 5-10>과 같다.

<표 5-1> adt_data_extractor() 신택스

Syntax	No.of bits	Mnemonic

adt_data_extractor( )
{
payload_available = FALSE;
do {
do {
for (i = PreambleLength-1; i > 0; i--){
preamble_bits[i] = preamble_bits[i-1];
}
preamble_bits[0];
payload_available = preamble_sync(preamble_bits);
 	
} while(payload_available);
 	
adt_data_frame( );
payload_available = FALSE;
 	
} while(payload_available);
}
	1	uimsbf

PreambleLength — 동기화를 위해 비트스트림 앞 단에 삽입된 프리앰블 데이터의 비트스트림 길이로 34 bit 값으로 정의된다.

preamble_bits[] — 프리앰블 데이터를 나타내며, 전후 {1,1,1,1} 비트를 포함하는 보수(complement)열로 다음과 같이 정의된다.

$$preamble_{bits} = \{\underbrace{1,1,1,1,0,1,0,1,0,1,0,1,0,1,\ldots,0,1,0,1,0,1,1,1,1,1,1}_{PreambleLength}\}$$

preamble_sync() — 수신된 비트스트림으로부터 프리앰블 데이터를 찾는 동기화 함수이다.

adt_data_frame() — 프리앰블 데이터에 대한 동기화가 완료된 후, 전송된 ADT 데이터 프레임을 추출하는 함수이다.

<표 5-2> ‘payload_available’ 정의

Index	payload_available
TURE	비트스트림 동기화 수행 완료
FALSE	비트스트림 동기화 진행 중

<표 5-3> preamble_sync() 신택스

Syntax	**No.of bits**	**Mnemonic**

preamble_sync(preamble_bits, payload_available)
{
corr_rx_amble = FastCrossCorrelation(preamble_bits);
 	
if (corr_rx_amble > PreambleLength * (1 - Allowed_BER)){
payload_available = TURE;
	1	uimsbf

}
else{
payload_available = FALSE;		

} }		

corr_rx_amble	수신된 프리앰블 데이터와 원본 프리앰블 데이터간 상관도 계수로, 0 ~ PreambleLength 범위의 값을 갖는다.
Allowed_BER	동기화의 신뢰도 기준을 나타내는 값으로, 0 ~ 1 사이 값으로 표시한다. 예를 들어, 수신된 프리앰블 데이터의 BER을 20%까지 허용한다면, Allowed_BER은 0.2로 설정한다.
FastCrossCorrelation	수신된 프리앰블 데이터와 원본 프리앰블 데이터간 상관관계를 구하는 함수이며, 상관도 값이 corr_rx_amble 범위에 있도록 한다.

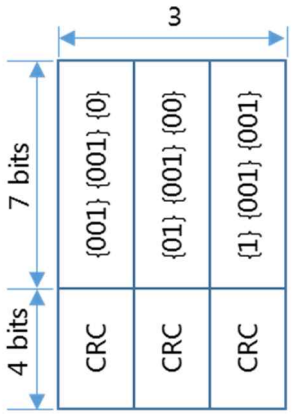
<표 5-4> adt_data_frame() 신택스

Syntax	**No.of bits**	**Mnemonic**
adt_data_frame( )		
{		
for (i = 0; i < 3; i++){		
TypeInfo[i·7**]**;	7 x 3	uimsbf
FixedHeaderCRC;	4	uimsbf
}		
for (i = 0; i < TypeInfoRepetition; i++){		
memcpy(&TypeOfinfoBuffer[i][0], &**typeInfo[**i·3**]**, 3);		
}		
TypeOfInfoIs = GetBitsSum(TypeOfinfoBuffer, 7, 3);		
for (i = 0; i < 4; i++){		
PayloadSize[i·7**]**;	7 x 4	uimsbf
memcpy(&PayloadSizeBuffer[i][0], &**PayloadSize[**i·7**]**, 7);		
FixedHeaderCRC;	4	uimsbf
}		
PayloadSizeIs = GetBitsSum(PayloadSizeBuffer, 4, 7);		
switch (TypeOfInfoIs){		
case ‘TextTypeInfo’		
NumChar = GetTextHeader( );		
TextPayloadData(PayloadSizeIs, NumChar);		
break;		
case ‘TableTypeInfo’		
TableType = GetTableHeader( );		
TablePayloadData(PayloadSizeIs, TableType);		
break;		
case ‘TimecodeInfo’		
TimecodePayloadData(PayloadSizeIs);		
break;		
}		
}		

TypeInfo	전송된 비트스트림의 부가정보에 대한 유형을 나타내며, 3 bit로 구성된다.
TypeInfoRepetition	부가정보 유형 관련 데이터에 대한 반복 전송 횟수를 알려주는 정보로 별도 전송되지 않으며, 본 표준에서는 7회 반복하여 전송하는 것으로 규정한다.
FixedHeaderCRC	CRC 코드값으로 4 bit로 구성된다.
PayloadSize	페이로드 크기에 대한 정보를 받기 위한 배열 변수이다.
GetBitsSum(A,B,C)	반복 전송된 비트스트림을 입력 받아 유효한 하나의 정보로 계산하여 제공하는 함수로, A는 수신된 비트스트림이며, B는 필드 데이터의 반복 횟수이며, C는 필드 데이터의 비트수를 나타낸다.
PayloadSizeIs	페이로드에 할당된 비트스트림의 실제 크기를 나타내며, 코드워드 11 bit 단위로 환산된 값이다
TextTypeInfo	전송된 부가정보가 텍스트 타입일 경우 정의하는 변수이다.
TableTypeInfo	전송된 부가정보가 테이블 인덱스 타입일 경우 정의하는 변수이다.
TimeCodeInfo	전송된 부가정보가 타임코드 타입일 경우 정의하는 변수이다.
NumChar	전송된 부가정보가 텍스트 타입일 경우, 유효한 텍스트 정보의 바이트 길이를 나타내는 변수이다.
TableType	전송된 부가정보가 테이블 인덱스 타입일 경우, 어떤 테이블의 인덱스인지를 알려주기 위한 테이블 유형을 나타내는 변수이다.
TextPayloadData(A,B)	텍스트로 전송된 부가정보를 로딩하는 함수로, A는 11 bit 단위로 환산된 페이로드 크기를, B는 텍스트 정보의 byte 단위 유효 길이를 인수로 입력 받는다.
TablePayloadData(A)	테이블 인덱스 유형으로 전송된 부가정보를 로딩하는 함수로, A는 11 bit 단위로 환산된 페이로드 크기를 인수로 입력 받는다.
TimecodePayloadData(A)	타임코드로 전송된 부가정보를 로딩하는 함수로, 11 bit 단위로 환산된 페이로드 크기를 인수로 입력 받는다.

<표 5-5> ‘TypeOfInfoIs’ 정의

Index	TypeOfInfoIs
000	‘TextTypeInfo’
001	‘TableTypeInfo’
010	‘TimeCodeInfo’
…	Reserved
111	Reserved



(그림 5-6) 테이블 타입({001}) 정보에 대한 ‘Type of Info’ 비트스트림 구성 예

‘Type of Info’ 필드는 <표 5-4>의 ‘TypeInfo’와 ‘TypeInfoRepetition’으로부터 (그림 5-6)과 같이 구성된다. 이는 부가정보 유형 정보가 7회 반복되어 21 bit로 전송되며, 7 bit씩 나누어 CRC 코드를 부가한다. 부가정보 유형에 대한 정보는 <표 5-5>와 같이 ‘TypeOfInfoIs’로부터 얻는다.

<표 5-6> GetTextHeader() 신택스

Syntax	No.of bits	Mnemonic
GetTextHeader( )		
{		
for (n = 0; n < 4; n++){		
NumCharSizeData[n·7**]**;	7 x 4	uimsbf
FixedHeaderCRC;	4	uimsbf
}		
for (i = 0; i < 4; i++){		
memcpy(&NumCharBuffer[i][0], &**NumCharSizeData[**i·7**]**, 7);		
}		
NumChar = GetBitsSum(NumCharBuffer, 4, 7);		
return(NumChar);		
}		

NumCharSizeData 전송된 텍스트 정보의 문자열 길이 정보를 받기 위한 구문이다. 7 bit 단위로 4회 반복되어 전송된 비트스트림을 얻는다.

NumChar 전송된 부가정보의 문자열 길이를 나타낸다.

<표 5-7> GetTableHeader() 신택스

Syntax	No.of bits	Mnemonic
GetTableHeader( )		
{		
for (n = 0; n < 4; n++){		
TableTypeData[n·7**]**;	7	uimsbf
FixedHeaderCRC;	4	uimsbf
}		
for (i = 0; i < 4; i++){		
memcpy(&TableTypeBuffer[i][0], &**TableTypeData[**i·7**]**, 7);		
}		
TableType = GetBitsSum(TableTypeBuffer, 4, 7);		
return(TableType);		
}		

TableTypeData	전송된 테이블 유형 정보를 얻기 위한 구문이다. 7 bit 단위로 4회 반복되어 전송된 비트스트림을 얻는다.
TableType	전송된 테이블 유형을 나타내는 변수이다.

<표 5-8> TextPayloadData() 신택스

Syntax	No.of bits	Mnemonic
TextPayloadData(PayloadSizeIs, NumChar)		
{		
for (n = 0; n < PayloadSizeIs; n++){		
TextDataBits[n**]**;	7xPayloadSize	uimsbf
FixedPayloadCRC;	4	uimsbf
}		
for (k = 0; k < NumChar; k++)		
TextDataRepetition = PayloadSizeIs/NumChar;		
for (n = 0 ; n < TextDataRepetition; n++){	1	uimsbf
memcpy(&CharBuffer[n][0],&TextDataBits[n*NumChar+k][0], 7);		
}		
TextData[k] = GetBitsSum(CharBuffer, TextDataRepetition, 7);		
}		
return(TextData);		
}		

TextDataBits	PayloadSizeIs x 7 형태의 2차원 배열 변수이다.
FixedPayloadCRC	페이로드의 CRC 값으로, 4 bit로 고정되어 수신된다.
TextDataRepetition	텍스트 정보가 페이로드에 반복되어 삽입된 횟수를 나타낸다.
TextData	텍스트 정보가 비트스트림 형태로 최종 복원되고, 이를 출력하기 위한 배열 변수이다.

<표 5-9> TablePayloadData() 신택스

Syntax	No.of bits	Mnemonic
TablePayloadData(PayloadSizeIs)		
{		
for (n = 0; n < PayloadSizeIs; n++){		
TableIndexBits[n];	7xPayloadSize	uimsbf
FixedPayloadCRC;	4	uimsbf
}		
TabletIndexData = GetBitsSum(**TableIndexBits**[n][0], TableDataRepetition, 7);		
return(TabletIndexData);		
}		

TableIndexBits	PayloadSizeIs x 7 형태의 2차원 배열 변수로, 텍스트 데이터를 7 bit 단위로 PayloadSizeIs 만큼 로딩한다.
TabletIndexData	텍스트 정보가 비트스트림 형태로 최종 복원되고, 이를 출력하기 위해 저장하는 배열 변수이다.

<표 5-10> TimecodePayloadData() 신택스

Syntax	**No.of bits**	**Mnemonic**

TimecodePayloadData(PayloadSizeIs)
{
for (n = 0; n < PayloadSizeIs/2; n++){
TimeCodeDataBits[n];
FixedPayloadCRC;
}
TimeCodeRepetition = PayloadSizeIs >> 1;
 	
for (n = 0; n < TimeCodeRepetition; n++){
memcpy(&TimeCodeBuffer[n>>1][0],
TimeCodeDataBits[n][0], 7);
memcpy(&TimeCodeBuffer[n>>1][7],
TimeCodeDataBits[n][0], 7);
}
 	
TimeCodeData = GetBitsSum(TimeCodeBits[n][0],
TableDataRepetition, 18);
 	
Memcpy(H_info,&TimeCodeData[0], 5);
Memcpy(M_info,&TimeCodeData[5], 6);
Memcpy(S_info,&TimeCodeData[5], 6);
 	
return(TimeCodeData);
}
	
	18xPayloadSize/2
	4
		
		uimsbf
		uimsbf

TimeCodeDataBits	PayloadSizeIs/2 x 18 형태의 2차원 배열 변수로, 타임코드 데이터를 18 bit 단위로 PayloadSizeIs 만큼 로딩한다.
TimeCodeBuffer	18 bit로 구성된 타임코드를 PayloadSizeIs/2개를 저장하는 2차원 배열이다.
TimeCodeRepetition	타임코드의 반복 전송 횟수로 PayloadSizeIs/2와 동일하다.
TimeCodeData	타임코드 정보가 비트스트림 형태로 최종 복원되고, 이를 출력하기 위해 저장된 배열 변수이다.
H_info	TimeCodeData로부터 시간 정보를 추출하여 저장하는 변수이다.
M_info	TimeCodeData로부터 분 정보를 추출하여 저장하는 변수이다.
S_info	TimeCodeData로부터 초 정보를 추출하여 저장하는 변수이다.

부 록 Ⅰ-1

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

지식재산권 확약서 정보

본 표준의 ‘지적 재산권 확약서’ 제출 현황은 차세대방송표준포럼 웹사이트에서 확인할 수 있다.

※ 상기 기재된 지식재산권 확약서 이외에도 본 표준이 발간된 후 접수된 확약서가 있을 수 있으니, 차세대방송표준포럼 웹사이트에서 확인하시기 바랍니다.

부 록 Ⅰ-2

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

시험인증 관련 사항

Ⅰ-2.1 시험인증 대상 여부

해당사항 없음

Ⅰ-2.2 시험표준 제정 현황

해당사항 없음

부 록 Ⅰ-3

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

본 표준의 연계(family) 표준

해당 사항 없음

부 록 Ⅰ-4

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

참고

해당사항 없음

부 록 Ⅰ-5

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

영문표준 해설서

해당 사항 없음

부 록 Ⅰ-6

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

표준의 이력

판수	채택일	표준번호	내용	담당 위원회
제1판	2017.1.06	제정 NGBF-STD-019	-	디지털라디오분과 위원회