HDLC(High-level Data Link Control) 프로토콜에서 효율적 문자부호 전송을 위한 문자부호화 규칙

홍완표*

Composition Rule of Character Codes to efficiently transmit the Character Code in HDLC(High-level Data Link Control) Protocol

Wan-Pyo Hong*

요 약

본 논문은 데이터 통신의 전송효율 측면에서 OSI 표현계층에서 수행되는 문자의 원천부호화에 대하여 연구하였다. 데이터링크 계층의 HDLC와 PPP 프로토콜은 프레임과 프레임간의 식별 및 수신기의 동기화 패턴용으로 프레임의 맨 앞뒤에 FLAG 바이트를 삽입한다. 이 FLAG 바이트는 “01111110”의 8비트열로 구성된다. 그러므로 데이터비트열에서 ″0″비트 이후 “1”의 비트가 연속하여 5개 이상 발생될 경우 데이터비트열이 플래그(flag)로 혼동되어 질 수 있다. 이를 방지하기 위해 HDLC에서는 데이터 비트열에 “1”의 비트가 5개 이상 연속될 경우 5번째 비트 다음에 “0”비트를 인위적으로 추가해 주고 있다. 그러므로 문자 부호에 연속 5개의 “1”비트열이 많이 발생하도록 부호화하게 되면 데이터 통신의 전송 효율에 영향을 주게 된다. 본 본문에서는 문자부호에 연속 5개 이상의 비트“1”이 발생 되지 않도록 하는 문자부호화 규칙을 제시하였다.

ABSTRACT

This paper is to show the character coding rule in computer and information equipment etc to improve the transmission efficiency in telecommunications. In the transmission system, the transmission efficiency can be increased by applying the proper character coding method. In datalink layer, HDSL ptotocol use FLAG byte to identify the frame to frame which consists of data bit stream and other control bytes. FLAG byte constits of ″01111110″. When data bit stream consist of the consecutive 5-bit ″1″ after ″0″, the decoder can not distinguish whether the data bit sequence is flag bit stream or data bit stream. To solve the problem, when the line coder in transmitter detects the consecutive 5-bits ″1″ after ″0″ in the input data stream, inserts violently the ″0″ after 5th ″1″ of the consecutive 5-bit ″1″ after ″0″. As a result, when the characters are decoded with the above procedure, the efficiency of system should be decreased. This paper shows the character code rule to minimize the consecutive 5-bits ″1″ after ″0″ when the code is given to each characters.

키워드

Source coding, Line coding, HDLC, PPP, FLAG
문자부호, 회선부호화, HDLC, PPP, 플래그

* 한세대학교 정보통신공학과(wphong@hansei.ac.kr)
접수일자 : 2012. 06. 13 심사일자(수정) : 2012. 07. 26 게재확정일자 : 2012. 08. 09

Ⅰ. 서 론

컴퓨터나 정보기기에서 OSI 표현계층에서 원천 부호화된 문자는 데이터링크 계층에서 프레임화 된다. 데이터링크 계층의 대표적인 프로토콜인 HDLC 프로토콜[1][2][3]과 PPP프로토콜[3][4]은 프레임과 프레임 간의 식별 및 수신기의 동기화 패턴용으로 프레임의 맨 앞뒤에 FLAG 바이트를 삽입한다. 이 FLAG 바이트는 “01111110”의 8비트열로 구성된다. 그러므로 표현계층에서 원천 부호화된 문자의 데이터 비트열에 ″0″비트 이후 “1”의 비트가 연속하여 5개 이상 발생될 경우 데이터비트열이 플래그(flag)로 혼동되어 질 수 있다. 이를 방지하기 위해 HDLC 프로토콜은 데이터 비트열에 “1”의 비트가 5개 이상 연속될 경우 5번째 비트 다음에 “0”비트를 인위적으로 추가해 준다. 문자중심 프로토콜인 PPP 프로토콜에서는 이러한 데이터 부호에 16진수 7D인 탈출문자를 부가하여 이를 구별해 준다. 그러므로 문자를 원천부호화 할 때 부호화된 비트열이 연속 5개의 비트 “1”을 발생하도록 구성할 경우 데이터 전송효율을 떨어뜨리게 될 수 있다. 본 논문에서는 OSI 표현계층에서 문자를 원천부호화 할 때 자주 사용되는 문자 부호에 비트 “1”이 연속하여 5개 이상 발생되지 않도록 하는 문자의 원천부호화 규칙을 제시하였다. 참고문헌[5][6][7]에서는 문자부호에 비트 “0”의 연속이 발생되지 않도록 하는 문자의 원천부호화에 대하여 논하였다.

Ⅱ. 문자 부호화 표준 규격

1.1 문자 부호화 주요용어정의

한국기술표준원은 2004년도에 정보교환용 부호계(한글 및 한자)에 대한 표준규격 KSX 1001을 개정 공표하였다[8]. 이 규격은 정보 처리 및 데이터를 전송하는 시스템에서 정보 교환에 사용하는 부호계의 표현 형식에 대한 것을 규정하고 있다. 이 규격에서는 문자, 도형문자, 특수문자, 숫자, 한글낱자, 한글글자마디, 로마문자, 한자, 그리스문자 그리고 바이트 등에 대한 정의를 하고 있다. 한글 낱자는 한글 글자 마디를 이루는 닿소리, 홀소리 글자 또는 첫소리, 가운뎃소리, 끝소리 글자(″?″ 이나 ″과″ 같은 겹글자도 하나의 낱자로 여긴다.)를 가리킨다. 한글 낱자는 닿소리 글자 30자, 홀소리 글자 21자, ″채움″(끝소리 글자 없음 표시) 1문자와 고어 42문자, 총 94문자로 한다.

한글 글자 마디는 첫소리와 가운뎃소리 글자, 또는 첫소리, 가운뎃소리, 끝소리 글자로 이루어진 한글의 단위이다. 한글 글자 마디는 사용 빈도에 의하여 선정된 한글 글자 마디 2,350자로 하고 있다.

한글낱자로 구성되는 문자를 조합형, 한글 글자마디로 구성된 글자를 완성형이라 한다. 문자의 코드는 행과 열로 구성된 2바이트조합으로 구성된다. 바이트는 정보 교환의 편의상 한 단위로 취급되는 8개의 비트를 의미한다. 행은 2바이트 부호 계에서 첫째 바이트에 의하여 구별되는 부호 값의 집합이다. 열은 지정된 행 안에서 둘째 바이트에 의하여 구별되는 각각의 부호 값을 의미한다. 조합형과 완성형한글문자의 구분은 부호 배열 표에 의하여 다음과 같이 식별할 수 있다. 완성형은 b0a1-c8fe, 조합형은 8861-d3b7이다[8].

1.2 문자 부호계 규격

참고문헌[8]에서는 부호계의 단위를 2바이트로 하고 있다. 각각의 바이트는 정보 교환용 부호계의 확장법 (ISO/IEC2022)에 따라 7비트 또는 8비트로 하고 있다. 2바이트 중 앞의 바이트를 첫째 바이트라고 하고 뒤의 바이트를 둘째 바이트라 하고 있다. 도형문자에 대한 부호계열은 7/8비트열 바이트 모두 b7−b1비트열로 구성된다[9]. 따라서 8비트열 바이트의 경우에는 최상위 비트인 b8비트가 항상 ″1″이 되게 한다. 이 규격에 포함되지 않는 한글 글자 마디는 첫소리, 가운뎃소리, 끝소리 글자로 분류하고 각 부분에 해당되는 특수 문자 영역의 한글 낱자 부호 값을 정보 교환에 사용한다. 한글 낱자만의 정보 교환과 구분하기 위하여 한글 글자 마디마다 맨 앞에 ″채움″ 문자를 1개씩 추가하도록 하고 있다.

1.3 문자부호의 분류와 배열

참고문헌[8]에서는 도형문자의 분류와 배열에 대하여 규정하고 있다. 사용 빈도가 높은 특수 문자, 숫자, 한글 낱자, 외국 문자 및 괘선 조각은 제1행~제12행

에 배열하고 있다. 한글글자마디는 총2350자를 가나다 순으로 제16행~제40행에 배열하고 있다.

1.4 2바이트 조합형 부호계 구성규격

참고문헌[8]부속서3에는 기본 부호계인 2바이트 완성형 부호계의 보조 부호계로서 2바이트 조합형 부호계를 규정하고 있다. 한글은 [8]부속서 3 표 1에 규정된 첫소리 글자 19자, 가운뎃소리 글자 21자, 끝소리 글자 27자로 조합 가능한, 모든 현대 한글 글자 마디(11 172자) 및 현대 한글 낱자(67자)에 대한 것이다. 한글글자마디에 대한 코드배치영역은 첫번째 바이트는 84-D3, 두번째 바이트는 41-7E, 81-FE로 규정하고 있다. 한글글자마디에 대한 부호 값은 2바이트 내에 첫소리 글자 5비트, 가운뎃소리 글자 5비트, 끝소리 글자 5비트로 하여 한글 낱자를 조합하여 표현한 값으로 하고 있다. 각 한글 낱자의 순서는 최상위 비트(MSB)를 1로 하고 첫소리, 가운뎃소리, 끝소리 글자가 순서대로 나오도록 구성하고 있다. 표 1는 이것에 대한 것이다[5].

표 1. 한글글자마디 구성 규칙
Table 1. Hangul sylable code composition rule

첫째 바이트 / 둘째 바이트

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
"1"	초 성							중 성			종 성				

1.5 7비트 한글낱자 부호계 규정

참고문헌[8]부속서 4에는 한글낱자 7비트 부호계에 대하여 규정하고 있다. 이 규격은 권장규격으로 원칙적으로 정보교환용으로는 사용하지 않는 것으로 규정하고 있다. 7비트 한글 낱자 부호 값은 한글 낱자 1자를 표현한다. 즉, 한 개의 글자마디는 첫소리 바이트, 가운뎃소리 바이트, 끝소리 바이트의 세 개의 바이트로 조합되어 나타낸다. 표 2에서 4/0의 채움 문자에 대한 용법은 첫소리, 가운뎃소리, 끝소리 글자가 없을 때 각각 그 자리에 넣는 부호이다[5].

표 2. 7비트 한글낱자 부호계
Table 2. Hangul jamo 7-bits code

구 분				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1	열 / 행	0	1	2	3	4	5	6	7
0	0	0	0	0	기능문자				(채움)	ᄀᆞᆭ		
0	0	0	1	1					ᄀ	ᄆ		
0	0	1	0	2					ᄁ	ᄇ	ᅡ	ᅭ
0	0	1	1	3					ᄀᆺ	ᄈ	ᅢ	ᅮ
0	1	0	0	4					ᄂ	ᄇᆺ	ᅣ	ᅯ
0	1	0	1	5					ᄂᆨ	ᄉ	ᅤ	ᅰ
0	1	1	0	6					ᄂᆫ	ᄊ	ᅥ	ᅱ
0	1	1	1	7					ᄃ	ᄋ	ᅦ	ᅲ
1	0	0	0	8					ᄃᆨ	ᄌ		
1	0	0	1	9					ᄅ	ᄍ		
1	0	1	0	10					ᄅᆨ	ᄎ	ᅧ	ᅳ
1	0	1	1	11					ᄅᆷ	ᄏ	ᅨ	ᅴ
1	1	0	0	12					ᄅᆸ	ᄐ	ᅩ	ᅵ
1	1	0	1	13					ᄅᆺ	ᄑ	ᅪ	
1	1	1	0	14					르	ᄒ	ᅫ	
1	1	1	1	15					ᄆᆞ		ᅬ	

이상에서와 같이 정보교환용 부호계는 국제 표준화기구 (ISO)의 정보 교환용 부호계의 확 장법(ISO/IEC 2022)을 따르는 2바이트 부호 계로 규정하고 있다. 부호 값의 범위는 2121 -7E7EH까지 8836문자를 수용할 수 있도록 구성하고 있다. 2바이트 조합형 부호계에서 모아 쓴 한글의 부호 값은 KS X 1003(KSC 5636)에 규정되어 있는 7비트 부호계와 8비트 부호계의 기능 문자와 중복되지 않도록 하고 있다. 첫소리와 가운뎃소리 글자 자리에도 "채움" 문자부호를 배열하여 2바이트 조합형 부호계에서 한글 낱자 부호 값도 한글 글자 마디와 같은 방법으로 만들 수 있도록 하고 있다. 또한 정보 교환용 한글 부호계에 포함되지 않은 한글 글자 마디는 그 출현 빈도가 매우 낮을 것으로 보고 보통 한글 낱자와 구분하기 위하여 각 한글 글자 마디마다 맨 앞에 "채움" 문자 1개씩 덧붙여 교환하도록 하고 있다[8]. 도형문자의 한글글자마디 부호표는 총2350 자를 가나다순으로 배열하고 있다. 이와

같이 현재의 한글 부호계 규격은 국제표준 부호배열 규격에 맞추어 가나다 또는 ㄱ, ㄴ, ㄷ 등의 순서로 단순 배열하고 있음을 알 수 있다[5].

III. 2진 문자부호와 플래그

3.1 플래그(FLAG)

데이터 링크계층에서는 바이트중심 프로토콜과 비트중심 프로토콜이 있다. 대표적인 바이트 중심 프로토콜로는 PPP(Point-to-Point) 프로토콜이 있다. 비트중심프로토콜로는 HDLC(High-lever Data Link Control) 프로토콜이 있다. 이 두 개의 프로토콜은 데이터 프레임을 구성하는데 모두 FLAG라고 하는 "01111110"의 8개 비트로 구성된 필드를 가지고 있다. 일반적으로 이 플래그 필드는 프레임의 맨 앞과 뒤에 배치된다. 즉 플래그 필드는 프레임의 시작과 끝을 나타내는 역할을 한다.

문제는 이러한 플래그 필드와 동일한 비트열을 가진 비트열이 데이터 비트열에 있을 때 시스템에서는 이 데이터비트열이 프래그인지 데이터인지 구분을 하지 못한다는 것이다. 다시 말하면 데이터 비트열에 플래그 필드와 동일한 비트열이 있을 경우 이 비트열을 플래그 필드 비트열로 오인을 한다는 것이다. 그러므로 데이터 비트열에서 플래그와 동일한 비트열이 발생되면 이 비트열을 플래그와 오인되지 않는 다른 비트열 패턴으로 바꾸어 주어야한다.

즉, 정보기기내에서 문자부호를 어떠한 형태로 구성하느냐에 따라서 플래그필드를 보호하기 위한 시스템내에서의 동작횟수를 감소시킬 수 있다. 결과적으로 문자부호를 구성하는 방법에 따라 시스템의 효율에 영향을 줄 수 있다는 것이다.

3.2 플래그 필드 보호규칙

가. 문자중심 프로토콜(PPP)

문자중심프토토콜[3][4]에서는 플래그 필드가 "01111110" 8비열로 구성되어 있다. 그림과 같이 프레임의 처음과 마지막 필드는 플래그로 채워진다. 만약 처음 플래그와 마지막 플래그 사이에 플래그 필드를 구성하는 비트열과 동일한 비트열이 있을 때는 플래그 필드와 오인되지 않기 위해 일정하게 정해진 비트열 즉 바이트를 추가한다. 이것을 바이트 채우기 또는 문자 채우기라고 한다. 이러한 바이트를 보통 탈출문자(ESC, Escape character)라고 한다. 프레임 중간에 있는 플래그 앞에 탈출문자가 있을 경우 디코더에서는 탈출문자 다음에 있는 플래그를 플래그 비트열로 인식하지 않고 데이터로 인식한다. 이것을 방지하기 위해 데이터열을 플래그와 구별하기 위해 채우기를 하는 경우에는 탈출문자를 중복하여 채워 넣는다. 즉 탈출문자필드가 1개인 것을 플래그로 인식하고 탈출문자가 2개인 경우에는 탈출문자 다음의 바이트를 데이터 비트열로 인식한다. 탈출문자는 "01111101"의 비트열로 구성되어 있다.

나. 비트중심 프로토콜(HDLC)

HDLC 프로토콜[1][2][3]은 대표적인 비트중심 프로토콜이다. 이 프로토콜에서도 플래그 필드 바이트의 비트열구성이 문자중심 프로토콜의 플래그와 같이 "01111110"로 되어있다. HDLC 프로토콜은 정보 프레임(I-frame), 감시프레임(S-frame), 무번호 프레임(U-frame) 등 3개의 프레임종류를 가지고 있다. 이 세 개의 프레임 모두 프레임의 맨 앞과 뒤에 플래그 필드를 가지고 있다. HDLC 프로토콜에서도 PPP에서와 같이 플래그 비트열과 동일한 데이터비트열이 있을 경우에는 두 비트열간에 오인이 발생되지 않도록 하기 위해 데이터 비트열을 인위적으로 플래그와 다른 형태의 비트열로 바꾼다. 이러한 방법은 데이터열에서 "0"비트 이후에 "1"의 비트가 연속하여 5개 이상 있을 경우 5번째 "1"비트 다음에 "0"비트를 강제로 삽입한다. 그래서 이러한 방법을 비트채우기(bit stuffing)방식이라고 한다.

즉 데이터 열에 비트 "1"이 연속하여 5개 이상 되도록 정보기기에서 문자를 부호화할 경우 시스템에서 비트 채우기 동작을 수행하여야 한다. 그러므로 문자부호가 비트 채우기를 얼마만큼 자주 일어나게 구성되느냐하는 것은 시스템과 데이터의 전송효율을 결정하는 요인이 되는 것이다.

Ⅳ. 문자 원천 부호화 규칙

4.1 부호화 범위와 조건

본 논문에서는 위에서 언급한 두 가지의 프로토콜을 기준으로 하여 정보기기내에서 문자를 원천 부호화하는 규칙으로 다음과 같은 범위와 조건을 적용하였다[5][6][7].

첫째 1개의 조합단위를 구성하는 3비트열, 4비트열 또는 단위조합 비트열과 단위조합비트열의 조합에 의한 바이트에서 "1"의 연속이 발생되지 않도록 한다. 둘째 여기서 1바이트는 행3 x 열4=7비트 또는 행4 x 열4=8 비트로 구성된다.

셋째 단위조합간의 결합은 행4,4 x열4,4=16비트 또는 행4,4,4 x열4=16비트로 구성된다. 넷째 문자부호의 단위바이트 또는 바이트의 조합에 있어서 플래그 바이트의 비트열이 구성되지 않도록 한다.

4.2 4비트 열 x 4비트 행 단위조합

표 3는 이러한 원칙을 적용한 4비트 x 4비트 단위조합에 대한 각 비트열 별 조합이 가능한 것과 조합이 제한되는 조합조건을 보여 주는 것이다[5]. 상위비트열이 16진수 2, 4, 6, 8, 10, 12, 14인 경우 하위 모든 비트열과 조합이 가능하다. 상위비트열이 16진수 1, 5, 9, D의 경우에는 하위 비트열 16진수 F를 제외한 모든 조합이 가능하다. 상위비트열이 16진수 3과 B의 경우에는 하위 비트열 16진수 E, F를 제외한 모든 조합이 가능하다. 상위 비트열이 16진수 7의 경우에는 하위 비트열 16진수 C, D, E, F외의 모든 조합이 가능하다. 상위비트열이 16진수 F인 경우에는 하위비트열 16진수 8, 9, A, B, C, D, E, F는 조합이 제한되고 O, 1, 2, 3, 4, 5, 6, 7은 조합이 가능하다.

표 3. 문자 원천부호화 규칙 ; 4 비트행 x 4비트열
Table 3. Character source coding rule; 4-bit rows x 4-bit ranks composition rule

HEXA	상위 비트열	하위 비트행	
		조합제한	조합가능
O	OOOO		O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
1	OOO1	F	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E
2	OO1O	-	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
3	OO11	E,F	O1,2,3,4,5,6,7,8,9,A,B,C,D,
4	O1OO	-	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
5	O1O1	F	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E
6	O11O	-	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
7	O111	C,D,E,F	O,1,2,3,4,5,6,7,8,9,A,B
8	1OOO	-	O,1,2,34,5,6,7,8,9,A,B,C,D,E,F
9	1OO1	F	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E
A	1O1O	-	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
B	1O11	E,F	O,1,2,3,4,5,6,7,8,9,A,B,C,D
C	11OO	-	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
D	11O1	F	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E
E	111O	-	O,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
F	1111	8,9,A,B,C,D,E,F	O,1,2,3,4,5,6,7

*하위비트열의 숫자는 16진수임

표 3와 같이 4비트 행 X 4비트 열의 조합으로 8비트열 바이트가 구성된다. 그 다음단계로 8비트열 각 바이트간의 조합이 고려되어야 한다. 표 4는 4비트열 16진수 조합으로 구성된 8비트 바이트간의 조합규칙에 대한 것이다[5][6]. 첫 번째 출력된 바이트의 최상위 비트열과 두 번째 출력된 바이트의 최하위 비트열과 조합되는 것이다. 예를 들어 첫 번째 바이트의 최상위 비트열이 16진수 1인 경우 최상위 비트열은 0001이 된다. 두 번째 출력된 바이트의 최하위 비트열이 16진수 2인 경우 비트열이 0010이 된다. 따라서 두 바이트의 조합은 0010 0001이 된다. 결과적으로 두 바이

트간의 조합에는 연속 5개 “1”의 비트열은 없다. 그러므로 조합이 가능하다. 표 4에서와 같이 16진수 0~7까지는 모든 조합이 가능하다. 16진수 8~F는 F를 제외한 모든 조합이 가능하다.

표 4. 8비트(1바이트)간 조합규칙
Table 4. Compositon between 8-bit bytes

16 진수	첫번째 바이트 최상위 비트열	두 번째 바이트 최하위비트열	
		조합제한	조합가능
0	0000	-	모두가능
1	0001	-	모두가능
2	0010	-	모두가능
3	0011	-	모두가능
4	0100	-	모두가능
5	0101	-	모두가능
6	0110	-	모두가능
7	0111	-	모두가능
8	1000	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
9	1001	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
A	1010	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
B	1011	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
C	1100	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
D	1101	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
E	1110	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14
F	1111	F	0,1,2,3,4,5,6,7,8,9,10,11,12 13,14

4.3 3비트 열 x 4비트 행 단위조합

표 5는 상위 3비트열 x 하위 4비트열 단위조합에 대한 각 단위조합별 조합이 가능한 것과 조합이 제한되는 조합조건을 보여 주는 것이다[6]. 이 규칙은 하위비트열과 상위비트열의 조합에 연속 5개 이상의 “1”이 발생되지 않도록 한 것이다.

예를 들면 상위비트열 “000”은 모든 하위 4비트열과 조합이 가능하다. “010”, “100”, “110”또한 모든 하위4비트열과 조합이 가능하다. 상위비트열 “001”“101”은 하위4비트열 16진수 F를 제외한 모든 비트열과 조합이 가능하다. 상위비트열 “011”은 하위 4비트열 16진수 E와 F를 제외한 모든 비트열과 조합이 가능하다. 마지막으로 상위비트열 “111”은 하위 4비트열 16진수 C, D, E, F를 제외한 모든 비트열과 조합이 가능하다.

표 6은 3x4비트열 조합으로 구성된 7비트열 바이트간의 조합규칙에 대한 것이다[6]. 표 6에 의하여 3x4비트열 조합에 의한 바이트가 구성되면 다음에는 표7과 같이 각 바이트간의 조합을 고려하여야 한다. 즉, 첫 번째 출력된 바이트의 최상위 비트열과 두 번째 출력된 바이트의 최하위 비트열과 조합에서 연속5개 이상의 “1”의 비트열이 생성되지 않도록 구성하는 것에 대한 것이다.

첫 번째 상위 비트열 “000”~“011”의 4개의 3비트열은 두 번째 바이트를 구성하는 하위 4 비트열 어느 것과도 조합이 가능하다. 첫 번째 상위 비트열이 “100”~“101”인 3비트열은 두 번째 바이트를 구성하는 하위 비트열 16진수 E, F를 제외한 모든 것과 조합이 가능하다. 첫 번째 상위 비트열 “110”은 두 번째 바이트를 구성하는 하위 비트열 16진수 7, F를 제외한 모든 것과 조합이 가능하다.

첫 번째 상위 비트열 “111”은 두 번째 바이트를 구성하는 하위 비트열 16진수 3, 7, B, F를 제외한 모든 16진수 비트열과 조합이 가능하다.

표 5. 원천 부호화 규칙; 3 비트행 x 4비트열
Table 5. Character source coding rule ; 3-bit rows x 4-bit ranks composition rule

3비트 열 (상위비트열)	4비트 행(하위 비트열)	
	조합제한	조합가능
000	-	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
001	F	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E
010	-	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
011	E, F	0,1,2,3,4,5,6,7,8,9,A,B,C,D
100	-	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
101	F	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E
110	-	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
111	C,D,E,F	0,1,2,3,4,5,6,7,8,9,A,B

* 4비트행의 숫자는 16진수임

표 6에서 한 바이트를 8비트 구성하기 위해 첫 번째 바이트를 구성하는 비트열의 최상위비트에 “1”를 추가하여 4비트로 했을 경우에 대한 조합이 제한되는 조건은 다음과 같다[5][6].

첫 번째 상위 비트열 “000”~“011”의 4개의 4비트열은 두 번째 바이트를 구성하는 하위 4 비트열에서 16진수 F를 제외한 어느 것과도 조합이 가능하다.

첫 번째 상위 비트열이 “100”~“101”인 3비트열은 두 번째 바이트를 구성하는 하위 비트열 16진수 7, F를 제외한 모든 것과 조합이 가능하다. 첫 번째 상위 비트열 “110”은 두 번째 바이트를 구성하는 하위 비트열 16진수 3, 7, B, F를 제외한 모든 것과 조합이 가능하다.

첫 번째 상위 비트열 “111”은 두 번째 바이트를 구성하는 하위 비트열 16진수 1, 3, 5, 7, 9, B, D, F와 조합이 제한되고 2, 4, 6, 8, A, C, E와는 조합이 가능하다.

표 6. 7비트(1바이트)간 조합규칙
Table 6. Compositon between 7-bit bytes

첫번째바이트 상위 비트열 (3비트)	두번째 바이트 하위비트열 (4비트)			
	하위 비트열 (4비트)	조합제한		조합가능[1]
		상위 3비트열기준	상위4[2]비트열 기준	상위 3비트열 기준
000	0000	-	F	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
001	0001	-	F	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
010	0010	-	F	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
011	0011	-	F	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F
100	0100	F	7,F	0,1,2,3,4,5,6,8,9,A, B,C,D,E
101	0101	F	7,F	0,1,2,3,4,5,6,8,9,A, B,C,D,E
110	0110	7,F	3,7,B,F	0,1,2,3,4,5,6,8,9,A, B,C,D,E
111	0111	3,7,B,F	1,3,5,7,9,B,D, F	0,1,2,4,5,6,8,9,A,C ,D,E
-	1000	-	-	-
-	1001	-	-	-
-	1010	-	-	-
-	1011	-	-	-
-	1100	-	-	-
-	1101	-	-	-
-	1110	-	-	-
-	1111	-	-	-

V. 연구결과 및 향후 연구내용

데이터 링크 계층의 문자중심 프로토콜인 PPP와 비트중심 프로토콜인 HDLC 프로토콜은 프레임을 구성하고 있는 플래그와 데이터 비트열간의 혼동을 방지하기 위해 각각 바이트 채우기와 비트 채우기를 한다. 플래그 필드의 비트열은 양 프로토콜에서 모두 “01111110”을 사용한다. 정보 기기내에서 문자부호를 구성할 경우 플래그와 오인될 수 있는 문자부호가 많이 만들어 질 경우 시스템과 데이터의 전송효율을 저하시킨다. 본 논문에서는 OSI 표현계층에서 정보 기기의 문자를 원천부호화 할 때 자주 사용빈도 문자부호에 플래그 비트열인 “01111110”과 오인될 수 있는 문자부호가 가능한 최소화 될 수 있도록 하는 문자의 원천부호화 규칙을 제시하였다. 본 논문에서 제시한 문자의 원천 부호화 규칙을 적용할 경우 데이터 전송효율을 크게 높이게 될 것으로 판단된다. 향후 본 논문에서 더 연구하여야 사항으로는 다음과 같다.

첫째, 본 논문에서 제시하는 규칙과 연속 4개 이상의 “0”비트열이 발생되지 않는 문자부호화 규칙과 연계한 원천부호화 규칙을 제시하는 것이다. 둘째 문자의 사용빈도에 따라 도형문자 및 낱글자에 대한 최적 문자부호의 배열을 최적화하는 것이다. 즉, 이 연구결과에 의거하여 현재 규격화되어 있는 도형문자 및 낱글자 부호체계에 대한 적정성을 분석하여 최적 안을 제시하는 것이다. 또한 이 경우에 전송효율을 어느 정도 개선할 수 있는가에 대한 정량적인 값을 도출해 내는 것이다.

1) 상위4비트열기준 생략
2) 상위3비트열의 최상위에 “1”비트 추가할 경우, 즉 상위3비트열이 4비트열이 되었을 경우에 대한 것이다.

참고 문헌

[1] http://en.wikipedia.org/wiki/High-Level_Data_Link_Control

[2] “Data Communication Lectures of Manfred Lindner-Part HDLC” http://en.wikipedia.org/wiki/High-Level_Data_Link_Control. Reference

[3] Behrouz A. Forouzan, “Data communications and Networking" Fourth Edition. McGraw Hill Korea, p. 339, May, 2007.

[4] Behrouz A. Forouzan, “Data communications and Networking" Fourth Edition. McGraw Hill Korea, pp. 345, May, 2007.

[5] 홍완표, “데이터 전송 효율을 고려한 4비트행x4비트열 2 바이트 문자 부호화 규칙에 관한 연구”, 한국항행학회논문지, 15권, 5호, pp. 749-756, 10, 2011.

[6] 홍완표,“데이터 전송효율을 고려한 3x4비트 1바이트 문자부호화 규칙에 관한 연구”, 한국전자통신학회논문지, 6권, 4호, pp. 499-504, 8, 2011.

[7] 홍완표,“데이터통신 전송효율과 ASCII 부호체계 고찰” 한국전자통신학회논문지, 6권, 5호, pp. 657-664, 10, 2011.

[8] 산업자원부 기술표준원,“KS C 5601 : 1987 (1987년 고침) : 정보 교환용 부호(한글 및 한자)” 12. 2004

[9] 산업자원부 기술표준원,"정보 교환용 부호계 (한글 및 한자) 부속서 3. 보조 부호계(2바이트 조합형 부호계", KS X 1001 : 2004. 12, 2004.

[10] ITU-T Recommendation G.703, "Physical/ electrical characteristics of hierarchical digital interfaces" pp. 24-41, Oct. 1998.

[11] TTA Standard, “Test Method for Telecommunication Terminal Equipment” TTAS. KO-05.0028/R1, pp. 306-451, Revised on 23 Dec. 2004.

[12] Behrouz A. Forouzan, “Data communications” McGraw Hill Korea, pp. 132-134, Jan, 2008.

저자 소개

홍완표(Wan-Pyo Hong)

1991년 서울과학기술대학교 전자공학과 졸업(공학사)
1994년 연세대학교대학교 공학대학원 산업공학과 졸업(공학석사)
1999년 광운대학교 대학원 전자공학과 졸업(공학박사)
1990년 전기통신기술사합격
1991년 정보통신부 5급특별채용고시합격 본부 통신정책실, 전파방송관리국, 정보화기획실
1997년 삼성전자(주) 통신사업부 전송영업그룹장
1999년 광운대학교 연구전담교수
2000년 한국정보통신기술협회장
2002년 한세대학교 IT학부 정보통신공학전공 교수
한세대학교 정보통신연구소장
※ 관심분야 : 위성통신방송, 문자코딩, 통신정책