情報処理学会 試行標準 IPSJ-TS 0015:2015
Information Processing Society of Japan - Trial Standard
IPSJ-TS 0015:2015

エラー訂正機能付き高信頼伝送路符号 4b/10b

Dependable Line Code with Error Correction Capability: 4b/10b

Publication of version 1 (this version)	2015-09-07
Publication of version 2	--
Publication of version 3	--
Cor.1 to this version	--
Comment to	TS desk of IPSJ/ITSCJ

Contents

Patent Statement

The IPSJ draws attention to the fact that it is claimed that compliance with this Trial Standard may involve the use of patents concerning Japanese patent application No. 2012-14181.

The IPSJ takes no position concerning the evidence, validity and scope of this patent right. The holder of this putative patent right has assured the IPSJ that it is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of the putative patent rights is registered with the IPSJ. Information may be obtained from:

Faculty of Science and Technology, Keio University
3-14-1, Hiyoshi, Kouhoku-ku, Yokohama, Kanagawa 223-8522, Japan.

Attention is drawn to the possibility that some of the elements of this Trial Standard may be the subject of patent rights other than those identified above. The IPSJ shall not be held responsible for identifying any or all such patent rights.

Information Processing Society of Japan - Trial Standard IPSJ-TS 0015:2015

Dependable Line Code with Error Correction Capability: 4b/10b

0. Introduction

This trial standard defines a line code that incorporates the error correction capability to communicate among components, I/O peripherals and computers reliably. Complex machines, such as robots, automobiles, and network routers, have a growing demand for distributed processing. In addition, modernization of facilities such as factories, offices, schools, and homes is creating a ubiquitous computing environment. Unlike conventional PC applications for documentation and Internet applications that exchange texts without hard time constraints, these types of cooperative computing require reliable real-time responses to physical events occurring in the real world. In order for distributed nodes to cooperate in real-time, an interconnecting network shall realize real-time and dependable communication without re-sending on noisy environments. The 4b/10b provides a dependable line code for such real-time communications between components, I/O peripherals and/or computers by providing embedded clock, DC balance, error detection and error correction features.

0.1 Real-time

The real-time means that the exactness of the system including operations and communications depends on not only the result but also the time it took to achieve the result. In the narrow sense, the real-time means that the time constraint including deadline or cycle must be met.

Real-time tasks with the time constraints are generally scheduled and executed by a real-time scheduler and a real-time operating system. Most real-time scheduling algorithms assume that the WCET (Worst Case Execution Time) of each task is given. A real-time scheduling algorithm converts a time constraint of each real-time task to a priority. Most real-time operating systems based on such real-time schedulers preempt and execute tasks in order of priority at every tick to meet the time constraint.

0.2 Real-time scheduling

As real-time scheduling algorithms, the Earliest Deadline First (EDF) scheduler, the Rate Monotonic (RM) scheduler, and their variations have been established, as explained in Annex A. These algorithms commonly schedule tasks based on priorities determined by the time constraints.

Most real-time scheduling algorithms assume that the WCRT (Worst Case Response Time) of each communication packet is given in case of communication. In order to apply real-time scheduling algorithms to real-time communications, preemptive communication, which is achieved by Responsive Link, and the error correction capability to prevent from re-sending a broken packet are required.

0.3 Line code

A line code is a lowest-level communication protocol on a communication line. Most current line codes have a few typical functions including embedded clock, DC balance

and basic error detection features. The 8b/10b codec is a major example, which is used for PCI Express, USB 3.0, SATA, IEEE1394b, and 10GbE. But no conventional line code has an error correction capability.

0.4 Demerits of the 8b/10b codec

When an encoded code (a 10b code) is broken during communication, the multi-bits of the decoded code (the 8b code) is corrupted. In other words, when a single bit error occurs in an encoded 10-bit code, the decoded 8-bit code (a byte) is completely broken.

When an error is detected on the decoder, the broken data is normally re-transmitted under an upper-level communication protocol. However, the re-transmission is not allowed to realize a real-time communication.

It is hard for a bit-level error correction code including the Hamming code and the BCH code to incorporate the error correction capability, because multi-bits of the decoded code is broken even if a single-bit error occurs on the encoded code.

In order to incorporate the error correction capability on the 8b/10b codec, a block-level error correction including RS (Reed Solomon) is required as a large packet level error correction. But the block-level error correction is not suitable for real-time communication, because the communication latency becomes longer as it is impossible to correct the corrupted data until all corresponding packet are received.

0.5 Important features

The line code 4b/10b has the following distinctive features for real-time communications:

- Embedded clock
- DC balance
- Error detection
- Error correction

No conventional line code supports the above features at one time. For example, the industry-wide standard 8b/10b codec can be easily replace with the 4b/10b line code for highly reliable communications.

0.6 Typical applications and operations

Figure 1 shows a distributed control configuration of a humanoid robot as one of typical applications of the 4b/10b. The electronic control part of the humanoid robot consists of several control nodes with local sensing and actuating devices. The distributed controllers are connected to each other by Responsive Link. In this figure, rectangles represent node controllers, and dotted lines show communication links such as the Responsive Link that is a point-to-point serial link.

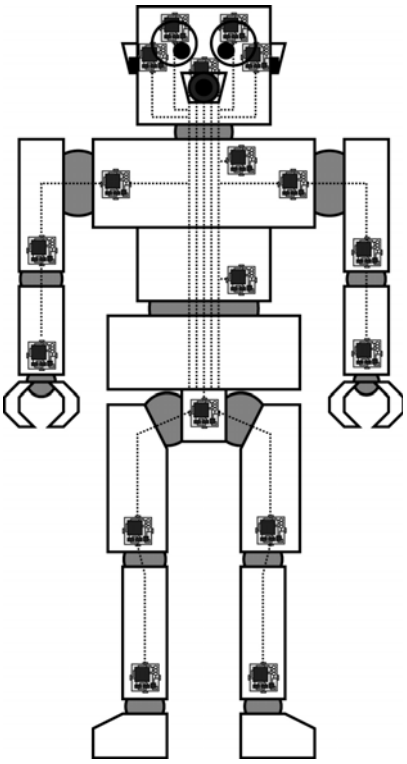

Figure 1 - A humanoid robot

For a humanoid robot to walk stably, a servo loop of 1 ms or shorter is needed. In this configuration, the farthest two nodes can exchange a 16-byte packet within 5 μ sec. Since the time is guaranteed not to fluctuate, the distributed control of a humanoid is considered to be sufficiently possible. Since many actuators that generate noises are embedded inside the robot, the line code is required for noise tolerance. The 4b/10b is the line code that has error correction capability.

Currently many I/O interfaces and communication standards including PCI Express (PCIe), USB 3.0, SATA, IEEE 1394b, and 10GbE use the 8b/10b codec as a line code. The 8b/10b has a lot of functions and its code rate is relatively high (about 80%). However if one bit error occurs in an encoded data (10b), the decoded data (8b) will be broken completely. Therefore when the 8b/10b codec is used on noisy environment such as inside the robot, an upper-level error correction code is required. In order for error correction, it is hard to apply any bit-level error codec including the Hamming code and the BCH code, because multiple decoded bits (1-byte) will be broken even if an encoded bit is inverted, so that block-level error correction including Reed-Solomon, which is long latency ECC that is not suitable for real-time applications, is required. Hence a reliable line code with ECC is highly required for such applications.

1. Scope

This trial standard specifies the line code 4b/10b for dependable communications. This standard corresponds to the functions specified in layer 1 to layer 2 of the OSI reference model.

The purpose of this standard is to facilitate the development and use of the 4b/10b in dependable systems by providing a line code protocol. This standard provides a line code protocol for interconnections among distributed real-time systems, including embedded systems, control systems, amusement systems, robot systems, and intelligent buildings. The 4b/10b can achieve the line code with ECC (error code correction). The 4b/10b is the line code that realizes embedded clock, DC balance, error detection and error correction at a time, which was not able to satisfy these functions in one codec by conventional schemes, so that the 4b/10b line code can achieve highly reliable and dependable digital communications.

2. Normative references

2.1 OSI reference model

ISO 7498, Open System Interconnection - Basic reference model

2.2 Responsive Link (RL)

ISO/IEC 24740:2008, the communications protocol and interface that inter-connect computers for distributed real-time control applications

2.3 8b/10b

Albert X. Widmer, 8b/10b encoding and decoding for high speed applications, IBM research report, 2004-1103

2.4 PCI Express (PCIe)

PCI Express base specification, PCI-SIG

http://pcisig.com/

2.5 USB 3.0

Universal Serial Bus Revision 3.0 Specification, USB Implementers Forum

http://www.usb.org/

2.6 SATA (Serial ATA)

Serial I/O interface for hard drive, SSD, and optical drive, Serial ATA working group

http://www.sata-io.org/

3. Terms and definitions

For the purpose of this standard, the following definitions apply:

3.1 byte (B)

A group of eight bits

3.2 half byte (HB)

A group of four bits

3.3 4b

Original half byte (4-bit) data

3.4 10b

Encoded 10-bit data for transmitting

3.5 symbol

A unit for encoding

The size of a symbol is 10 bits.

3.6 Frame

A unit for transmitting

The size of a frame is 10 bits.

3.7 ECC

Error Correction Code

3.8 DC

Direct-Current

3.9 WCET

Worst Case Execution Time

3.10 WCRT

Worst Case Response Time

4. 4b/10b line code

4.1 Overview

The line code 4b/10b shall handle a 10-bit frame encoded by 4-bit digits. The original 4 bits of information digits shall be encoded into a 10-bit frame by the look-up table shown in Table 1. A byte (8 bits) is divided into two half bytes (4b). A half byte (4b) is encoded to a symbol (10b). A frame consists of a symbol and is transmitted to the communication line. Since four bits are encoded to ten bits by the 4b/10b line code, the communication speed at 1,000 MHz is approximately equal to 400 Mbit/s.

4.2 Forward error correction (FEC)

The line code 4b/10b should provide error-free transmission for reliable real-time control. Error correction should be performed by hardware. The 4b/10b shall perform line-code-level error correction. Original four-bit data (4b) shall be encoded to 10-bit transmitting data with embedded clock, DC balance, 2bit error detection, and 1-bit error correction for a half byte data (4 bits of information digits).

The Hamming distance of any digits in a symbol (10b code) shall be longer than or equal to 4 for 1-bit error correction and 2-bit error detection.

4.3 Embedded clock

The line code 4b/10b keeps that successive 0 or 1 bits shall be within five bits, even if 1-bit error/symbol occurs.

In case of inside symbol digits, successive 0 or 1 bits shall be within five bits.

In case of inter-symbol digits, successive 0 or 1 bits shall be within five bits.

When each symbol has 1-bit error, if the distance of error bits are greater than four-bit, the successive 0 or 1 bits are within five bits.

In case of two bit errors, discontinuity of digits is not guaranteed.

4.4 DC balance

In order for the line code 4b/10b not to allow a current to flow in the communication cable, the numbers of 0 and 1 in a symbol shall be same for DC balance. But DC balance between successive symbols is not necessary.

If an error occurs, bit-level DC balance is not guaranteed among nearest neighbour error symbols.

4.5 4b/10b data encoding

In case of encoding, the look-up table that satisfies the above three conditions including embedded clock, DC balance, and error detection and correction, is used as shown in Table 1. Original 4-bit digits (4b) are encoded to 10-bit digits (10b).

Table 1 - The 4b/10b data transform table

4b	10b
0000	1100101100
0001	1011001100
0010	1100110010
0011	0110011100
0100	0111010001
0101	1100011001
0110	0101110100
0111	1101000101
1000	1001110001
1001	0111000110
1010	1010110100
1011	1101001010
1100	1011010010
1101	1001100110
1110	1010101001
1111	0110101010

4.6 Frame format

4.6.1 Frame

A frame shall consist of 10 bits, including 4 information bits and 6 redundant bits implicitly, as shown in Table 1.

4.6.2 Setup command

After the power is first applied, or after an unexpected burst link error occurs, the synchronization between the sender and the receiver can be lost. In such a situation, the link shall be initialized explicitly. The encoder in the initial mode shall send the setup pattern shown in Table 2. The decoder can distinguish the pattern from normal frames and thus switches to the initial mode. The initialized decoder interprets the first receiving frame after the initialization as the start frame of a new frame sequence.

Table 2 - Setup command

Setup pattern
0110100101

4.6.3 Idle command

When an encoder has no actual communication data, the encoder shall send the idle pattern shown in Table 3 in order to maintain the frame synchronization of the link.

Table 3 - Idle command

Idle pattern
0101101001

4.7 Encoding

The look-up table that satisfies embedded clock, DC balance, error detection and correction, and control commands including setup and idle command is shown in Table 4. In case of encoding, the 4b/10b look-up table shall be directly used.

Table 4 - The 4b/10b look-up table

4b (i)	10b (C_i)
0000	1100101100
0001	1011001100
0010	1100110010
0011	0110011100
0100	0111010001
0101	1100011001
0110	0101110100
0111	1101000101
1000	1001110001
1001	0111000110
1010	1010110100
1011	1101001010
1100	1011010010
1101	1001100110
1110	1010101001
1111	0110101010
setup	0110100101
idle	0101101001

4.8 Decoding

Decoding shall be based on the shortest distance decoding on the Hamming distance. An implementation of a decoder is illustrated in Annex D.

4.9 Error handling

4.9.1 1-bit error

The encoding scheme of the 4b/10b can automatically detect and correct any 1-bit error in a frame. However, when errors of greater than 2 bits are present in a frame, the error correction mechanism does not work. In such a situation, the calculation of the syndrome results in one of the following two cases:

4.9.2 2-bit error

In this case, the decoder can detect an unrecoverable fatal error. If a fatal error is detected in a frame, then the decoder shall not try to correct digits in the received frame and should interrupt the controller (processor).

4.9.3 Over 3-bit error

Although the probability of this case is very low, the decoder cannot detect the occurrence of the fatal error in this case. Since this error is indistinguishable from other correctable 1-bit errors, the received frame is inadequately modified by the decoder. This situation allows transmission of an incorrect packet and is highly undesirable. Therefore, when simple 1-bit errors are corrected in two successive frames, the decoder shall consider this to be a fatal error that cannot be corrected, and so handle the frame in the manner described above.

Annex A (informative)
Real-time scheduling

There are several real-time scheduling algorithms, including Earliest Deadline First (EDF), which is an optimal dynamic scheduling algorithm, and Rate Monotonic (RM), which is an optimal static scheduling algorithm.

The EDF algorithm translates the deadline to a priority. The priority of the task with the earliest deadline becomes the highest.

The RM algorithm translates the cycle time to a priority. The task with the shortest cycle is assigned to have the highest priority.

Many other real-time scheduling algorithms also translate the time constraint to a priority.

Figure A.1 shows an example of EDF scheduling. Priority-based scheduling is performed at every clock tick and at timings when tasks are released (invoked), as well as at execution finish.

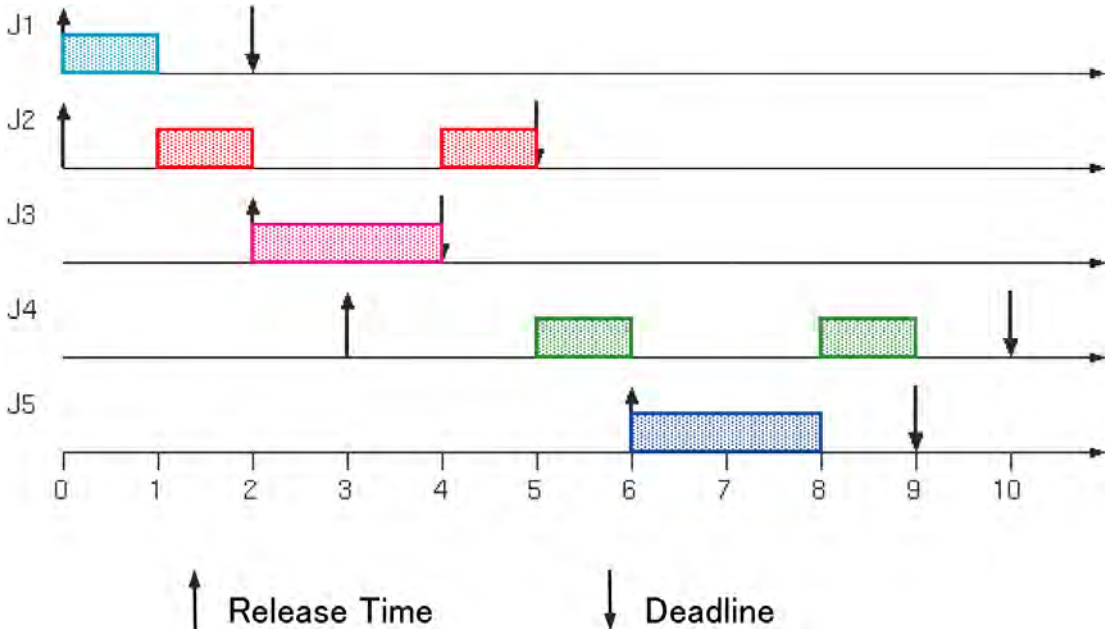

Figure A.1 - EDF scheduling

This real-time task scheduling process can be regarded as an overtaking process, i.e. tasks with higher priorities are executed earlier than tasks with lower priorities. In order to implement this idea in a distributed real-time system, communication of higher priority tasks should be able to overtake other communication. The Responsive Link does this at every node.

Annex B (informative)
Characteristics of embedded clock

The line code 4b/10b keeps that successive 0 or 1 bits shall be within five bits, even if 1-bit error/symbol occurs.

In case of inside symbol digits, successive 0 or 1 bits shall be within five bits. For example, if 1-bit error occurs on 1100110010 symbol, successive 0 or 1 bits are within five bits as shown in Table B.

Table B - The length of successive 0 or 1 in case of 1-bit error

The number of errors	Line code	The length of successive 0 or 1
0 (original)	1100110010	2
1	0100110010	2
1	1000110010	3
1	1110110010	3
1	1101110010	3
1	1100010010	3
1	1100100010	3
1	1100111010	3
1	1100110110	2
1	1100110000	4
1	1100110011	2

In case of inter-symbol digits, successive 0 or 1 bits shall be within five bits. For example, inter-symbol successive 0 or 1 bits of

1100110010 1001110001 symbols are within five bits.

When each symbol has 1-bit error, if the distance of error bits are greater than four-bit, the successive 0 or 1 bits are within five bits as follows.

- 1100110010 1001110001: Original
- 1100110000 1000110001: 1-bit error/symbol

In case of two bit errors, discontinuity of digits is not guaranteed.

Annex C (informative)
Characteristics of DC balance

In order for the line code 4b/10b not to allow a current to flow in the communication cable, the numbers of 0 and 1 in a symbol shall be same for DC balance. But DC balance between successive symbols is not necessary. In other words, the isomery of 0 and 1 inside a symbol is guaranteed. But the isomery of 0 and 1 in any connecting 10-bit window is not guaranteed as shown in Table C.

Table C - An example of the isomery of 0 and 1 in a successive 10-bit window

Window number	10-bit window	The number of 1
0	1100101100 0111010001	5
1	1100101100 0111010001	4
2	1100101100 0111010001	4
3	1100101100 0111010001	5
4	1100101100 0111010001	6
5	1100101100 0111010001	5
6	1100101100 0111010001	6
7	1100101100 0111010001	5
8	1100101100 0111010001	4
9	1100101100 0111010001	4

If an error occurs, bit-level DC balance is not guaranteed among nearest neighbour error symbols.

Annex D (informative)
Implementation of a decoder

1. Each Hamming distance HD_i between a received frame (10b) and each symbol C_i as shown in Table 3 shall be calculated.

2. Minimum value of the Hamming distance HD_{min} is calculated by the following equation:

   $HD_{min} = \min\{ HD_i \}$.

   When HD_k is equal to HD_{min}, the 10b C_k is decoded to the 4b k.

3. If HD_{min} is equal to 0, the 10b C_k is decoded to the 4b k without error.

   If HD_{min} is equal to 1, the 10b C_k is decoded to the 4b k. In this case, 1-bit error is corrected, and the corrected error should be informed to the upper layer.

   If HD_{min} is greater than 1, the 10b C_k is decoded to all 0. In this case, multiple bit error is detected, and the error should be informed to the upper layer. The decoded 4b k (all 0) is broken.

4. The 10b symbol C_k and corresponding decoded 4b k are determined.