

Multiplex Communication Protocol for Switch/Sensor/Actuator Network: “CXPI”

Eiji Taki, Yoshiro Hirata, and Yoshifumi Ohmori
Toyota Motor Corporation

Naoji Kaneko
DENSO Corporation

Hiroya Andou
Toyota Motor Corporation

ABSTRACT

The growing functionality and complexity of recent vehicle electronic systems have made inter-device communication (on-board LAN) technology vital to vehicle design. By field of application, the LAN (Local Area Network) systems currently in use are LIN (Local Interconnect Network) used for body systems, CAN (Controller Area Network) used for control systems, and MOST (Media Oriented Systems Transport) used for multimedia and camera systems, and work to standardize the next-generation communication technology for each of those fields is underway. This paper provides a technical overview of the CXPI (Clock Extension Peripheral Interface) communication protocol, which satisfies the body system requirements (rapid response, system extensibility, high reliability, and low cost). It also presents the progress made on standardization at SAE and other organizations. The current situation of standardization for CXPI is as follows: JASO standard [1][2][3]: May 2014 (1st edition), May 2015 (2nd edition), and May 2016 (final), SAE standard: November 2015 (IR), and ISO standard: May 2015 PWI, May 2016 (NWIP), August 2017 (WD), and August 2019 (IS).

CITATION: Taki, E., Hirata, Y., Ohmori, Y., Kaneko, N. et al., "Multiplex Communication Protocol for Switch/Sensor/Actuator Network: “CXPI”," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 9(1):2016, doi:10.4271/2016-01-0057.

INTRODUCTION

CXPI is a communication protocol established as a standard by JASO that is currently also undergoing SAE and ISO standardization. Although communication in body systems currently relies on LIN, high expectations are being placed on CXPI as a protocol capable of meeting the stringent (severe) requirements of body system communication. However, EMC resistance is an issue since CXPI uses the same single wire communication as LIN, making it necessary to increase said resistance in the low frequency band used for communication.

The basic functions and characteristics of CXPI, as well as measures to raise EMC performance, are presented in the following sections.

BASIC CXPI FUNCTIONS

The basic functions of CXPI will be described first. This paper limits itself to the fundamentals of the communication protocol. For information on its details, the JASO standards [1][2][3] should be consulted.

CXPI is a dedicated vehicle communication protocol with a transfer rate of up to 20 kbit/s. Its basic characteristics include the use of a single wire connection, the possibility of attaching up to 16 nodes, compatibility with multi-master access using the CSMA/CR (Carrier Sense Multiple Access with Collision Resolution) protocol, and high error detection functionality.

Communication System Configuration

Figure 1 presents an example configuration for a CXPI communication system that consists of one clock master node with a communication microcontroller, and a slave node with no microcontroller. A low-cost UART (Universal Asynchronous Receiver Transmitter) equipped microcontroller can be used as the communication microcontroller. The clock master node generates a constant clock signal on the bus. Here, the clock signal refers to a PWM (Pulse Width Modulation) waveform of a logical value of ‘1’ which always starts with a falling edge. That falling edge is considered as the clock signal. Using the clock signal transmitted on the communication bus makes it unnecessary to be equipped with an internal high precision oscillator in the slave node. The clock master and slave nodes superimpose the data to transmit with the clock signal for transmission on the communication bus.

In case the clock master node sends data to the bus, receiving the UART NRZ (Non-Return to Zero) code from the Microcontroller, encoding the data into PWM code, the transceiver sends the encoding data onto the bus.

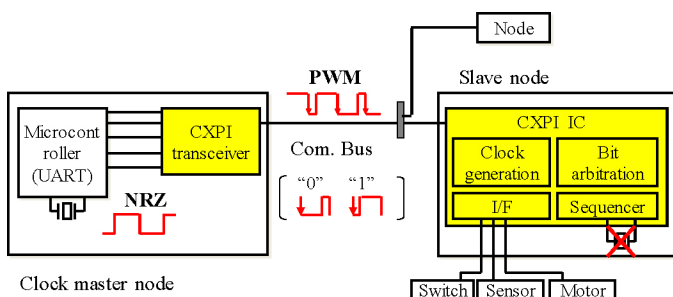


Figure 1. Example configuration for CXPI communication system.

Bus Access Method

The communication bus PWM waveform can take one of two logical values, '0' or '1'. Figure 2 shows how the PWM waveform is generated.

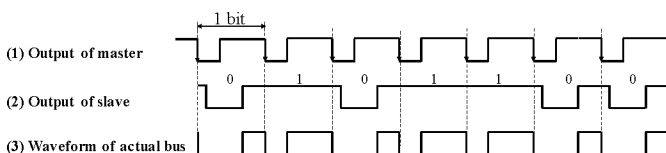


Figure 2. PWM generation method.

In wakeup mode, the clock master node always transmits a clock pulse as "0" or "1" data to the communication bus. Each node can synchronize for every bit by synchronizing with the falling edge of (1).

When a certain node transmits data "0," the node outputs a low potential to the communication bus only for a defined time, after it detects a falling edge. When transmitting data "1," it outputs nothing after it detects a falling edge.

This produces waveform (3) in the communication bus, allowing it to use a UART to transmit the data.

Arbitration Resolution

As stated earlier, CXPI is a multi-master system implemented with the CSMA/CR method, supposing the case that collisions between transmitted signals arise. In such cases, arbitration between nodes is performed by the way that transmission by the node with the higher priority (transmitting a logical '0') shall be continued and transmission by the lower priority node (transmitting a logical '1') shall be suspended. Figure 3 illustrates how arbitration is resolved.

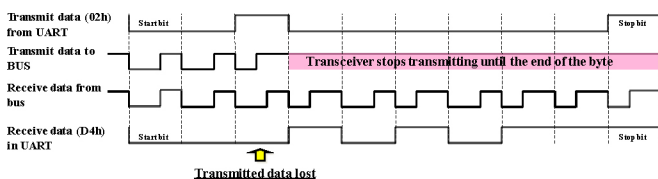


Figure 3. Arbitration method.

All nodes are synchronized with the clock signal provided by the clock master node. When a collision between logical '0' and logical '1' data occurs on the communication bus, as the dominant signal is received with priority. Logical '0' shall be received with higher priority than logical '1' on the communication bus. The transmitting node shall make a bit-by-bit comparison of the data it transmits and receives. If they match, it wins the arbitration and continues transmitting. Conversely, if they do not match, it loses the arbitration and stops transmitting from the next bit. In the figure, there is a mismatch between transmitted and received data at the second data bit, so transmission stops from the third bit.

Transmission Method

CXPI provides two methods of transmission. According to system requirements, either one method shall be selected by an implementer.

- Method 1: event-triggered method
- Method 2: polling method

Event-Triggered Method

A description of the sequence of operation for event-triggered method follows. When an ECU detects that the bus is idle (no ECUs currently transmitting data), it can transmit a frame (PID field).

Figure 4 provides an example of the event-triggered transmission sequence.

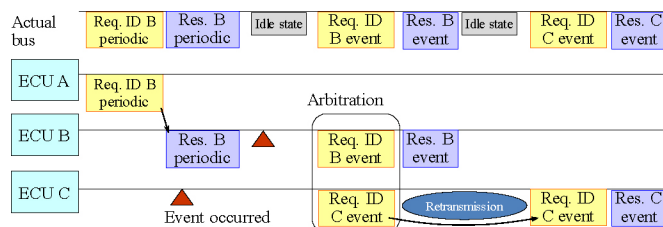


Figure 4. Example of event-triggered transmission.

ECU_A detects that the communication bus is idle and transmits request ID_B. ECU_B, its response is associated with request ID_B, transmits response B. Then, the internal event requesting transmission occurs in ECU_B and ECU_C at the same time. Immediately after the bus becomes idle next time, request ID_B from ECU_B and request ID_C from ECU_C are transmitted at the same time, resulting in a collision on the communication bus. After arbitration, request ID_B with higher priority from ECU_B is transmitted. Responding request ID_B, and ECU_B transmits a response associated with request_B. ECU_C retransmits request ID_C when the bus becomes idle again.

Figure 5 shows the normal frame for event-triggered method.

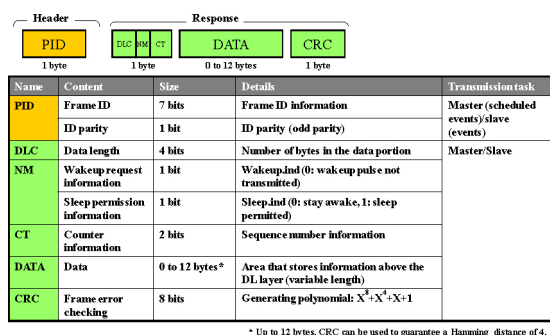


Figure 5. Normal frame in event-triggered method.

A normal frame includes a PID (Protected ID) field and a response field. The PID field is composed of a 7 bit frame ID and 1 bit parity. The response field consists of a 4 bit DLC (Data Length Code), a 2 bit NM (Network Management), a 2 bit counter (CT), up to 12 bytes of DATA, and an 8 bit CRC (Cyclic Redundancy Check). The DLC indicates the number of bytes of DATA in the frame. NM includes two bits of indication flags - wakeup.ind and sleep.ind. The wakeup.ind indicates the first wakeup node by setting '1'. The sleep.ind indicates whether the nodes can go to sleep. CT holds sequential number data representing frame continuity. The DATA fields stores data in accordance with the DLC. CRC is used for frame error detection.

Figure 6 shows the long frame for event-triggered transmission. The long frame is a format designed to increase transfer efficiency by allowing up to 255 bytes of DATA per frame.

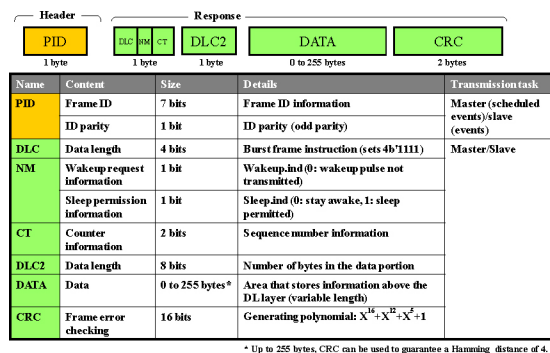


Figure 6. Long frame in event-triggered method.

It differs from the normal frame in the use of DLC2 (1 byte) to specify data length, and the extension of the CRC field to 16 bits. A DLC (4 bits) with a value of 1111b is handled as a long frame.

Polling Method

Next, a description of the operation sequence for polling method is presented. In polling method, according to the predefined communication schedule, the master node issues two types of request - PID field or PTYPE(Protected TYPE) field. Responding to the request issued by the master node, the master node as well as slave node transmits the response associated with the request.

Figure 7 provides an example of the polling method sequence.

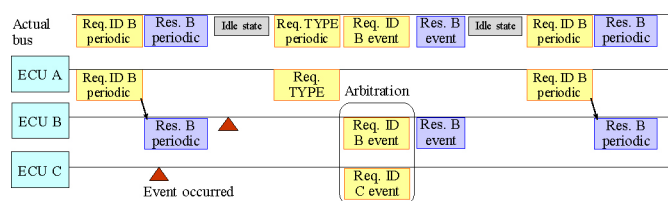


Figure 7. Normal frame in polling method.

After detecting that the communication bus is idle, master ECU_A transmits request ID_B in accordance with the communication schedule, and the corresponding slave ECU_B transmits response B. Before the master node issues next request, the internal event occurs in the slave ECU_B and slave ECU_C triggering request ID_B and request ID_C respectively. Next, a PTYPE field is transmitted by master ECU_A, and the slave ECU_B and slave ECU_C simultaneously transmits the request ID_B and request ID_C respectively on the communication bus. The resulting collision is arbitrated, and the request ID_B (Event) from slave ECU_B is transmitted. A response B(Event) is then transmitted from the corresponding slave ECU_B. Next, master transmits request ID_B (Periodic) requesting slave ECU_B response, and slave ECU_B transmits a response B (Periodic). ECU_C, queuing the event, can not respond until it receives request ID_C or PTYPE fields.

The normal and long frames for polling method are shown in Figures 8 and 9, respectively.

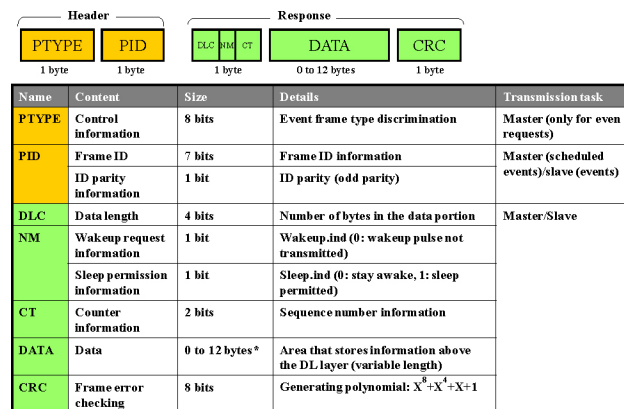


Figure 8. Normal frame in polling method.

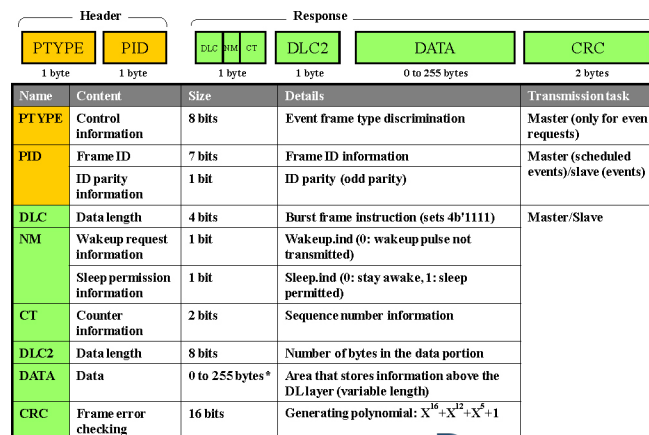


Figure 9. Long frame in polling method.