BBridge: Transport Layer Protocol

PEG-GX

Porsche Engineering
driving technologies

Version: 1.3
Status: Released

Version History

Date	Version	Change	Author / Role
29.03.2016	0.1	created	A. Perico / J. Dürr
30.03.2016	1.0	Review	J. Duerr / U. Schlieben
13.04.2016	1.1	Updated states and transition values in main state machine	A. Perico
21.04.2016	1.2	Updated SPI Frame Format and Examples	J. Dürr / A. Perico
11.05.2016	1.3	Corrected FCS calculation in section 2.1.1	A. Perico

Open points in this document

Open Point	Section	Tbd by

Review

Role	Name	Department	Date	Version	Signature

Table of contents

1 Introduction

1.1 Purpose of this document

This document describes the transportation layer of the data communication between Panther and BBridge.

1.2 Referenced Documents

Document	Description // SVN URL
BBridge_Architecture_Interfaces_Protocol.pdf	Documentation of the BBridge Architecture, software and hardware interfaces, communication protocol and API. PES document

Table 1 – Referenced Documents

1.3 Definitions, Acronyms and Abbreviations

Definition, Acronym, Abbreviaton	Desciption
Panther	µController host system that uses the BBridge software for networking
BBridge	Application that manages the networking over Bluetooth LE
Bluetooth LE	Bluetooth Low Energy , low range wireless personal area network technology
CC2540	µController subsystem for Bluetooth LE
PES	Porsche Engineering Services GmbH
TI	Texas Instruments
SPI	Serial Peripheral Interface
SPI Ready Line	Hardware line between µC host and subsystem
ACK	Acknowledge

Table 2: Definitions, Acronyms and Abbreviations

1.4 Motivation

The transport layer is responsible for transporting data between the Panther and BBridge over SPI. First implementation of the transport layer showed hardware limitations of the CC2540 and forced the design to implement a simple protocol to ensure a stable communication path.

Also, SPI/UART topics are often part of bugfixes by TI.

Fixes claimed to be solved by TI extracted from the release notes documents (BLE-Stack(tm) Version 1.3 – 1.4):

- Fixed CC254x UART DMA reception discontinuity
- Updated SPI and UART_DMA drivers for improved robustness and throughput.
- Fix for SPI reliability improvement
- Fix for HAL_DMA_CLEAR_IRQ() can be interrupted causing missed ISR cause

2 BBridge Transport Layer Protocol

The Transport Layer Protocol wraps the data (payload) needed to be transferred with extra control bytes. This extra information, together with the payload forms a data frame that is used by both sides of the SPI communication to verify the validity of the data received.

The following sections describe the protocol rules and data frames. Figure 1 shows the data flow from the Panther application through the Communication Protocol and the Transport Layer Protocol to the BBridge and vice versa. The Communication Protocol is described in the document "BBridge_Architecture_Interfaces_Protocol.pdf".

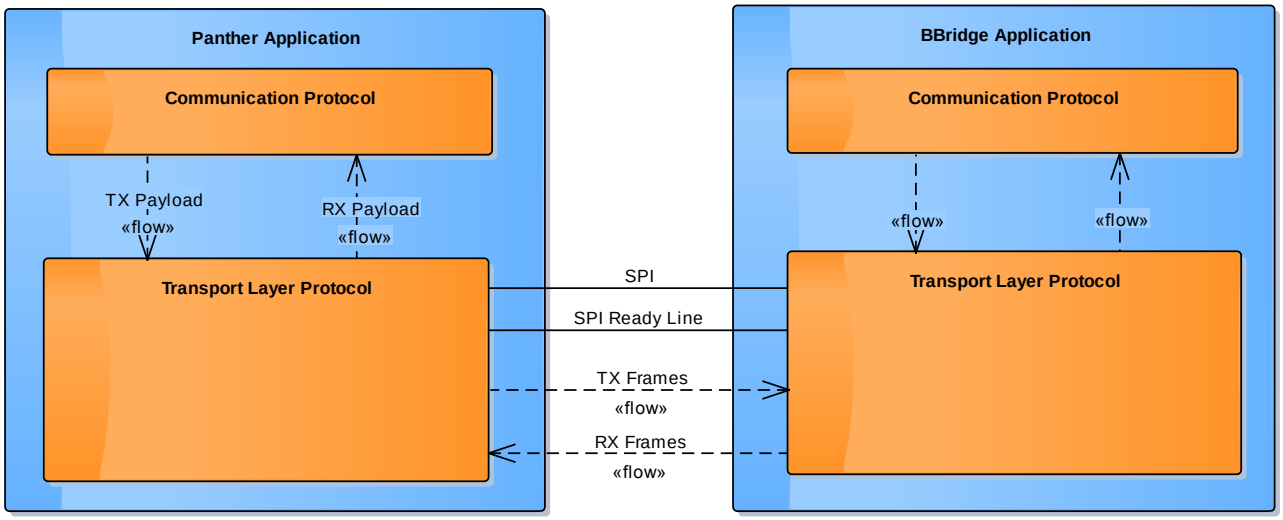

Figure 1 – Data flow between Panther Application and BBridge

2.1 SPI Frame format

The frame format is derived from the Texas Instruments Network Processor Interface included in the Texas Instruments OSAL which is used on the CC2540 device. For further information please refer to http://processors.wiki.ti.com/index.php/NPI.

SOF (1 Byte)	LEN (1 Byte)	CONTROL (1Byte)	PAYLOAD (n Bytes)	FCS (1Byte)
Start of Frame	Length	4 bits: Frame Counter; 4 bits: Frame Type	Application data to be transferred	Frame checksum

Table 2: SPI Frame format and fields

- **SOF**: 0xFE
- **LEN**: Length counting bytes of CONTROL and PAYLOAD
- **Frame Type:**

Frame Types	
0x1	ACK
0x3	DATA

Table 3: Frame Types

- **Frame Counter:** Used for synchronization
- **FCS**: Frame checksum, FCS = LEN XOR CONTROL XOR PAYLOAD

2.1.1 Frame examples

2.1.1.1 Data frame with payload = 0x55 and frame counter = 1

Data Frame: [0xFE 0x02 0x31 0x55 0x66]

Start of Frame:	0xFE
Length:	2
Packet Type:	((0x31 & 0xF0) >> 4): 3
Frame Counter:	((0x31 & 0x0F)): 1
Payload:	0x55
Frame Checksum:	0x66

2.1.1.2 ACK frame for frame counter = 1

Data Frame: [0xFE 0x01 0x11 0x10]

Start of Frame:	0xFE
Length:	1
Packet Type:	((0x11 & 0xF0) >> 4): 1
Frame Counter:	((0x11 & 0x0F)): 1
Payload:	- (none)
Frame Checksum:	0x10

2.2 Protocol Description

In general, the transport layer sends a DATA frame and waits for acknowledgement (ACK) frame, in case the ACK frame is not received within a given period of time, retransmissions of the DATA frame takes place until the DATA frame is transferred correctly and the ACK frame is received correctly.

For the scope of this section, take A and B as the communicating peers using the Transport Layer Protocol. Each peer should hold two internal counters (TX_Counter and RX_Counter) and a Timeout procedure as follows:

1. **TX_Counter** is increased every time the peer receives acknowledgement for a data frame sent. This value shall be used as the Frame counter parameter.
    i. Range: 0-15
    ii. Suggested initial value: 0x00
2. **RX_Counter** saves the frame counter of the last DATA frame received.
    i. Range: 0-15
    ii. Suggested initial value: 0x0F
3. **Timeout procedure** is used to control the time between a Data frame sent and the ACK frame received. Each peer must configure its **"Protocol Timeout"** variable to the same value, so both peers behave similarly.
    i. In case the time between a data frame sent and the ACK frame received exceeds the **"Protocol Timeout"** or the ACK frame does not come within **"Protocol timeout"** time, the data frame is to be retransmitted with the same Frame Counter value (TX_Counter).
    ii. The Timeout procedure is always reset right after a DATA frame has been sent.
4. Receiving a DATA frame while waiting for an ACK frame:
    i. If the Frame Counter received has already been answered with an ACK frame, retransmit ACK frame for the corresponding DATA frame.
    ii. Otherwise ignore data and keep waiting for the expected ACK
5. Receiving an ACK frame with Frame counter different than TX_Counter, ignore this frame.

2.2.1.1 Protocol State Machine

Figure 3 shows the protocol behavior as a state machine.

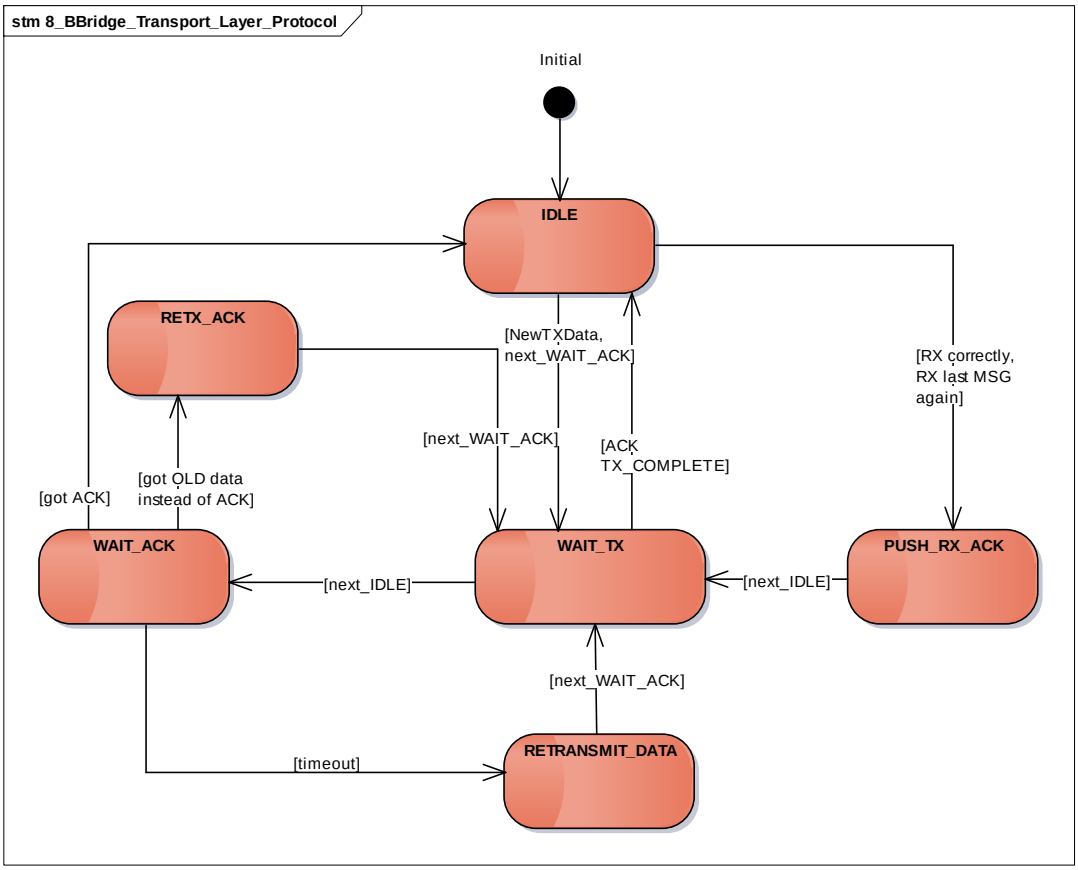

Figure 3: State machine abstraction for the Transport Layer Protocol

2.2.1.2 State transition examples:

Example of information flow when A sends DATA to B successfully:

A TX to B:
(init) -> IDLE -> WAIT_TX -> WAIT_ACK -> (received ACK within timeout window) -> IDLE

B RX from A:
(init)->IDLE-> PUSH_RX_ACK->WAIT_TX->IDLE

2.3 Sequence diagram

Figure 4 shows the sequences of a successful transmission of data from A (Amy) to B (Bob), as well as a corrupted transmission from Bob to Amy followed by a timeout and a retransmission of the data frame.

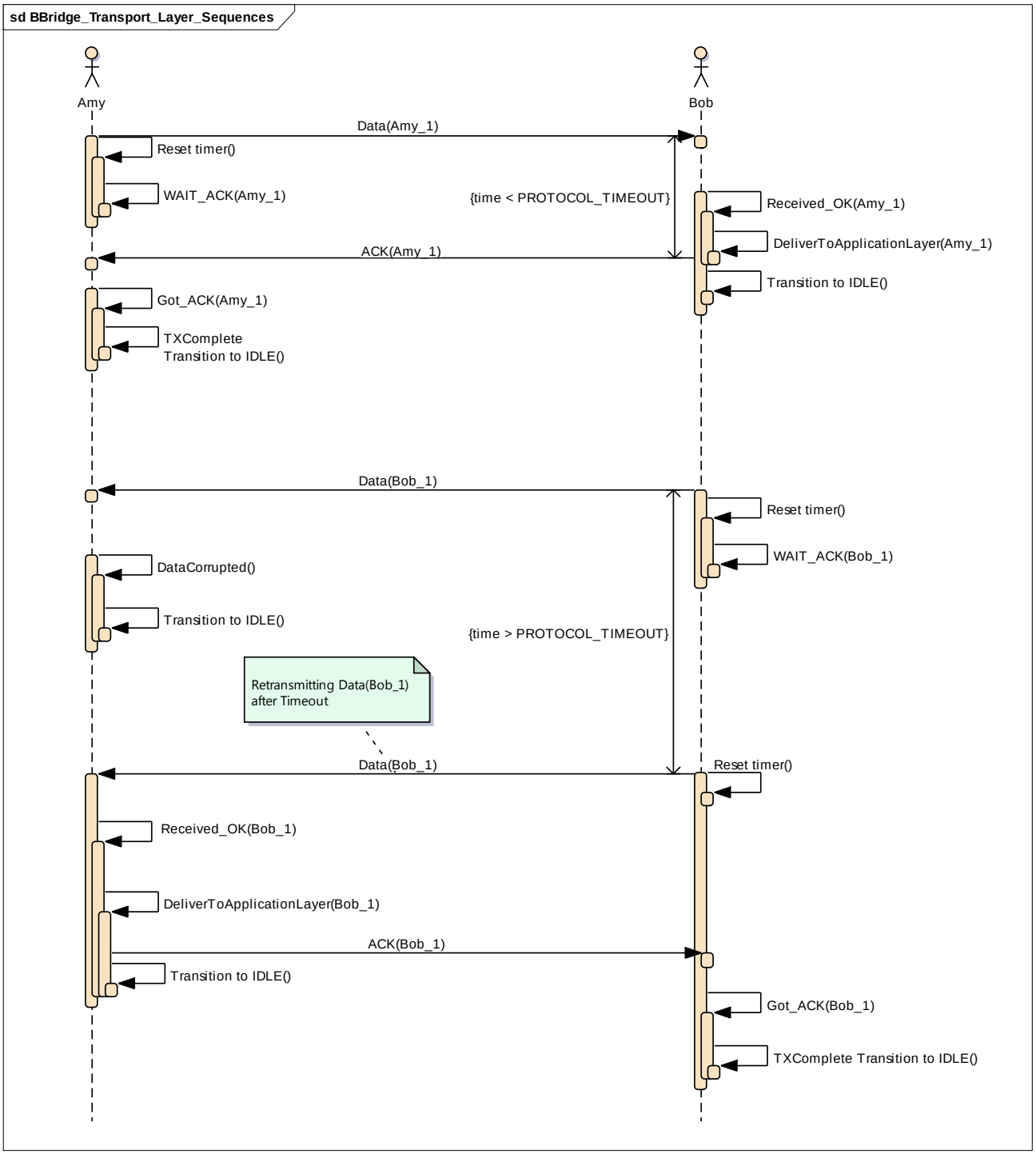

Figure 4: Sequence for successful transmissions and retransmissions after timeout