Telink Enhanced ShockBurst Engine User Guide

AN-18103100-E2

Ver.2.0.0

2021/11/30

Keyword

Enhanced ShockBurst

Brief

This document presents the guide on the usage of Telink Enhanced ShockBurst Engine, including timing sequence, transmission diagram, and packet format.

Published by

Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,

Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

Information

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinksales@telink-semi.com

telinksupport@telink-semi.com

Revision History

Version	Change Description	Date	Author
V1.0.0	Initial release.	2018/11	ZFP, Cynthia
V2.0.0	Added Chapter 3 Transmission Diagram Updated Chapter 4 Packet Format Deleted ESB APIs description chapter	2021/11	mingqian.ouyang

Contents

1. Overview

This document presents the guide on the usage of Telink Enhanced ShockBurst Engine, including timing sequence, transmission diagram, and packet format.

2. Timing Sequence

2.1 PRX Timing Sequence

Suppose all RF-related configurations have been completed before PRX RX mode, and RF is triggered to implement reception by writing "0x84" into the register "0xf00".

First, the state machine enters the RX Settle phase, so as to wait for the PLL in the RF RX Settle phase to become stabilized (need at least 85us). The duration (unit: us) of "RX Settle" is configured via the register 0xf04<7:0> and 0xf05<3:0>. After the end of the RX Settle phase, the state machine enters the Actual RX phase.

The Actual RX phase is the actual phase of packet reception, and its duration is configurable.

- ✧ When the register 0xf03<2> is set as 1b'1, the duration of the Actual RX phase is configurable as the RX Time minus the duration of the RX Settle phase. The RX Time (unit: us) is configured via the register 0xf0a and 0xf0b.
- ✧ When the register 0xf03<2> is cleared, the state machine will stay in the Actual RX phase, until a packet is received to enter the next phase.

When a packet is received in the RX phase, the state machine will judge whether this packet is a repeat packet.

- ✧ If it's a repeat packet, it won't be stored in related register.
- ✧ If it's not a repeat packet, the RF hardware will analyze to get the "NO_ACK" bit in the Packet Control field of the packet header, and implement the XOR operation for this bit and the local 0xf15<5> bit.

If the XOR operation result is 1, it will consider the peer does not need ACK, so the state machine will return to the RX Settle phase waiting to receive the next packet.

If the XOR operation result is 0, it will consider the peer does need ACK, so the state machine will send an ACK packet to the peer.

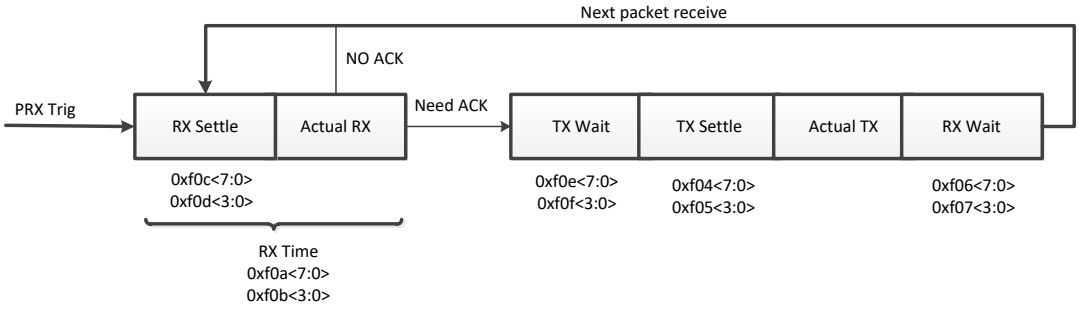
When the state machine needs to reply with an ACK, first it waits for the duration of "TX wait", which is configured via the register 0xf0e<7:0> and 0xf0f<3:0> (unit: us). Then the state machine enters the TX Settle phase, so as to wait for the PLL in the TX phase to become stabilized (need at least 112.5us). The duration of the TX Settle phase is configured via the register 0xf04<7:0> and 0xf05<3:0>.

After the end of the TX Settle phase, the state machine enters the Actual TX phase to send an ACK. The duration of the Actual TX phase is determined by the length of the ACK packet.

After the ACK packet is sent, the state machine waits for the duration of the "RX wait", which is configured via the register 0xf06<7:0> and 0xf07<3:0> (unit: us). Then it returns to the RX Settle phase so as to receive the next packet.

The figure below shows the PRX timing flow of the state machine.

Figure 2-1 PRX timing sequence



2.2 PTX Timing Sequence

Suppose before PTX TX mode all configurations have been completed and TX buffer is filled, and then RF is triggered to implement transmission by writing "0x83" into the register "0xf00".

First the state machine waits for the duration of "TX Settle", which is configured via the register 0xf04<7:0> and 0xf05<3:0>, so as to wait for the PLL in the RF TX Settle phase to become stabilized (need at least 112.5us).

Then the state machine enters the Actual TX phase, so that the RF will send packets over the air. The duration of packet transmission is determined by packet length.

After the end of packet transmission, first the RF state machine will calculate the XOR result of the register 0xf15<5> and 0x404<6>, and then judge whether it needs to wait for the ACK from the peer accordingly.

- ✧ When the XOR result is 1, the state machine will consider it does not need to wait for the ACK from the peer, so that it will end current packet transmission and wait for the next transmission to be triggered.
- ✧ When the XOR result is 0, the state machine will consider it does need to wait for the ACK from the peer. The state machine will wait for the duration of "RX Wait" configured via the register 0xf06<7:0> and 0xf07<3:0> (unit: us), and then switch to the RX mode, so that it can wait for the PRX to switch from RX to TX and respond with ACK.

The PTX will stay in the RX mode for the duration of "RX Time" configured via the register 0xf0a<7:0> and 0xf0b<3:0> (unit: us), waiting for the peer to respond with ACK.

Note: During the duration of "RX Settle" in the RX mode (configured via 0xf0c<7:0> and 0xf0d<11:8>, unit: us), the state machine stays in the RX Settle state, so as to wait for the RF PLL to become stabilized (need at lease 85us). During the RX Settle phase, since the RF RX function is not enabled actually, any coming packet cannot be received, i.e. the valid reception period during the RX Time is the Actual RX phase.

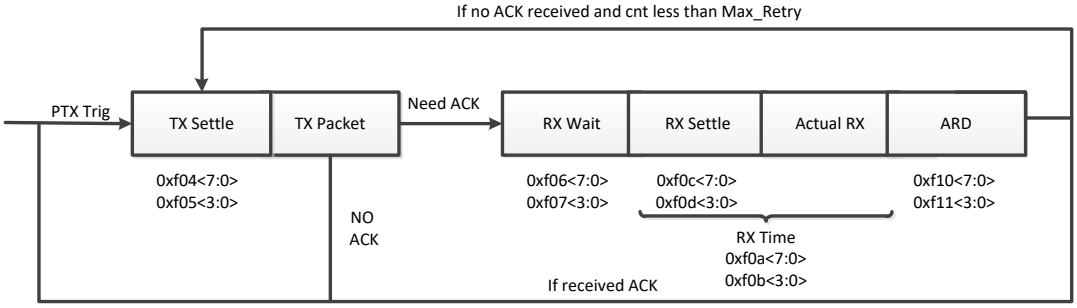
If an ACK packet is received in the Actual RX phase, current packet transmission is finished.

If no ACK is received from the peer when the RX Time expires, the PTX will attempt retransmission. First it will judge whether the retry count number reaches the maximum value: if not reached yet, it will implement "retry cnt++". Then it will wait for the duration of "ARD" configured via 0xf10<7:0> and 0xf11<3:0> (Auto Retransmit Delay, unit: us).

Then the state machine waits for the duration of "TX Settle", enters the TX mode, sends packet, waits for the duration of "RX Wait", enters RX, and then waits for ACK. The state machine repeats the flow until the retry count number reaches the maximum value.

The figure below shows the PTX timing flow of the state machine.

Figure 2-2 PTX timing sequence



2.3 Interrupt Description

For the PTX side, the TX interrupt will be triggered every time when it transmits a packet. User can check the flag bit to judge whether the packet is sent successfully.

If the PTX enables ACK, the following two cases apply when the ACK from the PRX is received:

- ✧ If the PRX responds with an empty ACK, the PTX will only trigger the TX_DS interrupt.
- ✧ If the PRX responds with an ACK with Payload, the PTX will generate the RX_DR and TX_DS interrupt.

If the PTX fails to receive any ACK within a specified duration, it will attempt to re-transmit the same packet, until an ACK is received or the retry count number reaches the preset threshold. If the PTX does not receive any ACK after the retry count number reaches the threshold, it will generate the RETRY_HIT interrupt.

For the PRX side, the RX interrupt will be generated every time when it receives a packet, but RX_DR interrupt can be generated only when the received packet has a correct CRC result and Payload_Len is unequal to zero. If it's the mode of ACK with Payload, the TX_DS interrupt will be generated starting from the reception of the second packet.

If PRX enables ACK, it generates TX and RX interrupts but does no generate RX_DR interrupt in a transmission. The reasons may be as follows:

- ✧ It receives a packet which has a error CRC result.
- ✧ It receives a empty packet that payload len is equal to zero.
- ✧ It receives a packet with the same pid as the previous packet.

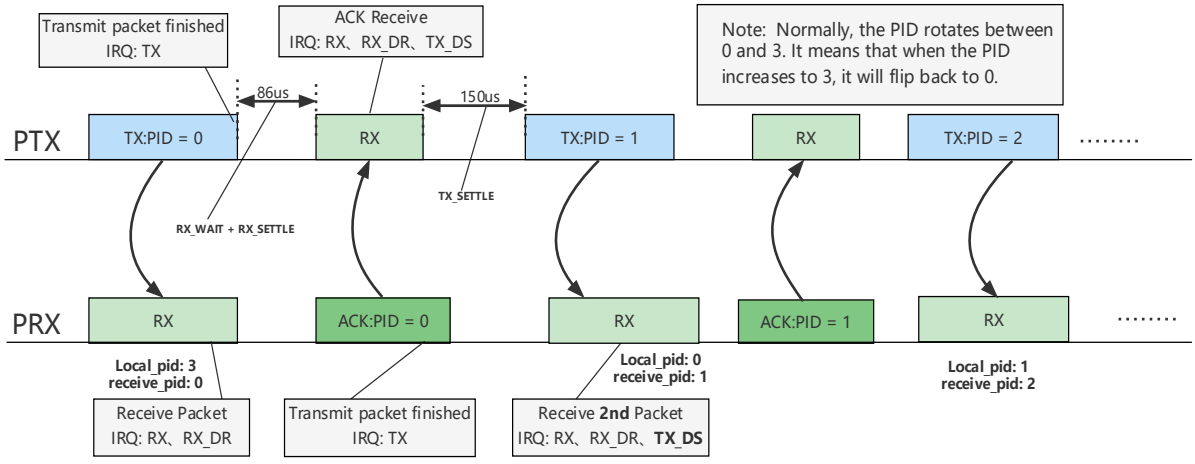
Notes:

- *Both PTX and PRX will generate INVALID_PID interrupt when the received packet which pid is neither local_pid nor local_pid + 1. PTX\PRX will generate CRC_2 interrupt when it receives two packets with wrong CRC result continuously.*
- *The PRX can be configured to receive packets continuously or only within a specified duration. If the PRX is configured to receive packets within the specific duration, and it fails to receive any packet within this duration, it will generate the RX_TIMEOUT interrupt.*

3. Transmission Diagram

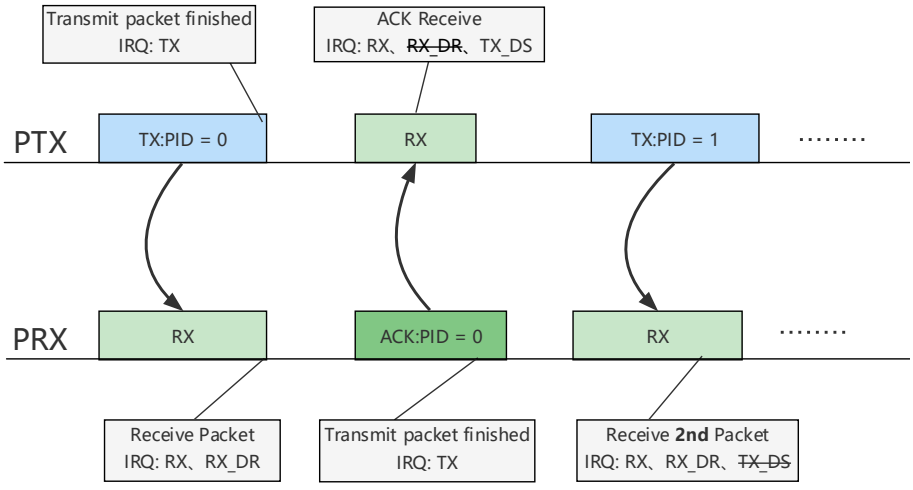
3.1 Normal communication process for ACK with payload

Figure 3-1 Normal communication process for ACK with payload



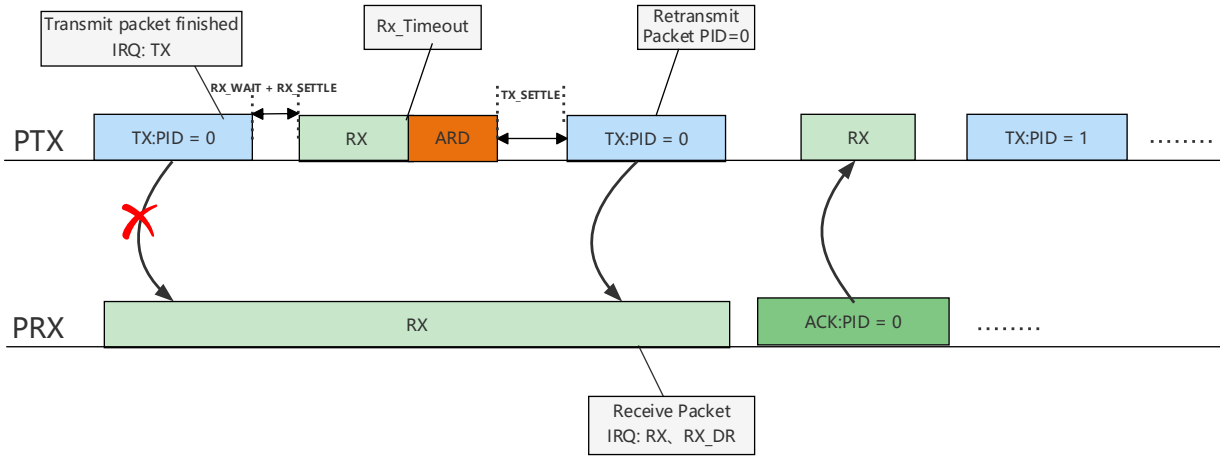
3.2 Normal communication process for empty ACK

Figure 3-2 Normal communication process for empty ACK



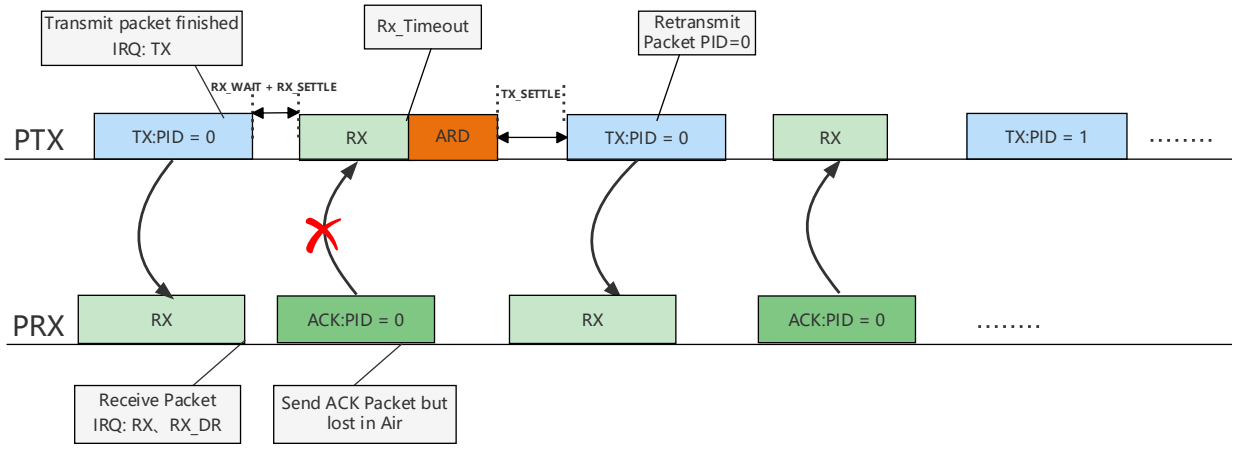
3.3 Single communication process with a lost packet

Figure 3-3 Single communicatioin process with a lost packet



3.4 Single communication process with a lost ACK packet

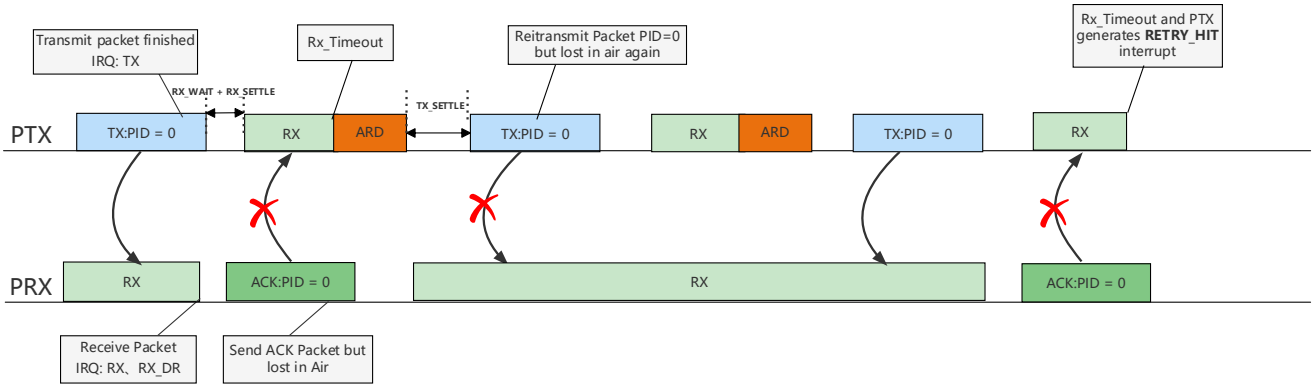
Figure 3-4 Single communication process with a lost ACK packet



3.5 Multiple communication process with max retransmission count

In this scene, preset ARC is 2. If PTX does not receive any ACK after the retry count number reaches the threshold (ARC=2), it will generate the **RETRY_HIT** interrupt.

Figure 3-5 Multiple communication process with max retransimission count

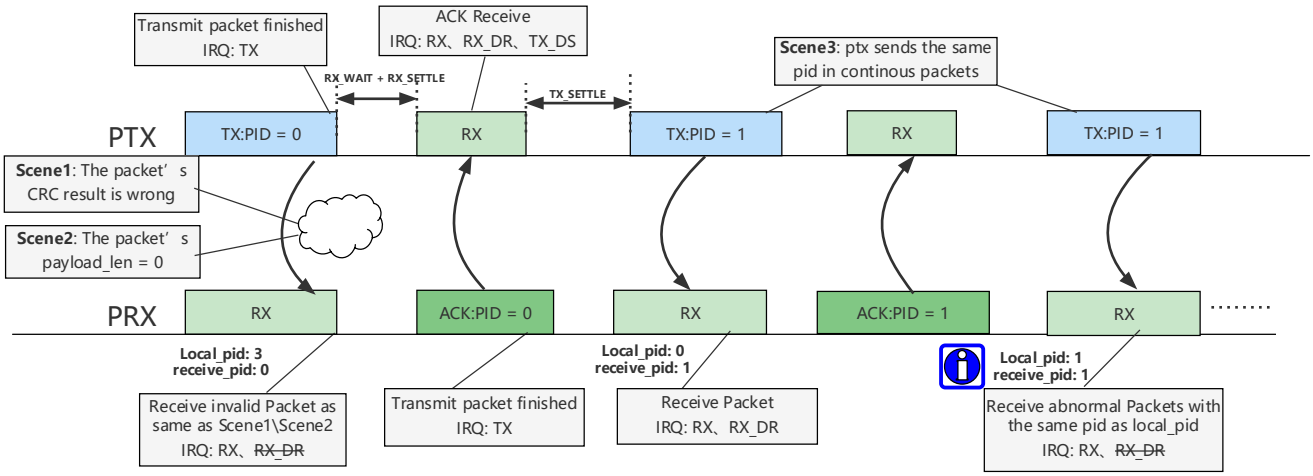


3.6 A Classic communication process

During communication, the transceiver as will have a situation that generates TX and RX interrupts, but does not generate RX_DR interrupts. The reasons for this appearance is non-uniqueness and it is caused by following issues mainly :

- ✧ Receives a packet with a wrong CRC result.
- ✧ Receives a packet without payload(payload_len = 0).
- ✧ Receives packet's pid as same as local pid

Figure 3-6 A classic communication process



Notes: Normally, Telink PTX will send continuous packet with the same pid only when a ack packet is lost.

4. Packet Format

4.1 Packet format for PTX TX buffer (payload len=8)

Example:

```
00000000: 09 00 00 00 08 01 02 03 04 05 06 07 08 00 00 00
00000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Description for packet data:

- tx_packet[0] : tx packet length (payload length +1)
- tx_packet[1] : 0x00
- tx_packet[2] : 0x00
- tx_packet[3] : 0x00
- tx_packet[4] : payload length
- tx_packet[5] : data[0]
- ...
- tx_packet[5 + length -1] : data[length -1]

4.2 Packet format for PRX RX buffer (payload len=8)

Example:

```
00000000: 13 00 00 00 08 01 02 03 04 05 06 07 08 81 66 12
00000010: cb 7d 14 18 11 58 10 10 cc cf f3 0c 03 c0 f3 30
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Description for packet data:

- rx_packet[0]: 0x13 (total number of bytes received and transferred to rx dma)

- rx_packet[1]: 0x00
- rx_packet[2]: 0x00
- rx_packet[3]: 0x00
- rx_packet[4]: bit[5:0] payload length, bit[7:6] pid
- rx_packet[5]: data[0]
- rx_packet[6]: data[1]
- rx_packet[7]: data[2]
- ...
- rx_packet[5 + length -1] : data[length -1]
- rx_packet[5 + length] : CRC(0)
- rx_packet[5 + length + 1] : CRC(1)
- rx_packet[5 + length + 2] : TimeStamp_Byte[0]
- rx_packet[5 + length + 3] : TimeStamp_Byte[1]
- rx_packet[5 + length + 4] : TimeStamp_Byte[2]
- rx_packet[5 + length + 5] : TimeStamp_Byte[4]
- rx_packet[5 + length + 8] : PACKET RSSI