Buetooth Module - UART Buffer Application Note

National Semiconductor
November 2006
Revision 1.0

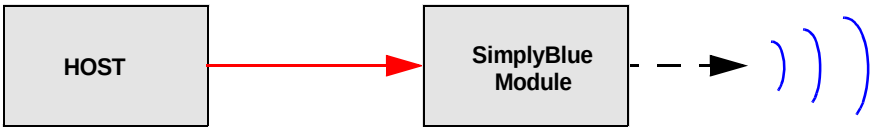
Introduction

This document explains the UART Buffer management of the SimplyBlue module in the different possible cases as receiving, transmitting, command mode and transparent mode.

1.0 Transmitting data through SimplyBlue (LMX UART Receiving buffers)

1.1 RTS HANDLING

The receiving buffers are in place to hold the data coming from the host. In this configuration the host sends to the module, the data to be sent on the Bluetooth level.

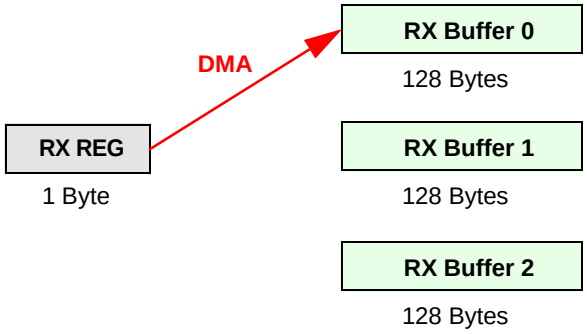


UART Receive

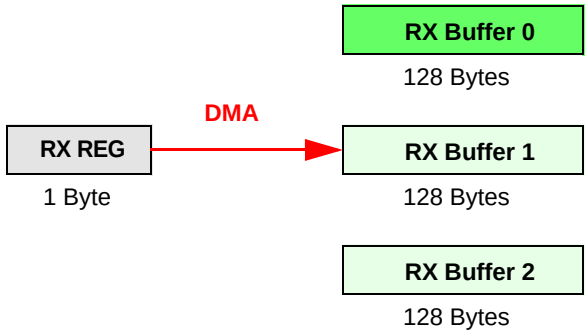
The module uses 3 buffers of 128 Bytes each, called RX Buffers, where the receiving data are copied to through DMA, and 2 buffers of 340 Bytes each, called RX Application Buffers, for holding the received commands and data.

The internal baseband of the module includes a UART RX register to receive the UART data. As the UART RX register can hold only 1 byte, a DMA channel is configured to copy the content from the UART RX register to the RX Buffers. Using DMA gives more resources to the CPU for other tasks.

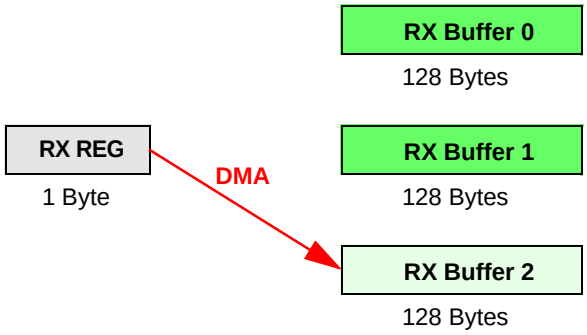
At start-up, the DMA channel is configured to point to RX Buffer 0. Once all 128 Bytes of RX Buffer 0 have been filled, a “Terminal Count Interrupt” is generated to set up the next RX Buffer to be filled. This interrupt will configure the DMA channel to point to RX Buffer 1. Once all 128 Bytes of RX Buffer 1 have been filled, the “Terminal Count Interrupt” will configure the DMA channel to point to RX Buffer 2. When the buffer RX Buffer 2 is full, the DMA will be configured back to RX Buffer 0.



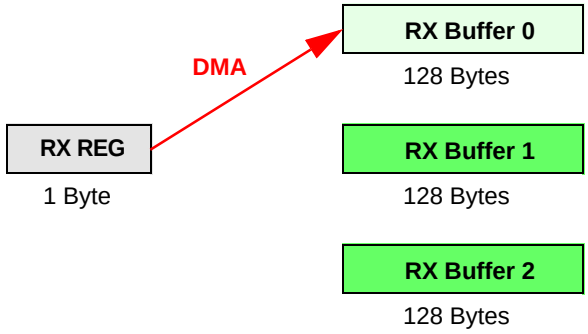
RX Buffers Empty



RX Buffer 0 Full

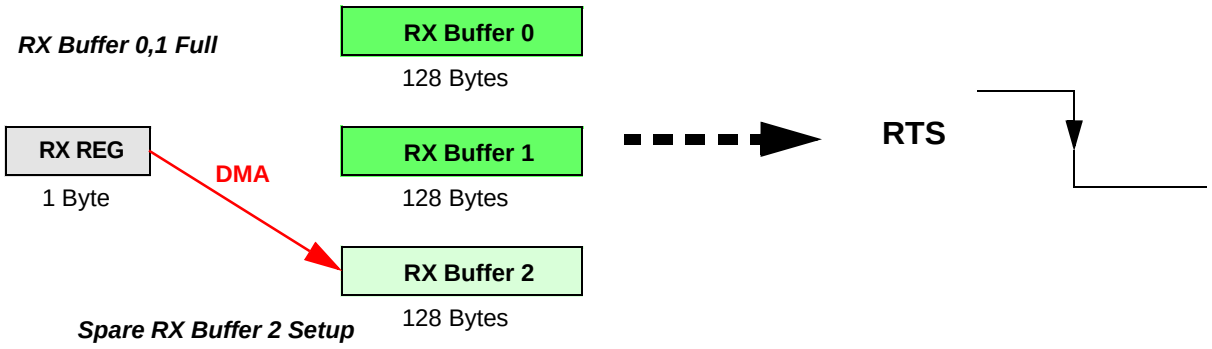


RX Buffer 1 Full



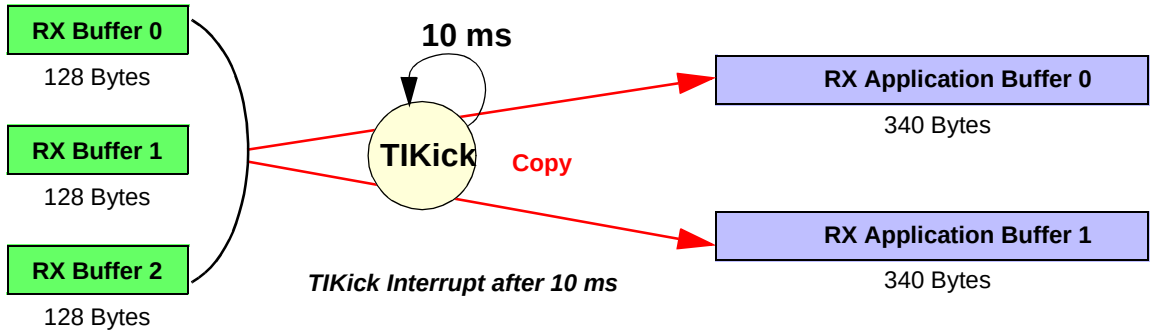
RX Buffer 2 Full

The flow control RTS is cleared when two buffers are full and a new spare buffer is setup to receive the data via DMA. This will stop the host and avoid an overrun when the last buffer is full as there would not be any buffer left available to switch to.



Note: In case RTS is ignored by the Host, an "Overrun Interrupt" is generated if the DMA has copied 128 Bytes to a buffer before a new spare buffer has been setup. Data will be lost in this case, and the module might run into a dead lock.

Every 10th ms the internal OS calls the TIKick function, managing the copy of the RX Buffers to the RX Application Buffers.



Once the data have been copied into the Application buffers, they can be treated depending on the module operation mode.

1.2 USING SEND DATA COMMAND

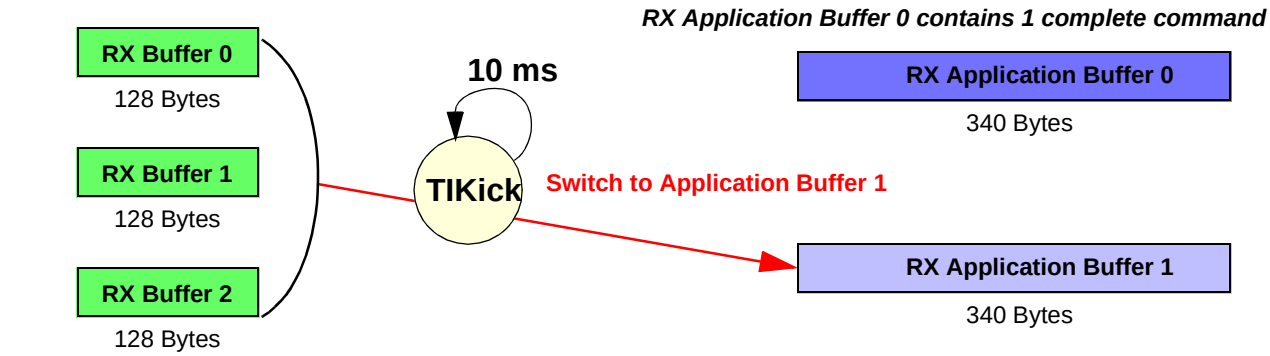
In Command mode the RX Application Buffers will get a complete command before being analyzed by the firmware.

The application parses on the way the first bytes of the command coming in to determine the command size. It then reads the complete command and stores it into one of the application buffer. There is no timeout to check whether a command is complete or not. If a command sent is not ended, the device will not be locked, but will not be able to recognize and interpret the following sent command. For example, the host wants to send 300 Bytes of data using the SPP_SEND_DATA command:

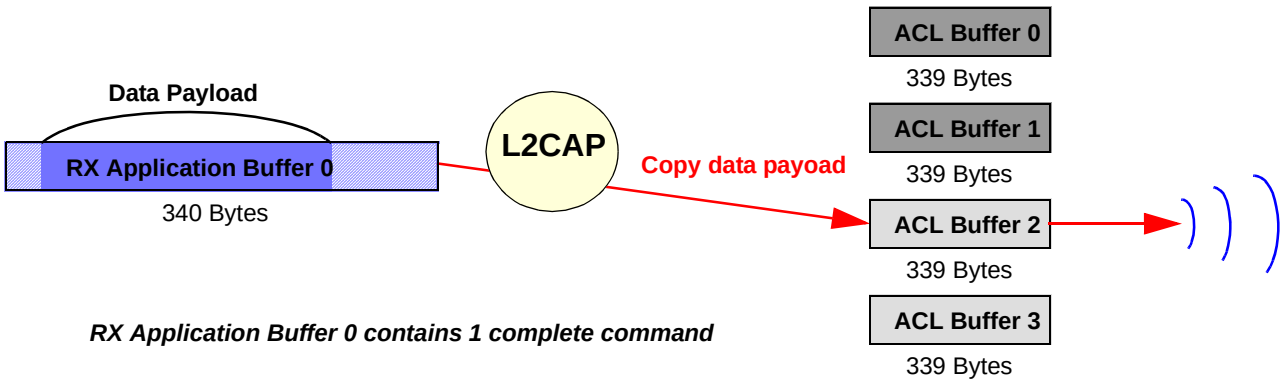
- The host first sends the command header and 250 bytes. At this point the SimplyBlue module expects another 50 bytes and the End Of Frame (0x03).
- The host sends the RESET command to put the module back to idle.
- The module expecting 50 data bytes and 0x03 will throw away both commands (SPP_SEND_DATA and RESET) as both will be unrecognized.
- At this stage the module should be able to accept new command even if the RESET has not been detected.

Once the first allocated RX Application Buffer has received one complete command, the Application buffer will be marked as Full, and the function will switch the copy and point to the second buffer. This allows the main task application to handle the packet just received while the TIKick task copy the next command to the second buffer.

Note: RX Application Buffers can hold only 1 command at a time, even if it is smaller than 340 Bytes. The application will switch to the second buffer as soon as the first buffer holds 1 complete command.



If the command requires sending data over the air, the buffer marked as Full will be passed to the Bluetooth stack for sending. The stack allocates one of the four 339 Bytes ACL buffers and copies the data payload from the command to the allocated ACL buffer.



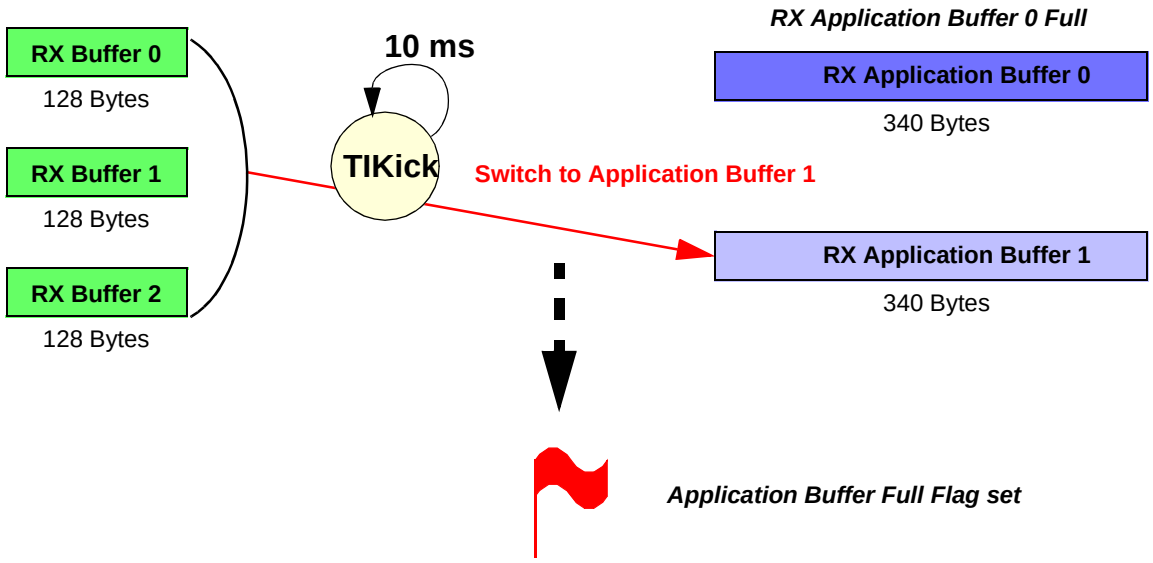
Once the L2CAP layer has finished copying the data to the ACL buffer, the Bluetooth stack will send the data over the air. The time needed to successfully send the data can not be predictable as it depends on the transmission states, the air interface, and the remote device.

The main task takes the control of the execution again, and analyses the eventual commands received, and forward the data to the Bluetooth stack depending on the event.

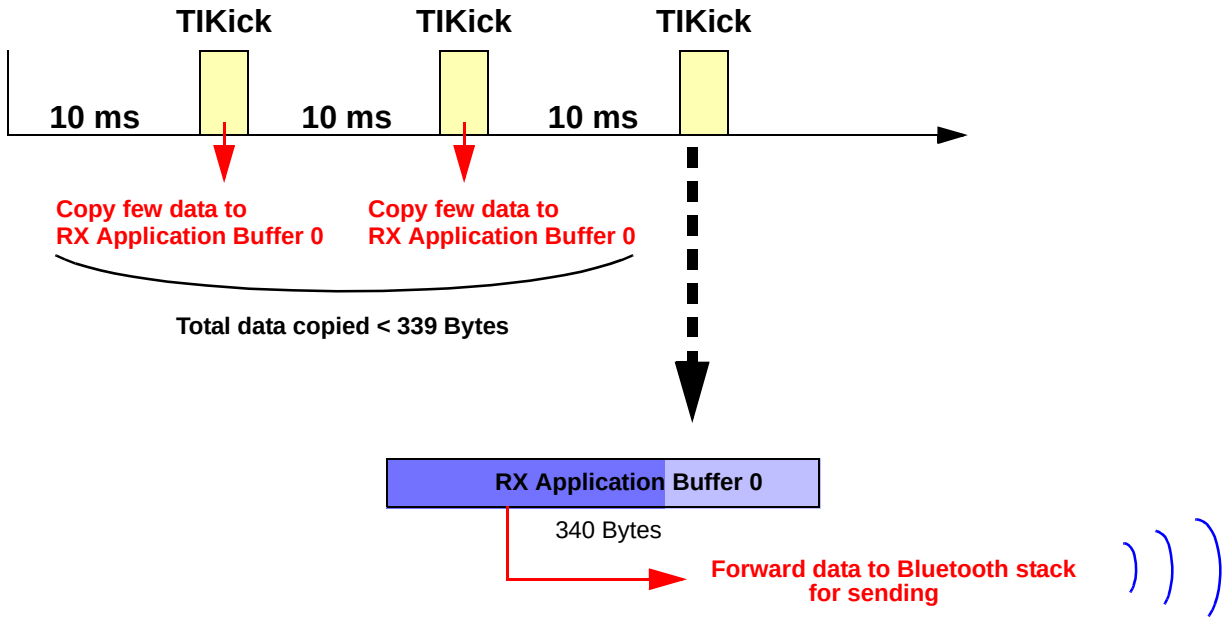
1.3 USING TRANSPARENT MODE

In Transparent mode the RX Application Buffers will hold the transparent data while it is being transmitted by the Bluetooth stack.

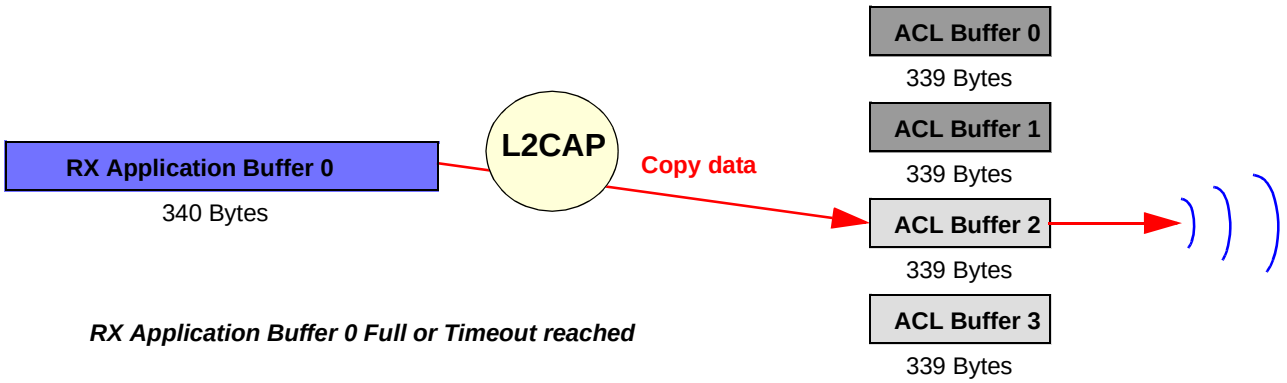
As soon as the SimplyBlue device gets 339 Bytes in one of the RX Application buffers, an *Application Buffer Full* flag will be raised to indicate the firmware that data are ready to be sent via Bluetooth. The function will as well switch the copy and point to the second buffer allowing the main task application to forward the bytes just received to the Bluetooth stack while the TIKick task copy the next data to the second buffer. In the Transparent mode case, there will be a buffer switch only when the active buffer is full.



To include the case where the number of data to be sent over Bluetooth is less than 339 Bytes, a timeout is set up at every beginning of RX Application buffer copy. If after 2 TIKick execution (this correspond to 20ms delay) the active RX Application Buffer is not full, the data will still be forwarded to the Bluetooth stack for sending on the next TIKick execution. The latency due to timeout can be up to 30ms in that case. Below is an example to illustrate this timeout case.



In both cases, RX Application Buffer is full or timeout, as for command mode, the stack allocates one of the four 339 Bytes ACL buffers and copies the transparent data to be sent to the allocated ACL buffer.



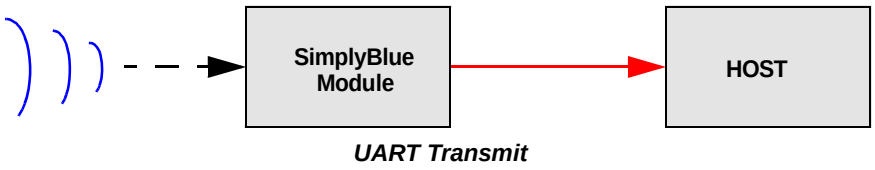
RX Application Buffer 0 Full or Timeout reached

Once the L2CAP layer has finished copying the data to the ACL buffer, the Bluetooth stack will send the data over the air. The time needed to successfully send the data can not be predictable as it depends on the transmission states, the air interface, and the remote device.

The main task takes the control of the execution again to continue an eventual data transfer.

2.0 Incoming data on SimplyBlue (LMX UART Transmitting buffers)

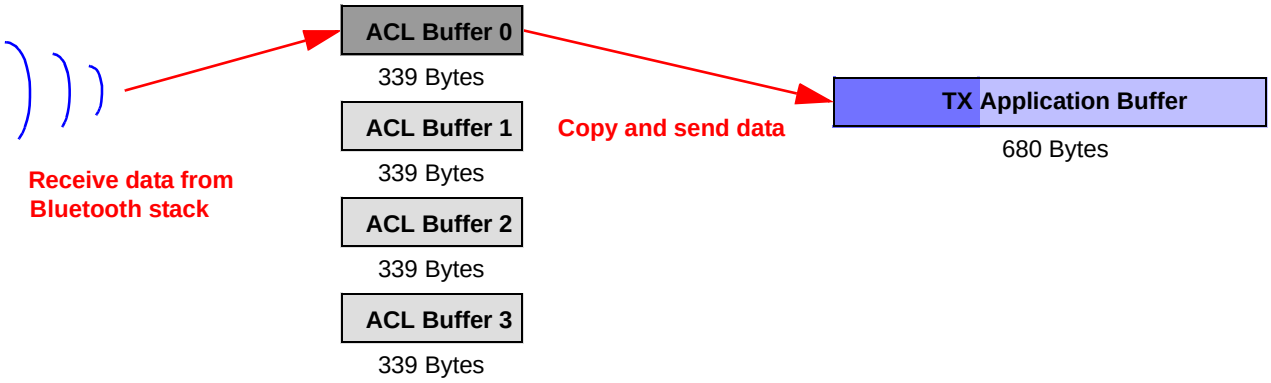
The transmitting buffers are in place to send the data received from the Bluetooth link to the host device. In this configuration the host receives the information from the Bluetooth module.



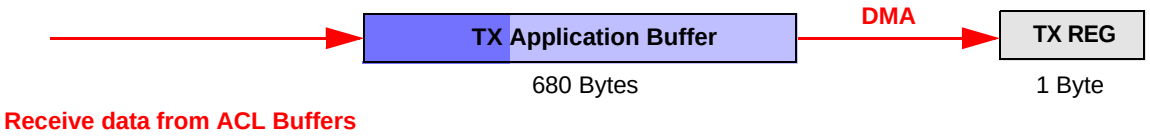
UART Transmit

The module uses 1 buffer of 680 Bytes, called TX Application Buffer, where the incoming data are copied by the main task.

Incoming air data will be received by the 4 ACL buffers then copied into the TX Application Buffer. As soon as some data are ready, it will copy and send them immediately.



As soon as the TX Application Buffer contains a data packet to be sent, it will be copied directly to the internal UART TX Register by the DMA channel. The transfer is started immediately without any delay when a packet is queued in the TX buffer and the UART Transmit is not busy.



2.1 USING COMMAND MODE

If the module operates in Command Mode the TX Application Buffer will hold complete events to send to the host device depending on the receiving frame from the Bluetooth stack.

2.2 USING TRANSPARENT MODE

If the module operates in Transparent Mode the TX Application Buffer will hold transparent data.