Socle Technology Corporation

P2S1542A Etherent Segment Delineation Software Specification

DOCUMENT REVISION HISTORY

Rev.	Date	By	Description
r0p0	2016/12/12	kenny	Preliminary version

TABLE OF CONTENT

List of Tables

List of Figures

Table 1. List of Terminologies

Symbol	Description
A-CPU	Application Subsystem CPU
APSS	Application Subsystem
AMR	Adaptive Multi-Rate
BSP	Board Support Package
FATFS	File Allocation Table File System
I2C	Inter-Integrated Circuit
IPC	Inter Processor Communication
LTE	Long-Term Evolution
M-CPU	Modem Subsystem CPU
OTP	One Time Programmable
S-CPU	Sensor Hub Subsystem CPU
SHSS	Sensor Hub Subsystem
SHUB	Sensor Hub
SPI	Serial Peripheral Interface
SDK	Software Development Kit
SOC	System On Chip

1. Introduction

This document introduces the priopriety Etherent segment delineation protocol between Mink SoC to extern wifi module via HS-UART. This document also applied to internal HS-UART for LTE to AP connection.

This document is divided into three parts:

1. Software architecture
2. Spftware specification
3. Protocol specification
4. Software design
5. Application interface
6. Demo system

2. Software Architecture

This section describes the segment delineation software architecture for Mink SoC to handle the communication between Mink and external wifi module via HS-UART. There is no detail software or ptotocol description addressed in this section.

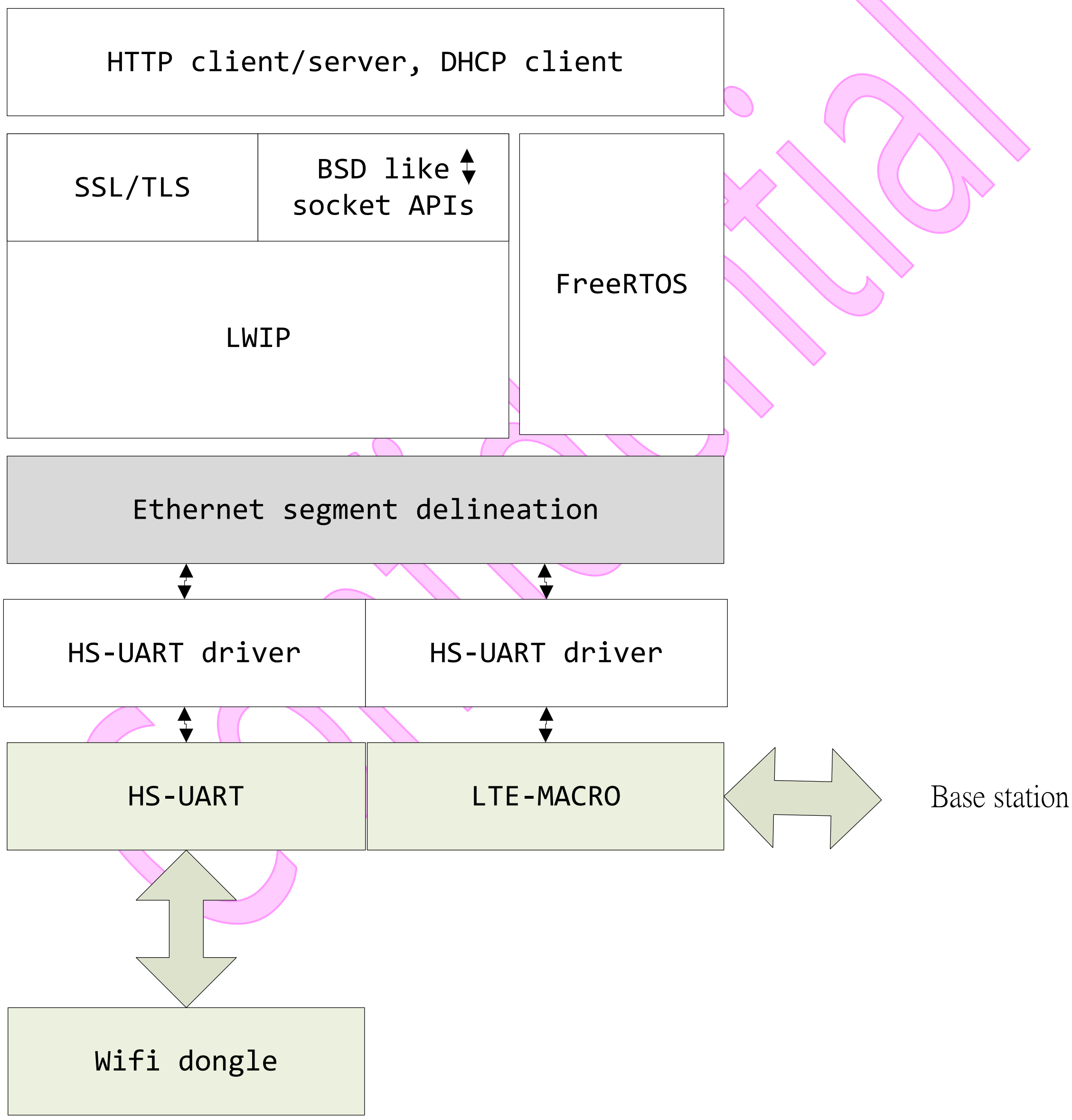


Figure 1. HS-UART Network Software Architecture

The Figure 1 shows the role of Etherent segment delineation software module. The gray block is the software module which uses physical UART device to transmittion of network packet to externl network device.

The wifi dongle also needs to implement the same Etherent segment delineation software module because not commercial module supports this propriety protocol.

3. Software Specification

This section describes the requirements of convertor software module. The HS-UART transmits bytes stream. It should do some encaptule to convey the network packet transmit through HS-UART.

Following issues should be considered before design the convertor module.

1. There should support two type of message: one for bearer the network packet, another one is propriety control message.
2. The protocol should detect bytes lost during transmission.
3. The transmission should be recoved if any error happens during transmission. The error should not affect next transmission.
4. The module allowed packet lost if any error happens.

4. Protocol Specification

This section describes the protocol beteween devices connected by HS-UART and messages for convey the network packets and control messages.

4.1 Message format

Figure 2 is the message format for convey the network packets.

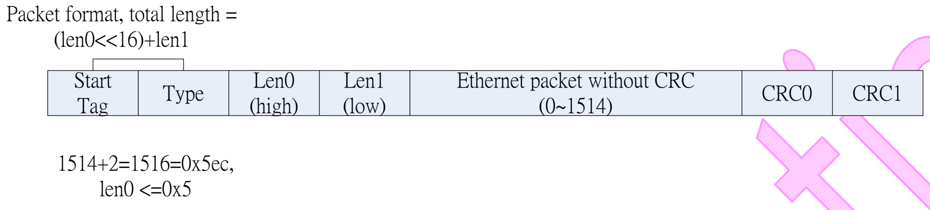


Figure 2. Message format

- Start tag: To be used to identify a start of a message. Define it “0xf0”
- Type: define the type of following message, 0: network packet, 1: control/response message
- Len0,Len1: length of packet, include start tag, type, length field, payload of message, and crc. It is short integer (2 bytes little endian). The total length is 1 (tag) + 1 (type) + 2 (length) + network payload (0 – 1514) + crc16 (2). Maximun number is 1520 (0x5f0). Len0 should not over 0x5.
- Payload: If type file is 0, the payload contains Ethernet packet, Ethernet packet comes from external device (wifi module or LTE module) without CRC.
  If type filed is 1, the payload contains control message to receive side, the receive side should have a handler to handle the control message. After process and doing the control, it may have response message need send back to the control message issuer. The type 0x81 should be applied in response message.
- CRC: CRC 16, algorithm in appendx A. the CRC calculates area from start tage to end of payload.

4.2 Receive State diagram

The state diagram describes the data received from HS-UART. It is design for guarantee the upper layer software module can received correct data of network packets. The received processing should follow the diagram to implement its handler function. The transmission function is simplier then receives function. It should encode the transmission message as a format descrived in 4.1.

There should have a time gap to identify 2 messages. It can be used to isolate error message and recover correct state for receiving next message.

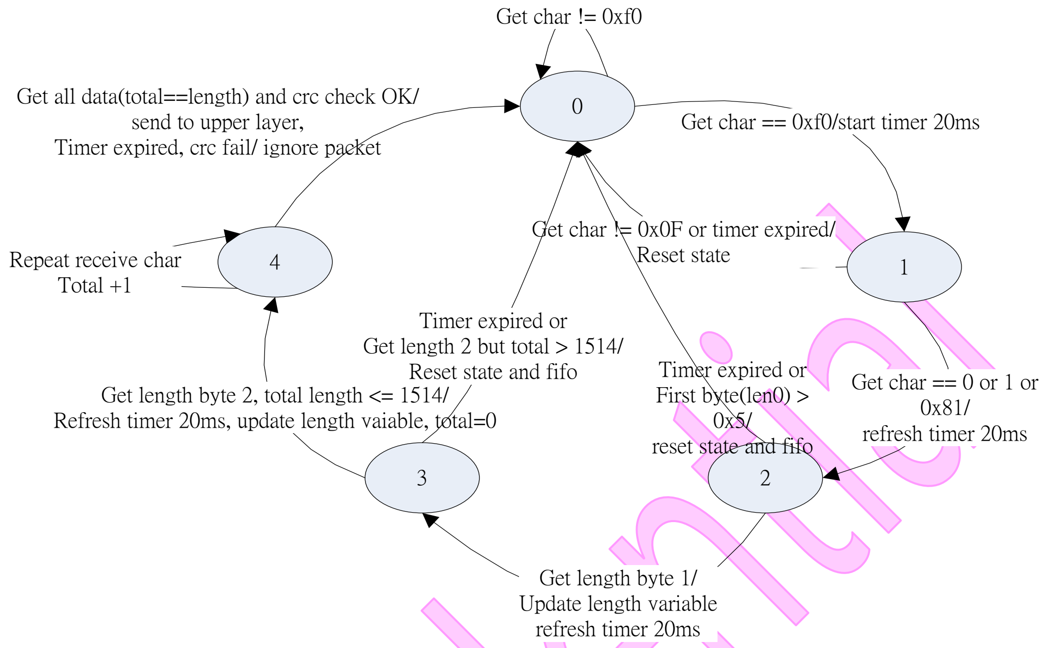
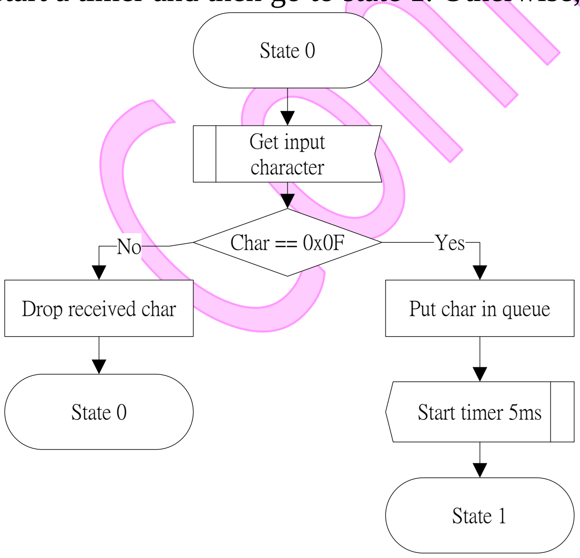


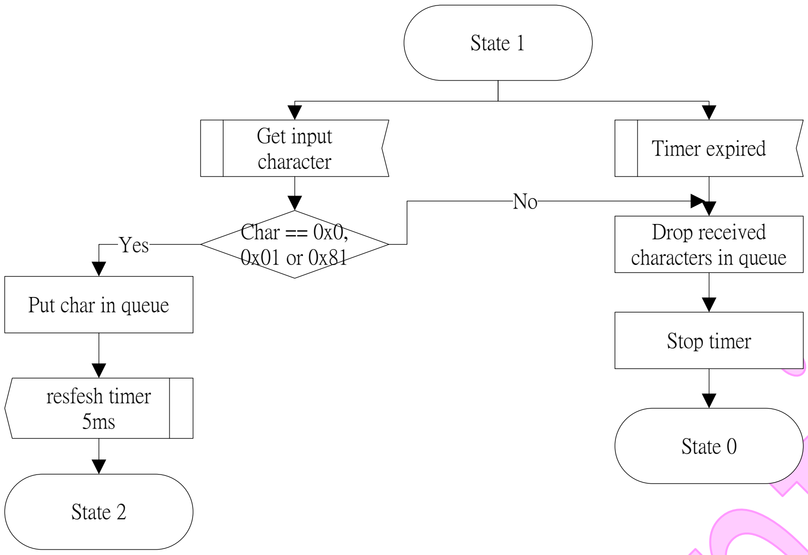
Figure 3. receive procedure state diagram

4.3 Receive Flow chart diagram

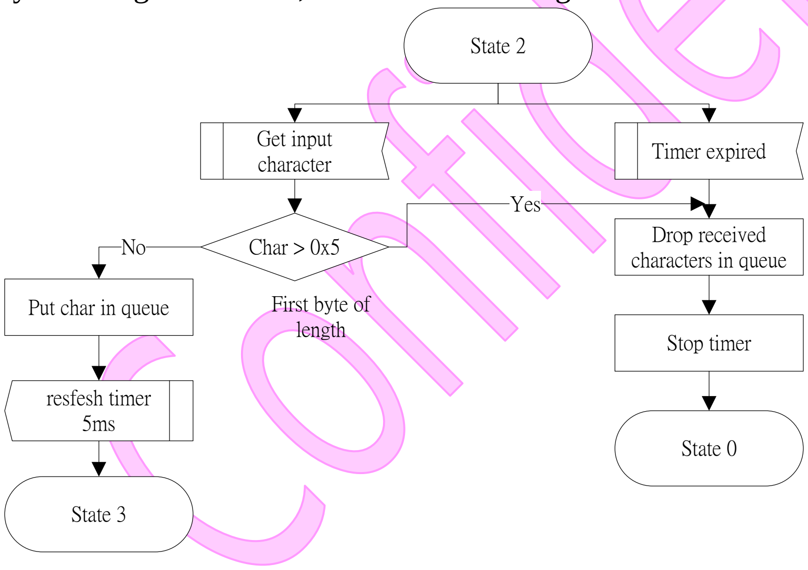
State 0: the initial state, continue waitting to get a character from driver. If get a character is start tag, start a timer and then go to state 1. Otherwise, ignore the character.



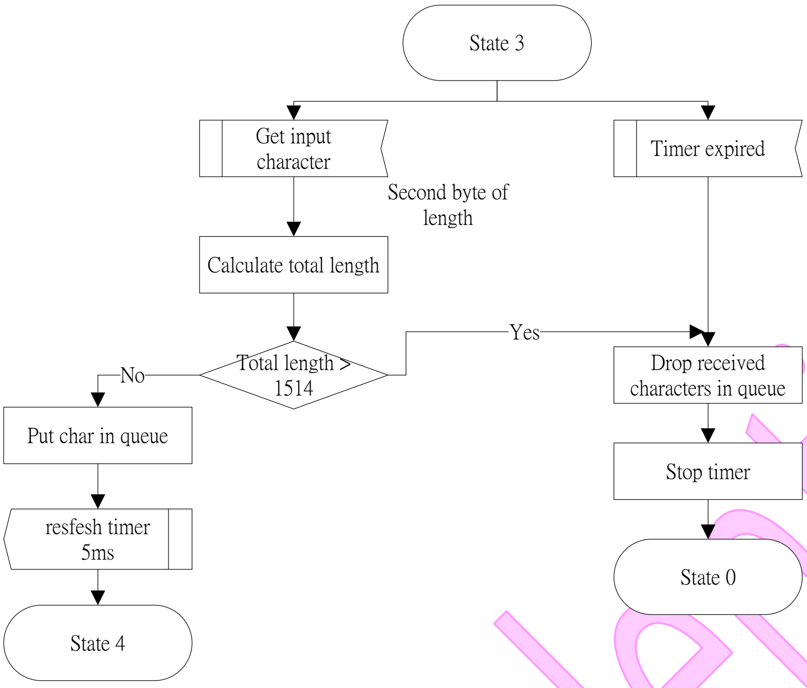
State 1: Continue waitting to receive next character. If not get character, or not get expect characters (0x01, 0x81, 0x00) and timer expired, stop timer and got to state 0. If get correct character, refresh timer and got to state 2.



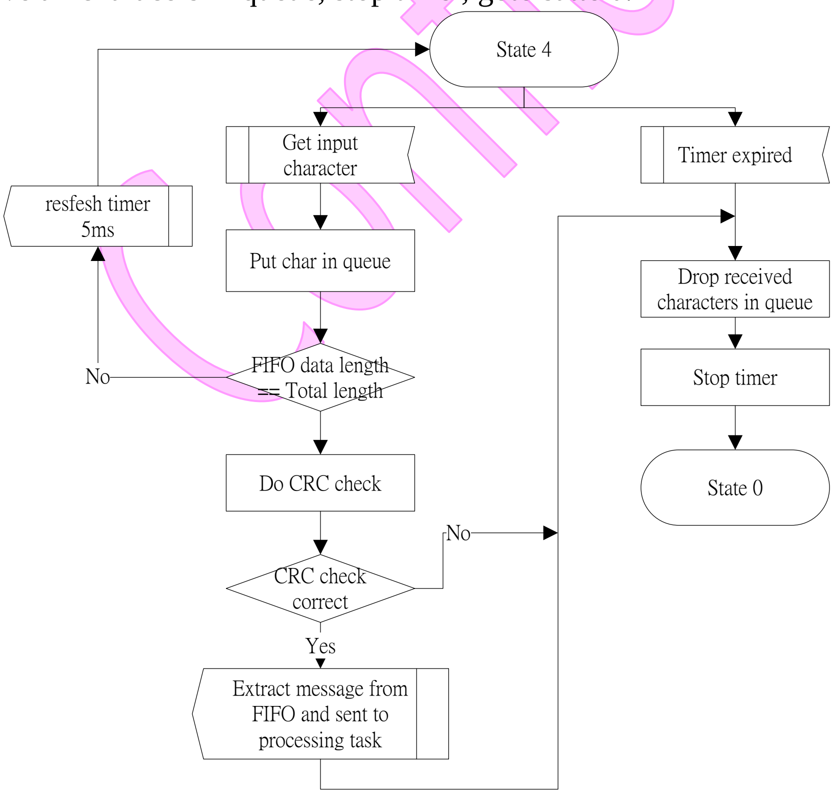
State 2: To get first byte of length field. The character should less or equal 0x05. If got correct first byte of length character, refreshe timer and goto state 3. Otherwise, stop timer and got to state 0.



State 3: Get length byte 2, calculate total length. If total length less then 1514, refresh timer, goto state 4. Otherwise, stop timer, clear buffer and goto state 0.

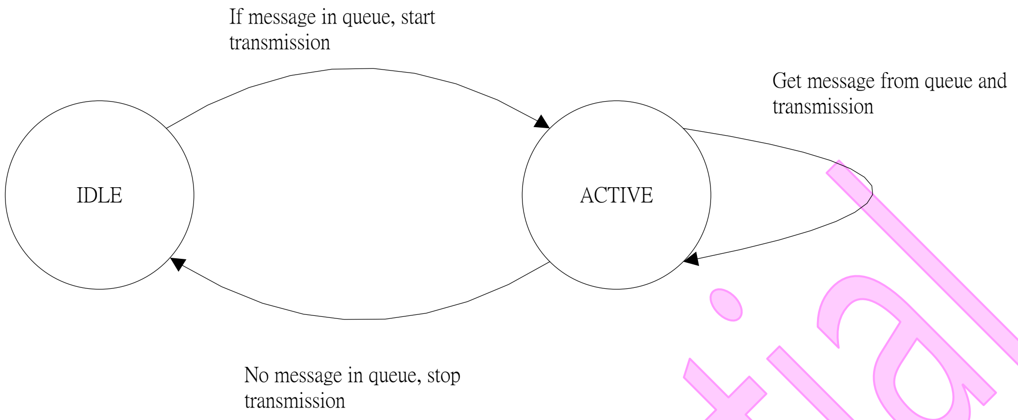


State 4: Repeat to get characters from input, and refersh timer. If total receive count >= total length +2, do CRC check. Correct message will send to processing task, then goto state 0, otherwise, remove all characers in queue, stop timer, goto satte 0.



4.4 Transmission State Diagram

The Transmission function of device driver can process transmission one by one. The transmission queue is applied in development to avoid block when previous message not been transmission complete.

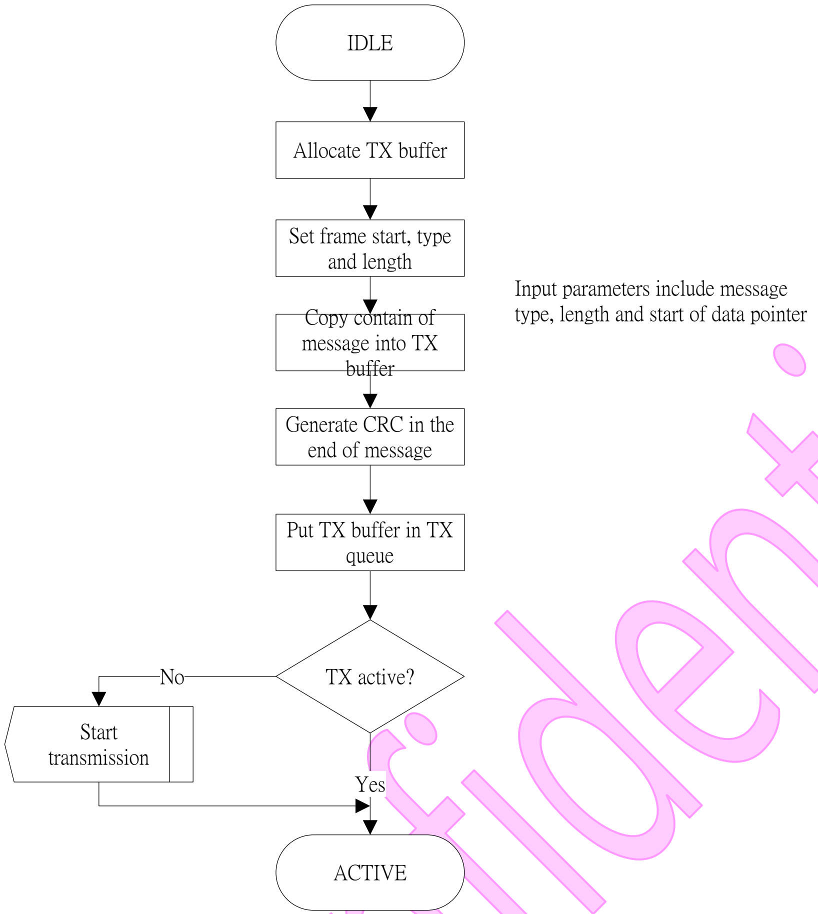


A state variable will be used to control the transmission flow. There are only two states for transmission, IDLE or ACTIVE. The IDLE state means that without messages in queue and driver's transmission function is not actived.

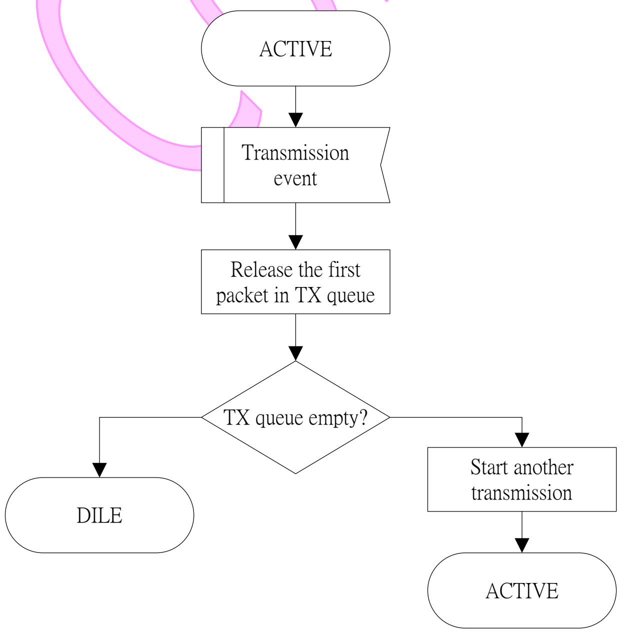
The ACTIVE state is doing transmission for device driver. A handler fuction to process the transmsiion complete event should be implemented. A new transmission will start in handler function if the transmission queue in not empty. The state will change to IDLE in handler function if transmission queue is empty.

4.5 Transmission Flow Chart Diagram

When application send Ethernet packet to Etherent segment delineation software module, it will first allocated a buffer, and then put correct value in message header, copy content into buffer and then calculate CRC value. The buffer will be put in a transmission queue and then star transmission if state variable is IDLE.



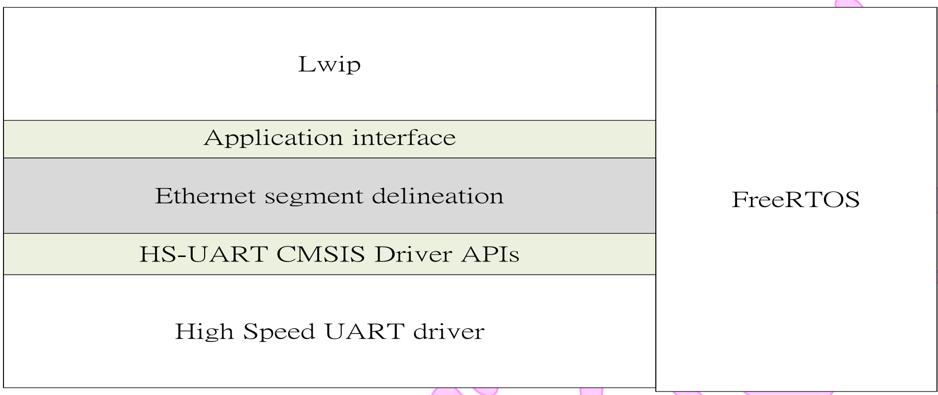
A handler function to process the transmission complete of HS-UART driver will check the transmission queue. If the queue is not empty, it should start another thransmission. Otherwise, it will back to IDLE state.



5. Application Interface

The Etherent segmenet delination module can use in both linux and RTOS system that creates a new network interface based on serial port. So, some APIs should separate for different operation system. Currently, it support FreeRTOS and Linux

For FreeRTOS, the Interface between device driver and Etherent segmenet delination module is HS-UART CMSIS driver APIs. The Interface between LWIP and Etherent segmenet delination module is defined in this section. The software interface with RTOS uses the FreeRTOS APIs.



For Linux, the software module is a kernel driver base on serial port.

5.1 Initial function

The initial function for FreeRTOS will init the basic data structure for Ethernet Segment delination module. For linux, the module will be a driver base on serial port.

5.2 Transmission function

The transmission will be used by LWIP transmission function.

5.3 Receive function

The receive function should register the receive handler functionof LWIP. The input packet will be sent to LWIP via registry handler function. For reducing memory copy, then buffer which used to put then received characters should allocated by LWIP memory pool, and the packet will be process and release by LWIP.

6. Demo System

The demo system shows the sceneriao of data traffic between wifi module and host system via UART port.

Now we plan to do two demo systems, one is Mink (FreeRTOS) plus wifi module; the other is PC (linux) plus wifi module.

The wifi module will use Realtek ameba.

6.1 Mink + Wifi module

6.2 PC(Linux) + Wifi module

~END~

