L20

Quectel GPS Engine

AGPS Application Note

L20_AGPS_AN_V1.0

Document Title	L20 AGPS Application Note
Revision	1.0
Date	2011-4-8
Status	Release
Document Control ID	L20_AGPS_AN_V1.0

General Notes

Quectel offers this information as a service to its customers, to support application and engineering efforts that use the products designed by Quectel. The information provided is based upon requirements specifically provided for customers of Quectel. Quectel has not undertaken any independent search for additional information, relevant to any information that may be in the customer's possession. Furthermore, system validation of this product designed by Quectel within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Contents

0. Revision History

Revision	Date	Author	Description of change
1.0	2011-4-8	Crystal HE/Ree ZHANG	Initial

1. What's CGEE?

The document describes how to use CGEE (Client Generated Extended Ephemeris) supported by SiRF starIV chipset. CGEE is the one type of the AGPS. Shorter time for cold/warm start, particularly warm start only about 10 seconds and faster positioning can be achieved by use of the CGEE. The stored ephemeris can be available within the next 3 days.

Note:
This AGPS application note is directed towards L20 R1.0 firmware version.

1.1. Related Documents

Table 1: Related documents

SN	Document name	Remark
[1]	L20_EVB _UGD	L20 EVB User Guide
[2]	L20_GPS_Protocol	L20 GPS Protocol Specification
[3]	L20_HD_V1.0	L20 Hardware Document

1.2. Abbreviations

Table 2: Abbreviations

Abbreviation	Description
CGEE	Client Generated Extended Ephemeris
GPS	Global Positioning System
GGA	GPS Fix Data
GLL	Geographic Position – Latitude/Longitude
GSA	GNSS DOP and Active Satellites
GSV	GNSS Satellites in View
NMEA	National Marine Electronics Association
OSP	One Socket Protocol
TTFF	Time-To-First-Fix
UART	Universal Asynchronous Receiver & Transmitter
VDOP	Vertical Dilution of Precision

2. How to Use CGEE?

2.1. Reference Design

The functional schematic diagram of CGEE is shown in Figure 1.

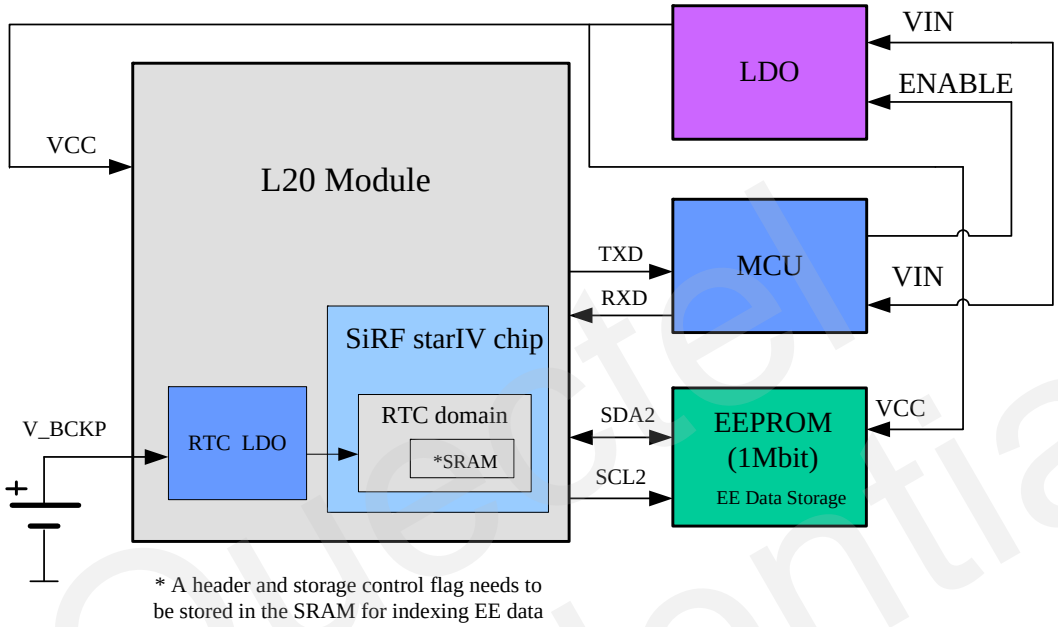

Figure 1: Functional Schematic Diagram of CGEE

The external 1Mbit EEPROM is used to store Client-Generated Extended Ephemeris (CGEE) data generated by SiRF starIV chip through I2C (1) port. Control flag and header of EE Data are written in the SRAM which is in the RTC domain in the chip. When the module is restarted, the information in the SRAM will function as the index of EEPROM. V_BCKP is used to supply power to the RTC domain. When V_BCKP is removed: the information saved in the SRAM will be lost and the CGEE data files stored in the serial EEPROM will not be accessed accordingly when L20 starts up next time. If L20 is powered on again, the system will generate new CGEE data files which will be stored in the eternal EEPROM and the corresponding new header and control flag will be written in the SRAM.

Thus, it is strongly recommended two separate voltage sources, VCC and V_BCKP, should be used in the design of the module. V_BCKP should be kept alive as long as possible.

(1) The I2C port is open drain output and supports up to 400kbps for accessing the EEPROM. The data line and clock line are internally pulled up to VCC by 2.2K resistors.

2.2. Recommended EEPROM

Some types of 1Mbit EEPROM have been tested to be available. The type and manufacturer are listed in table 3.

The reference circuit of EEPROM is shown in Figure 2.

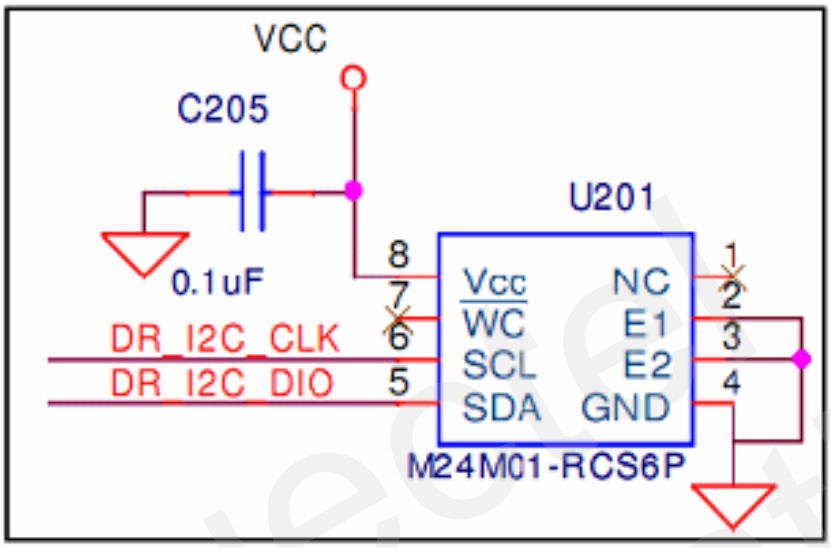

Figure 2: EEPROM Reference Circuit

Table 3: Recommended EEPROM

Manufacturer	Part Number
ST	M24M01
Seiko Instruments Inc.	S-24CM01C
Atmel	AT24C1024B

3. How to Implement CGEE?

The following introduces the guideline on how to operate CGEE when the L20 is powered on and how to restart CGEE when L20 is powered back.

Step 1: Switch NMEA protocol to OSP protocol mode

Click the menu: Receiver-Command-Switch Protocol. See Figure 3.

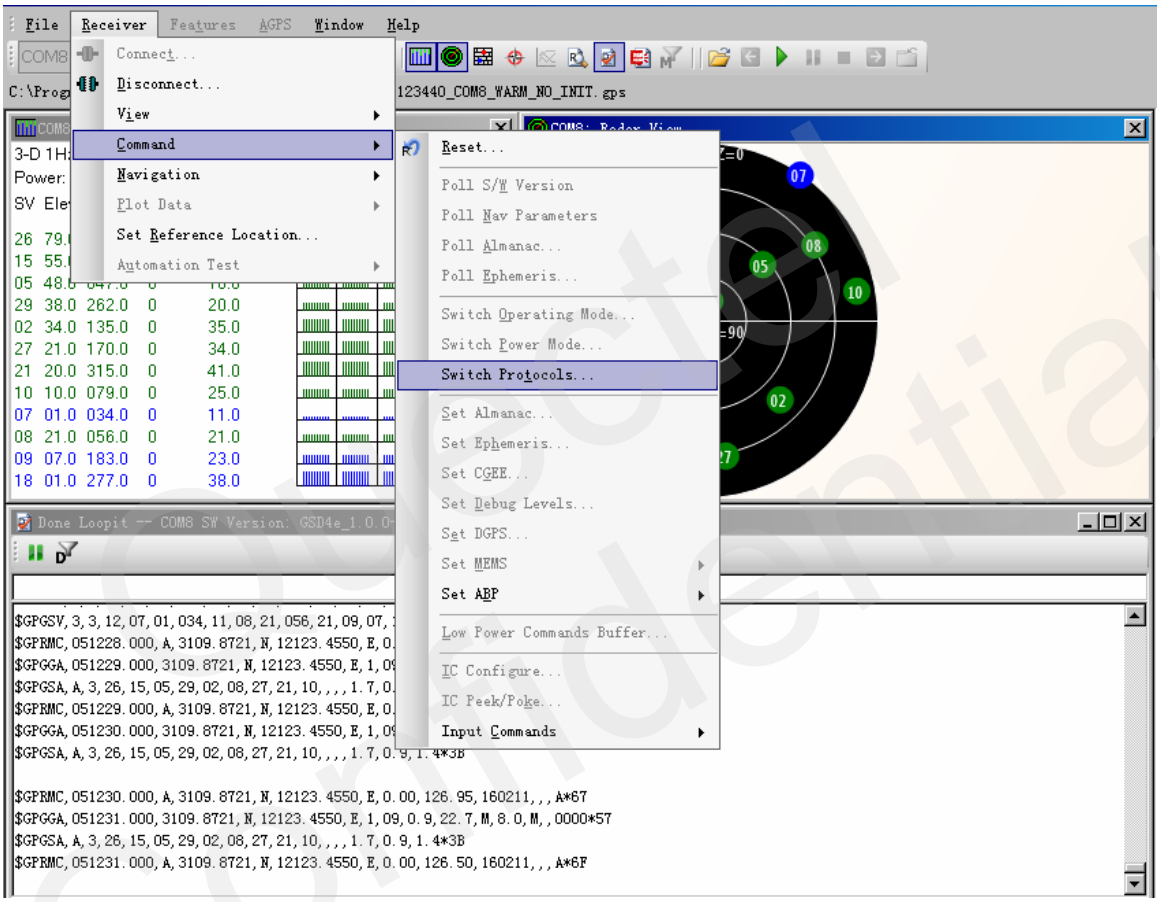

Figure 3

Then click Set, the module will switch to OSP protocol mode. See figure 4 on the Page 8.

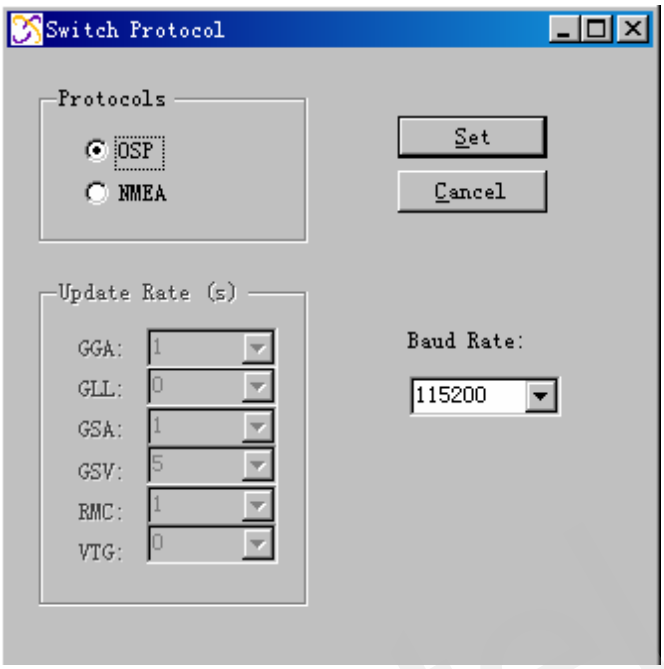

Figure 4

Step 2: Access EEPROM

Click the menu: Receiver-Command-Input Command-User Defined Message. See figure 5.

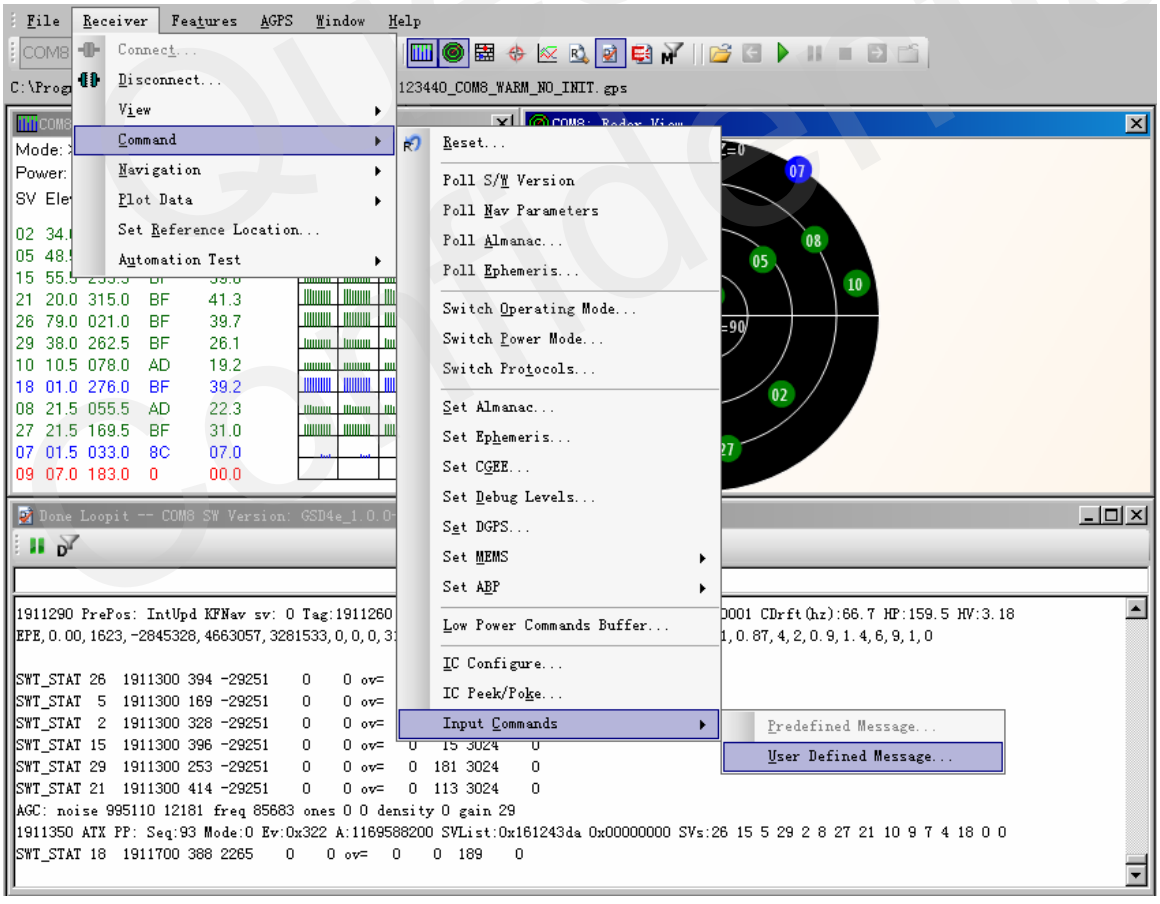

Figure 5

Input defined message, **A0 A2 00 03 E8 FD 01 01 E6 B0 B3**, which allows the module to access

the EEPROM. Then click the button “Send”, L20 will send the EE data generated locally to the EEPROM. See figure 6.

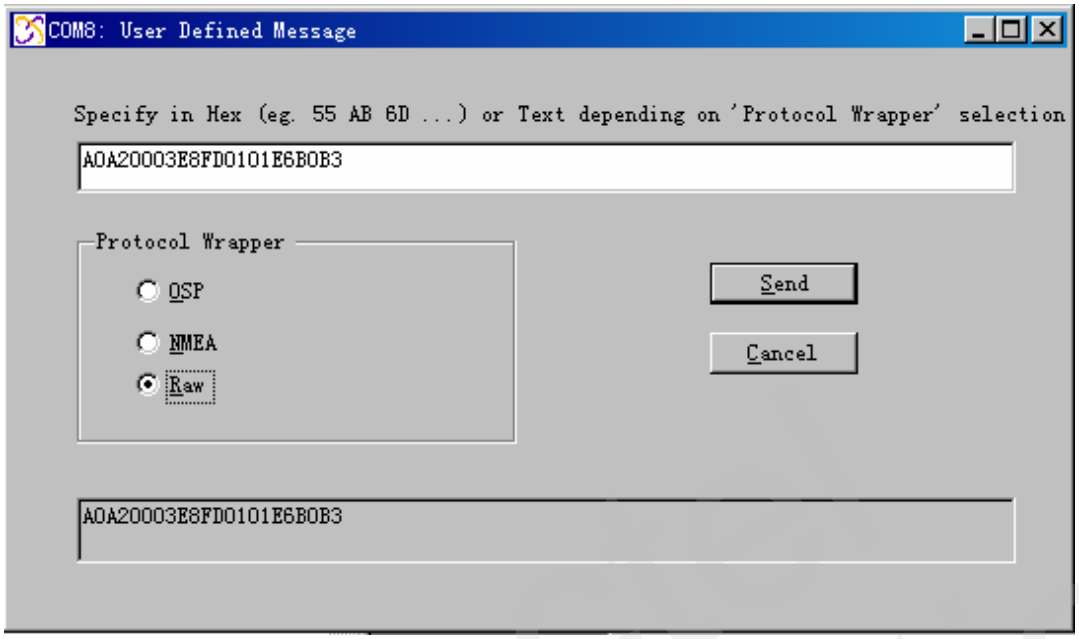

Figure 6

Note: If customers need to shut down the VCC power, Step 3 to 7 is required; otherwise, the following steps can be ignored.

Step 3: Acquire the index from EEPROM

Input defined message: **A0 A2 00 0A B2 03 02 04 80 04 25 28 02 5C 01 EA B0 B3.**

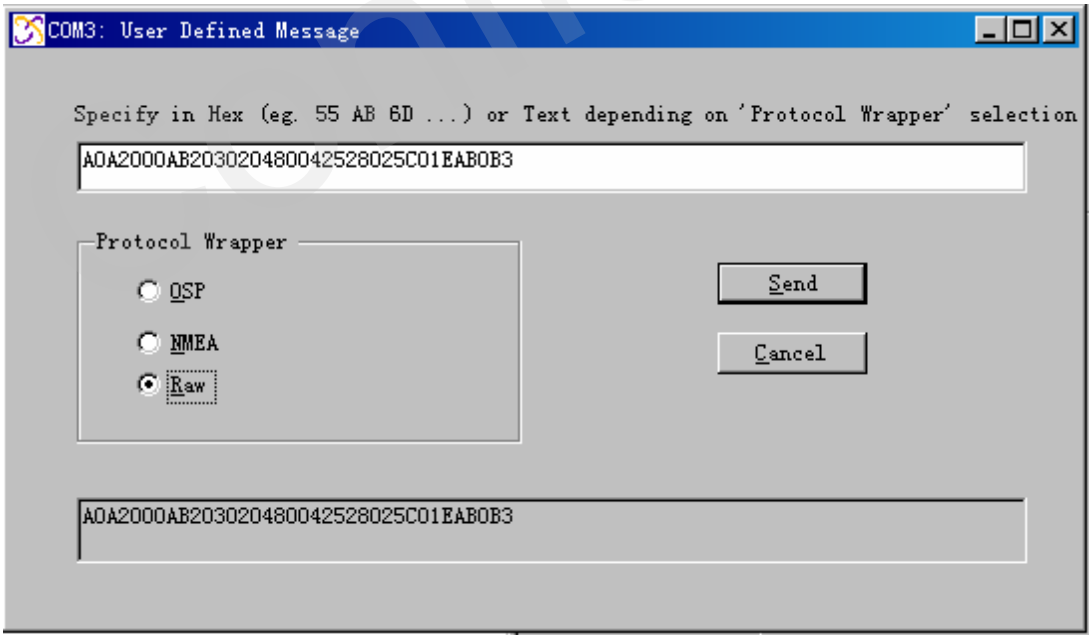

Figure 7

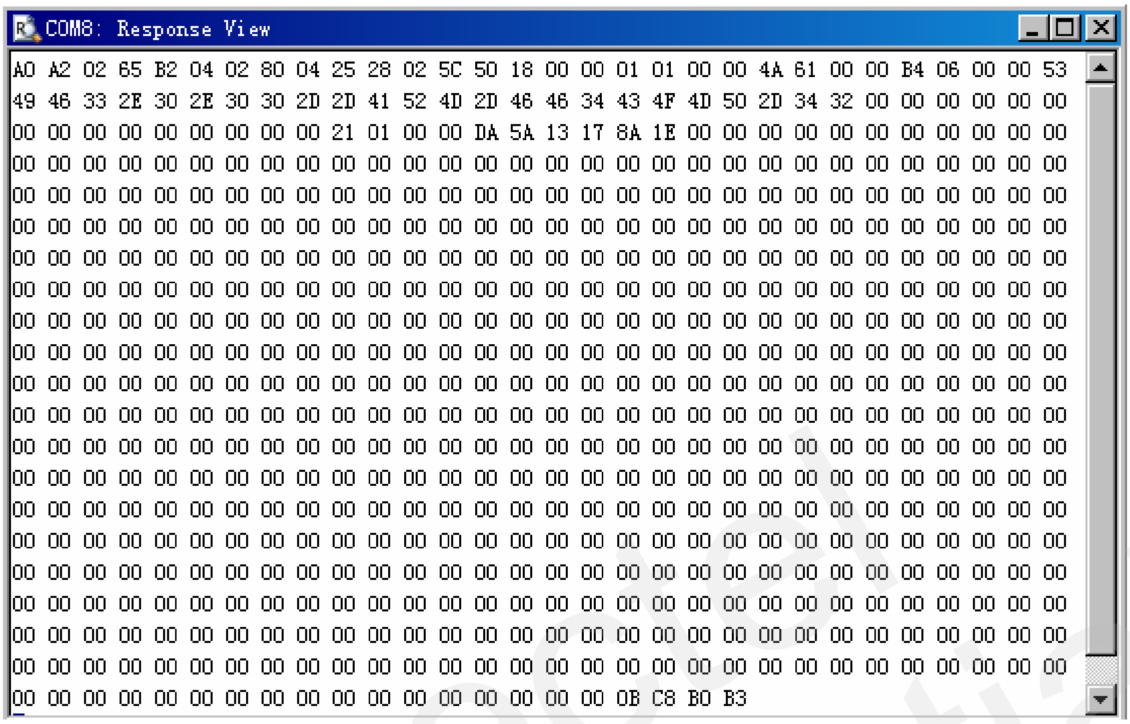

Figure 8

An incoming message, MID178 which has a variable length, will be found in the Response View window. An example of such message is listed in the following data. The payload is 613 bytes. The part in red between the start of the message "A0 A2 02 65 B2 04 02 80 04 25 28 02 5C" and checksum fields "0B C8 B0 B3", should be saved.

A0A20265B2040280042528025C5018000001010000 4A610000B40600005349 46332E302E30302D2D41524D2D4646
34434F4D502D3432000000000000000000000000000000 21010000DA5A13178A1E00000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000BC8B0B3

Step 4: Acquire another index data from EEPROM

Input defined message: **A0 A2 00 0A B2 03 02 04 80 04 27 90 00 24 02 1A B0 B3.**

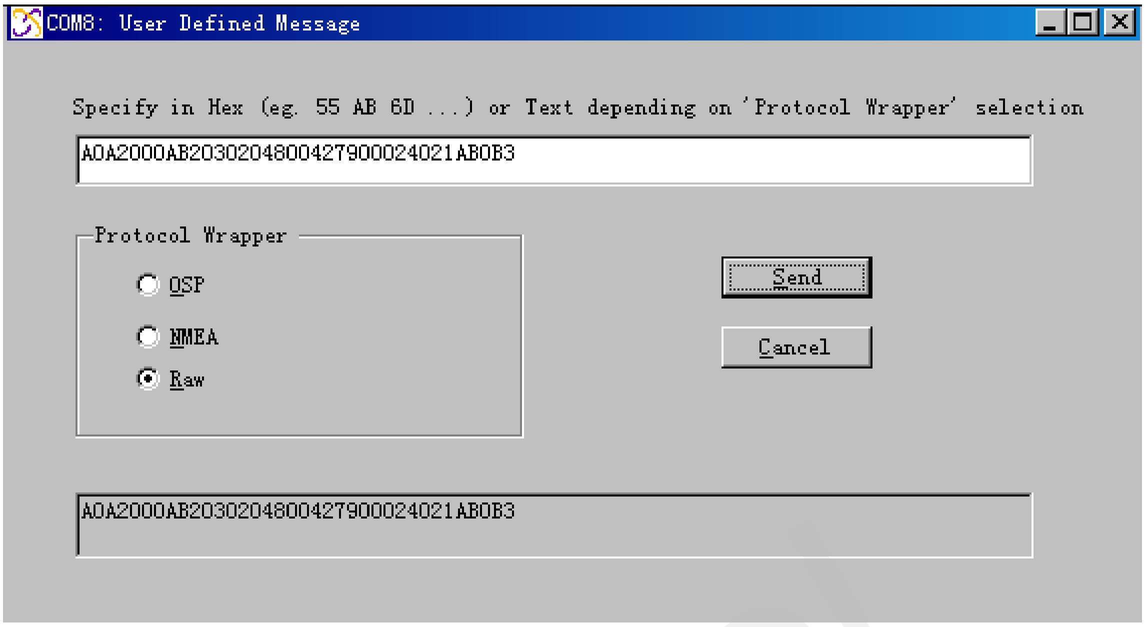

Figure 9

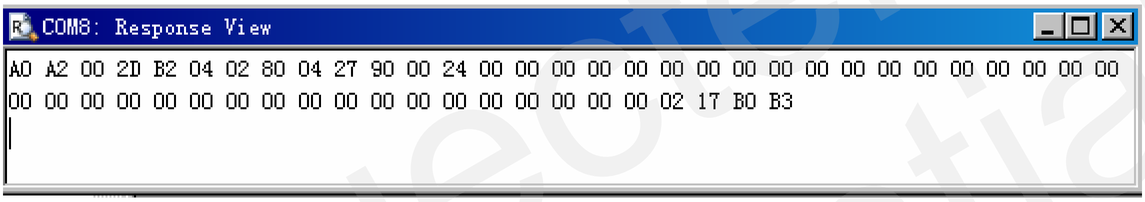

Figure 10

Another incoming message, MID178 which is shorter than the first one, can be found in the Response View window. The red payload bytes between the start of the message "**A0 A2 00 2D B2 04 02 80 04 27 90 00 24**" and checksum fields" **02 17 B0 B3**", should be saved in the message.

A0A2002DB204028004279000240000000000000000000000000000000000000000000000000000000000000000000000000000217B0B3

Step 5: Remove the power

Step 6: Input the processed message saved in step 3

When L20 is powered on again, please append the payload message saved in step 3 behind "**A0 A2 02 66 B2 03 03 04 80 04 25 28 02 5C**". Checksum is marked in blue below and needs to be calculated. After putting these three parts together, then click the button "Send".

*Note: the **Annotations** in the end of this document will introduce how the checksum calculates.*

A0A20266B20303048004252802 5C5018000001010000 4A610000B4060000534946332E302E30302D2D41524D2D464634434F4D502D3432000000000000000000000000000021010000DA5A13178A1E00

000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000BCCB0B3

Input the above message.

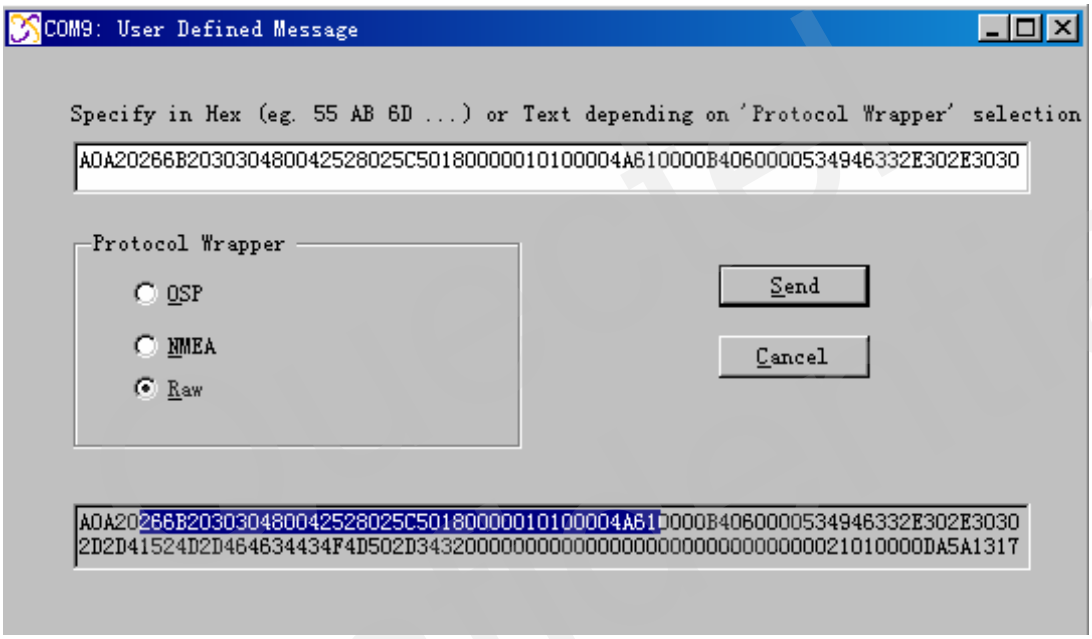

Figure 11

Step 7: Input the processed message saved in step 4

Similar to step 6, append another payload saved in step 4 behind "**A0 A2 00 2E B2 03 03 04 80 00 04 27 90 00 24**", calculate the Checksum, then put these three parts together, and the total message is shown as below:

A0A2002EB2030304800427900240000000000000000000000000000000000000000000000000000000000000000000000
000000000021BB0B3

Input the message, and click the button "Send".

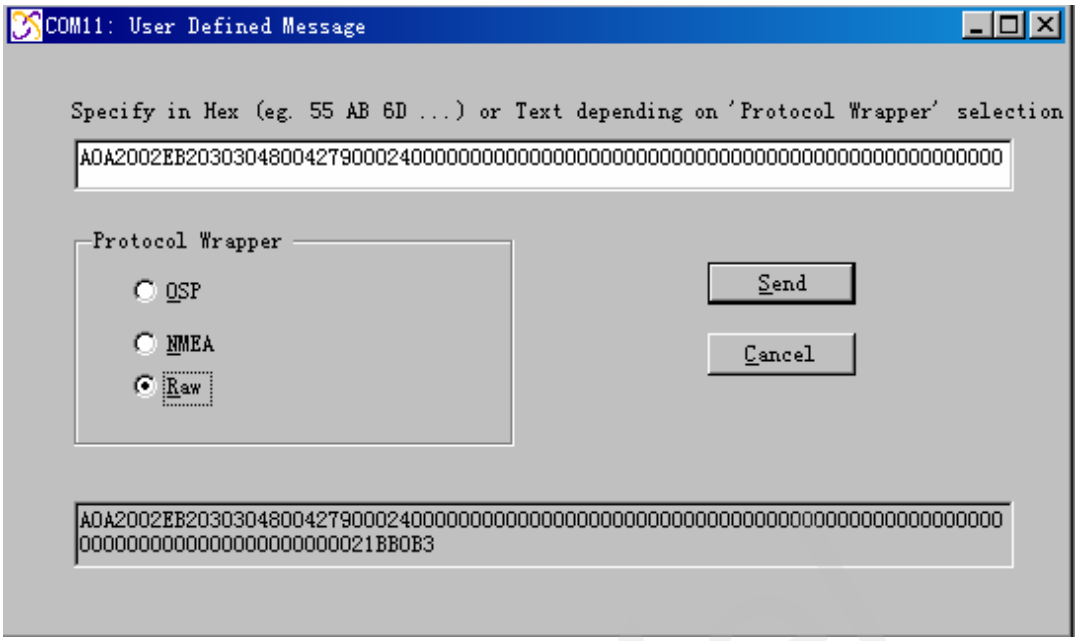

Figure 12

After finishing the steps above, the CGEE will be achieved.

***Note*:**

1) It would take about 70 ms to handle steps above at baud rate of 115200 bps.

2) The data generated from EEPROM is available within the next three days.

3) The incoming messages in Step 3 and 4 are always not the same in different time.

Annotations:

The checksum is transmitted with high order byte first and followed by the low byte. This is the so-called big-endian order.

High Byte	Low Byte
< 0x7F	Any value

The checksum is 15-bit checksum of the bytes in the payload data. The following pseudo code defines the algorithm used.

Enable message to be the array of bytes which will be sent by the transport.

Set msgLen to the number of bytes in the message array to be transmitted.
Index = first
Checksum = 0
While index < msgLen
checkSum = checkSum + message[index]
checksum= checksum AND（2^{15}-1）
checkSum = checkSum AND (2^{15}-1)

For more detailed information, please refer to document [2]

Shanghai Quectel Wireless Solutions Co., Ltd.

Room 501, Building 13, No.99, TianZhou Road, Shanghai, China 200233

Tel: +86 21 54453668

Mail: info@quectel.com