# Application Note for Serial LTE Modem

Nordic Semiconductor ASA

Version 0.X (last updated 2020-03-05)

This document explains how to use proprietary AT commands in Serial LTE Modem in various scenarios. It’s intended for Nordic internal usage only.

# Introduction

## 1.1 System Configuration

By default, the system is defined with

CONFIG\_SLM\_CONNECT\_UART\_0=y

CONFIG\_SLM\_GPIO\_WAKEUP=n

The interface GPIO is P0.6 by default, which is Button 1 on PCA10090. This is the default configuration for SLM to be used with terminal software on PC side. When used with nRF52 or other external MCU, please change to:

CONFIG\_SLM\_CONNECT\_UART\_2=y

CONFIG\_SLM\_GPIO\_WAKEUP=y

The interface GPIO is changed to P0.31 by default.

AT#XSLEEP or AT#XSLEEP=0 deactivate SLM to save power, which can always be re-activated by toggle the interface GPIO. AT#XSLEEP put nRF9160 into sleep, which can be woke up by the interface GPIO if CONFIG\_SLM\_GPIO\_WAKEUP is defined, or otherwise must do a Reset.

To save power, SLM has disabled console and logging over UART\_0. User ca change below to switch between UART and RTT in the prj.conf.

# Segger RTT

CONFIG\_USE\_SEGGER\_RTT=n

CONFIG\_RTT\_CONSOLE=n

CONFIG\_UART\_CONSOLE=y

CONFIG\_LOG\_BACKEND\_RTT=n

CONFIG\_LOG\_BACKEND\_UART=y

## 1.2 How to add own AT command

Refer to any of the exiting modules, tcpip/icmp/gps, etc.

1. Define own AT strings and handlers in own module file
2. Define \_init(), \_uninit(), \_parse() and \_clac(), similar to those in current modules, and expose them in a header file;
3. Call your APIs in slm\_at\_host.c
   1. In slm\_at\_host\_init(), call your \_init()
   2. In slm\_at\_host\_uninit, call your \_uninit()
   3. In handle\_at\_clac(), call your \_clac()
   4. In cmd\_send(), call your \_parse()
4. Pay special to your \_uninit() function, because SLM expects to call it before entering Idle and call \_init() after existing Idle status
5. New command’s name should start with “AT#X”
6. Bug fix in main.c and slm\_at\_host.c/h, please feedback to Nordic

## 1.3 Error Handling

SLM returns the error code that is defined in

<NCS>/zephyr/lib/libc/minimal/include/errno.h

A more complete of the error code list could be find in Google’s Android [repo](https://chromium.googlesource.com/native_client/nacl-newlib/+/master/newlib/libc/include/sys/errno.h).

In case of TCP/IP, the socket is closed for most case of error conditions. To understand the *errno* returned by each API, please refer to [POSIX Programmer's Manual](http://man7.org/linux/man-pages/dir_section_3.html)

# Generic AT commands

\* AT#XSLMVER

\* AT#XSLEEP[=<shutdown\_mode>]

\* AT#XCLAC

**at#xslmver**

#XSLMVER: 1.2

OK

**at#xclac**

AT#XSLMVER

AT#XSLEEP

AT#XCLAC

AT#XSOCKET

AT#XSOCKETOPT

AT#XBIND

AT#XCONNECT

AT#XLISTEN

AT#XACCEPT

AT#XSEND

AT#XRECV

AT#XSENDTO

AT#XRECVFROM

AT#XGETADDRINFO

AT#XPING

AT#XGPSRUN

OK

**at#xsleep=?**

#XSLEEP: (0, 1)

OK

**at#xsleep**

<exit Idle>

Ready

**at#xsleep=0**

<exit Idle>

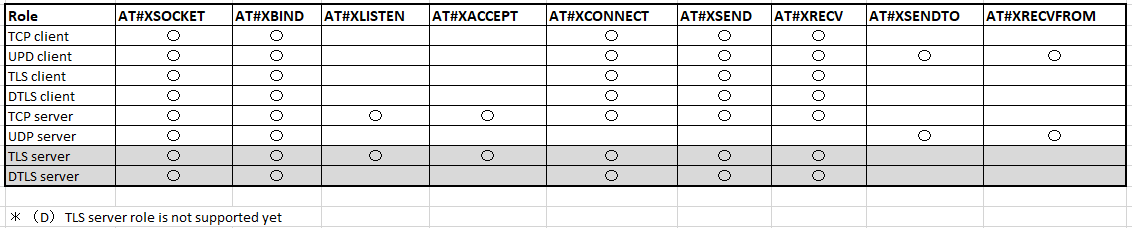
Ready

**at#xsleep=1**

<exit Sleep>

Ready

# TCP/IP AT commands



\* AT#XSOCKET=<op>[,<type>,<role>[,<sec\_tag>]]

\* AT#XSOCKETOPT=<op>,<name>[,<value>]

\* AT#XBIND=<port>

\* AT#XLISTEN

\* AT#XACCEPT

\* AT#XCONNECT=<url>,<port>

\* AT#XSEND=<data>

\* AT#XRECV[=<length>]

\* AT#XSENDTO=<url>,<port>,<data>

\* AT#XRECVFROM=<url>,<port>[,<length>]

\* AT#XGETADDRINFO=<url>

Below result is the case that **CONFIG\_SLM\_TEST\_MODE=y** is continued. As client role, it enables receive right after send; as server role, it enables send right after receive.

## 3.1 TCP client

**at#xsocket=?**

#XSOCKET: (0, 1), (1, 2), (0, 1), <sec-tag>

OK

**at#xsocket=1,1,0**

#XSOCKET: 2, 1, 0, 6

OK

**at#xsocket?**

#XSOCKET: 2, 6, 0

OK

**at#xconnect="remote.host",2442**

#XCONNECT: 1

OK

**at#xconnect?**

+XCONNECT: 1

OK

**at#xsend="Test TCP by hostname"**

#XSEND: 20

#XRECV: PONG: b'Test TCP by hostname'

#XRECV: 29

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

**at#xsocket?**

#XSOCKET: 0

OK

**at#xsocket=1,1,0**

#XSOCKET: 2, 1, 0, 6

OK

**at#xconnect="remote.ip",2442**

#XCONNECT: 1

OK

**at#xsend="Test TCP by IP address"**

#XSEND: 22

#XRECV: PONG: b'Test TCP by IP address'

#XRECV: 31

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

When tested with **CONFIG\_SLM\_TEST\_MODE=n**

**at#xsocket=1,1,0**

#XSOCKET: 2, 1, 0, 6

OK

+CEREG: 1,"106A","0184780A",7

**at#xsocketopt=1,20,30**

OK

**at#xconnect="remote.host",2442**

#XCONNECT: 1

OK

at#xsend="Test TCP by hostname"

#XSEND: 20

OK

**at#xrecv**

#XRECV: PONG: b'Test TCP by hostname'

#XRECV: 29

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

## 3.2 UDP client

Connectionless UDP scenario

**at#xsocket=1,2,0**

#XSOCKET: 2, 2, 0, 17

OK

**at#xsendto="remote.host",2442,"Test UDP by hostname"**

#XSENDTO: 20

#XRECV: PONG: Test UDP by hostname

#XRECV: 26

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

**at#xsocket=1,2,0**

#XSOCKET: 2, 2, 0, 17

OK

**at#xsendto="remote.ip",2442,"Test UDP by IP address"**

#XSENDTO: 22

#XRECV: PONG: Test UDP by IP address

#XRECV: 28

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

Connection-based scenario

**at#xsocket=1,2,0**

#XSOCKET: 2, 2, 0, 17

OK

**at#xconnect="remote.host",2442**

#XCONNECT: 1

OK

**at#xsend="Test UDP by hostname"**

#XSEND: 20

#XRECV: PONG: Test UDP by hostname

#XRECV: 26

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

And the case of **CONFIG\_SLM\_TEST\_MODE=n**

**at#xsocket=1,2,0**

#XSOCKET: 2, 2, 0, 17

OK

**at#xsocketopt=1,20,30**

OK

**at#xconnect="remote.host",2442**

#XCONNECT: 1

OK

**at#xsend="Test UDP by hostname"**

#XSEND: 20

OK

**at#xrecv=32**

#XRECV: PONG: Test UDP by hostname

#XRECV: 26

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

## 3.3 TLS client

Before the test, CA certificate, client certificate and private key has been stored in modem by LTE Link Monitor, with the sec\_tag 1682755.

**AT%CMNG=1,16842755**

%CMNG: 16842755,0,"0000000000000000000000000000000000000000000000000000000000000000"

%CMNG: 16842755,1,"0101010101010101010101010101010101010101010101010101010101010101"

%CMNG: 16842755,2,"0202020202020202020202020202020202020202020202020202020202020202"

OK

**at#xsocket=1,1,0,16842755**

#XSOCKET: 2, 1, 0, 258

OK

**at#xconnect="remote.host",2443**

#XCONNECT: 1

OK

**at#xsend="Test TLS client"**

#XSEND: 15

#XRECV: PONG: b'Test TLS client'

#XRECV: 24

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

PSK-based TLS client has not been tested yet (no TLS server yet)

## 3.4 DTLS client

Before the test, PSK has been stored on the modem side by AT command like below.

**AT%CMNG=0,16842756,3,"6e7266393174657374"**

**AT%CMNG=0,16842756,4,"nrf91test"**

**AT%CMNG=1,16842756**

%CMNG: 16842756,3,"0303030303030303030303030303030303030303030303030303030303030303"

%CMNG: 16842756,4,"0404040404040404040404040404040404040404040404040404040404040404"

OK

The same PSK and Identity has been registered on server side, too.

**at#xsocket=1,2,0,16842756**

#XSOCKET: 2, 2, 0, 273

OK

**at#xconnect="**remote.host**",5684**

#XCONNECT: 1

OK

at#xsocket=0

#XSOCKET: 0, closed

OK

*at#xconnect="remote.host",5684 // This is LwM2M server, just connection test*

*at#xconnect="remote.host",2444*

*at#xconnect="remote.host",2445*

*at#xsend="Test DTLS client"*

Certificate-based DTLS client has not been tested yet (no DTLS server)

## TCP server

To test as Server role, please make sure that the nRF9160 has a global private address and the radio network would route the incoming IP packet to your UE. This could be test by (1) use AT+CGDCONT? To the local IP address allocated by network and check whether it’s reserved Type-A/B/C private address ([Wiki reference](https://en.wikipedia.org/wiki/IP_address#Private_addresses)), if not (2) do a ping from the destination server to your UE.

**at#xsocket=1,1,1**

#XSOCKET: 2, 1, 1, 6

OK

**at#xsocket?**

#XSOCKET: 2, 6, 1

OK

**at#xbind=3442**

OK

**at#xlisten**

OK

**at#xaccept**

#XACCEPT: connected with remote.ip

#XACCEPT: 3

#XRECV: Hello, TCP!

#XRECV: 11

#XSEND: 11

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

A simple python-based TCP client connect with nRF91 for an Echo test.

$ $ **python client\_tcp.py**

Sending: 'Hello, TCP!

Hello, TCP!

Closing connection

$ $

## UDP Server

Like the TCP server, nRF9160 needs to have a global private address.

**at#xsocket=1,2,1**

#XSOCKET: 2, 2, 1, 17

OK

**at#xbind=3442**

OK

**at#xrecvfrom="remote.host",3442**

#XRECV: Hello, UDP!

#XRECV: 11

#XSENDTO: 11

OK

**at#xsocket=0**

#XSOCKET: 0, closed

OK

A simple python-based UDP client connect with nRF91 for an Echo test.

$ **python client\_udp.py**

Sending: 'Hello, UDP!

Hello, UDP!

('202.238.218.44', 3442)

Closing connection

$ $

Please NOTE UDP packet could be lost in either way. For example, lost in downlink to nRF9160,

**at#xrecvfrom="remote.host",3442**

#XSOCKET: -60

ERROR

## (D)TLS Server

(D)TLS server role is currently not supported (modem limitation).

**at#xsocket=1,1,1,16842755**

#XSOCKET: (D)TLS Server not supported

ERROR

**at#xsocket=1,2,1,16842755**

#XSOCKET: (D)TLS Server not supported

ERROR

## 3.8 DNS lookup

**at#xgetaddrinfo="www.google.com"**

#XGETADDRINFO: 172.217.174.100

OK

**at#xgetaddrinfo="remote.host"**

#XGETADDRINFO: 5.189.130.26

OK

**at#xgetaddrinfo="5.189.130.26"**

ERROR

If there is problem to get the IP address, make sure network has assigned DNS server to UE by AT+CGCONTRDP command.

## 3.9 Socket Options

Requires a client-role socket is opened first.

**at#xsocketopt=?**

#XSOCKETOPT: (0, 1), <name>, <value>

OK

**at#xsocket=1,1,0**

#XSOCKET: 2, 1, 0, 6

OK

**at#xsocketopt=1,20,30**

OK

at#xsocketopt=0,20

ERROR // to be ivestigated

**at#xsocketopt=0,2**

#XSOCKETOPT: ignored

OK

**at#xsocketopt=1,2,1**

#XSOCKETOPT: ignored

OK

**at#xsocketopt=0,61**

#XSOCKETOPT: not supported

OK

**at#xsocketopt=1,61,30**

#XSOCKETOPT: not supported

OK

# ICMP AT commands

Ping remote host, also can be used as MO Data call, e.g. during eDRX or PSM

**AT#XPING="remote.ip",45,5000,5,1000**

#XPING: 0.873

#XPING: 0.576

#XPING: 0.599

#XPING: 0.623

#XPING: 0.577

#XPING: average 0.650

OK

# GPS AT commands

GPS standalone sample

**AT%XSYSTEMMODE=0,0,1,0**

OK

**AT%XMAGPIO=1,0,0,1,1,1574,1577 // required ~ DKv0.9.0**

OK

**AT%XCOEX0=1,1,1570,1580 // required to DK v0.10.x ~**

OK

AT+CFUN=1

OK

**at#xgpsrun=1,3**

#XGPSRUN: 1,3

OK

#XGPSS: tracking 1 using 1 unhealthy 0

#XGPSS: tracking 2 using 2 unhealthy 0

…

**at#xgpsrun?**

#XGPSRUN: 1,3

OK

**at#xgpsrun=0**

#XGPSRUN: 0

OK

**at#xgpsrun?**

#XGPSRUN: 0

OK

GPS in PSM mode sample

**AT+CEREG=5**

OK

**AT%XSYSTEMMODE=1,0,1,0**

OK

**AT%XMAGPIO=1,0,0,1,1,1574,1577**

OK

**AT%XCOEX0=1,1,1570,1580**

OK

**AT+CPSMS=1,,,"00100100","00100001"**

OK

**AT+CFUN=1**

OK

+CEREG: 2,"106A","00A3965D",7,0,0,"11100000","11100000"

+CEREG: 1,"106A","00A3965D",7,,,"00011110","00011000"

**at#xgpsrun=1,3**

#XGPSRUN: 1,3

OK

#XGPSS: tracking 1 using 1 unhealthy 0

#XGPSS: tracking 2 using 2 unhealthy 0

…

**at#xgpsrun=0**

#XGPSRUN: 0

OK

GPS in eDRX mode sample

**AT+CEREG=2**

OK

**AT%XSYSTEMMODE=1,0,1,0**

OK

**AT%XCOEX0=1,1,1570,1580**

OK

**AT+CEDRXS=2,4,"1010"**

OK

**AT%XPTW=4,"0001"**

OK

**AT+CFUN=1**

OK

+CEREG: 2,"106A","00A3965D",7

+CEREG: 1,"106A","00A3965D",7

+CEDRXP: 4,"1010","1010","0001"

**at#xgpsrun=1,3**

#XGPSRUN: 1,3

OK

#XGPSS: tracking 1 using 1 unhealthy 0

#XGPSS: tracking 2 using 2 unhealthy 0

…

**at#xgpsrun=0**

#XGPSRUN: 0

OK

A-GPS sample

To be added

# Modem FOTA AT Commands

To be added

AT#XFOTA=<host>,<port>,<uri>

# MQTT AT Commands

To be added

AT#XMQTTCON=<op>[,<url>,<port>[,<sec\_tag>]]

AT#XMQTTSUB=<op>,[<qos>,<topic>[,<qos>,<topic>[,...]]]

AT#XMQTTPUB=<qos>,<retain>,<topic>,<msg>

AT#XMQTTABORT

# Know Issues and Limitations

TCP/IP known limitation in this version

\* - Multiple concurrent sockets

\* - Socket type other than SOCK\_STREAM(1) and SOCK\_DGRAM(2)

\* - IP Protocol other than TCP(6) and UDP(17)

\* - TCP server accept one connection only

\* - Receive more than IPv4 MTU one-time

\* - IPv6 support

\* - does not support proxy

Low-layer UART still have issue in receiving, some warning message like below:

[00:01:10.448,944] <wrn> at\_host: UART RX buf rsp: -16

After this following AT command could result in ERROR. To recover, input more AT command like “AT” until OK is received, or set **CONFIG\_LOG=n.** This is currently under investigation together with driver team.

## Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Comments** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |