

How_to_send_AT_Commands

```
#include <c32comm.h>
Link Against : c32.lib

TRequestStatus status;
RCommServ commServer;
User::LeaveIfError(commServer.Connect());

_LIT(KCsyName, "dataport");
User::LeaveIfError(commServer.LoadCommModule(KCsyName));

_LIT(KDataPort, "DATAPORT::0");

RComm comm;
User::LeaveIfError(comm.Open(commServer1, port, ECommShared));

//Send AT Commands to MODEM.

_LIT8(cmd, "AT\r\n");
comm.Write(status, cmd);
User::WaitForRequest(status);
User::After(5000000);
comm.Close();
commServer.Close();
```

How to Write and Read AT Commands

To pass AT commands to the GSM modem of the mobile device, we need to use the **Dataport CSY module** and read/write commands on **Port 1** of this module.

This port is not available on the emulator so this code will work only on a mobile device.

This code is well tested on Nokia 6600, Nokia 6670 and Nokia 3230.

Header needed:

```
#include <c32comm.h>
```

Library needed:

```
LIBRARY c32.lib
```

Capabilities needed:

```
Capability NetworkControl CommDD
```

```
_LIT(KCsyName, "DATAPORT");
_LIT(KDataPort, "DATAPORT::1");
RCommServ commServer;
RComm comm;

// Connect the server
```

How_to_send_AT_Commands

```
User::LeaveIfError(commServer.Connect());

// Load the CSY module
User::LeaveIfError(commServer.LoadCommModule(KCsyName()));

// Open the port
User::LeaveIfError(comm.Open (commServer, KDataPort, ECommShared));

// Write any AT command
_LIT8(KWriteCommand, "AT+CGSN\r\n");
TRequestStatus writeStatus;
comm.Write(writeStatus, KWriteCommand());
User::WaitForRequest(writeStatus);

// Put some delay before reading the response
const TInt KTimeDelay = 1000000;
User::After(KTimeDelay);

// Here I am reading the port twice since the first read will be an echo
// i.e the same command will be received in response

// Second read will give the actual response, in this case, the IMEI number of
// the device
// Putting in a while loop hangs the application since there is nothing to read
// on the third read, and so the port keeps on waiting for a response

TBuf8<1024> des;// Response will be received in this buffer.
// 1024 is the default buffer size of the receive buffer
TInt count = 0;
TRequestStatus readStatus;
while( count < 2 )
{
    Zero(des);
    ReadCommOrMore(readStatus, des);
    ::WaitForRequest(readStatus);
    ::LeaveIfError(readStatus.Int());
    ++count;
    ::After(500000);
}

// Close the port and the server
comm.Close();
commServer.Close();
```

Note

Logic for reading of AT commands can be implemented in a much better manner according to the requirement. Here I have used a makeshift method of counter just to demonstrate the type and number of responses received.

Related Links

- [AT Commands](#)
- [File:Send AT Commands.zip](#)