# Nokia F-Bus Protocol

**Do you wish to control appliances from your mobile?**

**Sure you do! Read below now**


- A Introduction to Nokia F-Bus
- The F-Bus Protocol
- Sending an SMS message
- Receiving an SMS message
- Deleting an SMS message

- The F-Bus Demo Board
- F-Bus Libraries for Codevision AVR (The Firmware)
- The SMS telemetry Programmable logic Controller (PLC)
- The SMS Car tracking System

- Very useful F-Bus Links


# A Introduction to Nokia F-Bus

**Disclaimer:**

**The author takes no responsibility for the use of this information. Some commands can be very hazardous to your phone and can make your phone useless! Use at your own risk, but have fun! Please Note: No Nokia fones were injured or kill in the production of this page!!**

### How to connect microcontrollers to your Nokia 3310

Most Nokia phones have F-Bus and M-Bus connections that can be used to connect a phone to a PC or in our case a microcontroller. The connection can be used for controlling just about all functions of the phone, as well as uploading new firmware etc. This bus will allow us to send and receive SMS messages. Want to turn your air-conditioner on remotely?


The very popular Nokia 3310/3315 has the F/M Bus connection under the battery holder. This is a bit of a pain to get to and requires a special cable to make the connection. The left picture above shows the 4 gold pads used for the F and M Bus. The right picture shows the F-Bus cable connected to my Nokia 3310.


Nokia download cables are available from most mobile phone shops and some electronics stores. I brought my cable from Dick Smith electronics in Australia. The cost of the Nokia 3310 download cable is around $30AUD and is an excellent investment to get the most out of your phone. You can use PC software like Logomanager and Oxygen Phone Manager (See links below) to upload ringtones, graphics, phone numbers etc. No more paying for those cool ringtones, just download them off the internet or record your own! I have even designed a ringtone doorbell, just what every house needs! The cable contains electronics to level convert 3V signals to RS232 type signals. There are also M and F bus switching in most cables.

Some pages have schematics for making your own cable, however I think it's easier to buy one, as the phone connectors are very hard to come by.

### The differences of M-Bus and F-Bus

M-Bus is a one pin bi-directional bus for both transmitting and receiving data from the phone. It is slow (9600bps) and only half-duplex. Only two pins on the phone are used. One ground and one data. M-Bus runs at 9600bps, 8 data bits, odd parity, one stop bit. The data terminal ready (DTR) pin must be cleared with the request to send (RTS). This powers the electronics in the cable and I think it sets it for M-Bus operation.

F-Bus is the later high-speed full-duplex bus. It uses one pin for transmitting data and one pin for receiving data plus the ground pin. Very much like a standard serial port. It is fast 115,200bps, 8 data bits, no parity, one stop bit. For F-Bus the data terminal ready (DTR) pin must be set and the request to send (RTS) pin cleared.

### The F-Bus Protocol and Commands

The F-Bus is bi-directional serial type bus running at 115,200bps, 8 data bits. The serial cable contains electronics for level conversion and therefore requires power. The first thing to do is supply power to the cable electronics and this is done by setting the DTR (Data Terminal Ready) pin and clearing the RTS (Request to Send) pin.

Added 29-Dec-2003 - For all you microcontroller DIYer's out there, connect the DTR pin to a +3 to 12 Volt supply and RTS to a -3 to -12Volt supply. The easy way to achieve this is by using a Max232 or similar transceiver for the RS232 TX and RX pins and then connecting the DTR pin on the serial cable to the V+ pin on the Max232. Do the same for the RTS, however connect it to the V- pin on the Max232. The V+ and V- pins are derived from internal charge pumps that double the input voltage. ie. for a 5V Max232, the V+ will +10V and the V- will be -10V. I hope this clears up this issue for most people!

The next step is to synchronize the UART in the phone with your PC or microcontroller. This is done by sending a string of 0x55 or 'U' 128 times. Simple! The bus is now ready to be used for sending frames.

The Nokia protocol has a series of commands that allow the user to make calls, send and get SMS messages and lots more. If you have not already go to The Linux Gnokii Project and download the source code. This group has excellent documentation on the commands used with different Nokia phones. This is great work guys! Please do not email me for the Nokia Commands - If you want them download the Linux Gnokii Project Source

Sample frame sent to my Nokia 3310 (showed as a Hex dump)

```
Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
Data: 1E 00 0C D1 00 07 00 01 00 03 00 01 60 00 72 D5
```

This sample frame is used to get the hardware and software version from a Nokia phone. It is a good starting point to test if our implementation of the protocol is working.

Byte 0: All frames sent by cable will start with the character 0x1E first. This is the F-Bus Frame ID. Cable is 0x1E and InfraRed is 0x1C. Byte 1: This is the destination address. When sending data, it's the phone's device ID byte. In our case it's always 00 for the phone.
Byte 2: This is the source address. When sending data, it's the PC's device ID byte. In our case it's always 0x0C (Terminal).
Byte 3: This is the message type or 'command'. 0xD1 is Get HW & SW version.
Byte 4 & 5: Byte 4 & 5 is the message length. In our case it is 7 bytes long. Byte 4 is the MSB and byte 5 is the LSB. Thanks to Ron Reiter for comfirming this!
Byte 6: The data segment starts here and goes for 7 bytes in our case. As The Nokia is a 16 bit phone and therefore requires an even number of bytes. As ours is odd the last byte will be a padding byte and the message will end at location 13.
The last byte in the data segment (Byte 12 above) is the sequence number. The last 3 bits of this byte increments from 0 to 7 for each frame. This part needs to be sent back to the phone in the acknowledge frame. The other bits I am unsure about what they mean!

Bytes 14 & 15: The second to last byte is always the odd checksum byte and the last byte is the even checksum byte. The checksum is calculated by XORing all the odd bytes and placing the result in the odd Checksum location and then XORing the even bytes and then placing the result in the even byte.

Well that is our first frame for our Nokia Phone. If the phone received it is show reply with the following data

```
1E 0C 00 7F 00 02 D1 00 CF 71
1E 0C 00 D2 00 26 01 00 00 03 56 20 30 34 2E 34 35 0A 32 31 2D 30 36 2D 30 31 0A 4E 48 4D 2D 35 0A 28 63 29 20 4E 4D
50 2E 00 01 41 3F A4
```

The first line is an Acknowledge command frame. Notice how the destination and source addresses are now swapped. This is because the Nokia phone is now talking. This message is two bytes long with the two bytes representing the message type received (0xD1) and the sequence number (0x00). The last two bytes are the checksum and should be checked to make sure the data is correct. The 3310 will be waiting for an acknowledge frame after these two frames were sent. If the acknowledge frame is not sent the 3310 will retry sending the data. The 3310 will only send the data 3 times and then gives up.

The second frame from our Nokia 3310 is the data we requested. The message type is 0xD2. This is 'receive Get HW&SW version'. This 38-byte (0x26) message should show 0x0003 "V " "firmware\n" "firmware date\n" "model\n" "(c)NMP." The last byte in the data is the sequence number. As with standard F-bus frames, the last two bytes in the frame are checksum bytes.

The received data without f-bus frame

```
01 00 00 03 56 20 30 34 2E 34 35 0A 32 31 2D 30 36 2D 30 31 0A 4E 48 4D 2D 35 0A 28 63 29 20 4E 4D 50 2E 00 01 41

      0003  V     0  4  .  4  5  \n 2  1  /  0  6  /  0  1  \n N  H  M  -  5  \n (  c  )     N  M  P  .        Sequence no.
```

All that is required now is to send a acknowledge frame back to the phone to say 'I got it!'

```
1E 00 0C 7F 00 02 D2 01 C0 7C
```

0x7F is the acknowledge frame's command. We are only required to send a two-byte message so length is set to 0x02. The message contains the acknowledged message type (0xD2) and the sequence no. (0x01). The sequence number is made from the last 3 bits of the sequence number in the previous frame. The checksum needs to be calculated and sent.

Well that's easy now! We will look at sending SMS messages next, (part two) and look at the AVR hardware for the project in part three, coming soon!!

---

# How to send a SMS message with F-Bus?

Now that we know how to send frames on the bus, we should look at sending an SMS message. It is not a hard process, once you can pack the message into 7-bit characters!

### SMS Point-to-Point Character Packing

The first thing you should do is download GSM 03.38 - Alphabets and language-specific information. This is the Technical Specification that describes the packing of 7-bit characters and shows the standard character map. Packing ASCII into 7-bit characters is a pain, but is quite easy to do. Here is a quick example.

Let's say we want to decode the string 'hello'. First I have displayed 'hello' in hexadecimal using the character map provided in GSM 03.38. For A to Z and numbers its just the standard ASCII conversion.

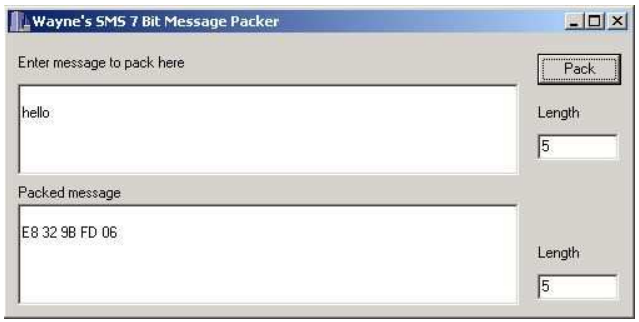
```
h       e       l       l       o       (ASCII characters)
68      65      6C      6C      6F      (In hexadecimal)
1101000 1100101 1101100 1101100 1101111 (In Binary)
```

When dealing with binary, it makes life easier to write everything backwards. The first byte in the string is on the right. The least significant bit is then displayed on the left with the most significant bit on the left. Shown below is the same string of 'hello' just displayed backwards. Then it's just a matter to dividing the binary values into bytes starting with the first character in the string. (Start from right and go to left.) The first decoded byte is simply the first 7 bits of the first character with the first bit of the second character added to the end as shown below. The next decoded byte in then the remaining 6 bits from the second character with two bits of the third byte added to the end. This process just keeps going until all characters are decoded. The last decoded byte is the remaining bits from the last character with the most significant bits packed with zeros.

```
6F      6C      6C      65      68
1101111 1101100 1101100 1100101 1101000 (The ASCII characters shown in binary)

110 11111101 10011011 00110010 11101000 (The above binary just split into 8 bit segments)
06      FD      9B      32      E8      (The 8 bit segments decoded into hex)
```

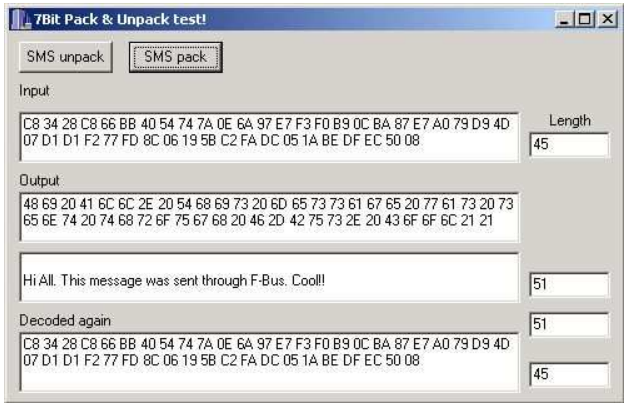
The message hello is therefore E8 32 9B FD 06 when packed.

When playing with sending SMS messages a lot of decoding is required. Therefore I wrote a few small programs to automatically pack and unpack the data for me. The first program allows me to input the string to be packed in the top memo box and when the button 'Pack' is pressed the top string is packed ready to go!




The full program with source for downloading SMSPacker.zip

This program below, will both unpack and then pack the message. It was used to read the packed messages from my serial protocol analyzer. It also tests my subroutines in C for both pack and unpack. Already for my AT90S8535 microcontroller!




Now we know how to pack and unpack the message lets look at the full frame.

### The Full SMS Message Frame

I would highly recommend downloading the GSM 03.40 - Technical realization of the Short Message Service (SMS) Point-to-Point (PP). This specification describes the following SMS fields in detail. Refer to this specification for further information.

Sample frame sent to my Nokia 3310 (showed as a Hex dump) 98 Bytes

```
Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
Data: 1E 00 0C 02 00 59 00 01 00 01 02 00 07 91 16 14 91 09 10 F0 00 00 00 00 15 00 00 00 33 0A 81 40 30 87 00 47
               SMS                              message centre --                                  Phone number -

Byte: 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
Data: 00 00 00 00 00 A7 00 00 00 00 00 00 C8 34 28 C8 66 BB 40 54 74 7A 0E 6A 97 E7 F3 F0 B9 0C BA 87 E7 A0 79 D9
                                        Start of Message - Hi All. This message was sent through F-Bus. Cool!!

Byte: 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
Data: 4D 07 D1 D1 F2 77 FD 8C 06 19 5B C2 FA DC 05 1A BE DF EC 50 08 01 43 00 7A 52
```

**F-Bus Frame Header**
Byte 0: F-Bus Frame ID. We are on Cable (0x1E) See Part 1.

Byte 1: Destination address.
Byte 2: Source address.
Byte 3: Message Type or 'command'. 0x02 (SMS Handling).
Byte 4 & 5: Message length. In our case it is 0x0059 bytes long or 89 bytes in decimal.

**(SMS) Short Message Service Frame Header**
Byte 6 to 8: Start of the SMS Frame Header. 0x00, 0x01, 0x00
Byte 9 to 11: 0x01, 0x02, 0x00 = Send SMS Message

**(SMSC) Short Message Service Centre (12 Bytes)**
Byte 12: SMS Centre number length. 0x07 is 7 bytes long. This includes SMSC Number Type and SMS Centre Phone Number
Byte 13: SMSC number type e.g. 0x81-unknown 0x91-international 0xa1-national

```
1XXX IIII : Where I is the Numbering-plan-identification (Refer to GSM 03.40 - 9.1.2.5 Address fields)
1TTT XXXX : Where T is the Type-of-number (Refer to GSM 03.40 - 9.1.2.5 Address fields)
```

Byte 14 to 23: (Octet format) SMS Centre Phone Number In this case +61 411990010

**(TPDU) Transfer Protocol Data Unit**
Byte 24: Message Type

```
XXXX XXX1 = SMS Submit - The short message is transmitted from the Mobile Station (MS) to the Service Centre (SC).
XXXX XXX0 = SMS Deliver - The short message is transmitted from the SC to the MS.
```

(Refer to GSM 03.40 - 9.2.3 Definition of the TPDU parameters) In our case it is 0x15 = 0001 0101 in binary. The message is SMS Submit, Reject Duplicates, and Validity Indicator present.

Byte 25: Message Reference if SMS Deliver & Validity Indicator used (Not used in this case). Refer GSM 03.40 - 9.2.3.6 TP-Message-Reference (TP-MR)
Byte 26: Protocol ID. Refer to GSM 3.40 - 9.2.3.9 TP-Protocol-Identifier (TP-PID)
Byte 27: Data Coding Scheme. Refer to GSM 03.38 & GSM 3.40 - 9.2.3.10 TP-Data-Coding-Scheme (TP-DCS)
Byte 28: Message Size is 0x33 in hex or 51 bytes long in decimal. This is the size of the unpacked message.
Refer to GSM 03.40 - 9.2.3.16 TP-User-Data-Length (TP-UDL)

**Destination's Phone Number (12 Bytes)**
Byte 29: Destination's number length. Is this correct?. Byte 30: Number type e.g. 0x81-unknown 0x91-international 0xa1-national
Byte 31 to 40: (Octet format) Destination's Phone Number

**Validity Period (VP)**
Byte 41: Validity-Period Code. Time period during which the originator considers the short message to be valid.
Byte 42 to 47: Service Centre Time Stamp ?? For SMS-Deliver

**The SMS Message (SMS-SUBMIT)**
Byte 48 to 92: This is the SMS message packed into 7 bit characters. SMS Point-to-Point Character Packing
Byte 93: Always 0x00

**The F-Bus usual ending**
Byte 94: Packet Sequence Number
Byte 95: Padding Byte - String is old and requires to be even!
Byte 96 & 97: Odd & even checksum bytes.

If the phone receives a valid frame it should reply with something like this below, to say it got the message.
Reply frame sent from my Nokia 3310 (showed as a Hex dump)

```
Byte: 00 01 02 03 04 05 06 07 08 09
Data: 1E 0C 00 7F 00 02 02 03 1C 72
```

This is just like the above Acknowledge command frame. The destination and source addresses are swapped, as this is a frame from the phone to the PC. This message is two bytes long with the first byte representing the message type received (0x02) and the next byte, the sequence number (0x03). The last two bytes are the checksum and should be checked to make sure the data is correct.

After a short time the phone will reply with a 'Message sent' frame shown below.

```
Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17
```

```
Data: 1E 0C 00 02 00 09 01 08 00 02 64 12 00 01 44 00 3F 1E
```

Byte 03: Message Type = 0x02 - SMS Handing
Byte 04 & 05: Message Length = 0x0009 - 9 Bytes long
Byte 09: 0x02 = Message Sent
Byte 10 to 14: Unsure of what this is?

The PC must then acknowledge the frame. No explanation required here!

```
Byte: 00 01 02 03 04 05 06 07 08 09
Data: 1E 00 0C 7F 00 02 02 04 10 79
```

---

# How to receive a SMS message with F-Bus?

This part is really quite simple. When the phone gets sent a SMS message it sends a 'SMS Message Received Frame' with the message attached. All you have to do is decode it!

The SMS send example above sends a SMS message back to my phone (The same phone sending the message). Therefore below is the SMS message that was sent above, but now getting received by the phone.

```
Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
Data: 1E 0C 00 02 00 59 01 08 00 10 02 10 00 07 91 16 14 91 09 10 F0 00 10 19 38 04 00 00 33 0B 91 16 04 73 08 70
                                          Message Centre --                                 Phone Number
Byte: 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
Data: F4 70 40 32 25 30 30 82 22 74 45 4C 25 30 30 82 22 74 45 4C 74 7A 0E 6A 97 E7 F3 F0 B9 0C BA 87 E7 A0 79 D9

Byte: 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97
Data: 4D 07 D1 D1 F2 77 FD 8C 06 19 5B C2 FA DC 05 1A BE DF EC 50 08 01 45 00 4A 5C
```

Byte 03: Message Type = 0x02 - SMS Handing
Byte 04 & 05: Message Length = 0x0059 - 89 Bytes long
Byte 09: 0x10 = SMS Message received
Byte 10: 0x02 = Memory Type = SIM
Byte 11: 0x10 = Location where SMS message stored - required to delete SMS message

**(TPDU) Transfer Protocol Data Unit**
Byte 24: 0x38
Byte 25: 0x04
Byte 26: Protocol ID
Byte 27: Data Coding Scheme
Byte 28: Message Length. 0x33 = 51 Bytes long!

The PC must then acknowledge this frame like normal

```
Byte: 00 01 02 03 04 05 06 07 08 09
Data: 1E 00 0C 7F 00 02 02 05 10 78
```

Well that's the received message above, the only last thing that can be done, is to delete the message from the phone. The phone can only hold so many messages so this should be required, or the memory with fill up.

---

# How to delete a SMS message with F-Bus?

This part is really quite simple as well. When the phone gets sent a SMS message it sends a 'SMS Message Received Frame' with the message attached. In this frame is the location where the message is stored. All you have to do is tell the phone to delete it!

```
Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
Data: 1E 00 0C 14 00 08 00 01 00 0A 02 02 01 41 11 54
```

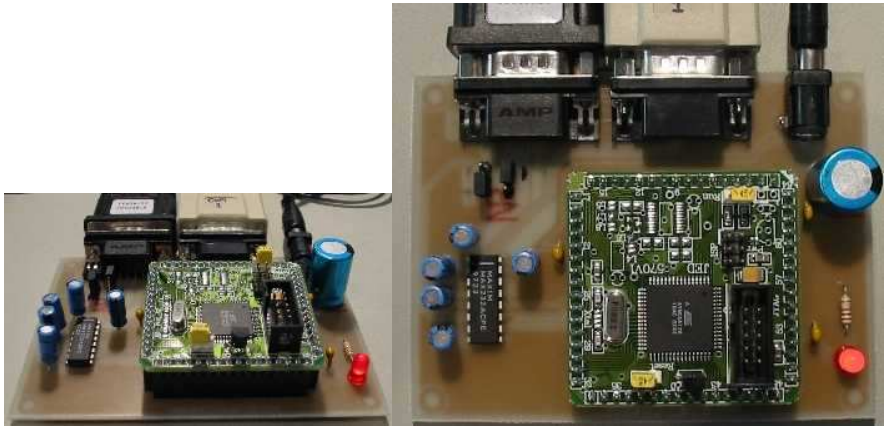
Byte 03: Message Type = 0x14 - SMS Functions
Byte 04 & 05: Message Length = 0x0008 - 08 Bytes long
Byte 6 to 8: Start of the SMS Frame Header. 0x00, 0x01, 0x00
Byte 9: 0x0A Delete SMS Message
Byte 10: 0x02 = Memory Type = SIM - Make sure message is store in this type (0x03 = phone)
Byte 11: 0x02 = Location where SMS message stored. This location can be found in the 'receive SMS frame' (Byte 11)
Byte 12: 0x01
Byte 13: Packet Sequence Number
Byte 14 & 15: Odd & even checksum bytes.

---

# The first hardware demo - Nokia F-Bus on a ATMega128

Sample code will be written with Codevision AVR standard for the ATMega128. Only basic code will be supplied to send an SMS message. A full version on the firmware and PCB will be available for sale. This will send SMS messages from a number of alarm inputs and receive SMS messages from a number of logic level outputs or relays.

Below is my first demo interface board. It has two serial ports, one goes to the nokia phone with the other going to a terminal for control. The AVR570 Processor Board is the heart of the system. This was supplied by JED Micro for $66AUD in one off quantities. A very quick way of rapid development!



The two jumpers near the F-Bus connection sets the levels for RTS and DTR. The 6 Pin header on the AVR570 is the In-Circuit Programming (ISP) port with the 10 Pin IDC used for JTAG programming and debugging. The board was designed with hole-through components as I am trying to get rid of my old stock!

24/05/2003 Well I have been able to receive and display hardware and software information from my Nokia Phone (see below). This is the first step to getting the frame send and receive routines working with the frame acknowledge. I have been able to send a SMS from my Mega128 using a fixed frame of data stored in flash. The next step is to build the frame on the fly, so any text can be sent.

I am currently writing a Command Interpreter so simple commands can be sent over the serial interface to setup the processor and send messages etc.

A dump from the terminal

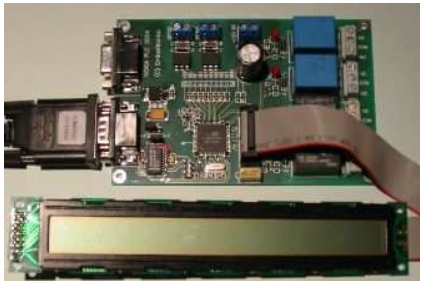
```
Welcome to Wayne's Nokia 3310 F-Bus Command Manager Version 1.0
S->GetHWSW

Sending Frame: 1E 00 0C D1 00 07 00 01 00 03 00 01 60 00 72 D5
Received: 1E 0C 00 7F 00 02 D1 00 CF 71
Received: 1E 0C 00 D2 00 26 01 00 00 03 56 20 30 34 2E 34 35 0A 32 31 2D 30 36 2D 30 31 0A 4E 48 4D 2D 35
0A 28 63 29 20 4E 4D 50 2E 00 01 43 3F A6
Sending Ack: 1E 00 0C 7F 00 02 D2 03 C0 7E

Hardware & Software Versions
Firware: V 04.45 21-06-01
Model: NHM-5

OK->
```

02/06/2003 We this is it! I can now receive SMS messages and turn I/O pins on and off remotely. Below is a screen dump from the RS232 port used for debuging my project.

```
Welcome to Wayne's Nokia 3310 SMS Manager MK1
Sending Frame: 1E 00 0C D1 00 07 00 01 00 03 00 01 60 00 72 D5
Received: 1E 0C 00 7F 00 02 D1 00 CF 71
Received: 1E 0C 00 D2 00 26 01 00 00 03 56 20 30 34 2E 34 35 0A 32 31 2D 30 36 2D 30 31 0A 4E 48 4D 2D 35 0A 28 63 29 20 4E 4D 50 2E 00 01 45 3F A0
Sending Ack: 1E 00 0C 7F 00 02 D2 05 C0 78

Hardware & Software Versions
Firware: V 04.45 21-06-01
Model: NHM-5
Sending Frame: 1E 00 0C 02 00 08 00 01 00 33 64 01 01 42 77 7B
Received: 1E 0C 00 7F 00 02 02 82 1C F3
Received: 1E 0C 00 02 00 2A 01 08 00 34 01 FD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 91 16 14 91 09 10 F0 00 00 00 00 53 4D 53 43 31 30 00 01 46 5D 02
Sending Ack: 1E 00 0C 7F 00 02 02 06 10 7B

SMS Centre Details
SMS Number : +61411990010
SMS Name : SMSC10
Received: 1E 0C 00 02 00 33 01 08 00 10 02 18 00 07 91 16 14 91 09 10 F0 00 10 1F C8 04 00 00 07 0B 91 16 04 73 95 72 F9 30 60 10 22 30 60 10 22 45 83 6B CC 32 39 06 7A BB 01 01 47 00 0D 0A
Sending Ack: 1E 00 0C 7F 00 02 02 07 10 7A

Receiving incoming SMS message!
SMS Centre Number : +61411990010
Senders Phone Number : +61403XXXXXX
Message Size = 7
Message = Led1 on

We have a Match LED ON
Received: 1E 0C 00 0A 00 15 01 08 00 71 01 00 01 0B 01 02 C3 F2 22 75 05 F5 20 00 00 01 40 00 9A 10
Sending Ack: 1E 00 0C 7F 00 02 0A 00 18 7D
Received: 1E 0C 00 02 00 33 01 08 00 10 02 19 00 07 91 16 14 91 09 10 F0 00 10 1F 48 04 00 00 08 0B 91 16 04 73 95 72 F9 30 60 10 22 30 60 10 22 55 81 6B CC 32 39 06 7A 9B CD 01 41 00 4A 3B
Sending Ack: 1E 00 0C 7F 00 02 02 01 10 7C

Receiving incoming SMS message!
SMS Centre Number : +61411990010
Senders Phone Number : +61403XXXXXX
Message Size = 8
Message = Led1 off

We have a Match LED OFF
Received: 1E 0C 00 0A 00 15 01 08 00 71 01 00 01 0B 01 02 C3 F2 22 75 05 F5 20 00 00 01 42 00 98 10
Sending Ack: 1E 00 0C 7F 00 02 0A 02 18 7F
```

---

# The SMS telemetry Programmable logic Controller (PLC)



### The Nokia SMS I/O Controller

It's currently raining and your still at work and at least 1 hour from home, wouldn't it be great to turn on your heater or disable the watering system while at work or across the other side of the state. Well you can with this Nokia based Interface module. If you need to turn pumps, lights etc. on remotely or need to remotely monitor equipment alarms then this project is for you. There are thousands of uses, from turning on remote pumps, lights, monitoring cool room temperatures, open doors etc.

With production of mobile phones topping 600 million, world wide in a year, I thought it be great to find a use for that old mobile phone, that still works fine but just not up with the latest gizmos. There are commercial SMS units around, however they contain an embedded modem and are very costly. This board contains four relays, two optos and 8 analog to digital channels or general logic level input/output pins. When the correct SMS message is sent to the phone connected to this board, a relay can be turned on or off. These relays can used to switch mains equipment of a few amps. The two opto-isolated inputs are for alarm condition inputs. When a voltage is applied to these inputs, a changeable alarm message is sent to the nominated phone number. A extra 8 pin can be used as analog to digital input sampling or just standard logic level inputs and outputs.

An optional 2x 40-character LCD panel can be added to give instant status of the unit. The unit runs from a 12 volt supply and is portable so it can be used in moving equipment like portable generators and farm equipment.

The systems can work in many ways, however you will need you own mobile phone plus a spare one to connect to the Nokia PLC board. If you need to turn on a Pump connected to Relay 2 you just setup the message string for relay2 to 'pump' using the terminal port on the Nokia PLC. All settings are stored in the Mega128's internal EEPROM so none are lossed during a period of power loss. Once this is done, just type 'pump on' into you phone and send a SMS to the phone connected to the system. The relay should then turn on. To turn it off, just use your phone to SMS 'Pump off' to the Nokia PLC. If enabled the Nokia PLC will send a reply back to you phone, to say the command was successful. As anyone who has sent a message on new years eve will know, there can be big delays or even message loss when the network is overworked.

If one of the Opto-isolated inputs become active the Mega128 will send a programmed SMS message to the phone who's number is programmed into the Nokia PLC. This message can be anything like 'over temperature alarm' etc. For the more advanced projects the temperature reading could be sent.

---

## The SMS car tracking system

5th of March 2004 - I have my first prototype boards for this project due in two weeks. It's a complete GPS based tracking system for cars, boats, containers and anything the needs tracking. The board has provision for the Lassen SQ GPS module, Atmel DataFlash and the SHT11 Humidity and Temperature sensor. A LCD panel can be added for a real time status display.

# The GPS Car Tracker

Well this is the circuit board for my new GPS based car tracker. If your car gets stolen, just SMS this board inside your car and it will SMS it's position back! It can also SMS you once the cars moves over a set speed. Complete with inbuilt data logger to log your position with car temperature. Just the thing you need if you run a fleet of cars. More coming soon!

### Nokia 3310 Battery connections

So you have got this far! All we have left is to run our Nokia Phone from a power supply, no more recharging batteries! If you just connect a power supply to the B+ and B- terminals the phone will not work, as the phone needs to know the battery type and battery temperature. The battery temperature is detected by a simple thermistor connected between the thermistor pin and the 0V pin. Please see photo below. The thermistor is around 50K for ambient temperature. Because most Nokia phones can work with both Lithium Ion and Nickel Metal Hydride (Ni-MH), a battery type / sense pin is needed. A Ni-MH battery has a 5.6Kohm resistor to the 0V pin, where the Lithium Ion has a 74.5Kohm resistor.

Now we know how all this works, the best way to connect an external supply is to get an old Nokia battery and disassemble it. Then just connect your power supply on the B+ and B- pins, the rest is done for you. If you have ever worked with embedded GSM modems, you will be aware of the high peak currents involved when the phone is transmitting. Your power supply must be able to supply the current required by the phone with very little voltage drop. A regulator of 2Amps should be enough. Therefore keep your wiring very short, even better place the regulator in the battery holder and have very good, low ESR Tantalum capacitors near by. This is not for the beginner!


### Links

- **The Linux Gnokii Project - A linux based programme to send SMS from a linux box.**
  An Excellent source of information on the F-Bus protocols - A must for developing systems for F-Bus. Have a look at the doc/protocol/nk6110.txt file for commands
- **Nokia Datacables Page by Adrian Dabrowski - Great information of M-Bus Protocols**
- **European Telecommunications Standards Institute (ETSI) Telecom Standards**
- **ETSI GSM UMTS 3GPP Numbering Cross Reference Page - Get the standards for GSM Digital cellular telecommunications system**
  GSM 03.38 - Alphabets and language-specific information (Information on packing 7 bit characters in SMS messages)
  GSM 03.40 - Technical realization of the Short Message Service (SMS) Point-to-Point (PP)
- **Advanced Wireless Planet - Useful links and resources**
- **Nokia 61xx SMS send/receive functions by Justin Karneges - A special thanks to Andrew Dunn for showing me this one!**
- **LogoManager - A programme to send SMS, edit ringtones, graphics etc (For Microsoft Windows)**
- **Oxygen Phone Manager - A programme to send SMS, edit ringtones, graphics etc (For Microsoft Windows)**
- **Sysinternals Download the free PortMon.**
  This is a powerful port monitor written by Mark Russinovich that greatly helps in debugging serial and parallel port activity.
  Sysinternals have some very powerful utilities for debugging just about anything PC related. A must bookmark site!
- **Sales of Nokia Data Cables in Australia**

## Hardware Solutions

For all you people who can't be bothered to make your own, here's some you can buy now!

### Great SMS Project Ideas

- My favorite idea - A SMS based weather station. Just SMS it a blank message and it replies with current temperature, humidity, wind speed, direction and atmospheric pressure. Just think of the message costs!!
- A SMS based programmable logic controller (PLC). Gives you the freedom of turning pumps and mains equipment on and off by SMS. Every house and workplace needs one!! Air-cond on, Air-cond off.
- Add SMS to your alarm system - Reports sectors and type of alarm when tripped.
- A SMS based car GPS tracking system. Just SMS your car with the string "Tracking on" and it starts SMSing you the position of your car every minute etc. The ideal car alarm system. It can even SMS you when the alarm goes off.

Back to main page

---

*Last Updated: 9th July, 2010*