AN07 - Sending Arbitrary CAN Messages

We are often asked about how to send arbitrary CAN messages. Typically the user wants to send data bytes of their own choosing and does not want PCI bytes inserted for them, or to have receive bytes removed because the ELM IC assumed that they were ISO 15765 PCI (formatting) bytes. This application note shows how to send CAN messages of your own choosing, with our ELM327, ELM327L, ELM329 or ELM329L products (version 1.3 or newer).

The ELM327 and ELM329 integrated circuits have evolved over the years, and for this reason it is best to choose the following settings (even though it may not be necessary for your particular chip):

AT D, AT WS, or AT Z

If you are about to send a message, but are unsure of the state of the integrated circuit, it is always advisable to force a reset to the default settings. How you do this is typically through one of the above commands (although you should be aware that the AT WS and AT Z commands will close any protocols that you are currently set to).

AT SP

You may wish to choose a protocol, rather than have the ELM IC look for an active one for you. This is done with the AT SP command (for example AT SP 8, to set the IC for 11/250 CAN). Be careful if using the automatic search on fail option (eg. AT SP A8), as that may cause the chip to go off looking for a valid protocol if it does not like a particular response (or lack of one).

AT AL

This command really only applies to non-CAN protocols (and protocol 0, when searching). You should not normally need to use it, but for compatibility with older ELM327 ICs, it is best to send it once. Note that the ELM329 and ELM329L ICs do not support the AT AL command, and will show an error if presented with it.

AT CEA

Turn off CAN Extended Addressing when trying to send arbitrary messages (it is off by default), unless you are absolutely certain of what you are doing. If you enable extended addressing, it may limit the allowed number of bytes that the IC will send. The ELM327 v1.3 and v1.3a do not support the CEA commands.

AT CAF0

Turn off CAN automatic formatting so that PCI bytes are not inserted in the messages, or expected in the responses.

AT V1 or V0

Only you know how many bytes that you wish to send. ISO 15765 always sends 8 data bytes, inserting filler (padding) bytes as required in order to make a total of 8 bytes transmitted. It is the PCI byte that normally tells how many of the 8 data bytes are to be used. If you do not wish to have your message padded out with 00's (i.e. the byte as set by PP 26), then send AT V1, or ensure that the appropriate Programmable Parameter bit is set when defining your own protocol.

AT BI

Choosing a protocol does not immediately make it active (i.e. valid). The ELM ICs will normally look for a response to a particular request (typically this is 01 00 for ISO 15765) before considering a protocol to be valid. If a reply is not obtained, and the automatic search feature is enabled, the IC will likely start sending search messages using other protocols, perhaps finding one that might work, or stopping with an error message. If you do not want to chance this happening, send the AT BI command in order to bypass the initiation step, and accept the current protocol.

AT SH

If you are choosing your own data bytes to send, you may also want to choose your own ID bits (we call them header bytes). Since the header values are not likely to change as often as the data bytes, they are assigned with a separate AT SH command, and not with the data bytes. It is not necessary to assign your own header bytes, but if you do not then the default OBDII values will be used.

As an example, we will set an ELM327 to use protocol 8 for sending messages of 1 to 8 bytes in length. This requires typically sending the following setup commands at the prompt (we also show the ELM327 responses, so that you know what to expect):

```
>AT Z

ELM327 v2.2
```

```
>AT SP 8
OK

>AT AL
OK

>AT CAF0
OK

>AT V1
OK

>AT BI
OK
```

The chip is now ready to send a message of your choosing. First, assign the header (ID bits) for your message. We'll use 777:

```
>AT SH 777
OK
```

Now present the data bytes that you wish to send. We will provide these eight:

```
>11 22 33 44 55 66 77 88
NO DATA
```

The ELM327 responds with 'NO DATA' because we had no ECU reply with a message that matched the receive filters. You must define the receive ID bits for the ELM IC to look for, using either the CAN Receive Address command (AT CRA) or the CAN Filter/CAN Mask commands.

In order to not receive this NO DATA response while testing, we can turn off the ELM IC's search for a response. This is done by either sending AT R0 in the setup, or by providing a 'number of responses' digit after the data bytes. To send the previous message but retrieve no replies, we would send:

```
>11 22 33 44 55 66 77 88 0

>
```

which eliminates the NO DATA response. Note that in both of the above cases, the ELM327 transmits exactly the same CAN message:

```
777 11 22 33 44 55 66 77 88
```

For this configuration, the ELM327 will always send only the data bytes provided, without any extra filler bytes being added. For example, if you had only wanted to send the data bytes 11 22 33 44, then at the prompt, you would provide only those four bytes:

```
>11 22 33 44
```

In this case, the ELM327 actually sends the following:

```
777 11 22 33 44
```

for you, since we chose AT V1 to enable a variable data length. If we had chosen AT V0, then the message sent would have had eight data bytes, as follows:

```
777 11 22 33 44 00 00 00 00
```

The extra bytes are all 00's as the filler byte is 00 by default (it is set by PP 26).

Using the above as a guideline, you should now be able to configure the ELM327 (and ELM327L, ELM329 or ELM329L) for sending any arbitrary group of one to eight data bytes. Sending more than eight bytes is somewhat more complicated, but not impossible. It involves splitting the data into smaller chunks that each fit into one message, then sending a First Frame message, receiving a Flow Control message, and sending Consecutive Frames. This is beyond what the average user might require and is left to the advanced users to work out.