Chell Instruments Ltd
Folgate House
Folgate Road
North Walsham
Norfolk NR28 0AJ
ENGLAND

Tel: 01692 500555
Fax: 01692 500088

nanoDAQ-LT

Pressure Sensor Acquisition System

USER PROGRAMMING GUIDE

e-mail:- **info@chell.co.uk**

Visit the Chell website at:
http://www.chell.co.uk

900222-1.0

Please read this manual carefully before using the instrument.

⚠ **Use of this equipment in a manner not specified in this manual may impair the user's protection.**

Chell Document No. : 900222 Issue 1.0
ECO: ---- Date:4th January 2019

Contents

1. Introduction.

From power up, the nanoDAQ-LT reads its non volatile setup information and calibration, and after applying these settings is then ‘up and running’, reading the sensors and delivering calibrated data. Although for many applications this is enough, more demanding applications may need to control the data delivery, request data rezero operations and more.

The nanoDAQ-LT supports the same user interface protocol as the microDAQ pressure scanner system, allowing remote access to the essential commands required when integrating the unit into an instrumentation system. Commands may be sent over any of the unit’s communication channels, whether the current active data channel or not. A simple block parity check adds security to the command protocol, and a correctly received command may be acknowledged if required.

The following sets out the essentials of the command protocol, the data packet format for all channels in addition to the message identifier arrangement for the CAN channel.

This document version supports V1.1.8 of the nanoDAQ-LT firmware (and it’s 16 channel variants – -LTS, -LTM, -LTR). If using earlier firmware then some user commands and/or parameters may be invalid.

2. User Command Protocol.

2.1 Command Packet.

The command protocol is based around a simple delimited control packet, allowing easy identification of command start and end. The packet includes a block parity byte increasing the robustness of transmission; the packet format is shown in Figure 2.1.

> (ASCII 62)	Command	Parameter	Parity	< (ASCII 60)

Figure 2.1 Command Frame Format.

The command byte values are defined in the following section, and may or may not require a parameter byte, for example data rate. The parity byte is an even block parity of all bytes (other than itself), including the delimiters. Calculation for parity bit *n* is therefore the sum of each bit *n* using modulo 2 arithmetic.

For a command not requiring a parameter, an arbitrary dummy byte should be used. This can be any value as the number is not processed other than in determining the parity of the command packet.

2.2 Acknowledgement.

A command packet is positively acknowledged if it is correctly formatted and the parity byte is correct. A positive acknowledgement is denoted by the transmission of ASCII 42 ('*'), a negative acknowledge indicating an invalid command by ASCII 33 ('!'). Recognised command values are then acted upon, though unrecognised values are discarded.

2.3 User Commands.

The available user command set is summarised in Figure 2.2.

Command	**Byte, Char/ ASCII**	**Parameter**	**Description**
Standby	S/83	None	Set all data streaming off.
Reset	R/82	None	Request a soft device reset – nanoDAQ-LT is reinitialised.
Rezero	Z/90	None	Request a rezero.

Rate	V/86	byte = 0xab
		
		a = 4 TCP / UDP
		a = 8 CAN
		
		b = 0 : Off
		b = 1 to 6 : Invalid
		b = 7 : 200 Hz
		b = 8 : 150 Hz
		b = 9 : 100 Hz
		b = 10 : 50 Hz
		b = 11 : 25 Hz
		b = 12 : 20 Hz
		b = 13 : 10 Hz
		b = 14 : 5 Hz
		b = 15 : 1 Hz
			Adjust the data delivery rate for each communication channel. The upper four bits of the parameter byte selects which communication channel, while the lower four bits selects its data rate.
			
			Note that the numbering used is the same as the microDAQ system that it is based upon, for historical reasons. Selecting a number not displayed could have unexpected results.
			
			It should be noted that the available data rates are based upon the selected oversampling rate as indicated in the nanoDAQ-LT operating manual. i.e. greater oversampling will limit the available rates.

Protocol	P/80	byte = 0xab
		
		a = 1 TCP / UDP
		a = 2 CAN
		
		b = 0 - 16 bit LE
		b = 1 - 16 bit BE
		b = 2 -
		Engineering Units
			Permits the changing of the data protocol. The abbreviations LE and BE in the table refer to little and big endian respectively, denoting whether the less significant byte is sent first or last.
			
			Pressure data are available in engineering units over the TCP connection, though only binary data are available with CAN.
			
			Binary data provides pressure scaled as unsigned integers to the operating full scale, eg for 16 bit resolution, 0 to 65535 represents -FSD to +FSD for the differential pressure type.

Stream ON	1/49	1 - TCP / UDP 2 - CAN	Enable the data delivery for the chosen channel. The delivery rate is as selected in webserver, unless it is modified via the Rate command.
Stream OFF	0/48	1 - TCP / UDP 2 - CAN	Disable the data delivery for the chosen channel.
Get Status	?/63	0 : Short 1 : With temp. 2 : Full 3 : Pressure reading 4 : Temp. readings *5 : Reserved* *6 : Reserved* 7 : Firmware ID 8 : Unit serial number *9 : Reserved*	Return a status packet from the nanoDAQ. Three main versions are available, '*short*', '*with temp*.' and '*full*'. Short returns status byte information showing current operating state only, whereas '*full*' returns webserver options and temperature data in addition. '*With temp*.' returns the status and temperature information intended for continuous temperature monitoring applications. The data format is documented in a separate section. Additional readings poll raw values for temperature and pressure as read from the sensors, as well as other fields as indicated in the table.
Poll	O/79	1 - TCP / UDP 2 – CAN	Request a single data packet (in the current active format) for the channel selected. Data streaming should be set off before using this command. Note that there is no positive acknowledge for this command.

Hardware Trigger	T/84	byte = 0xab
		
		a = 0 Disable
		a = 1 Enable
		
		b = 1 TCP / UDP
		b = 2 CAN
			Enables or disables the hardware trigger on the selected channel. Note that there is no positive acknowledge for this command.

Datastream Timestamp	t/116	0 – None 1 – Start of Cycle 2 – Every Channel	Allows timestamping to be added/removed from the datastream. *'Start of Cycle'* sets a timestamp in front of channel 1 only (to indicate when the first channel of a cycle was acquired). *'Every Channel'* sets a timestamp in front of every channel. This command is only relevant for the TCP/UDP data stream.

Figure 2.2, The Available User Command Set for the nanoDAQ-LT.

3. Status Data Format.

Status data is not currently supported over the CAN channel, however for TCP connections, status data may be requested from the nanoDAQ-LT as three main forms – ‘short’, ‘full’ and ‘with temp’. The short form returns 4 bytes, the 16 bit status word delimited by the ASCII characters ">" and "<", the less significant byte is returned before the more significant. The bit assignment is shown in fig 3.1.

bit 15 8

			PTP sync. active		HW Trigger Error detected		Harware Trigger active
		CAN stream active	TCP stream active				Rezero in progress

7 bit 0

Figure 3.1, Status word bit assignment.

Requesting the status 'with temp.' returns the short status 4 bytes, followed by the temperature data. Temperatures for all active channels are returned in ascending order as comma delimited ASCII engineering units (ie degrees C).

The 'full' status data contains the above, followed by fields for the setup options of the nanoDAQ-LT. Each field is comma delimited, and its function is indicated in plain text between square parentheses. On/off is indicated by 1/0. An example 'full' status string is shown in figure 3.2.

```
>@.<,19.88,20.01,20.07,20.23,20.25,20.35,20.37,20.28,20.19,20.26,20.33,20.37,20.
33,20.32,20.18,20.16,[Serial] 1810801,[Full scale] 2.50000000,[Active channels]
16,[CAN channels] 16,[TCP channels] 16,[CAN rate] OFF,[TCP rate] OFF,[CAN
message] Multiple,[CAN protocol] 16 LE,[TCP protocol] 16 LE,[Press. input
impulse] 0,[Press. input power] 4,[IP] 192.168.3.190,[Mask] 255.255.0.0,[Gateway]
0.0.0.0,[CAN timing] (BRP) 4 (TSEG1) 11 (TSEG2) 4 (SJW) 3,[CAN message]
100,[IENA key] 0x3101,[IENA end word] 0xDEAD,[Ethernet power] Auto,[CAN
power] Auto,[Press. units] psi,[Press. type] Differential,[PTP sync] Off,[Stream
timestamp] None,[Time format] UTC,
```

Figure 3.2, Example full status information.

4. nanoDAQ-LT Communication Channels.

4.1 TCP

4.1.1 Overview.

The nanoDAQ-LT's TCP channel affords it the ability to stream real time pressure data at high speed over standard 100Mbit Ethernet connections.

4.1.2 Connection.

When using the TCP/IP channel, the nanoDAQ-LT is programmed to listen on its local port number 101. It will respond to a connection request, connect and immediately start streaming data if it has been setup to use the TCP channel. Note that the nanoDAQ-LT only supports one TCP connection.

Setup requires the nanoDAQ-LT to be given an IP address, and the network's subnet mask. It is possible either to run it over a local Ethernet connection, or directly from a PC's network card, as long as a crossover cable is used. A dual network card (or 2 cards) may be used, however care should be taken about network routing, as similar subnets for two network connections on a single Windows ® machine can cause the nanoDAQ-LT not to be seen on the second card. The nanoDAQ-LT will respond to a 'ping' instruction for the purposes of setup and troubleshooting.

4.1.3 TCP Protocol

With TCP channel selected, calibrated pressure data are streamed continuously over the TCP interface. TCP supports 3 different data formats as shown in Figure 4.1. The binary data formats scaling are dependant on the Pressure Type setting. For the Differential setting, the binary 0 to 65535 represents -/+ full scale respectively, whereas for the Absolute setting the binary 0 to 65535 represents 15000Pa (approx. 2.175psia) to 115000Pa (approx. 16.679psia). Binary protocols require less communications bandwidth as well as less processor overhead within the nanoDAQ-LT; it is recommended that engineering unit conversions be applied at the client.

The engineering unit protocol should be avoided if possible, due to the need to declock the nanoDAQ-LT's internal clock and scanner addressing to half the chosen scanner acquisition frequency from the chosen scanner acquisition frequency (due to internal issues regarding string handling). The 16 bit little ended protocol is most efficient from the nanoDAQ-LT's viewpoint, the data being calibrated and spanned to 0 to 65535. Scaling to floating point full scale is better achieved within a PC client application where there is more processing power available. Figure 4.1 shows the available data packet formats over TCP.

Protocol	**Example data format**	**Note**
16 bit LE	0x00 0xFF 0x00 CH1LSB CH1MSB CH2LSB CH2MSB CH3LSB…..CH16LSB CH16MSB	3 byte (00, FF, 00) header identifies start of packet. All active channels (CH1 to CH16) then sent LSB first, no delimiters.
16 bit BE	0x00 0xFF 0x00 CH1MSB CH1LSB CH2MSB CH2LSB CH3MSB….. CH16MSB CH16LSB	16 bit data, 3 byte header, no delimiters, MSB first.
Eng. Units	*,#.#####,#.#####,#.#####,#.#####,#.#####,#.#####……..#.#####	ASCII readable string, * (ASCII 42) as header, active channels in ascending order, 5 decimal places, comma delimited.

Figure 4.1, Data Packet Protocol, TCP Channel.

4.1.4 TCP Data Rate.

The data delivery rate is selected from the webserver, and the following values are available (Hz) – 1, 5, 10, 20, 25, 50, 100, 150, 200. Although the system endeavours to deliver the rate with maximum accuracy, ultimate responsibility for data timing lies with the user's host system.

The available data rates are also dependant on the oversampling setting in the nanoDAQ-LT. The more oversampling that is set, the lower the maximum data rate is available. Figure 4.2 shows the oversampling settings and the subsequent maximum data rate allowed.

Oversampling setting	Maximum data rate available
High Speed	200 Hz
Low Resolution	150 Hz
Standard Resolution	100 Hz
High Resolution	100 Hz
Ultra-High Resolution	50 Hz

Figure 4.2, Oversampling vs Max Data Rate.

The data rate over TCP is vulnerable to low level operating system considerations, in particular any 'delayed acknowledgement' algorithm. Windows® attempts to suppress too many acknowledgments for small data packets swamping a network by inserting a 200ms (default) delay in the generation of a second acknowledgement to a communicating device. Since the nanoDAQ-LT's communication consists of many small packets at a high repeat rate, this algorithm has a catastrophic effect on the nanoDAQ-LT's delivered bandwidth (effectively limiting it to tens of Hz).
The bottleneck can be removed by altering/adding a value of the registry key: HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{*adapter GUID*}
(HKLM = HKEY_LOCAL_MACHINE; {*adapter GUID* = the network adapter being used by the Windows PC), though this should be done only in consultation with the network administrator.
In Windows 2000, the DWORD value TcpDelAckTicks should be set to 0 in the registry key.
In Windows XP (must have SP1 or later installed) and Windows 7/8/10, the DWORD value TcpAckFrequency should be set to 1 in the registry key.

Also note that the TCP channel of the nanoDAQ-LT is subject to buffering both within the unit itself (the size of the buffer depending on number of channels and data rate), and within Windows ® itself. At low data rates, it is possible to receive single complete data packets, however as the data rate increases Windows ® is more likely to deliver chunks of data 2k to 4k bytes in length, with arbitrary boundaries with respect to nanoDAQ-LT's data packets. User software should take this into account to avoid losing data.

Please note that if streaming at high data rates it is essential that a good network infrastructure is used. It is recommended that any streaming is performed over a 'private' network (i.e. disconnected from any corporate network structure) to reduce the number of packets flying around the network. It is also highly recommended that a high speed managed network switch with a large store and forward buffer is used between the nanoDAQ-LT and client PC, particularly if several nanoDAQ-LTs are being used together to acquire data.

4.1.5 Control Via TCP.

A positive acknowledgement is returned as '**' and a negative acknowledgement as '!!'.

The argument regarding loss of acknowledgements in the data stream holds good for the TCP channel, and it is recommended that a 'Stream Off' or 'Standby' is sent before any other command.

4.2 UDP

4.2.1 Overview.

As with TCP the nanoDAQ-LT's UDP channel affords it the ability to stream real time pressure data at high speed over standard 100Mbit Ethernet connections.
The nanoDAQ-LT supports two data stream formats, the Chell data stream format and an IENA specification output which can be setup from the web server.

4.2.2 Connection.

When using the UDP/IP channel, the nanoDAQ-LT is programmed to listen on its local port number 101. It will respond to a connection request, connect and immediately start streaming data if it has been setup to use the UDP channel with remote IP address and port configured in the settings.

Setup requires the nanoDAQ-LT to be given an IP address, and the network's subnet mask - these will be the same as for the TCP - however in addition to those, UDP also needs a remote IP address and remote port number. As with TCP, it is possible either to run the nanoDAQ-LT over a local Ethernet connection, or directly from a PC's network card, as long as a crossover cable is used. A dual network card (or 2 cards) may be used, however care should be taken about network routing, as similar subnets for two network connections on a single Windows ® machine can cause the nanoDAQ-LT not to be seen on the second card.

4.2.3 Chell UDP Protocol

With UDP channel selected, calibrated pressure data are streamed continuously over the UDP interface. UDP supports 3 different data formats, the binary options are shown in Figure 4.3. As with TCP, the binary data scaling is dependant on the Pressure Type setting (see section 4.1.3).
The engineering unit protocol is also supported with the same packet format as with TCP. The system effects & points highlighted regarding the engineering unit protocol on TCP also apply for UDP (again, detailed in section 4.1.3).

Protocol	Example data format	Note
16 bit LE	nanoDAQ-LT_serial Packet_Num CH1LSB CH1MSB CH2LSB CH2MSB …..CH16LSB CH16MSB	32 bit representation of serial number identifies start of the packet then a 32 bit packet number, autoincrementing with each packet. All active channels (CH1 to CH16) then sent LSB first, no delimiters.
16 bit BE	nanoDAQ-LT_serial Packet_Num CH1MSB CH1LSB CH2MSB CH2LSB ….. CH16MSB CH16LSB	32 bit representation of serial number identifies start of the packet then a 32 bit packet number, autoincrementing with each packet. 16 bit data, no delimiters, MSB first.
Eng. Units	*,#.#####,#.#####,……..#.#####	ASCII readable string, same as TCP.

Figure 4.3, Data Packet Protocol, UDP Channel.

4.2.4 UDP Data Rate.

The UDP data rate is much the same as the TCP data rate in its configuration and use, so all the TCP data rate information is applicable for UDP.

4.2.5 Control Via UDP.

A positive acknowledgement is returned as '**' and a negative acknowledgement as '!!'.

The argument regarding loss of acknowledgements in the data stream holds good for the UDP channel, and it is recommended that a 'Stream Off' or 'Standby' is sent before any other command.

4.2.6 IENA specification

The nanoDAQ-LT supports the option to output data in the IENA specification (aerospace test spec.) data packet format. This specialised format arranges the data in a specific manor containing various different information.

Data packets formatted in this way start with a specific header format before any data in the packets.

Key	Size	Time			Status	Seq	Data 0	-	Data 15	Temperature	Scanner Status	End
16 Bits	16 bits	16 bit	16 bit	16 bit	16 bits	16 bits	32 bits	-	32 bits	32 bits	16 bits	16 bits
	Frame size	Time in microseconds			Set at 0x0000	Rolling 16 bit counter	Byte order 3-2-1-0		Byte order 3-2-1-0	Byte order 3-2-1-0	See description below	0x DEAD

Starting with the 16 bit key field, this key tells the user information about this packet, in this case the key is created so that the most significant 4bits is the manufacturers ID, the next 4 bits is the ID of the device (2 for nanoDAQ-LT), the 8 bits denotes which IENA datastream this data is from – on the Chell data acquisition family of devices this is always 1.

The next 16 bits of the data packet is the size of the packet, this is the total number of bytes in the IENA header and the data blocks.

The next section of data in the data packet is the time, this is 48 bits long and contains the time that has elapsed in microseconds from the 1st of January of the current year. For this to work accurately IEEE 1588 has to have been enabled and the device has to be synchronised with a grandmaster. It is also possible for the user to get the time from a PC on the embedded webserver, this is however less accurate than using a IEEE 1588 synchronised time.

The next 16 bits is a reserved STATUS word with a value fixed at 0x0000.

Following this is a rolling 16 bit sequence number that counts how many packets have been sent in this data stream. This is a rolling counter, so if the packet sent is greater than the maximum value this field can hold then it will reset to 0 and start counting again.

After the header is finished the next part of the packet is the main data in channel order, this is 32 bit floating point big endian data. There is one 32 bit value for each channel of data.

The data is followed by a single 32 bit big endian scanner temperature value.

Note the byte ordering is currently always big endian and as such will ignore any data protocol format setting chosen on the webserver.

The scanner status is a 16 bit field of which is used to show various details about the scanner and device. The least significant bit is currently reserved and will always be set to 0, the second bit will show if the device has been time synchronised and the third bit will denote if the device has been IEEE 1588 synchronised or not. Further bits are reserved for future use and are set as 0.

The final 16 bits are an end field, the default value of this being 0xDEAD – this signals that all data has been transmitted. (Note that this value is configurable in the webserver, but typically should be left as the default setting).

4.3 CAN

4.3.1 Overview.

The CAN channel is somewhat different to the above channels, in that it is not a simple serial communications channel, the data being sent in discrete chunks on specified message identifiers.

4.3.2 CAN Baudrate.

The nanoDAQ-LT offers a single 'standard' CAN bus connection running at a selectable baudrate, the user having access to the values used for the nanoDAQ's microcontroller CAN timing registers to enable customising of sample point and jump width. The values should be calculated based on the microcontroller CAN peripheral clock of 90MHz and users should bear this in mind when calculating suitable values for BRP, TSEG1, TSEG2 and SJW for their own physical network implementation. Factory defaults set the CAN baud rate to 1MHz with a sample point of 72%

4.3.3 CAN Protocol.

Two separate protocols are available from the setup – either multiple or single message. For the former, the nanoDAQ-LT is allocated a fixed message for each group of 4 pressure channels, ie for a 16 channel scanner, 4 discrete CAN message identifiers are required. Alternatively for a more economical use of identifiers within a system, a single message id may be used, with all channels sent over this message sequentially. Channel numbers are positively identified with a channel identifier byte within the message.

For the multiple message option, data are sent on 8 byte messages, 4 channels per message – the pressure channels associated with a particular message number being fixed. The 3 digit message identifier is user selectable via the webserver. The most significant digit can be configured between 0-7 hex, the 2nd digit between 0-F and the least significant at fixed settings of 0, 4, 8, or C – so for example for a setting of 0x220, the identifier for channels 1-4 would be 0x220, and the identifiers for channels 5-8, 9-12 etc. follow on incrementally from this first identifier as shown in Figure 4.4. It is the user's responsibility to avoid overlap of message identifiers in a multiple nanoDAQ installation.

Data are two bytes binary unsigned and again the scaling is dependant on the Pressure Type setting. Differential gives a scaling of +/- full scale, ie 0x0000 might represent –2.5psi, 0xffff +2.5psi. Absolute gives a scaling of 15000Pa to 115000Pa (as described in TCP/UDP previously).

The data are user selectable as 16 bit big or little ended. Use of the 16 bit little ended option is recommended as it requires least processor overhead within the nanoDAQ-LT unit.

Note that there is a configurable setting within the Advanced section of the webserver, to show the reference pressure in the CAN data stream. This currently exists for Multiple message mode only and adds an extra CAN message after the channel data messages which contains the reference pressure (16bit encoded – LE or BE as with channel data, but always scaled as Absolute irrespective of Pressure Type setting – so scaling is 15000Pa to 115000Pa for 0 – 65535 binary) and also the device temperature (16bit integer representing 100ths of a degree – e.g. a value of 2405 = 24.05 degC …Note values > 32767 will represent a twos compliment encoding of a negative temperature; e.g. 63131 = -2405 = -24.05 degC). The firmware version is also included in this message as three integers.

Figure 4.4 shows an example Multiple message packet, including the additional message for the reference pressure, temperature, etc. Note that it is the users responsibility to ensure that CAN message ID's do not overlap, particularly when using this additional configurable setting.

The single message identifier option is included to reduce the number of message id's required within a system. Data are packed in 7 byte messages as 3 channels per message with one channel identifier byte. This identifier byte is incremented for each message in the sequence, starting at 0x00 for channels 1,2 3, then 0x01 for channels 4,5,6 etc. Odd leftover channels (eg 16 channels/3 = 5 full messages plus a message with a single channel and two ‘leftovers’) are set to 0x0000 and should be discarded. Figure 4.5 shows the structure of a message for the single message identifier protocol. A selectable inter message delay of between 1ms and 200ms is available to match message generation timing to the user’s system. Note that it is the user’s responsibility to select appropriate delays with respect to channel rate and CAN bus baudrate.

	Message ID					
Data Byte	**0x220**	**0x221**	**0x222**	**0x223**		***0x224***
7	CH4 MSB	CH8 MSB	CH12 MSB	CH16 MSB		
6	CH4 LSB	CH8 LSB	CH12 LSB	CH16 LSB		*F/W Minor Revision*
5	CH3 MSB	CH7 MSB	CH11 MSB	CH15 MSB		*F/W Minor Version*
4	CH3 LSB	CH7 LSB	CH11 LSB	CH15 LSB		*F/W Major Version*
3	CH2 MSB	CH6 MSB	CH10 MSB	CH14 MSB		*Temp MSB*
2	CH2 LSB	CH6 LSB	CH10 LSB	CH14 LSB		*Temp LSB*
1	CH1 MSB	CH5 MSB	CH9 MSB	CH13 MSB		*Ref MSB*
0	Ch1 LSB	Ch5 LSB	Ch9 LSB	Ch13 LSB		*Ref LSB*

Figure 4.4, Example of CAN multiple message packing using the 16 bit little ended data protocol – base identifier 0x220, with reference pressure setting enabled, to include in data stream as extra message (ID 0x224)

Data Byte	Content
6	CH3 - MSB
5	CH3 - LSB
4	CH2 - MSB
3	CH2 - LSB
2	CH1 - MSB
1	CH1 - LSB
0	0x00

Figure 4.5, Example of the CAN single message protocol using 16 bit little ended data– first message (ie channels 1,2,3)

4.3.4 CAN Data Rate.

The data delivery rate is selected from the setup program, and the following values are available (Hz) – 1, 5, 10, 20, 25, 50, 100, 150, 200. Although the system endeavours to deliver the rate with maximum accuracy, ultimate responsibility for data timing lies with the user's host system.

As with TCP & UDP, the available data rates are also dependant on the oversampling setting in the nanoDAQ-LT. The more oversampling that is set, the lower the maximum data rate is available. Figure 4.2 in the TCP section earlier, shows the oversampling settings and the subsequent maximum data rate allowed.

4.3.5 Control Via CAN.

The user command set is supported over the nanoDAQ-LT's CAN channel. The specification and function of the commands are detailed in section 2 above, however the CAN implementation is as follows.

The incoming message number is selected by the user from the front end software, though is constrained to be relative to the outgoing data base message identifier. The offset from the base identifier may be selected as being +0x10, +0x20, +0x30, +0x40 or +0x50. For example the base data message identifier of 0x220 might be set up to receive commands over CAN on message 0x230 (0x220 + 0x10).

In addition to the incoming message offset, the user may select whether the incoming command is acknowledged or not. The user command is a delimited 5 byte message that includes a block parity check, as shown in Figure 4.6. If the command is received without detected error, and the acknowledge option has been selected, nanoDAQ-LT will respond to a user command with a positive acknowledge byte ('*' or ASCII 42). Alternatively, if the command is received incorrectly it will respond with the negative acknowlege byte ('!' or ASCII 33). The acknowledgement is sent as a single byte message with identifier one greater than the receiving message. For the above example, the acknowledge will be sent on message 0x231.

Data Byte	Content
4	< (ASCII 60)
3	Parity
2	Parameter
1	Command
0	> (ASCII 62)

Figure 4.6, Data Structure of the CAN Incoming Control Message.