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

Tel: 01692 500555
Fax: 01692 500088

nanoDAQ-LTC

Pressure Sensor Acquisition System

USER PROGRAMMING GUIDE

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

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

900230-1.0

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. : 900230 Issue 1.0
ECO: ---- Date: 21st January 2020

Chell's policy of continuously updating and improving products means that this manual may contain minor differences in specification & functionality from the actual instrument supplied.

Contents

1. Introduction.

From power up, the nanoDAQ-LTC 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-LTC supports the same user interface protocol as the nanoDAQ-LT, allowing remote access to the essential commands required when integrating the unit into an instrumentation system. A major difference to the LTC over the original LT variant is that commands can only be sent over the CAN communications channel as this is the only form of communications outside of factory level debugging. 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 and the message identifier arrangement for the CAN channel.

This document version supports V1.0.0 of the nanoDAQ-LTC firmware.

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.

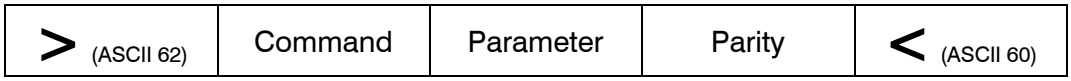

Figure 2.1 Command Frame Format.

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.

Some commands are used to configure settings within the unit and these settings can also be read by setting the most significant bit in the command byte (known from now on as the read bit).

E.g. To set the streaming rate, the command byte is ASCII 86 (0x56). To read the current streaming rate, the command byte is ASCII 214 (0xD6)

2.2 Acknowledgement.

A command packet is positively acknowledged if it is correctly formatted and the parity byte is correct.

Response Byte 1	Response Byte 2	Acknowledge Byte

Figure 2.2 Acknowledgement Frame Format.

The acknowledgment packet consists of 3 bytes. If a command has been issued with the read bit set, then the first two bytes will contain an appropriate response, depending on the command issued. In all other instances, these two bytes will be 0x00. The third byte is the acknowledgement byte which is either ASCII 42 ('*') for a positive acknowledgement, or ASCII 33 ('!') for a negative acknowledgement, indicating an invalid command parameter or parity error (or a read bit set on a command that is an action and has no value to read back).

2.3 User Commands.

The available user command set is summarised in Figure 2.3. The R/W field indicates those commands that configure settings that can be read as well (by setting the read bit, as discussed previously).

Please note that any change of settings will not be remembered after a power cycle on the nanoDAQ-LTC, unless the Burn To EEPROM command (ASCII 101) is issued prior to the power cycle.

Also note that some settings will not be in effect until after a power cycle & EEPROM save (e.g. CAN ID changes).

Command	Byte, Char/ ASCII	R/W	Parameter	Description
Standby	S/83		None	Set all data streaming off.
Reset	R/82		None	Request a soft device reset
Rezero	Z/90		None	Request a rezero.
Rate	V/86	R/W	byte = 0xab a = 2 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 the communication channel. Historically, the upper four bits of the parameter byte selects which communication channel, while the lower four bits selects its data rate. In the case of the nanoDAQ-LTC, the comms are CAN only, so the upper nibble is always 2. 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 will be ignored and negatively acknowledged. It should be noted that the available data rates are based upon the selected oversampling rate as indicated in section 3.14 below. Selecting a rate that is out of range for the chosen oversampling setting will cause the command to be negatively acknowledged. Setting a non-zero rate and burning it to EEPROM will result in auto data streaming output when the unit is power cycled.
Input Filters	F/70	R/W	0 – Off 1 - 2 2 - 4 3 - 8 4 - 16 5 - 32 6 - 64 7 - 128 8 - 256 9 - 512 10 - 1024 11 - 2048 12 - 4096 13 - 8192 14 - 16384 15 - 32768 16 – 65536 +128 – Enable Impulse	Adjust the input filter settings. The first five bits indicate the number of samples of the input moving average filter. The most significant bit can be set (+128) to enable the input impulse filter (median)
Protocol	P/80	R/W	byte = 0xab a = 2 CAN b = 0 – 16 bit LE b = 1 – 16 bit BE	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.
Stream ON	1/49		2 – CAN	Enable the data streaming. The delivery rate is as configured by the Rate command.

Stream OFF	0/48		2 – CAN	Disable the data streaming
Poll	O/79		2 – CAN	Request a single data packet, in the current active protocol format. Data streaming should be set off before using this command. Note that there is no positive acknowledge for this command.
Pressure Type	a/97	R/W	0 – Absolute 1 - Differential	Set the pressure type for data output streaming.
Reference Channel	K/75	R/W	0 – No Reference Channel 1-16 – Channel to be used as reference.	Select the sensor channel to use as a reference channel. Required for Differential pressure readings.
Sensor Oversampling	G/71	R/W	0 – High Speed 1 – Low Resolution 2 – Standard Resolution 3 – High Resolution 4 – Ultra High Resolution	Set the oversampling setting of all sensors. Changes to this will affect the maximum available data streaming rates (see 3.1.4 below for more details)
Base CAN ID (LSB)	c/99	R/W	byte = 0xab a = 0 to F b = 0, 4, 8 or C	Set the LSB of the Base CAN ID
Base CAN ID (MSB)	d/100	R/W	Byte = 0x0a a = 0 to 7	Set the MSB of the Base CAN ID
Ref CAN ID (LSB)	r/114	R/W	byte = 0x00 to 0xFF	Set the LSB of the Reference CAN ID
Ref CAN ID (MSB)	s/115	R/W	Byte = 0x0a a = 0 to 7	Set the MSB of the Reference CAN ID
CAN Message Scheme	v/118	R/W	0 – Multiple messages 1 – Single message, dynamic delay 2 – Single message, 1ms delay 3 – Single message, 2ms delay 4 – Single message, 3ms delay 5 – Single message, 4ms delay 6 – Single message, 5ms delay 7 – Single message, 10ms delay 8 – Single message, 15ms delay 9 – Single message, 20ms delay 10 – Single message, 25ms delay 11 – Single message, 50ms delay 12 – Single message, 100ms delay 13 – Single message, 150ms delay	Set the CAN Message scheme Multiple messages fits four channels of data into a message and increments the CAN ID for subsequent channel blocks Single message fits three channels of data into a message (one CAN ID) with an incremental channel packet counter. See section 3.1.3 below for more detail.
Burn to EEPROM	e/101		None	Burns the setup block to EEPROM. Any of the user commands that change the current settings (i.e. filters, data rate, data protocol, CAN IDs) will be stored to EEPROM. Note that some settings will require a system reset to become active (e.g. CAN IDs)

Figure 2.3, The Available User Command Set for the nanoDAQ-LTC.

3. nanoDAQ-LTC Communication Channel.

3.1 CAN

3.1.1 Overview.

The nanoDAQ-LTC only has CAN as a valid comms channel for data streaming and command control, but its use is the same as the nanoDAQ-LT devices it is based upon.

3.1.2 CAN Baudrate.

The nanoDAQ-LTC 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%

3.1.3 CAN Protocol.

Two separate protocols are available from the setup – either multiple or single message. For the former, the nanoDAQ-LTC 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. The default protocol for the nanoDAQ-LTC is single 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 an appropriate user command. 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 3.2. It is the user's responsibility to avoid overlap of message identifiers in a multiple nanoDAQ installation.

Data are two bytes binary unsigned with the scaling 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 repesentation is dependant on the full scale setting as follows (**bold** indicates default for nanoDAQ-LTC):

Range	Absolute scaling
< 5 psi (< 345 mbar)	150 to 1150 mbar
5 psi (345 mbar)	**0 to 1310.72 mbar**
> 5 psi (> 345 mbar)	130 to 1600 mbar

Figure 3.1, Absolute scaling range

The data are user selectable as 16 bit big or little ended.

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. Figure 3.3 shows the structure of a message for the single message identifier protocol. In general, odd leftover channels (eg 32 channels/3 = 10 full messages plus a message with the two final channels and one 'leftover') are set to 0x0000 and should be discarded.

The nanoDAQ-LTC provides an inter message delay when using the single message protocol. By default, this delay is dynamic, meaning that it is calculated based on the configured CAN data rate so that messages are consistently sent with the same delay between each. E.g. if the data rate is 1Hz then the 6 messages required for all 16 channels are sent with 166.7ms delay between them. If the data rate is increased to 10Hz then the delay becomes 16.67ms between messages.

As with the nanoDAQ-LT there are also a number of fixed delays (between 1ms & 200ms) that can be configured instead, if required. Note that in this instance, 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**
7	CH4 MSB	CH8 MSB	CH12 MSB	CH16 MSB
6	CH4 LSB	CH8 LSB	CH12 LSB	CH16 LSB
5	CH3 MSB	CH7 MSB	CH11 MSB	CH15 MSB
4	CH3 LSB	CH7 LSB	CH11 LSB	CH15 LSB
3	CH2 MSB	CH6 MSB	CH10 MSB	CH14 MSB
2	CH2 LSB	CH6 LSB	CH10 LSB	CH14 LSB
1	CH1 MSB	CH5 MSB	CH9 MSB	CH13 MSB
0	Ch1 LSB	Ch5 LSB	Ch9 LSB	Ch13 LSB

Figure 3.2, Example of CAN multiple message packing using the 16 bit little ended data protocol – base identifier 0x220

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

Figure 3.3, Example of the CAN single message protocol using 16 bit little ended data– first message (ie channels 1,2,3)

3.1.4 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.

The available data rates are also dependant on the oversampling setting in the nanoDAQ-LTC. The more oversampling that is set, the lower the maximum data rate is available. Figure 3.4 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 3.4, Oversampling vs Max Data Rate.

3.1.5 Control Via CAN.

The specification and function of the commands are detailed in Section 2 above, the CAN implementation is as follows.

Currently the incoming message identifier is fixed to 0x590 and always acknowledges on identifier 0x591. This is currently not user configurable The user command is a delimited 5 byte message that includes a block parity check, as shown in Figure 3.5. The acknowledgement packet is 3 bytes as shown in Figure 3.6 – where a response is not valid or required, both response bytes will be 0x00.

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

Figure 3.5, Data Structure of the CAN Incoming Control Message.

Data Byte	Content
2	Ack ('*' or '!')
1	Response byte 2
0	Response byte 1

Figure 3.6, Data Structure of the CAN Outgoing Control Acknowledge Message.

3.1.6 CAN Status Message.

The nanoDAQ-LTC also has an additional multiplexed CAN status message that is sent every 500ms, irrespective of any settings for CAN data rate, etc. The message identifier for this is configurable with 2 user commands as detailed in section 2 above. Note that it is the user's responsibility to ensure that this identifier does not clash with any other identifiers based on the nanoDAQ-LTCs configuration.

Currently this status message is 8 bytes and cycles through three sets of details as shown in Figure 3.6 below. Byte 0 is the sequence identifier which increments from 0x00 to 0x02 on subsequent message transmissions:

Data Byte	1st Message	2nd Message	3rd Message
7	Current Data Streaming Rate	*unused*	*unused*
6	Range Index (see Fig.3.7 below)	*unused*	*unused*
5	Hardware Revision Number ('10' = V1.0)	*unused*	*unused*
4	Firmware Minor Revision Number	Unit serial number (32bit integer encoded)	Life Counter (8 bit incremental counter)
3	Firmware Minor Version Number		Diagnostics Value bitfield (*1)
2	Firmware Major Version Number		Diagnostics Type bitfield (*1)
1	*unused*		Unit temperature (to nearest degC)
0	0x00	0x01	0x02

Figure 3.6, Structure of the CAN Status Message.

Absolute Range	Range Index
150 mbar to 1150 mbar	0x00
0 mbar to 1310.72 mbar	0x01
130 mbar to 1600 mbar	0x02

Figure 3.7, Range Index Byte

(*1) The Diagnostics Type & Value bitfields allow for indication of any system problems that may be occurring at that time, The type field indicates the part of the system that the problem relates to (currently this is only 0x01 for CAN diagnostics, but allows for more to be added in future firmware releases). The value field indicates the problem – currently this is 0x01 for a CAN Passive error and 0x02 for a CAN Bus Off state.

If there are no problems to report, these bitfields will be set to 0x00.