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

Tel: 01692 500555
Fax: 01692 500088

MicroDaq-8

Pressure Scanner Acquisition System

USER PROGRAMMING GUIDE

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

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

900236-1.0

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. 900236 : Issue 1.0
ECO: 3472 Date: 23rd June 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 MicroDaq-8 reads its non volatile setup information and calibration, and after applying these settings is then ‘up and running’, reading the connected scanners 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 MicroDaq-8 has the same user interface protocol taken from the microDAQ-Mk2 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 V1.0.2 of the MicroDaq-8 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.

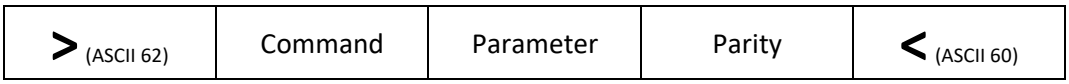

Figure 2.1 Command Frame Format.

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.

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 – MicroDaq-8 is reinitialised.
Rezero	Z/90	1-8 – Scanner to rezero 255 – Rezero All	Request a rezero, either for an individual scanner, or for all connected scanners
Derange	D/68	None	Calibration is rebuilt with the appropriate deranging factor applied, the DTC scanner deranging gain is switched into circuit.
Rebuild Calibration	C/67	1-8 – Scanner to rebuild	The calibration is rebuilt for the current temperature, using the active MicroDaq-8 settings, for the selected scanner.
Rate	V/86	byte = 0xab a = 1 TCP / UDP a = 2 CAN b = 0 : Off b = 7 : 200 Hz b = 8 : 150 Hz b = 9 : 100 Hz b = 10 : 50 Hz b = 11 : 25 Hz b = 12 : 20 Hz	Adjust the data delivery rate for each communication channel. The upper nibble of the parameter byte selects which communication channel, while the lower selects its data rate. Available rates are the same for both TCP/UDP and CAN.

		b = 13 : 10 Hz b = 14 : 5 Hz b = 15 : 1 Hz	
Protocol	P/80	byte = 0xab a = 1 TCP / UDP a = 2 CAN b = 0 - 18 bit LE b = 1 - 18 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. Binary data provides pressure scaled as unsigned integers to the operating full scale, eg for 18 bit resolution, 0 to 262143 represents -FSD to +FSD.
Stream ON	1/49	1 – TCP / UDP 2 - CAN	Enable the data delivery for the chosen channel. The delivery rate is as selected in the 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.
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.
Span	A/65	1-8 – Scanner to span cal.	Request a 'span' operation. The unit should have been recently rezeroed, then the appropriate span pressure (as selected from the webserver) applied to the scanner. Requesting a span operation will then calculate the span correction coefficient for each channel. All rezero values and span coefficients are then written to the unit's EEPROM (NOT to the scanner), and the calibration tables rebuilt using these new values.
Reset Linear Calibration	E/69	1-8 – Scanner to reset cal.	Resets the linear calibration (ie zero and span values) held within the unit to 0 and 1.0 respectively for all channels. A calibration table rebuild is then performed.
Hardware Trigger	T/84	byte = 0xab a = 0 Disable a = 1 Enable with TTL b = 1 TCP / UDP b = 2 CAN	Enables or disables the hardware trigger on the selected comms channel. Note that there is no positive acknowledge for this command.
Get Status	?/63	byte = 0xab a = Scanner number (0-indexed) b = Detail level: 0 : Short 1 : With temp. 2 : Full 3 : Pressure reading 4 : Temp. readings 5 : Excitation reading 6 : Hall sensor read 7 : Firmware ID 8 : Unit serial number 9 : Scanner serial	Return a status packet from the MicroDaq-8 for a given scanner (note the upper four bits of the parameter are the scanner number, but 0-indexed). Three main versions are available, '*short*', '*with temp*.' and '*full*'. Short returns status byte information showing current operating state only, whereas '*full*' returns setup options and temperature data in addition. '*With temp*.' returns the status and temperature information only. Additional readings poll raw values for temperature and pressure as read from the scanner, as well as other fields as indicated in the table. Where an actual scanner is not connected, some fields will be zeros and the scanner serial number will be replaced with <scanner number>:NOT_CONNECTED.

Figure 2.2, The Available User Command Set for the MicroDaq-8.

3. Status Data Format.

Status data is not currently supported over the CAN channel, however for TCP/UDP connections, status data may be requested from the MicroDaq-8 for any scanner. There are three main detail levels – '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 figure 3.1.

15						I-daq conn.	Hardware Trigger Active	Derange acvtive	DTC conn.	CAN Active	TCP Active	*Reserved*	Cal. table	Span	Rezero

Figure 3.1, Status word bit assignment.

Requesting the status 'with temp.' returns the short status 4 bytes, followed by the temperature data for all active channels, 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 MicroDaq-8 for the requested scanner. Each field is comma delimited, and its function is indicated in plain text between square parentheses. On/off is indicated by 1/0. If a scanner channel is requested where an actual scanner is not connected then some field values will be zeros and the scanner serial number will be replaced with 'NOT_CONNECTED'. An example 'full' status string is shown in figure 3.2 for a scanner channel where no scanner is connected. Note that the figure for full scale includes the derange constant (ie scanner full scale x derange constant) if the derange option has been selected in setup.

*>Mó<0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,[Serial] 1820402,[Sc Number] 1,[Scanner] NOT_CONNECTED,[Valve] NA,[Full scale] 0.00000000,[Active channels] 0,[DTC active] 0,[CAN channels] 64,[TCP channels] 64,[CAN rate] OFF,[TCP rate] OFF,[CAN message] Multiple,[CAN protocol] 18 LE,[TCP protocol] 18 LE,[Temp. compensation] cont.,[IP] 192.168.1.190,[Mask] 255.255.0.0,[Gateway] 0.0.0.0,[CAN timing] (BRP) 5 (TSEG1) 11 (TSEG2) 3 (SJW) 3,[CAN message] 10n,[Rezero order] 7

Figure 3.2, Full status information.

Other detail levels return specific values, including the pressure or temperature values as read from the scanner (18bit ADC readings) for each channel, the excitation and hall sensor readings (from the DTC scanner), the firmware revision and the MicroDaq-8 and scanner serial numbers. As with the main detail levels, if an actual scanner is not connected then readings like the excitation,etc. will be zero and the scanner serial number will return: <scanner number>:NOT_CONNECTED

4. MicroDaq-8 Communication Channels.

4.1 TCP

4.1.1 Overview.

The MicroDaq-8 TCP channel affords it the ability to stream real time pressure data for all connected scanners at high speed over standard 100Mbit Ethernet connections.

4.1.2 Connection.

When using the TCP/IP channel, the MicroDaq-8 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 MicroDaq-8 only supports one TCP connection on that port.

Setup requires the MicroDaq-8 to be given an IP address, and the network's subnet mask. It is possible either to run the MicroDaq-8 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 MicroDaq-8 not to be seen on the second card. The MicroDaq-8 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 2 different binary data formats – 18bit little ended and 18bit big ended. The MicroDaq-8 can support up to 8 scanners of 64 channels at once and as such the data output format is fixed to reflect this. Where a scanner is not present or has less than 64 channels then the data stream packet is filled with 0 for all 'non-existent' channels of that scanner.

Due to the large amount of data that needs to be sent in a packet for all scanners, the MicroDaq-8 packs the 18bit data for four channels into 9 bytes. This format repeats for all channels of a scanner and all scanners of the system.

Figure 4.1 shows an example section of the protocol. Where two channels share a streaming byte, the following of the two channels is bit shifted left by 2, 4, or 6 bits and the lowest 6, 4, or 2 bits are bitwise-ORed with the left over bits from the preceding channel.

The 18 bit little ended protocol is most efficient from the MicroDaq-8's viewpoint, the data being calibrated and spanned to 0 to 262143 (zero at 131071), representing –ve to +ve full scale of the scanner. Scaling to floating point full scale is best achieved within a PC client application where there is more processing power available.

Figure 4.1 Example 18bit data packing in streaming bytes (using 18-bit LE protocol):

Data		Goes into streaming bytes …
3 byte header (0x00, 0xFF, 0x00)		0, 1 & 2

… then…

18bit data (LSB first)	Is bit shifted left by …	Goes into streaming bytes …
Scanner 1 Ch1	0	3, 4 & 5
Scanner 1 Ch2	2	5, 6 & 7
Scanner 1 Ch3	4	7, 8 & 9
Scanner 1 Ch4	6	9, 10 & 11
Scanner 1 Ch5	0	12, 13 & 14

...

Scanner 1 Ch61	0	138, 139 & 140
Scanner 1 Ch62	2	140, 141 & 142
Scanner 1 Ch63	4	142, 143 & 144
Scanner 1 Ch64	6	144, 145 & 146
Scanner 2 Ch1	0	147, 148 & 149

... and so on for all channels of all scanners (channel data filled with 0 for those scanners that are not 64 channels or not connected at all).

Note each block of 512 channels (i.e. 8 scanners at 64 channels per scanner) is preceded by the three byte header (0x00, 0xFF, 0x00) to indicate the start of the channel block (this is the same header as used in the data streaming of the microDAQ-Mk2, for historical and familiarity purposes).

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.

Note that since the scanners are addressed at a fixed channel rate, requesting a data delivery of more than 14.285k/All Scanner Channels - i.e. 205Hz (8 scanners at 64 channels per scanner) - wastes resources and in some cases can actually cause the MicroDaq-8 to hang and require a power cycle.

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 MicroDaq-8's communication consists of many small packets at a high repeat rate, this algorithm has a catastrophic effect on the MicroDaq-8'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 and later, the DWORD value TcpAckFrequency should be set to 1 in the registry key.

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 MicroDaq-8 and client PC.

4.1.5 Control Via TCP.

The command packet has been detailed previously in section 2.1 above. 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 MicroDaq-8's UDP channel affords it the ability to stream real time pressure data at high speed over standard 100Mbit Ethernet connections.

4.2.2 Connection.

When using the UDP/IP channel, the MicroDaq-8 is programmed to listen on its local port. 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 webserver settings.

Setup requires the MicroDaq-8 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.

4.2.3 UDP Protocol

The protocol for streaming over UDP is almost exactly the same as that for the TCP protocol (detailed in 4.1.3 above) and so is not repeated here. The one difference is that the 3 byte header (0x00, 0xFF, 0x00) is replaced by a 32bit representation of the MicroDaq-8 serial number followed by a 32bit packet number, to identify the start of the packet (so 8 bytes are used at the start of the packet before the channel data, rather than the 3 bytes of a TCP packet). This header layout is identical to that used for UDP streaming on the microDAQ-Mk2, again for historical and familiarity purposes.

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.

4.3 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 MicroDaq-8 offers a single 'standard' CAN bus connection running at a selectable baudrate, the user having access to the values used for the MicroDaq-8's microcontroller CAN timing registers to enable customising of sample point and jump width.

4.3.3 CAN Protocol.

Note that currently the exact structure of the CAN protocol has yet to be agreed and as such is not detailed here.

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.

Care should be taken in selection of delivery rate, as it is possible to select unfeasibly fast data delivery for a particular bus baudrate, resulting in data loss. Similarly since the scanners are addressed at a fixed channel rate, requesting a data delivery of more than 14.285k / All Scanner Channels - i.e. 205Hz (8 scanners at 64 channels per scanner) - wastes resources and in some cases can actually cause the MicroDaq-8 to hang and require a power cycle.

4.3.5 Control Via CAN.

The command set is supported over the MicroDaq-8'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 webserver, 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 4.2. If the command is received without detected error, and the acknowledge option has been selected, the MicroDaq-8 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 4.2, Data Structure of the CAN Incoming Control Message.