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

Tel: 01692 500555
Fax: 01692 500088

flightDAQ-TL

USER PROGRAMMING GUIDE

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

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

900224-1.1

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. 900224 : Issue 1.1
ECO: 3924 Date: 16th September 2019

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.

This manual covers the communication protocols available for the flightDAQ-TL product. The flightDAQ-TL supports a subset of Chells native command set found in our other DAQ systems (in particular the microDAQ-Mk2 & flightDAQ-Mk2) so users familiar with those other systems will notice many similarities between those and the flightDAQ-TL. Additionally the flightDAQ-TL supports a Netscanner emulation mode where it operates with a sub-set of the Netscanner commands available on a 9016/9022 Netscanner system from TE Connectivity

Furthermore the flightDAQ-TL supports the DDS command protocol but that will be covered in a separate document.

From power up, the flightDAQ-TL reads its non volatile setup information and calibration, and after applying these settings is then 'up and running', reading the (pre-configured) transducers 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.

As mentioned previously, the flightDAQ-TL 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. 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 data packet format for all channels.

This document version supports V1.0.1 of the flightDAQ-TL firmware. If using earlier firmware then some user commands and/or parameters may be invalid.

2. Native User Command Protocol.

2.1. Command Packet.

The native 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.

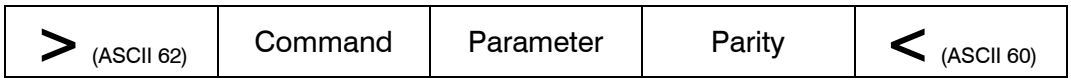

Figure 2.1 Command Frame Format.

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
Reset	R/82	0: Soft reset 1: Hard reset	Request a device reset. The parameter determines whether it is a hard reset (OS reboot) or soft reset (application restart)
Rezero	Z/90	0 to 31: rezero specified channel 255: All channels	Request a rezero. For a single channel rezero, the parameter is zero-indexed – ie. for channel 1, specify a 0 in the parameter byte. 0-15 are for primary channels, 16-31 are for secondary channels.
Rate	V/86	byte = 0xab b = 0 : Off b = 1 : n/a (250Hz) b = 2 : n/a (250Hz) b = 3 : n/a (250Hz) b = 4 : n/a (250Hz) b = 5 : 250 Hz b = 6 : 200 Hz b = 7 : 150 Hz b = 8 : 100 Hz b = 9 : 50 Hz b = 10 : 33 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 TCP/UDP communication channel. The lower nibble of the parameter byte (b) selects the data rate, while the upper nibble (a) is not used (could be set to 1 in keeping with the same command found on the microDAQ-MK2 system, but is actually ignored on the flightDAQ-TL). Note that rate values of 1 to 4 are currently not applicable to the flightDAQ-TL and are reserved for future use.

Protocol	P/80	byte = 0xab b = 0 : 32 bit LE b = 1 : 32 bit BE b = 2 : ASCII Engineering Units	Permits the changing of the data streaming protocol. The lower nibble of the parameter byte (b) selects the protocol, while the upper nibble (a) is not used. 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 encoded data is a 32bit representation of the floating point EU value for a channel conforming to IEEE754. ASCII representation is a comma separated channel data string.
Stream ON	1/49	None	Enable the streamed data delivery, at the rate preconfigured in the flightDAQ-TL webserver or via the Rate command.
Stream OFF	0/48	None	Disable the streamed data delivery.
Get Status	?/63	0 : Short 1 : Internal pressure and temperature 2 : Full 3 : undefined 4 : undefined 5 : Excitation reading 6 : Sensor type 7 : Firmware ID 8 : Unit serial number 9 : Connector serial numbers	Return a status packet from the flightDAQ-TL. Three main versions are available, '*short*', '*internal pressure/temp*' and '*full*'. Short returns status byte information showing current operating state only, whereas '*full*' returns all setup options and internal pressure & temperature data in addition. ' *internal pressure/temp* ' returns the status and internal pressure & temperature informatio. The data format is documented in a separate section. Additional status variants return other specific information values as listed in the table. *Note at time of writing this command is currently Work In Progress, and as such may not return the requested status packet. This will be addressed in a future firmware release.*
Channels	H/72	byte = 0xab b = 0 - 16 b = 1 - 32	Set the number of active channels returned to the user for a data stream. The lower nibble (b) sets the number of channels which can be either 16 (for all primary channels) or 32 (for all primary and secondary channels). The upper nibble (a) is not used.
Poll	O/79	None	Request a single data packet in the current active format. Data streaming should be set off before using this command. Note that there is no positive acknowledge for this command.

Figure 2.2, The Available User Command Set for the flightDAQ-TL.

3. Status Data Format.

Status data is currently Work In Progress on the flightDAQ-TL and such the three main forms are not fully defined. This section will be updated when Status is fully implemented in a future firmware release.

4. Native Communication Channels.

4.1. TCP

4.1.1. Overview.

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

4.1.2. Connection.

When using the TCP/IP channel, the flightDAQ-TL 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 previously setup to use the TCP channel. Note that the flightDAQ-TL only supports one TCP connection on this port.

It is possible either to run the flightDAQ-TL 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 flightDAQ-TL not to be seen on the second card. The flightDAQ-TL will respond to a 'ping' instruction for the purposes of setup and troubleshooting.

4.1.3. TCP Protocol

The different data protocol formats supported are as shown in Figure 4.1. The binary data formats take the floating point values and encode them as 32bit IEEE754 representation. Binary protocols require less communications bandwidth as well as less processor overhead within the flightDAQ-TL; it is recommended that engineering unit conversions be applied at the client.

32bit IEEE754 float, little endian

Header			Channel1 Data				Channel 2 Data				Channel 3 Data			
0x00	0xFF	0x00	LSB			MSB	LSB			MSB	LSB			MSB

….

Channel 'N' Data			
LSB			MSB

3 byte header identifies the start of the packet, followed by all channels (1 to 'N' where 'N' = 16 or 32 depending on setting), LSB first, with no delimiters.

32bit IEEE754 float, big endian

Header			Channel1 Data				Channel 2 Data				Channel 3 Data			
0x00	0xFF	0x00	MSB			LSB	MSB			LSB	MSB			LSB

….

Channel 'N' Data			
MSB			LSB

3 byte header identifies the start of the packet, followed by all channels (1 to 'N' where 'N' = 16 or 32 depending on setting), MSB first, with no delimiters.

ASCII comma separated

Hdr		Channel 1 Data		Channel 2 Data		….		Channel 'N' Data
*	,	#.######	,	#.######	,		,	#.######

Single character header followed by all channels (1 to 'N' where 'N' = 16 or 32 depending on setting), 6 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 setup page of the webserver or via the user commands detailed previously (see Section 2.3). Although the system endeavours to deliver the rate with maximum accuracy, ultimate responsibility for data timing lies with the user's host system.

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 flightDAQ-TL's communication consists of many small packets at a high repeat rate, this algorithm has a catastrophic effect on its 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 7, 8 & 10 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 flightDAQ-TL and client PC, particularly if several units 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 flightDAQ-TL's UDP channel affords it the ability to stream real time data at high speed over standard 100Mbit Ethernet connections.
The flightDAQ-TL supports two data stream formats, the native Chell data stream format and an IENA specification output, which can be setup from the web server.

4.2.2. Connection.

UDP is known as a 'connectionless' protocol, but it will listen for incoming commands to it's local port (101 – same as TCP) and will respond accordingly back to the host IP & port that sent the command packet.

To facilitate auto streaming over UDP, setup requires a remote IP address and remote port number to be configured, so it knows where the streaming data is to be sent to. It is possible either to run the flightDAQ-TL 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 flightDAQ-TL not to be seen on the second card. The flightDAQ-TL will respond to a 'ping' instruction for the purposes of setup and troubleshooting.

4.2.3. Chell UDP Protocol

The different data protocol formats supported are shown in Figure 4.2 (same as with the TCP protocol above). The binary data formats take the floating point values and encode them as 32bit IEEE754 representation. Binary protocols require less communications bandwidth as well as less processor overhead within the fligthDAQ-TL; it is recommended that engineering unit conversions be applied at the client. Note the difference between UDP & TCP in that the header for UDP consists of a 32 bit number indicating the flightDAQ-TL serial number followed by a further 32 bit packet number!

32bit IEEE754 float, little endian

Header (serial num.)				Packet Number				Channel 1 Data				Channel 2 Data			
LSB			MSB	LSB			MSB	LSB			MSB	LSB			MSB

....

Channel 'N' Data			
LSB			MSB

4 byte header containing the unit serial number identifies the start of the packet, followed by a packet number, followed by all channels (1 to 'N' where 'N' = 16 or 32 depending on setting), LSB first, with no delimiters.

32bit IEEE754 float, big endian

Header (serial num.)				Packet Number				Channel 1 Data				Channel 2 Data			
MSB			LSB	MSB			LSB	MSB			LSB	MSB			LSB

....

Channel 'N' Data			
MSB			LSB

4 byte header containing the unit serial number identifies the start of the packet, followed by a packet number, followed by all channels (1 to 'N' where 'N' = 16 or 32 depending on setting), MSB first, with no delimiters.

ASCII comma separated

Hdr		Channel 1 Data		Channel 2 Data		....		Channel 'N' Data
*	,	#.######	,	#.######	,		,	#.######

Single character header followed by all channels (1 to 'N' where 'N' = 16 or 32 depending on setting), 6 decimal places, comma delimited.

Figure 4.2, 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.

4.3. Timestamping

In addition to the above, standard data streaming over both TCP & UDP can include time stamping information to aid with synchronisation of data packets. The device keeps track of time to microsecond resolution since it was powered on but this can be turned into a real time by either synchronising the time with a PC (via the webserver) or for a more accurate time sync, by using a PTP grandmaster to provide a constant PTP sync message.

In both cases the time stamp is represented as two 32 bit values, the first being the Unix Epoch time (i.e. the number of seconds since 1st January 1970) and the second being the number of nanoseconds in that second.

In the flightDAQ-TL's data streaming, the timestamps can be placed at the start of a channel cycle (ie. after the header but before channel 1's data) or in front of every channel. This is configurable from the Advanced page of the embedded webserver. Note that the byte order of the timestamp is the same as that of the channel data, as configured by the protocol selection (32 bit LE or BE) and timestamps are not added if the Engineering units protocol is selected instead. Adding timestamps (particularly on a per channel basis) can significantly affect the speed of the data streaming – some of the high data rates may not be achievable due to the extra data being transmitted per packet to facilitate the time stamps in the stream.

4.4. IENA specification

The device has the option to output data in the IENA specification data packet format. This specialised format arranges the data in a specific format containing various different information. The data format is only available when using the UDP comms protocol.

Please Note: As of writing, this format is not supported in the current release of flightDAQ-TL firmware and will appear in a future release. As this is a pre-defined format, the following description will stand true when implementation is complete.

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 63	Temperature	Reserved	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				Rolling 16 bit counter	Byte order 3-2-1-0		Byte order 3-2-1-0	Byte order 3-2-1-0		

Starting with the 16 bit key field, this is a user defined key that will appear at the start of every IENA packet. The default value is 0x3101.

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. This will be in UTC or TAI depending on the users choice.

The next 16 bits is a STATUS word. This word is used to show the status of various parameters in the device, as follows:

Bit	**Meaning**
1(LSB)	Synced to external time
2	Synced to PTP master clock
3	*Reserved for future use*
4	*Reserved for future use*
5	*Reserved for future use*

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 data and can be in big endian or little endian format depending on the users selection. There is one 32 bit value for each channel of data.

The data is followed by a single 32 bit unit temperature value.

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 will show if the scanner is in purge mode or not, the second bit will show if the device has been time synchronised. Further bits are reserved for future use and are set as 0.

The final 16 bits are the end field, This value is a user settable value that signifies the end of an IENA packet. The default value of this field is 0xDEAD.

5. NetScanner Emulated Communication

5.1. Overview

The flightDAQ-TL has the ability to emulate a NetScanner instrument from TE Connectivity, supporting a specific sub-set of the commands available to that family of instruments. Emulation is enabled by setting the appropriate configuration item on the Advanced page of the embedded webserver.

5.2. Connection

On startup, the flightDAQ-TL will open two sockets, one for receiving of broadcast UDP discovery commands and the other for connection over TCP.

The flightDAQ-TL can be directly connected via TCP if the IP address is already known, but typically, the host PC will broadcast the 'psi9000' discovery command over UDP to port 7000 and the flightDAQ-TL will respond with a status packet on UDP port 7001 containing the IP address, listening TCP port, etc. of the unit. The host can then open a TCP connection to that IP address/port for sending of NetScanner commands and to receive streamed data.

5.3. Supported Commands

The table in Figure 5.1 lists the NetScanner commands & sub-commands supported by the flightDAQ-TL in emulation mode. Please note that unless indicated, each command should be issued without any spacing between the command letter, sub-command and any/all parameters.

All supported commands will respond like a NetScanner with either a positive acknowledge ('A') or an error response ('Nxx' where xx is an error number – see section 5.5 below for more information)

UDP broadcast

Command	Sub Command	Parameters	Description
psi9000	N/A	None	Broadcast discovery. Response is a comma-separated string of status values: `ipadr,ethadr,sernum,mtype,sfwver,connst,ipadrst,lisport,subnet,iparpst,udpast,pwrst,` where: `ipadr` is the TCP/IP Address `ethadr` is the MAC Address `sernum` is the Serial Number `mtype` is the Model Type `sfwver` is the Firmware Revision `connst` is the TCP/IP Connection Status `ipadrst` is the RARP TCP/IP Address Assignment Status (always set to 1 on the flightDAQ-TL `lisport` is the TCP/IP port on which the flightDAQ-TL is listening for connections `subnet` is the Subnet Mask `iparpst` is the TCP/IP Address Resolution Method (always set to 0 on the flightDAQ-TL) `udpast` is the UDP/IP Broadcast when Re-Boot status (always set to 1 on the flightDAQ-TL) `pwrst` is the Power Up Status

TCP

A		None	NOP. Used to verifiy proper communication.
B		None	Reset. Resets some of the internal operating parameters to their default state. Note this does nor perform a full system reset. Average samples setting is set to its last stored value (in EE) Any data streaming is stopped and currently configured data streams are cleared (must be setup again with the 'c 00' command).

c
(Note: this command, sub-command and parameters are space separated)
	00	st pppp trig per f num	Stream Data – Configure Stream
			Configures a steam for the flightDAQ-TL to stream data to the host.
			Parameters are:
			`st` = stream number – always 1 on flightDAQ-TL
			`pppp` = channel bit field – always FFFF (all transducers) on flightDAQ-TL
			`trig` = trigger type - always 1 (s/w trigger) on flightDAQ-TL
			`per` = data stream interval (in milliseconds) – the following are valid values for flightDAQ-TL:
			`0, 4, 5, 10, 20, 30, 40, 50, 100, 200, 1000`
			`(note 0 sends a single packet`
			`f` = format field – valid values are:
			`0` (ASCII), `7` (32bit BE), `8` (32bit LE)
			`num` = number of packets – always 0 (all) on flightDAQ-TL

c *(Note: this command, sub-command and parameters are space separated)*	01	st	Stream Data – Start Streaming Starts a pre-configured stream at the rate set in 'c 00'. Parameter is: `st` = stream number – always 1 on flightDAQ-TL
c *(Note: this command, sub-command and parameters are space separated)*	02	st	Stream Data – Stop Streaming Stops a pre-configured stream Parameter is: `st` = stream number – always 1 on flightDAQ-TL
c *(Note: this command, sub-command and parameters are space separated)*	05	st bbbb	Stream Data – Select Data to be transmitted Selects whether the stream will include primary EU channel data or primary & secondary EU channel data. Parameters are: `st` = stream number – always 1 on flightDAQ-TL bbbb = channel data to stream – valid values are: `0010` (primary) or `0090` (primary & secdonary)
h		pppp	Calibrate EU offset Performs a user rezero on the calibrated data.

			Parameter is: pppp = channel bit field – always FFFF (all transducers) on flightDAQ-TL
q	00	None	Read Status & Information – Model Number Returns the instrument model number as a string, e.g. 9016
q	05	None	Read Status & Information – Averaging Returns the current averaging samples setting as a hex string, e.g. 0010 = 16 samples avergaing
r		ppppf	Poll Data Returns a single packet of data for all transducers in the format specified. Parameters are: pppp = channel bit field – always FFFF (all transducers) on flightDAQ-TL f = format field – valid values are: 0 (ASCII), 7 (32bit BE), 8 (32bit LE)
u		0aa07	Read Cal Data – Last Calibration Date Returns the last calibration date in the format 'yymmdd'. Parameter is: aa = 01-10 (hex) for the transducer for which the date is being requested – Currently the flightDAQ-TL only supports the single calibration date held for the instrument and hence the tranducer channel parameter is ignored – any value within the given range will return the unit last calibration date.
w	10	yy	Write Configuration – Averaging Sets the averaging samples for ADC measurement. Note this change is volatile and will be returned to EE stored value on a reset or power cycle. Parameter is: yy = number of samples – valid values (in hex) are: 2, 4, 8, 10, 20, 40
w	16	dd	Write Configuration – Frame Header Enables or disables the frame header (aka Message Length field) that can be placed at the beginning of every response packet. For flightDAQ-TL is currently recommended that this be disabled. Parameter is: dd = enable/disable – valid values are: 00 (disable) or 01 (enable)

Figure 5.1, NetScanner Emulation supported commands

5.4. Streamed Data Output format

The format of the received data stream is determined by a number of NetScanner commands – in particular, the 'w16' command which determines whether there is a two byte frame header at the start, 'c 05' which determines whether only the primary channels (1-16) are in the data stream, or primary & secondary channels are included (1-32) and the 'c 00' command which determines the format of the channel data (ASCII, or 32bit encoded float in big endian & little endian).

With all formats, the channel data is preceeded by a single byte stream number (always 1 for flightDAQ-TL) and a four byte packet number which is always represented in big endian format irrespective of the channel data format specified. Also note that the channel data is streamed in reverse order (16 to 1 or 32 to 1).

ASCII (format '0')

Strm	Packet Number					Channel 'N' Data		Channel 'N-1' Data		....		Channel 1 Data
1	MSB			LSB		#.######		#.######				#.######

Single byte stream number (always 1), followed by a packet number, followed by all channels ('N' to 1 where 'N' = 16 or 32 depending on setting), 6 decimal places, space delimited.

32 Bit, big endian channel data (format '7')

Strm	Packet Number				Channel 'N' Data				Channel 'N-1' Data				....	Channel 1 Data			
1	MSB			LSB	MSB			LSB	MSB			LSB		MSB			LSB

Single byte stream number (always 1), followed by a packet number, followed by all channels ('N' to 1 where 'N' = 16 or 32 depending on setting), MSB first, with no delimiters

32 Bit, little endian channel data (format '8')

Strm	Packet Number				Channel 'N' Data				Channel 'N-1' Data				....	Channel 1 Data			
1	MSB			LSB	LSB			MSB	LSB			MSB		LSB			MSB

Single byte stream number (always 1), followed by a packet number, followed by all channels ('N' to 1 where 'N' = 16 or 32 depending on setting), LSB first, with no delimiters

Figure 5.2, Data Packet Protocol, NetScanner Emulation

5.5. NetScanner Error Codes

The table in Figure 5.3 lists the error responses that may occur as a result of an invalid/incorrectly received/processed command. The response begins with the letter 'N' indicating a Negative Reponse, followed by a two digit hex code as follows – taken directly from the TE Connectivity NetScanner Programmers Reference.

Code	Description
01	Undefined Command Received
03	Input Buffer Overrun
04	Invalid ASCII Character Received
05	Data Field Error
07	Specific Limits Invalid
08	Invalid Parameter

Figure 5.3, NetScanner Error Codes