Quantum System Advanced Serial Command Manual

Updated 12/10/2019

1. Introduction

Thank you for your purchase of the Unist Quantum system. The Quantum system is a programmable Minimum Quantity Lubrication (MQL) system designed to interface with a variety of machines. The system has been designed for maximum flexibility while still maintaining simplicity in its controls and programming. This manual details the serial command capabilities contained in the Quantum system. Please take time to review this manual to ensure you receive the maximum performance and flexibility attainable by your Quantum through the use of serial control.

1.1.What is serial control?

Serial control refers to the Quantum's capability to listen, react, and respond to a specific serial command language. Users can program their machine to send commands to the Quantum that it will carry out in real time. Serial control uses serial ports to communicate information between devices one bit at a time, or serially. Information is sent between the devices and the devices are programmed to react to specific commands. Unist has developed a serial command language that allows end use machines to tell the Quantum system how to behave. The Quantum has also been developed to send information back to the end use machine for information gathering.

1.2.Why use serial control?

The use of serial control allows users maximum flexibility in controlling their Quantum systems. Using serial commands, an end use machine can tell the Quantum exactly when to turn on and at what specific rate. This ensures that the amount of output from the Quantum can be adjusted as each process requires. The Quantum's job function also allows users to fine tune output for specific processes, but using jobs can become cumbersome when performing many different processes in applications such as CNC machining.

1.3.What is required to use serial control?

To use serial control, users must have the ability to program the end use machine to send text based (ASCII) commands out of a serial port in sequence with the process. This means that the end use machine must have an RS232 serial port available, the port settings must be able to be configured to match the settings on the Quantum serial port, and the machine must be able to be programmed to send commands over this serial port. For CNC Machines, this is usually accomplished using the DPRNT command. For other PLC controlled devices, an ASCII command to output a text string can be used to send commands to the Quantum.

2. Serial Connection Terminology

The following section summarizes various terminology related to serial connections between devices.

2.1.RS232

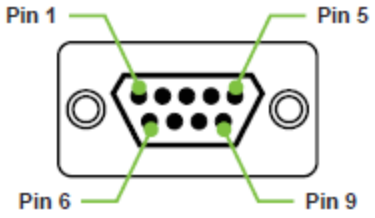
RS232 is a standard for serial transmission of data. It is a standard that defines the signal parameters between to equipped devices. The RS232 standard is commonly used on computer serial ports, CNC machines, and PLCs. In its most basic form, an RS232 connection uses three wires: transmit, receive, and ground. Other wires can be used for controlling the flow of data between devices, but the Quantum only requires the three connections.

2.1.1. DB9

A DB9 connector is one common type of serial port connector that has 9 pins. According to the RS232 standard, the pins are utilized as shown in the figure below. DB9 connectors are usually found on computers and PLCs. A DB9 serial cable that has nine-pin connectors on both ends and can usually be obtained in any combination of male and female ends.

RS232 DB9

Pin 2	Receive (RX)
Pin 3	Transfer (TX)
Pin 5	Signal Ground (GND)
Pin 7	Request To Send (RTS)
Pin 8	Clear To Send (CTS)



2.1.2. DB25

A DB25 connector is another common type of serial port connector that has 25 pins. According to the RS232 standard, the pins are utilized as shown in the figure below. DB25 connectors are commonly found on CNC machines. A DB25 cable has a 25 pin connection on either end and can usually be obtained in any combination of male and female ends.

RS232 DB25

Pin 2	Transfer (TX)
Pin 3	Receive (RX)
Pin 4	Request To Send (RTS)
Pin 5	Clear To Send (CTS)
Pin 7	Signal Ground (GND)

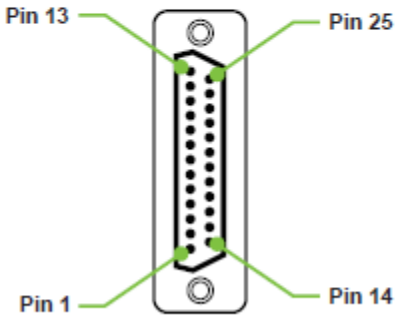
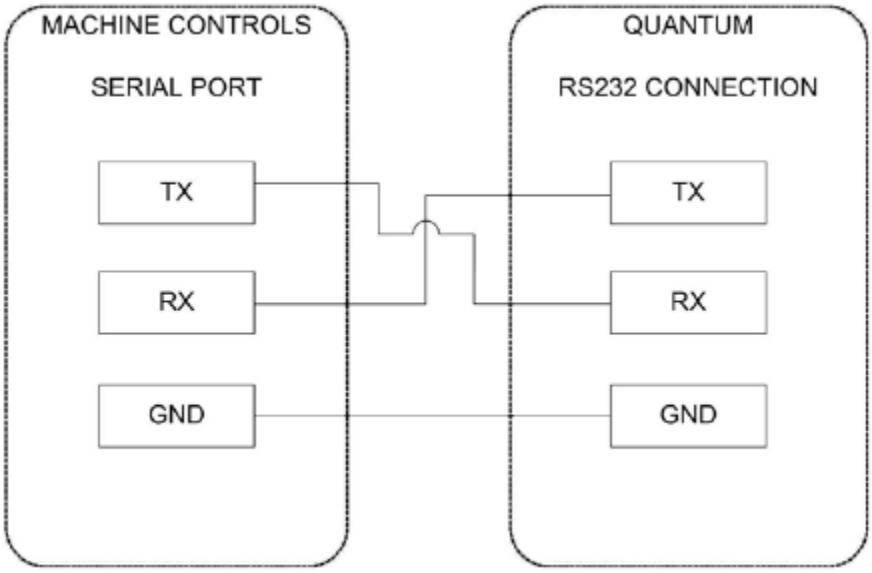


Figure 41: Typical Male DB 25 Connector

2.1.3. Null Modem and Straight Through

Other terms often encountered when dealing with serial cables are null modem and straight through. As mentioned before, RS232 serial connections require three wires: transmit (TX), receive (RX), and ground (GND). For two devices to communicate, the transmit pin of the first device must be connected to the receive pin of the second device. The transmit pin of the second device must be connected to the receive pin of the first device. The grounds between both devices must also be connected. A serial cable that is wired to achieve this crossover of transmit to receive is referred to as null modem. A cable that does not do this crossover is referred to as straight through. The diagram below shows a null modem connection between a machine and a Quantum.



2.2.Baud Rate

As previously mentioned, serial communication passes data between two devices one bit at a time. Baud rate refers to the frequency at which the bits are transmitted. If two devices are to communicate, the baud rate settings on the serial ports must match. Most devices have a number of baud rate values; refer to the Quantum User's Manual for the baud rate settings allowable on the Quantum system.

2.3.Data Bits

The serial commands that the Quantum reacts to are made up of ASCII characters. The data bit setting refers to the number of bits (1's or 0's) that are used to represent one ASCII character. The

Quantum can support both 7 and 8 bit ASCII characters. Both machines must be set to the same data bits for successful communication.

2.4.Parity

Parity bits are used in serial communications to check for errors in data communication. Serial ports can be configured to add an odd parity bit, an even parity bit, or no parity bits. The Quantum supports all three types of parity. Both the Quantum and the interfacing machine must have the same parity settings for successful communication.

2.5.Stop Bits

Stop bits are a pattern of bits added to the end of a serial communication indicating the end of the whole transmission. The Quantum supports both single and double stop bits; however this setting must match the settings on the commutating device.

2.6.Flow Control

Flow control is sometimes used in serial communication to control the flow of data between two devices. This is especially useful with a high rate of data transfers back and forth between devices. Flow control provides a mechanism for coordinating ingoing and outgoing messages between devices. Both devices must be configured to use the same flow control method.

2.6.1. No Flow Control

The Quantum supports the option of no flow control. For the majority of applications this is acceptable for the Quantum due to the limited number of commands. The sending device will just send a command whenever it is ready to send.

2.6.2. Software Flow Control (XON/XOFF)

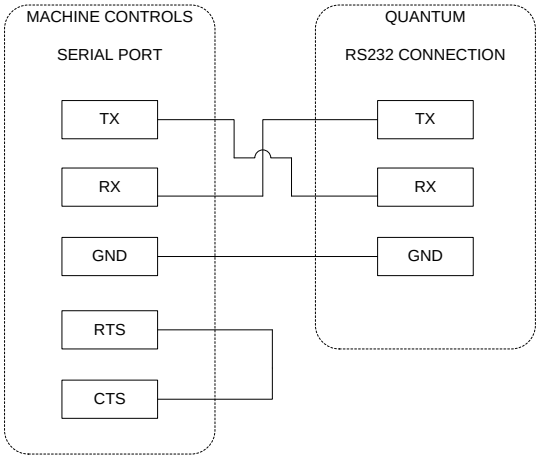
Software flow control uses special ASCII characters (XON and XOFF) to control the flow of information between devices. When a device is ready to receive a command, it sends an XON character to the other device and that device will start sending. If a device cannot receive a command, it sends an XOFF character and the device sending information will pause transmission. The Quantum supports XON/XOFF software flow control.

2.6.3. Hardware Flow Control (RTS/CTS)

Hardware flow control, sometimes referred to as RTS and CTS (Request to Send and Clear to Send), uses two additional physical wires to control the flow of information between devices. This is why it is

called hardware flow control. When a device wants to send data, it turns the RTS line on and waits for the CTS signal to come back from the other device before sending data.

The Quantum does not support hardware flow control. If the interfacing device must be set to use hardware flow control, hardware flow control can be imitated through wiring so that the commanding device believes the Quantum is practicing hardware flow control. However, communication with this setup will only be one way; from the device to the Quantum.



2.7.End of Block

Serial ports also usually have a setting available for which character is appended to the end of transmitted text. Options are usually nothing, a carriage return, a line feed, or a carriage return and line feed. According to the Quantum serial command language, each command must end with a carriage return. The machine interfacing with the Quantum must be set so that a carriage return is sent at the end of each transmitted command.

3. Serial Communications

Once a successful serial connection is established, users can begin to send serial commands to their Quantum system. The Quantum serial command language is composed of text based commands made up of ASCII characters. The commands contain a descriptive base command and then can be followed by parameters.

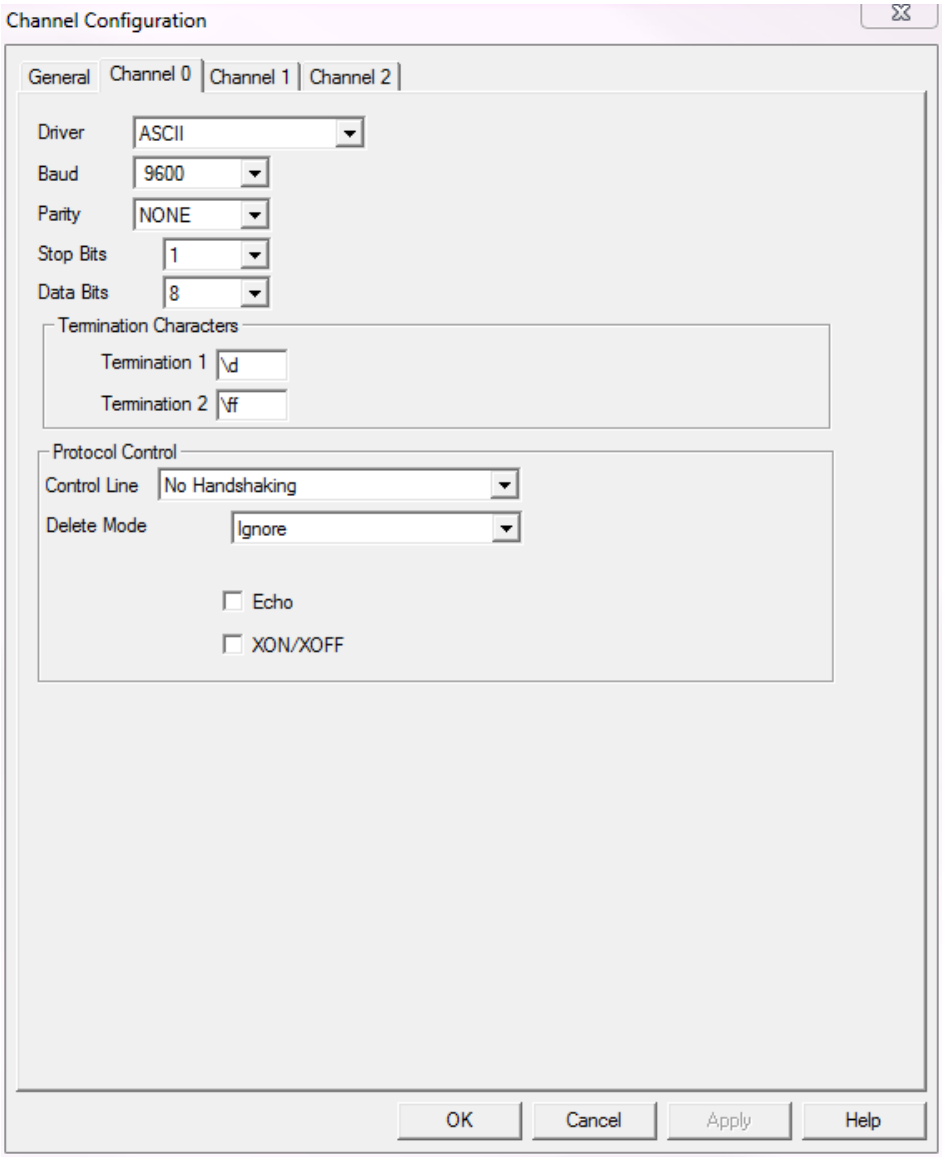
3.1.ASCII characters

ASCII stands for the “American Standard Code for Information Interchange.” Since computers only understand numbers, the ASCII standard dictates what numbers are used to represent specific characters. Users see letters, numbers, and special characters, but the computer sees the ASCII numeric code behind each character. Since the Quantum conforms to the ASCII standard, commands sent to it must also utilize the ASCII standard.

There are ASCII characters for number digits 0-9, all letters A-Z, special characters such as a -, +, ., and others. ASCII codes are also used to transmit not printable characters such as carriage returns and line feeds. These characters make up the Quantum serial control language.

3.2.ASCII Commands (PLC)

Most industrial PLC’s allow users to send ASCII strings to other devices. For this to be done, the PLC communication port must be configured to the ASCII settings and the connection specifics must be configured so that the PLC’s communication parameters match the Quantum’s parameters. ASCII command blocks can then be programmed to write out the desired command at the desired times during operation. The figure below shows the configuration display for a PLC communication channel configured to ASCII.



3.3.DPRNT Commands (CNC)

Most CNC controllers support the use of the DPRNT (D print) command for communication with external devices. The DPRNT command is intended for synchronizing controls between the CNC machine and peripheral devices. Each CNC machine has some nuances about how the DPRNT command is executed. It is important that the user fully understands these as they implement serial control of the Quantum system with a CNC machine. When writing a CNC program, the programmer can insert DPRNT commands that will turn the Quantum on at specific rates exactly when needed. DPRNT is generally considered a Macro command, so your CNC machine must have Macros enabled.

3.3.1. Synchronization with machining

CNC machine controllers utilize look-ahead to read ahead a certain number of lines of code for smoothing out machining operations. DPRNT commands usually execute during the look-ahead time so synchronization of the Quantum operation with the machining operation can be challenging. Understanding your machine's look-ahead procedures is critical for successful operation. For example, if the programmer wanted the Quantum to turn on after a tool change, he or she must make sure that the DPRNT command to turn the Quantum on operates after the tool change. Some machine controllers allow temporarily limiting of the look-ahead and others simply only look a few lines a head so a few blank lines of code will produce the desired result.

3.3.2. Special CNC observations

Commands in the Quantum serial control language are made up of the base command followed by parameters. The command and parameters are separated by spaces. When entering DPRNT commands in a CNC program, the space is not a valid character to enter. Rather, most CNC machines print out the * character as a space. So if the CNC machine was to send out the sentence "This is a serial command.", the DPRNT command would have to say DPRNT [This*is*a*serial*command.]. The CNC will send spaces instead of the asterisks.

3.3.3. CAM Integration

For CNC programmers writing programs with CAM software, it would be beneficial to edit the post processor in their CAM program to automate the addition of the special syntax required for using the Quantum during machining operations. Integration like this will allow for ease of selection of fluid and air flow rates for each specific operation, and the post processor can take care of the specific syntax required for completing the DPRNT command. This will greatly increase the ease of programming.

4. Testing Serial Connections

When setting up the Quantum to run serially, it can be beneficial to test the connection using a PC for more clarity about the commands being sent. Using a terminal program configured to the appropriate settings, a user can send commands to a Quantum system to ensure that the port is configured and working correctly. Users could also use a PC to interface with a CNC or PLC controller and test DPRNT or ASCII commands to ensure the device is sending out commands. A computer with a serial port and terminal program can be very beneficial when troubleshooting serial communications.

5. Quantum Serial Commands

The previous sections in this manual were about the generalities associated with making and establishing a serial connection between the Quantum and a controlling device. This section and the following are about the specific serial commands that are included in the Quantum serial control language and what the commands can allow users to do via a serial connection.

5.1.Syntax

Each Quantum command is made up of the command name that can be followed by one or more parameters. The command and parameters are separated by one or more spaces. Commands and parameters are case-insensitive and only one command can be transmitted at a time. A command must be followed by a carriage return. The carriage return indicates to the Quantum that the command is complete. The structure is shown below.

COMMAND NAME [SPACE] PARAMETER_1 [SPACE] PARAMETER_2 [CARRIAGE RETURN]

5.2.Command Types

The Quantum supports several broad types of serial commands. The description of each type of command follows below. For most commands, there is a SET and a GET version. The SET will write values to the Quantum system and the GET will return values from the Quantum.

5.2.1. System Information

System information commands are used to gather information about the Quantum such as system outputs and output types available for use.

5.2.2. User Configuration

User configuration commands can be used to get and control user specified settings on the Quantum system

5.2.3. Output Control

Output control commands directly control the state of fluid and air outputs on the Quantum system. These are the most common commands to use when controlling the Quantum serially.

5.2.4. Job Related

Job related commands allow users to view and configure jobs over a serial connection.

5.2.5. Alarms

Alarm commands allow the user to get the status of the system alarm registers.

5.3.Quantum Response

Whenever a serial command is sent to the Quantum it will respond. The responses can be as simple as an ASCII acknowledge character, or they can include a specific message such as a reason for a command failure, or the value that the command asked for. If a command is invalid the Quantum will respond with an ASCII cancel and the message "*COMMAND NOT VALID*".

6. Quantum Command Descriptions

The following sections outline available commands on the Quantum system.

6.1.System Information Commands

System information commands are used to gather information about the system.

6.1.1. GET-FLUID-TOTALIZER

6.1.2. RESET-FLUID-TOTALIZER

GET-FLUID-TOTALIZER

Syntax

GET-FLUID-TOTALIZER Output

Description

Gives back the totalizer value for the specified output. Outputs are any of the fluid outputs on the system.

Return Values

On success: the current value for the specified output totalizer

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_OUTPUT_NUMBER

RESET-FLUID-TOTALIZER

Syntax

RESET-FLUID-TOTALIZER Output

Description

Resets the totalizer value to zero for the specified output. Outputs are any of the fluid outputs on the system.

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_OUTPUT_NUMBER

6.2.User Configuration Commands

User configuration commands allow for setting of some settings on the controller.

6.2.1. SET-SYSTEM-DATE

6.2.2. GET-SYSTEM-DATE

6.2.3. SET-SYSTEM-TIME

6.2.4. GET-SYSTEM-TIME

6.2.5. SET-LOW-PRESSURE-ALARM-POINT

6.2.6. GET-LOW-PRESSURE-ALARM-POINT

SET-SYSTEM-DATE

Syntax

SET-SYSTEM-DATE DateString

Where DateString has the format of DDMMYYYY. This is a single string with no spaces, dashes separate the 4 digit year (YYYY), the two digit month (MM) and the two digit day (DD)

Description

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

UNABLE TO SET DATE

INVALID DATE

Examples

GET-SYSTEM-DATE

Syntax

GET-SYSTEM-DATE

Description

Gets the current date stored in the system

Return Values

On success: The date in DDMMYYYY format. This is a single string with no spaces, dashes separate the 4 digit year (YYYY), the two digit month (MM) and the two digit day (DD)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

UNABLE TO GET DATE

Examples

SET-SYSTEM-TIME

Syntax

SET-SYSTEM-TIME TimeString

Where TimeString has the format of HHMMSS. This is a single string with no spaces, colons separates the two digit hour (HH) which is specified in the 24 format (e.g. 3:00 pm is 15),the two digit minute, and the two digit seconds.

Description

The time does time update for Daylight Savings Time and will have to be updated manually.

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

UNABLE TO SET SYSTEM TIME

Examples

01:15:00 has the hour at 1 am, minutes at 15 after the hour, and no seconds.

14:20:30 has the hour at 2 pm. Minutes at 20 after the hour, and 30 seconds after the minute.

GET-SYSTEM-TIME

Syntax

GET-SYSTEM-TIME

Description

Gets the current time in the system

Return Values

On success: A string with the format of HHMMSS. This is a single string with no spaces, colons separates the two digit hour (HH) which is specified in the 24 format (e.g. 3:00 pm is 15), the two digit minute, and the two digit seconds.

On error: ASCII CAN (0x18) followed by error message

Error Messages:

UNABLE TO GET SYSTEM TIME

Examples

01:15:00 has the hour at 1 am, minutes at 15 after the hour, and no seconds.

14:20:30 has the hour at 2 pm. Minutes at 20 after the hour, and 30 seconds after the minute.

SET-LOW-PRESSURE-ALARM-POINT

Syntax

SET-LOW-PRESSURE-ALARM-POINT SetPoint

Where *SetPoint* is the pressure at which the alarm will sound. The value of *SetPoint* depends on the units selected. The following table gives the valid range

Units	Min	Max
PSI	0	100
KPA	0	700
BAR	0.0	7.0

Description

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

LPA_OUT_OF_RANGE

Examples

GET-LOW-PRESSURE-ALARM-POINT

Syntax

GET-LOW-PRESSURE-ALARM-POINT

Description

Returns the current setpoint for the low pressure alarm

Return Values

On success: the pressure at which the low pressure alarm will sound

On error: ASCII CAN (0x18) followed by error message

Error Messages:

NO_LPA_SETPOINT

Examples

6.3.Output Control Commands

Output control commands are the commands used to directly control outputs on the Quantum system. The most common is the SET-FLUID-OUTPUT-STATE command.

6.3.1. SET-FLUID-OUTPUT-STATE

6.3.2. GET-FLUID-OUTPUT-STATE

6.3.3. SET-AIRBLOWOFF-PARAMETERS

6.3.4. GET-AIRBLOWOFF-PARAMETERS

SET-FLUID-OUTPUT-STATE

Syntax

SET-FLUID-OUTPUT-STATE ControlID FluidRate AirflowRate

Where
ControlID - the output to be controlled.

FluidRate – specifies the outputs rate if this is a fluid or oil output On an ABO type output this value is unused

AirflowRate – sets the flow rate for air, if it is a proportional air pump output. This can be a value between 000 and 100. If the output is a an oil only, ABO, or metering screw output, this field is not used

Description

Programmatically turns on or off the specified output. To turn it off set the fluid rate to zero. Linked outputs require a ranged output designator (e.g., 3-5). It is the responsibility of the software and/or programmer to keep track of the linked outputs and use the proper designator. If an improper designator is sent – one that does not exactly match the linked set - an INVALID_CONTROL_ID error should be sent.

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_CONTROL_ID

FLUIDRATE_OUT_OF_RANGE

AIRRATE_OUT_OF_RANGE

Examples

GET-FLUID-OUTPUT-STATE

Syntax

GET-FLUID-OUTPUT-STATE ControlID

Where
ControlID - the output to be controlled.

Description

Gives the status of the specified output. If it is off the values are zero. If an improper designator is sent – one that does not exactly match the linked set - an INVALID_CONTROL_ID error should be sent.

Return Values

On success: FluidRate AirflowRate

FluidRate –the outputs rate if this is a fluid or oil output. *AirflowRate* –the flow rate for air, if it is a proportional air pump output. This can be a value between 000 and 100. If the output is a an oil only, ABO, or metering screw output, this field zero

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_CONTROL_ID

FLUIDRATE_OUT_OF_RANGE

AIRRATE_OUT_OF_RANGE

Examples

SET-AIRBLOWOFF- PARAMETERS

Syntax

SET-AIRBLOWOFF-PARAMETERS ControlID Frequency PulseLength

Where *Output* is the output designator for the output, *Frequency* is how often the pulse should occur in pulses per minute, and PulseLength length of the pulse in seconds. Valid ranges for the Frequency and PulseLength are:

Parameter	Min	Max
Frequency (min^{-1})	0	100
PulseLength (seconds)	0.00	99.9

Description

Determines how and if an air blow off pulses its output. works. If the pulse length multiplied by the frequency is less than 60, the output will pulse. If it is over 60 then the output will be continuous.

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_CONTROL_ID

INVALVID_CONTROL_ID

Examples

GET-AIRBLOWOFF- PARAMETERS

Syntax

GET-AIRBLOWOFF-PARAMETERS ***ControlID***

Where
ControlID specifies which output to control

Description

Returns the frequency and the pulse length set for the air blow off output.

Return Values

On success: a newline separated string contain the value of the frequency and the pulse length for the air blow off.

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_OUTPUT_TYPE

INVALVID_CONTROL_ID

Examples

0.05 20

6.4.Job Related Commands

Jobs are established to allow the function of the unit based on discrete inputs. See the user manual for more details.

6.4.1. SET-ACTIVE-JOB

6.4.2. GET-ACTIVE-JOB

6.4.3. EDIT-JOB-STEP

6.4.4. GET-JOB-STEP

6.4.5. INSERT-JOB-STEP

6.4.6. DELETE-JOB-STEP

SET-ACTIVE-JOB

Syntax

SET-ACTIVE-JOB ***JobID***

Where

JobID - the number that uniquely identifies a job. If the JobID is NONE, then no job is active and the device will not respond to inputs.

Description

Start the specified job

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_JOB_NUMBER

Examples

GET-ACTIVE-JOB

Syntax

GET-ACTIVE-JOB

Description

Returns the active job id.

Return Values

On success: *JobID* of active job.

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_JOB_NUMBER

Examples

EDIT-JOB-STEP

Syntax

EDIT-JOB-STEP JobID Step ControlID FluidRate

AirflowRate Count Delay Duration

Where

JobID - the number that uniquely identifies a job. This number will be unique for each job created.

Step - - the step in the job being set. This Step must already exist in the job

ControlID - the output to be controlled.

FluidRate – specifies the outputs rate if this is a fluid or oil output On an ABO type output this value is unused

AirRate – sets the flow rate for air, if it is a proportional air pump output. This can be a value between 000 and 100. If the output is a an oil only, ABO, or metering screw output, this field is not used

Count - The number of times to do the step. The step will be done each time the discrete input assigned to the output is received until the count is reached.

Delay –The time between when the discrete signal is received and the output is turned on at the specified rate. This can also be set to END , which has the job step start when the associated discrete turns off, and CONT for continuous on.

Duration – How long the output is active once it starts

Description

The job must have already been created through the user interface or via the CREATE-JOB command.

A definition must have already been given for that job and step, this will overwrite the previous settings.

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_JOB_NUMBER

NO_ROOM_FOR_STEP

INVALID_CONTROL_ID

COUNT_OUOT_OF_RANGE

DURATION_OUT_OF_RANGE

FLUIDRATE_OUT_OF_RANGE

AIRRATE_OUT_OF_RANGE

DELAY_OUT_OF_RANGE

Examples

GET-JOB-STEP

Syntax

GET-JOB-STEP JobID Step

Where

JobID - the number that uniquely identifies a job. This number will be unique for each job created.

Step - - the step in the job to get

Description

The job must have already been created through the user interface or via the CREATE-JOB command.

If a definition has already been given for that job and step, this will overwrite the previous settings.

If *Step* is greater than the number of steps in the job, then step is appended to the end of job. Thus the step number assigned may not match that given. To know the specific step number the returned value should be checked.

Return Values

On success: a whitespace separated string containing the ControlID to be controlled, the output rate of the fluid, the output rate of the air, the count of *t*he number of times to do the step, the delay between when the discrete signal is received and the output is turned on at the specified rate, and the duration of how long the output is active once it starts

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_JOB_NUMBER

INVALID_STEP

Examples

INSERT-JOB-STEP

Syntax

INSERT-JOB-STEP JobID Step ControlID FluidRate AirRate
Count Delay Duration

Where
JobID - the number that uniquely identifies a job. This number will be unique for each job created.

Step - - Where the step is to be inserted. All previously existing steps from this point to the end of the job will have their step numbers incremented by one. So to insert at the top use step 1. To append at the bottom use a number greater than the number of steps currently in the job

ControlID - the output to be controlled.

FluidRate – specifies the outputs rate if this is a fluid or oil output On an ABO type output this value is unused

AirRate – sets the flow rate for air, if it is a proportional air pump output. This can be a value between 000 and 100. If the output is a an oil only, ABO, or metering screw output, this field is not used

Count - The number of times to do the step. The step will be done each time the discrete input assigned to the output is received until the count is reached.

Delay –The time between when the discrete signal is received and the output is turned on at the specified rate.

Duration – How long the output is active once it starts

Description

The job must have already been created through the user interface or via the CREATE-JOB command.

If *Step* is greater than the number of steps in the job, then step is appended to the end of job. Thus the step number assigned may not match that given. To know the specific step number the returned value should be checked.

Return Values

On success: The step number assigned. In cases where it is inserted into an existing job this will match the number in the parameter. If this was inserted at the end (i.e. appended) and a number larger than the current number of steps was used then it will be the actual step number (old number of steps plus one).

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_JOB_NUMBER

COUNT_OUT_OF_RANGE

DURATION_OUT_OF_RANGE

NO_ROOM_FOR_STEP

INVALID_CONTROL_ID

FLUIDRATE_OUT_OF_RANGE

AIRRATE_OUT_OF_RANGE

DELAY_OUT_OF_RANGE

Examples

DELETE-JOB-STEP

Syntax

DELETE-JOB-STEP ***JobID*** ***Step***

Where

JobID - the number that uniquely identifies a job. This number will be unique for each job created.

Step - - The step is to be deleted. All previously existing steps from this point to the end of the job will have their step numbers decremented by one.

Description

Return Values

On success: ASCII ACK (0x06)

On error: ASCII CAN (0x18) followed by error message

Error Messages:

INVALID_STEP_NUMBER

Examples

6.5.Alarms

The alarms commands are activated with event commands that display the specific alarm message over the serial port.

1. *LOW FLUID LEVEL*

If the system is not configured with the auto refill feature, this alarm will pop up if the fluid low level switch is triggered, indicating that the fluid level in the reservoir is low.

2. *REFILL TIMEOUT*

If the system is configured with the auto refill feature, this alarm will pop up if it takes too long for the tank to refill (high level switch made) from when the low level switch is triggered. The time it should take to refill the reservoir will be set in the “Configure” display. This variable will only be visible if the system has the auto refill feature.

3. *LOW AIR PRESSURE*

This alarm will pop up if the air pressure monitor on the board reads less than the low pressure setting indicated on the “Configure” display.

4. *LOW FLOW OUTPUT ##*

If the system is configured with flow sensing on the outputs, this alarm will indicate that there is not flow with a specific output (## will be populated with the problematic output). When flow sensing is active, the flow sensing input for an output must see a change in state when the valve is cycled that strokes the pump for that particular output. If this is not seen, then the low flow alarm must pop up.

5. *INPUT TIMEOUT OUTPUT ##*

This alarm will pop up if the number of counts required to start a program step is counted before the previous step is finished. For example, a job has the settings shown below:

OUT	FLUID	AIR	CNT	DEL	DUR
1	10	90	2	5.00	8.00

In this job, output one waits until it has two inputs, delays 5 seconds, and then turns on for 8 seconds at 10 and 90. It starts again after two more inputs. If two more input signals are counted during the 13 (5+8) seconds of the step, an input timeout for output 01 alarm would occur.

When an alarm condition is present, the Alarm LED must illuminate and the active alarm will display on the alarm display with its name, date, and time, along with an “*” indicating that it is active. Also, the alarm relay must turn on while an alarm is active.

CLEAR-ALARM

Syntax

CLEAR-ALARM ***AlarmID***

Where

AlarmID – is the alarm to be cleared.

Description

Turn off the specified alarm

Return Values

On success: List of alarms currently active

On error: ASCII CAN (0x18) followed by error message

Error Messages:

Examples

EVT_ALARM_ACTIVATED

Message Format

EVT_ALARM_ACTIVATED	***AlarmID***	***State***

Description

The specified alarm has occurred or stopped