

Cadent™ 6 Syringe Pump

Hardware User Manual
Level 3

Engineering
GREAT Solutions



Cadent™ 6 Syringe Pump

Revisions Record

Date	Revision Number	Description
4 March 2020	1	Initial release

References

Title	Document Number
IMI Norgren 6cm Syringe Product Data Sheet	PDS-0010
IMI Norgren 6cm Syringe for Multi-channel Product Data Sheet	PDS-0011
IMI Norgren 6cm Valve Product Data Sheet	PDS-0032
IMI Norgren Cadent 6 Syringe Pump Product Data Sheet	PDS-0036
IMI Norgren Cadent 6 Multi-channel Pump Product Data Sheet	PDS-0037

Notice

Norgren Kloehn, Inc. maintains a continuous process of product evaluation and improvement. This is done through in-house engineering evaluations, market research, and customer feedback. As a result of this process, Norgren Kloehn, Inc. offers the most capable and sophisticated syringe pumps and valve drives in the world today. To sustain this effort, Norgren Kloehn, Inc. reserves the right to evolve and upgrade its standard products at its discretion.

Norgren Kloehn, Inc. has a policy of making every effort to ensure all upgrades are fully backwards compatible with earlier versions. However, differences may exist in internal parts, materials, or, rarely, firmware commands.

If such differences may impact your product certifications, or if a unique configuration must be maintained over long production runs, contact us to inquire about assigning a unique part identification number for your product.

Customer Service:

Phone: (702) 243-7727 or 1-800-358-4342 (1-800 FLUIDIC)

Email: IMIKloehnCustomerService@imi-precision.com

Table of Contents

1	Safety Messages.....	1
2	Pump Overview.....	2
2.1	Introduction and Specifications.....	2
2.1.1	Environmental	2
2.1.2	Mechanical.....	2
2.1.3	Power.....	3
2.1.4	Communications	3
2.1.5	I/O Interface	4
2.1.6	User Program Memory	5
2.2	Drivers	5
2.2.1	Syringe Driver	5
2.2.2	Valve Driver	6
2.3	Mounting	6
2.3.1	Mounting Surface Requirements	6
2.3.2	Base Mounting	7
2.3.3	Non-Standard Mounting Positions	7
2.3.4	Instrument Enclosures	7
2.3.5	Air and Ventilation Considerations	7
2.4	Wiring and Connectivity.....	8
2.4.1	Rear Panel Connectors	8
2.4.2	Home Button and LED	10
2.4.3	Inputs	11
2.4.4	Outputs	14
2.5	Use of Commands	14
2.5.1	General Command Structure	14
2.5.2	Command Addressing.....	14
2.5.3	Pump Responses.....	15
2.6	Communication via RS-485/232.....	15
2.6.1	The RS-232 Console and Network Bridge.....	15
2.6.2	Device Addressing	16
2.6.3	Communication Protocols	17
2.6.4	Connecting Multiple Devices (RS-485).....	22
2.6.5	Communication Checks	23
2.6.6	RS-485 Line Turnaround.....	24
2.7	Communication Via CAN Bus.....	24
2.7.1	CAN Cattro Communication	25

2.7.2	CAN Frame Types	26
2.7.3	Handling of Pump Boot Requests	27
3	Programming Techniques	28
3.1	Script Memory	28
3.1.1	Temporary Memory	28
3.1.2	Non-Volatile Memory	28
3.1.3	Saving and Erasing a Script	29
3.2	Pump Programming Tips	29
3.2.1	Programming Error Traps	29
3.2.2	Setting Syringe Movement Parameters	32
3.2.3	Counting Program Cycles	34
3.2.4	Converting Volume to Steps	34
3.3	I/O Interface Programming	35
3.3.1	Waiting for an Input	35
3.3.2	Analog Input 1 as a Selector Switch	36
3.3.3	Position Snapshots	36
3.3.4	A Binary Input Selector	37
3.3.5	Connecting the User Outputs	38
4	Pump Commands	39
4.1	Foreground vs. Background Commands	39
4.2	Syringe Commands	39
4.2.1	Position Commands	39
4.2.2	Movement Parameters	41
4.2.3	Initialization Commands	43
4.2.4	Syringe Queries	45
4.3	Valve Commands	46
4.3.1	Valve Type Setting	46
4.3.2	Valve Position Commands	47
4.3.3	Valve Queries	48
4.4	I/O Commands	48
4.4.1	Output Commands	48
4.4.2	Input Commands	49
4.4.3	The Home Button	51
4.4.4	The Tamper Pin	51
4.4.5	Handshaking Commands	52
4.5	User Program Commands	52
4.5.1	Program Storage Commands	53

4.5.2	Program Execution Commands.....	54
4.5.3	Program Control Commands.....	55
4.6	Variables.....	57
4.6.1	Setting Variables.....	57
4.6.2	Accumulators.....	57
4.6.3	Scaling Variables.....	59
4.6.4	The User Timer.....	60
4.6.5	Commands Using Variables.....	60
4.7	Configuration Commands.....	62
4.8	Query Commands.....	66
4.9	Error Trapping Commands.....	66
4.9.1	Trap Declarations.....	66
4.9.2	Trap Exits.....	67
4.9.3	Error Trap Query.....	67
4.10	Miscellaneous.....	67
4.10.1	Alternate Pin Functions.....	67
4.10.2	Flags.....	68
4.10.3	Motor Power Control.....	69
4.10.4	Repeat Command String.....	69
4.11	Command Reference.....	69
4.11.1	Flags.....	69
4.11.2	Variables.....	71
4.11.3	Standard Commands.....	74
4.11.4	Program Storage and Execution Commands.....	80
4.11.5	Configuration Commands.....	81
4.11.6	Query Commands.....	85
5	Status and Error Messages.....	88
5.1	Status and Error Messages.....	88
5.2	Status and Error LED Codes.....	90
Appendix A : Cadent 6 Syringe Pump Dimensions.....		92
Appendix B : Recommended Panel Clearance for Panel Mounting.....		93
Appendix C : Certificate of Conformity.....		94
Appendix D : Errata.....		95

Figures

Figure 2-1: Primary Connector Location and Pin Definition	8
Figure 2-2: Secondary Connector Location and Pin Definition.....	9
Figure 2-3: Home Button	10
Figure 2-4: LED	10
Figure 2-5: Analog Input Equivalent Circuit	11
Figure 2-6: Digital Input Equivalent Circuit.....	12
Figure 2-7: Tamper Input Equivalent Circuit	13
Figure 2-8: Digital Output Equivalent Circuit (Open Collector).....	14
Figure 2-9: CAN Cavo Use of Standard CAN Message 11-Bit Identifier	25
Figure 2-10: Action Command Message	26
Figure 2-11: Multi-Frame Command Message	27
Figure 2-12: Boot Response Message	27
Figure 3-1: Acceleration Example.....	32
Figure 3-2: Slow Acceleration Example	33
Figure 3-3: Binary Input Selector Example	37
Figure 3-4: Examples of User Output Connections.....	38
Figure 4-1: User Output Byte.....	49
Figure 4-2: User Input Byte	50

Tables




Table 1-1: General Safety Messages	1
Table 1-2: Safety Messages.....	1
Table 2-1: Valve and Syringe Datasheets	2
Table 2-2: Environmental Specifications.....	2
Table 2-3: Mechanical Specifications	2
Table 2-4: Power Specifications	3
Table 2-5: RS-232 Specifications	3
Table 2-6: RS-485 Specifications	3
Table 2-7: CAN Bus Specifications.....	3
Table 2-8: Digital Output Specifications (Secondary Connector)	4
Table 2-9: Digital Output Specifications (Primary Connector)	4
Table 2-10: Digital Input Specifications	4
Table 2-11: Analog Input Specifications	5
Table 2-12: Program Memory Specifications.....	5
Table 2-13: Accuracy and Precision Specifications	5
Table 2-14: Syringe Speed Specifications.....	6
Table 2-15: Primary Connector Pin Descriptions	8
Table 2-16: Secondary Connector Pin Descriptions	9
Table 2-17: Individual Device Addressing.....	16
Table 2-18: Dual Device Addressing	17
Table 2-19: Quad Device Addressing.....	17
Table 2-20: Packet Description for DT Command Protocol.....	18
Table 2-21: Packet Description for DT Response Protocol.....	19
Table 2-22: Packet Description for OEM Command Protocol	20
Table 2-23: Packet Description for OEM Command Protocol	21
Table 2-24: RS-485 Baud Rate and Termination.....	22
Table 4-1: Position Control Commands	40
Table 4-2: Handshake Dispense Commands	41

Table 4-3: Movement Parameter Commands	42
Table 4-4: Initialization Commands	44
Table 4-5: Syringe Query Commands	45
Table 4-6: Valve Type Setting Commands	46
Table 4-7: Valve Query Commands	48
Table 4-8: Output Query Commands	49
Table 4-9: Input Query Commands	50
Table 4-10: Handshaking Commands	52
Table 4-11: Program Storage Commands	53
Table 4-12: Program Execution Commands	54
Table 4-13: Jump and Label Commands	55
Table 4-14: Repeat Commands	56
Table 4-15: Accumulator 0 Commands	57
Table 4-16: Commands that Scale with Flag 33 Set	59
Table 4-17: Commands that Use Variables	60
Table 4-18: Configuration Commands	62
Table 4-19: Flag Commands	68
Table 4-20: Flags	69
Table 4-21: Variables	71
Table 4-22: Standard Commands	74
Table 4-23: Program Storage and Execution Commands	80
Table 4-24: Configuration Commands	81
Table 4-25: Query Commands	85
Table 5-1: Status/Error Messages	89
Table 5-2: LED Fault Codes	90

1 Safety Messages


The following hazard statements pertain to the Cadent 6 Syringe Pump. Read and understand these messages before continuing. Misuse or use in a manner not consistent with the User Manual is not authorized and may result in a loss of warranty coverage. General safety messages are listed in Table 1-1.

Table 1-1: General Safety Messages

	CAUTION:	Care should be taken in instrument design and usage to prevent the exposure of flammable fluids to the valve or syringe motors during operation or service due to the elevated temperatures at which these devices may operate. Fluid lines should be routed to minimize such risk.
	CAUTION:	Pump motors can reach temperatures of >65°C (149°F) during extended operation when IMI Norgren electronics are not used. It is therefore recommended that the operator use caution when working around the pump.
	CAUTION:	<p>When dispensing fluids other than water, cleaning procedures are recommended. Solutions that produce or contain particulates, such as saline solutions or the products of reactions between different solutions, should not be allowed to stand for extended periods of time. As sharp-edged particulates accumulate on the inner surfaces of the valve and syringe parts, damage to diaphragms or syringe piston seal materials may occur, greatly shortening their operating life. Saline solutions should never be allowed to stand long enough to crystallize.</p> <p>To prevent a problem with particulate accumulation, flush the pump with a buffer solution or de-ionized (DI) water after each use. Flushing is also advised between different solutions if cross-contamination is a potential issue. Sufficient buffer or DI water should be used to adequately dilute the possible residues from previous solutions.</p> <p>Do not attempt to disassemble the pump or its valve or syringe parts for cleaning. This will void the warranty and may result in damage to the assembly.</p>

Safety messages are listed in Table 1-2.

Table 1-2: Safety Messages

	CAUTION:	There are several pinch points and rotating mechanical parts. Keep hands and fingers clear of moving parts.
---	-----------------	---

2 Pump Overview

2.1 Introduction and Specifications

The IMI Norgren Cadent 6 Syringe Pump is a fluid dispense pump with exceptional volume dispense accuracy and precision. Additionally, this pump can be easily configured with multiple options to meet specific application or instrument requirements.

The IMI Norgren Cadent 6 Pump has two base models:

- Level 1 – Syringe drive
- Level 3 – Syringe drive assembled with valve and advanced controller

IMI Norgren offers an extensive list of valves and syringes for use with the IMI Norgren Cadent 6 pump. For more information on compatible valves and syringes, please review the IMI Norgren datasheets listed in Table 2-1. These documents are available on the [website](#).

Table 2-1: Valve and Syringe Datasheets

Title	Document Number
IMI Norgren 6cm Syringe Product Data Sheet	PDS-0010
IMI Norgren 6cm Syringe for Multichannel Product Data Sheet	PDS-0011
IMI Norgren 6cm Valve Product Data Sheet	PDS-0032

2.1.1 Environmental

Table 2-2: Environmental Specifications

Operating Temperature	2°C to 50°C (36°F to 122°F)
Storage Temperature	-33°C to 71°C (-27°F to 160°F)
Humidity	<80% relative humidity at 5°C to 30°C and <50% at 31°C to 50°C
Operating Noise	<55dBA

2.1.2 Mechanical

Table 2-3: Mechanical Specifications

Plunger Drive Force	12K resolution	120lbf
	24 and 48K resolution	140lbf
Panel/Base Mounting Screw Torque	339mN-m (3in-lbf), max	

2.1.3 Power

Table 2-4: Power Specifications

Voltage	24VDC nominal, 720mV peak-to-peak maximum ripple
Current Consumption	Idle: 0.07 to 0.15A Syringe or valve in motion: 1.5A (5A peak)
Power Consumption	Idle: 3W Syringe or valve in motion: 32W max
Power Backup for Backup Memory	At least 2 weeks when backup capacitor is charged

2.1.4 Communications

The Cadent 6 has three different methods of communication: RS-232, RS-485, and CAN bus.

2.1.4.1 RS-232

Table 2-5: RS-232 Specifications

Baud Rate	1200, 2400, 4800, 9600(default), 19200, 38400, 57600, 115200
Data Bits	8
Parity	None
Stop Bits	1
Handshake	None
Flow Control	None
Logical Protocols	Cavro: DT (default), OEM

2.1.4.2 RS-485

Table 2-6: RS-485 Specifications

Baud Rate	2400, 4800, 9600(default), 9200, 38400, 57600, 115200, 230400, 460800
Data Bits	8
Parity	None
Stop Bits	1
Handshake	None
Flow Control	None
Logical Protocols	Cavro: DT (default), OEM

2.1.4.3 CAN Bus

Table 2-7: CAN Bus Specifications

Baud Rate	10000, 20000, 125000, 250000 (default), 500000, 1000000
-----------	---

2.1.5 I/O Interface

User Inputs 1 through 6 and User Outputs 1 through 4 can be accessed through the Secondary Connector on the back of the pump. User Inputs 1 and 2 and User Outputs 1 through 3 can also be accessed through the Primary Connector. In addition, the push button on the PCB and the Tamper Pin on the Primary Connector can be reconfigured as User Inputs 7 and 8, respectively. The pinouts for the mentioned connectors, as seen from the back of the pump, are shown in Table 2-15 and Table 2-16. Detailed descriptions of input and output functionality are found in 2.4.

In total, the Cadent 6 provides:

- 6 digital inputs
- 1 tamper switch (reconfigurable as a digital input)
- 1 push button (reconfigurable as a digital input)
- 4 digital outputs
- 2 analog inputs

Table 2-8: Digital Output Specifications (Secondary Connector)

Output Current	300mA max
User Clamp Current	300mA max
User Clamp Voltage	5-24V
Output Voltage	48V max (external supply voltage)
Output Leakage	100µA max
Logic true/On level	0V (ground)
Logic false/Off level	Open collector

Table 2-9: Digital Output Specifications (Primary Connector)

Output Current	25mA max
Output Voltage	3.3V CMOS
Logic true/On level	3.3V (>2.4V)
Logic false/Off level	0V (<0.4V)

Table 2-10: Digital Input Specifications

Logic Compatibility	TTL, 5V CMOS
Logic true/On level	<0.95V
Logic false/Off level	>2.31V, open circuit

Table 2-11: Analog Input Specifications

Input Impedance	22.23kΩ
Max Input Voltage	5V
Resolution	12-bit, 1.22mV/LSB
Conversion Time	16.4μs
Input Filter	13.3Hz lowpass

2.1.6 User Program Memory

Table 2-12: Program Memory Specifications

User Program Capacity	Up to 99 stored scripts
User Program Size	Up to 99 commands
Program Retention	20 years minimum

2.2 Drivers

2.2.1 Syringe Driver

The syringe driver is designed to drive a syringe with a full-stroke length of 6cm and with pressures of up to 80 psi, at speeds between 0.0625 and 10,000 steps per second. The driver is available in three resolutions: 12,000 steps, 24,000 steps, and 48,000 steps for the same 6cm stroke length.

2.2.1.1 Performance

The performance of the pump is described by two parameters:

- **Accuracy** measures how closely a dispensed amount of fluid corresponds to an ideal programmed value.
- **Precision** describes the ability of the drive to deliver the same quantity of fluid for the same size programmed dispenses.

For both, the given value is expressed as a percentage of 1/10th of a syringe stroke.

Table 2-13: Accuracy and Precision Specifications

Accuracy (% error)	Precision
0.3% (1/10 th stroke), 0.2% (full stroke)	0.5% CV (1/10 th stroke), 0.05% CV (full stroke)

Additional factors that contribute to system accuracy are the total syringe size, any air bubbles or gaps, and any elasticity in the fluid path.

The syringe tolerance is a maximum of the total volume. This error contribution is proportional to the amount dispensed as a fraction of syringe volume. Air bubbles, gaps, and tubing elasticity can contribute errors due to compressibility or expansion of their volumes. Such errors are proportional to the positive or negative fluid pressures in the fluid path.

For small dispensed volumes, the accuracy of the volume can be sensitive to the means by which the volume is removed from the probe or tubing tip. Any meniscus can contribute several microliters of dispense error. To minimize these errors, submerge the tip into the destination fluid or “touch off” the tip against the container.

2.2.1.2 Speed

Syringe speeds are measured in steps per second. The definition of a step is one increment of motion in either the aspirate or dispense direction.

Table 2-14: Syringe Speed Specifications

Normal Range	5 to 10,000 steps per second
Extended Range*	0.0625 to 10 steps per second

*Extended range is achieved with the “V_” command; see 4.2.2 and Table 4-3.

2.2.1.3 Syringe Thrust and Pressure

Syringe thrust is related to syringe fluid pressure according to the following relation:

$$P = 19.35 \times (T - F)/V$$

Where:

- P is the fluid pressure
- V is the total rated syringe volume in milliliters (cc)
- T is the drive force in pounds
- F is the syringe piston friction force in pounds. Syringe friction is greater for larger syringes.

The 12,000, 24,000, and 48,000 step models can operate at up to 80psi at all speeds (see 2.2.1.2). The syringe will stall if the necessary syringe force to drive the fluid exceeds the pump’s capabilities.

2.2.2 Valve Driver

The valve driver controls a user-selected valve mounted to the faceplate of the pump. The valve driver is configurable for different types of valves. This allows the Cadent 6 Pump to accommodate any of the available valve types without modification.

2.3 Mounting

The mounting dimensions of the Cadent 6 Pump are shown in Appendix A. Recommendations for panel mounting are shown in Appendix B.

2.3.1 Mounting Surface Requirements

It is recommended that the pump be mounted to a solid base. If the pump is mounted to an instrument panel, reinforce the panel to create a stiff, rigid surface.

Where possible, use vibration isolation material between the pump and the mounting surface to improve acoustic isolation from the mounting structure.

NOTE: If these precautions are not observed, vibrations from operating the drive may result in the instrument face resonating with the pump acoustics and amplifying them.

2.3.2 Base Mounting

The Cadent 6 pump accommodates for mounting using M4 x 0.7mm tapped holes on the top and bottom and 6-32 UNC on the front of the pump. For exact positions and mounting configuration, see Appendix A.

2.3.3 Non-Standard Mounting Positions

The pump can also be mounted in an inverted orientation (upside down) or a horizontal orientation (sideways). However, when the pump is mounted in either of these orientations, trapped air in the system is likely to occur.

2.3.4 Instrument Enclosures

An instrument enclosure should have good electrical conductivity to the system's chassis ground. This reduces radiated emissions from the equipment. A wire from the pump chassis to the equipment chassis does not provide a satisfactory system ground because it does not provide the high-frequency transient conductivity required. If possible, a metallic or plastic enclosure with radio frequency (RF) shielding is preferred.

2.3.5 Air and Ventilation Considerations

The motor has been designed to operate below 90°C (194°F). This temperature is an absolute maximum. Therefore, a good, natural, convection air flow across the pump motors is recommended.

Adequate air venting for an enclosure is required for system reliability. In most applications, a large cooling air inlet at the bottom of an enclosure and an adequate hot air vent near the top can provide adequate ventilation.

NOTE: If air flow is inhibited, the motors may overheat and fail prematurely.

If a good, natural, convection air flow cannot be assured, it may be necessary to place a small fan at the lower rear part of the pump. Unloaded flow ratings of 8 cubic meters per hour or higher are sufficient. Place the fan so that it faces the motor at the bottom of the pump. Cool air flow will enter from the bottom, and heated air will exit at the top.

NOTE: Do not exceed 50°C (122°F) ambient.

2.4 Wiring and Connectivity

2.4.1 Rear Panel Connectors

There are two connectors on the back of the pump. The Primary Connector is a standard 15-pin D-Sub and is the lower connector of the two (see Figure 2-1). The Secondary Connector is a high-density, 15-pin D-Sub and is the higher connector of the two (see Figure 2-2).

The Primary Connector provides the connection to power, all communication pins, and a few input and output pins. The communication options are RS-232, RS-485, and CAN Bus. There are two inputs (digital or analog) and three outputs (digital, 3.3V CMOS).

Table 2-15: Primary Connector Pin Descriptions

Pin	Description
1	24VDC
2	RS-232 TxD line
3	RS-232 RxD line
4	Tamper
5	CAN high signal line
6	CAN low signal line
7	Input 1
8	Input 2
9	Ground
10	Ground
11	RS-485 A line
12	RS-485 B line
13	Output 1 (3.3V)
14	Output 2 (3.3V)
15	Output 3 (3.3V)

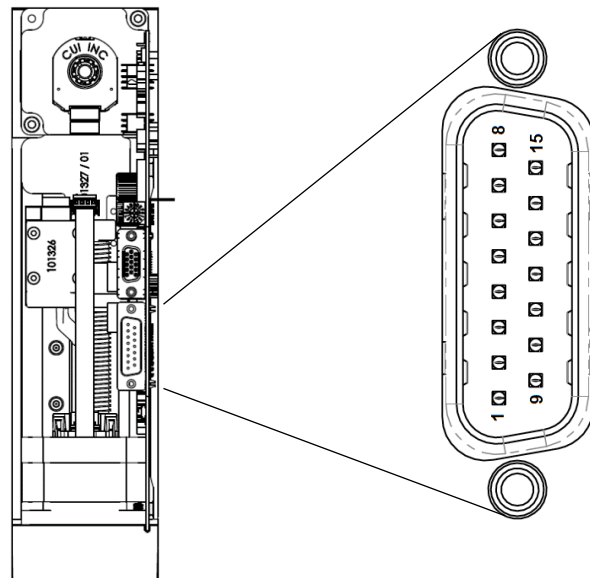


Figure 2-1: Primary Connector Location and Pin Definition

The Secondary Connector is used for inputs and outputs only. A 5V, 100mA supply is provided from the pump to support external sensors and electronics. There are a total of 6 inputs (inputs 1 and 2 can be analog or digital, whereas 3 through 6 are digital only), 4 digital outputs (open collector), and a selector pin (pin 7). The selector pin, if pulled to 5V, will convert inputs 5 and 6 to 3.3V USART pins. This can allow USART communication with external low-voltage devices using the same bus and protocol specifications given for RS-232 (see Table 2-5 and 2.6).

Table 2-16: Secondary Connector Pin Descriptions

Pin	Description
1	Clamp (5-24V)
2	Output 1 (Open Collector)
3	Output 2 (Open Collector)
4	Output 3 (Open Collector)
5	Output 4 (Open Collector)
6	5VDC
7	Selector
8	Input 1
9	Input 2
10	Input 3
11	Ground
12	Ground
13	Input 4
14	Input 5 or USART-Rx
15	Input 6 or USART-Tx

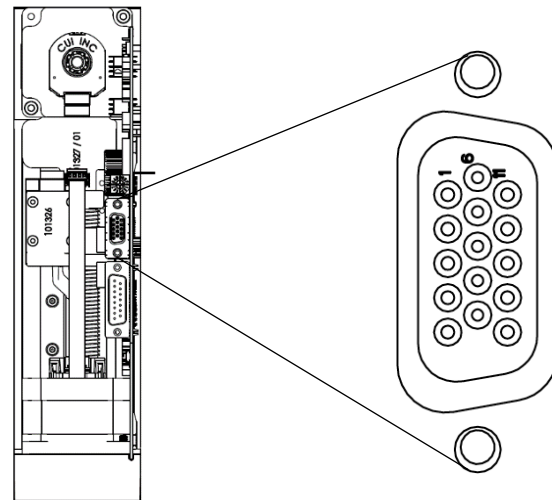


Figure 2-2: Secondary Connector Location and Pin Definition

2.4.2 Home Button and LED

2.4.2.1 Home Button

The pump includes a hardware pushbutton in the case that a software reset is unsuccessful. This button is located on the backfoot of the PCBA (see Figure 2-3).



Figure 2-3: Home Button

2.4.2.2 LED

The onboard tri-color LED is used to indicate the pump's current status, including errors (see Figure 2-4). For more information on the LED's status and error codes, refer to 5.2.



Figure 2-4: LED

2.4.3 Inputs

2.4.3.1 Analog Inputs

Two analog inputs are built into the Cadent 6. They are accessible from both the Primary Connector and the Secondary Connector on the pins designated “Input 1” and “Input 2”.

NOTE: Input 1 and Input 2 share analog and digital functionality. Therefore, the circuits described in Figure 2-5 and Figure 2-6 are both attached to these pins.

The analog inputs allow for 12-bit measurements of external analog voltages. These measurements, expressed in units of millivolts (mV) between 0mV and 5000mV, can be accessed through the variables “@3” (for Input 1) and “@23” (for Input 2). For Analog Input 1, the measured value is also available as a percentage (0-99) of 5V though the variable “@4” if flag 33 is set (value of 1). Analog Input 1 can also be used for script flow control though the Cavro commands “i>np” and “i<np” (see 4.4.2.3).

It is important to restrict the input voltage range to 5V or lower to avoid damaging the pump. Both analog inputs have the following equivalent circuit:

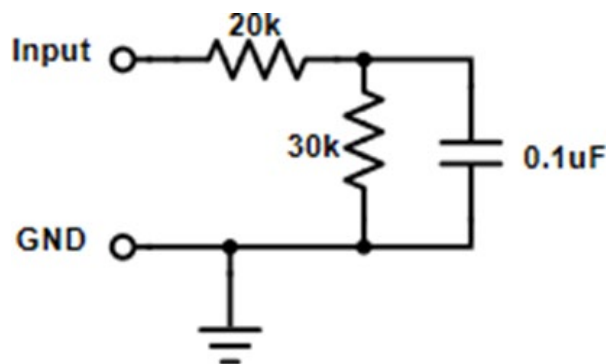


Figure 2-5: Analog Input Equivalent Circuit

NOTE: Do not apply voltages greater than 5V to a user input. Doing so can damage the circuit board.

2.4.3.2 Digital Inputs

Six digital inputs are built into the Cadent 6. They are accessible from the connectors on the back of the pump on the pins designated “Input 1” through “Input 6”. All six inputs are available on the Secondary Connector, but only Inputs 1 and 2 are available on the Primary Connector. In addition, the pushbutton on the PCB (see 2.4.2.1) and the Tamper Pin (see 2.4.3.3) on the Primary Connector can be reconfigured as Inputs 7 and Input 8, respectively. Inputs are compatible with CMOS and TTL logic operating from 5V supplies, with other pumps’ digital outputs, and with external switches. An On input is less than 0.95V. An Off input is more than 2.31V, or an open circuit. Each input can be queried at any time, including during pump operation or while an internal script is executing (see 2.4.3.4 and 4.4.2.2). Digital inputs are commonly used to control pump operation.

The effective circuit for all digital inputs is shown below.

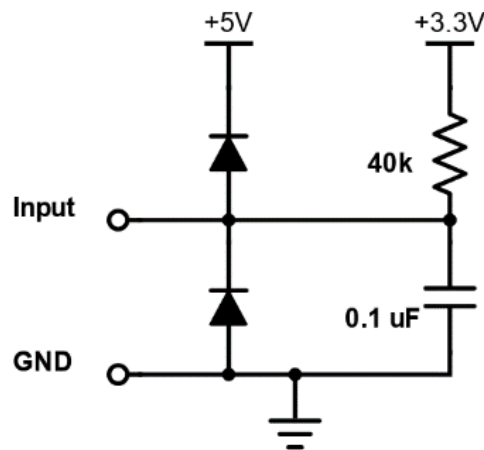


Figure 2-6: Digital Input Equivalent Circuit

NOTE: Do not apply voltages greater than 5V to a user input. Doing so can damage the circuit board.

2.4.3.3 Tamper Pin

The Tamper Pin will clear the backup memory when a normally open switch, connected to the Tamper Pin and to ground, is closed. It can also be configured as User Input 8 by setting flag 32, where it will also record the last time that the input was pulled low into the variable “@90”. The timestamp function (as well as the Tamper Pin function) will continue to work after the main power has been removed as long as the backup supply remains charged. In order for the timestamp function to work, the real time clock needs to be set. To set the time of day and the date, set the variable “@83” using the format **HHMMSS**. For example, “/1z#@83=083500” will set the time of day to 8:35:00 AM. The date can be set by setting the variable “@84” the same way using format **YYMMDD**. For example, the command “/1z#@84=150629” will set the date to June 29th, 2015.

NOTE: When the backup power has been discharged and the pump is unplugged, the real time clock will no longer be accurate and will need to be reset.

NOTE: A save (“!”) command must be sent to ensure settings are retained after a power cycle.

With the date and time set, pulling the Tamper Pin low will record the date of that event. The effective circuit for connecting to this input is shown below.

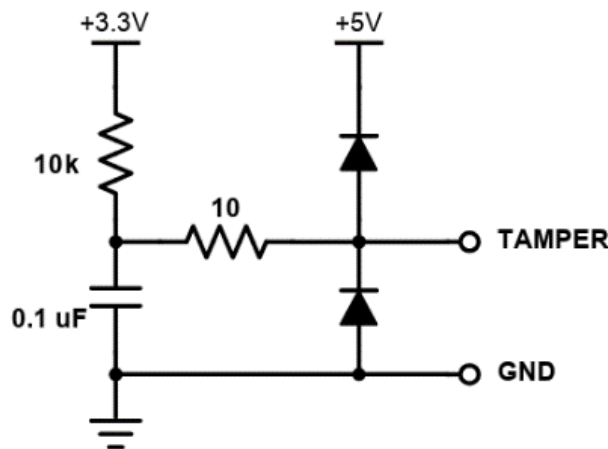


Figure 2-7: Tamper Input Equivalent Circuit

2.4.3.4 Input Triggering

For the digital user inputs, a transition from logic low to high (the falling edge from 5V to ground), triggers a logic high value to be latched. The input can go low again, but the input will be read as a “1” until the latch has been cleared. Except where otherwise noted, the user input latch is cleared when it is read or used by a command. All input latches can be reset simultaneously by clearing flag 40. Some queries or variables, such as “?4”, read the actual value at the input; generally, however, the latched value is read.

2.4.4 Outputs

Four user outputs are provided on the Secondary Connector. Each output is an open collector output that can sink up to 300mA continuously. There is also a User Clamp Pin on the Secondary Connector that can limit the output to a voltage provided by the user. The effective circuit for connecting to each digital output is shown below.

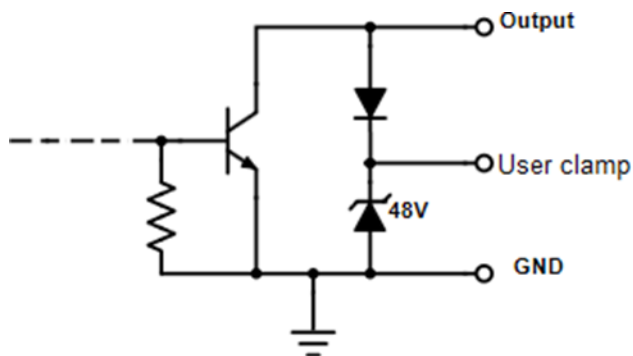


Figure 2-8: Digital Output Equivalent Circuit (Open Collector)

User Outputs 1, 2, and 3 are also accessible from the Primary Connector; there, however, each output is 3.3V CMOS (see Table 2-9).

2.5 Use of Commands

2.5.1 General Command Structure

A command is an instruction to the pump to perform a single action (e.g. to move the syringe or turn the valve). Multiple commands can be combined to form scripts. Scripts, or command strings, can perform complex tasks consisting of many operations, including decision making.

A command consists of ASCII characters and contains two parts:

- > **The Command** – A case-sensitive letter that represents a specific type of action to perform.
- > **The Argument** – Follows the command letter and determines how the command will be executed. For example, the command “**D1200**” tells the pump to dispense 1,200 steps.

2.5.2 Command Addressing

All commands and scripts must begin with a device address. The device address determines which devices will respond to a script. In this way, many devices can be connected on a single communications line without interfering with each other. The character that signifies an address is the forward slash (“/”). When the forward slash is seen by a pump, the pump reads the ASCII character that follows as an address to determine whether that pump should accept the script. An individual pump’s address is set via the address switch. For example, if the address switch is set to 3, any script that begins with “/3” will be accepted by the pump. If the string were to begin with “/2”, the pump would ignore the string. Pumps may be addressed individually or in groups. Groups can be in pairs, groups of four, or all pumps on a single communications line. The details of pump addressing are given in 2.6.2.1 and 2.6.2.2.

2.5.3 Pump Responses

When the pump receives a script, it checks the string for correctness and sends a response. The response always begins with “/0”, which is the address of the PC or controlling device. At least one character follows immediately after the “/0”. This character is the status byte. The status byte informs the controller of the current status of the pump.

2.5.3.1 Status Byte Types

The two status types are “OK” and “Error”. There is a unique letter assigned to each type of error the pump can recognize. For every error, the status letter can be upper or lower case. If the status byte is capitalized, the pump is busy doing something. If the status byte is lower case, the pump is not busy and is ready for another command.

The OK status has two special characters to indicate “busy” or “ready”. The accent grave mark (“`”) indicates a ready status, and the “at” symbol (“@”) indicates a busy status. A typical response is “/0`” or “/0@”. These responses indicate the pump and the script are OK.

2.5.3.2 Queries

A query asks the pump to report information, not perform an action. Enter a query at any time and it will be answered when it is received, even if the pump is busy.

2.6 Communication via RS-485/232

2.6.1 The RS-232 Console and Network Bridge

Although both RS-485 and RS-232 can be used as either a user console (pump-to-PC communication) or a network between up to 15 pumps, it is important to note that each bus has been optimized for a specific role. The RS-485 bus has been optimized for use as a network between multiple pumps (higher baud rates and optimized packet transfers), while the RS-232 bus has been optimized as a user console (emphasis on compatibility with common terminal programs, including character-by-character transmission).

Users can take advantage of the best features of both busses by making use of the Network Bridge. The Network Bridge is automatically enabled on a pump with address “1”. When enabled, the Network Bridge takes any commands on the RS-232 bus addressed to a pump other than “1” (including commands addressed to multiple pumps) and relays them across the RS-485 network. It also takes any responses on the RS-485 network addressed to the host (address “0”) and relays them to the RS-232 bus.

So, multiple pumps can be connected in an RS-485 network, and the user can still take advantage of the user console features of RS-232 by connecting to the pump addressed as “1” using RS-232.

NOTE: It is recommended that RS-232 be used as the user console and RS-485 be used for networking between pumps.

2.6.2 Device Addressing

The RS-485 and RS-232 buses support up to 15 devices, plus a PC host controller. Each device can be addressed individually, in pairs, in groups of four, or all at once. A response from a device only occurs for individual addressing. In the multiple device addressing modes, no device provides a status response. Status messages are saved until an individual pump is addressed.

Commands always originate from the PC host controller at address 0. They are never sent to the host.

Commands are sent using the ASCII code shown in the tables in Section 4. For example, if you wanted to have a pump with the address switch set to 1 report its status, you would send the command `/1`. Similarly, if you wanted to have a pump with the address switch set to E report its status, you would send the command `/E`.

NOTE: The first pump must be address 1 when using multiple pumps.

2.6.2.1 Individual Device Addressing

Individual device addressing is used to communicate with one pump at a time. The addresses and codes related to this mode can be seen below.

Table 2-17: Individual Device Addressing

Switch Setting	Hex Address	ASCII Character	Switch Setting	Hex Address	ASCII Character
0	Reserved for controller	Reserved for controller	8	38	8
1	31	1	9	39	9
2	32	2	A	3A	:
3	33	3	B	3B	;
4	34	4	C	3C	<
5	35	5	D	3D	=
6	36	6	E	3E	>
7	37	7	F	3F	?

2.6.2.2 Multiple Device Addressing

2.6.2.2.1 Dual Device Addressing

In the dual device addressing mode, a group of two pumps is addressed. In this mode, individual devices do not provide status responses to commands.

Table 2-18: Dual Device Addressing

Pump Group	1,2	3,4	5,6	7,8	9,A	B,C	D,E
Hex Address	41	43	45	47	49	4B	4D
ASCII Character	A	C	E	G	I	K	M

2.6.2.2.2 Quad Device Addressing

In quad device addressing, a group of four pumps is addressed. In this mode, individual pumps do not provide status responses to commands.

Table 2-19: Quad Device Addressing

Pump Group	1,2,3,4	5,6,7,8	9,A,B,C	D,E,F
Hex Address	51	55	59	5D
ASCII Character	Q	U	Y]

2.6.2.2.3 Global Mode

In the global device addressing mode, all devices on the bus are addressed simultaneously. In this mode, individual devices do not provide status responses to commands. The address character is the underscore (" _ ") or hexadecimal 5F.

NOTE: The first pump must be address 1 when using multiple pumps.

2.6.3 Communication Protocols

The communications software protocols are the command and response format used to send commands and receive responses from pumps over RS-485 and RS-232. There are two protocols:

- > **Data Terminal (DT)** – The DT protocol is a simple data terminal protocol that is compatible with nearly all terminal emulation programs and basic communications drivers. This is the preferred protocol in most situations.
- > **Original Equipment Manufacturer (OEM)** – The OEM protocol provides explicit error checking and a command sequencing algorithm. These features are not implemented in standard terminal programs.

2.6.3.1 DT Command Protocol

This section describes the command packet format of the DT protocol. A command packet is a sequence of bytes sent by a host computer to a device.

Table 2-20: Packet Description for DT Command Protocol

Byte Number	Description	ASCII	Hex
1	Start character	/	2F
2	Device address	See Sections 2.6.2.1 and 2.6.2.2	
3 to 3+N-1	Command characters	See Section 4	
3+N	End (carriage return)	<CR>	0D

Explanation of Bytes:

Byte 1: The starting character signals the beginning of the new packet. It is the forward slash character ("/"), 2F hex.

Byte 2: The device address is an address number for a device or a group of devices.

Byte 3: The command or sequence of commands starts with byte 3. A command or a command sequence with length N bytes uses byte 3 to byte 3+N-1.

Byte 3+N: The ending character indicates the end of a packet. It is 0D hex, or the carriage return.

2.6.3.2 DT Response Protocol

This section describes the device response packet format of the DT protocol. The device response packet is a sequence of bytes sent by a device to a host computer after receiving a command packet.

Table 2-21: Packet Description for DT Response Protocol

Byte Number	Description	ASCII	Hex
1	Start character	/	2F
2	Controller address	0	30
3	Status byte	See 2.5.3.1	
4 to 4+N-1	Response (if required)	See 5.1	
4+N	End-of-text	<ETX>	03
4+N+1	Carriage return	<CR>	0D
4+N+2	Line feed	<LF>	0A
4+N+3	End (blank)	<Blank>	FF

Explanation of Bytes:

Byte 1: The starting character signals the beginning of a new packet. It is the forward slash character ("/"), or 2F hex.

Byte 2: The host address, ASCII 0 or 30 hex, is the address number for the host computer.

Byte 3: The status and error byte describes the device status and errors. See 5.1.

Byte 4: The response byte(s) contain a response to the preceding command or a detailed error description, if applicable. Not all commands invoke a response.

Byte 4+N: The end-of-response mark is 03 hex.

Byte 4+N+1: The carriage return is 0D hex.

Byte 4+N+2: The end-of-packet character is the line feed character, 0A hex.

Byte 4+N+3: The extra ending character, FF hex, is there to ensure the packet is properly sent. This character might not be displayed by the host terminal.

2.6.3.3 OEM Command Protocol

This section describes the command packet format of the OEM protocol. The command packet is used to send commands from a host computer to a device. Explicit synchronization and error checking are key aspects of this protocol.

Table 2-22: Packet Description for OEM Command Protocol

Byte Number	Description	ASCII	Hex
1	Line synchronization	<Blank>	FF
2	Start transmit character	<STX>	02
3	Device address	See Sections 2.6.2.1 and 2.6.2.2	
4	Sequence number	See explanation below	
5 to 5+N-1	Command characters	See Section 4	
5+N	End of command	<ETX>	03
5+N+1	Check sum	See explanation below	

Explanation of Bytes:

Byte 1: The line synchronization character, FF hex, indicates a command packet is coming.

Byte 2: The start transmit character, 02 hex, signals the beginning of a new packet.

Byte 3: The device address is an address number for a device or a group of devices.

Byte 4: The sequence number if an error occurs during the communication. If this happens, the host sends the last packet again to the device with a new sequence number. The sequence number starts with 31 hex (ASCII). When repeating a command, the host sets bit 3 of the sequence number byte to 1 and increases the sequence number by 1. The valid sequence numbers are hexadecimal 31 for the first packet, hexadecimal 3A for the second packet (the first repeated packet), 3B for the third packet, and so forth. The maximum number of repeats is 7 with a sequence number of 3F.

Byte 5: The command or sequence of commands starts with byte 5. A command or a command sequence with length N bytes uses byte 5 to byte 5+N-1.

Byte 5+N: The end-of-command character, 03 hex, indicates the end of a command or command sequence.

Byte 5+N+1: The check sum is calculated by an exclusive OR operation on all bytes except the line synchronization byte and the check sum byte.

2.6.3.4 OEM Response Protocol

This section describes the response packet format in the OEM protocol. The response packet is used to send responses from the device to a host computer.

Table 2-23: Packet Description for OEM Command Protocol

Byte Number	Description	ASCII	Hex
1	Line synchronization	<Blank>	FF
2	Start transmit character	<STX>	02
3	Host address	0	30
4	Sequence number	See explanation below	
5 to 5+N-1	Response (if required)	See 5.1	
5+N+1	Check sum	See explanation below	
5+N+2	End (blank)	<Blank>	FF

Explanation of Bytes:

Byte 1: The line synchronization character, FF hex, indicates a response packet is coming.

Byte 2: The start transmit character, 02 hex, signals the beginning of a new packet.

Byte 3: The host address, 30 hex, is the address for the host computer.

Byte 4: The sequence number if an error occurs during the communication. If this happens, the host sends the last packet again to the device with a new sequence number. The sequence number starts with 31 hex (ASCII). When repeating a command, the host sets bit 3 of the sequence number byte to 1 and increases the sequence number by 1. The valid sequence numbers are hexadecimal 31 for the first packet, hexadecimal 3A for the second packet (the first repeated packet), 3B for the third packet, and so forth. The maximum number of repeats is 7 with a sequence number of 3F.

Byte 5: The response byte(s) contain a response to the preceding command or a detailed error description, if applicable. Not all commands invoke a response.

Byte 5+N: The end-of-response mark, 03 hex, indicates the end of the response.

Byte 5+N+1: The check sum is calculated by an exclusive OR operation on all bytes except the line synchronization byte and the check sum byte.

Byte 5+N+2: The extra ending character, FF hex, is there to ensure the packet is properly sent. This character might not be displayed by the host terminal.

2.6.4 Connecting Multiple Devices (RS-485)

Up to 15 devices can be connected to the same RS-485 communications bus. The bus consists of three wires: A, B, and a signal ground. A proper bus structure consists of the twisted-pair bus wiring and termination resistors at each end of the bus.

2.6.4.1 Bus Wiring

The bus wiring should connect all RS-485 A pins to one wire, all B pins to another wire, and all ground pins to a third wire. The connections begin on the device with address 1 and proceed to the other devices on the bus.

2.6.4.2 Bus Terminations

Each end of the bus must be terminated in a resistor network. Terminations are made only on the first and last devices along the bus. Terminating networks are provided internally on each pump and are enabled or disabled in the firmware using the “~Bn,t” command. In the command, n sets the baud rate for the RS-485 network, and t sets the termination.

Table 2-24: RS-485 Baud Rate and Termination

n	Baud Rate	t	Termination Mode
1	38400	0	Termination off
2	19200	1	Termination on
3	9600 [default]	2	Termination off, except nodes 1 and 15 [default]
4	4800		
5	2400		
6-8	Invalid		
9	57600		
10	115200		
11	230400		
12	460800		

Changing the termination mode requires a reset after the change has been saved to non-volatile memory (NVM) using the “!” command. This allows you to turn the pumps off and change the order or addresses to be properly terminated.

NOTE: Only the pumps at each end of the RS-485 bus should have termination on. All pumps between the two must have termination off.

Since the default has termination off for all pumps that are not assigned the addresses 1 or 15, an easy way to terminate a planned connection of pumps is to disable termination for 1 and 15 if they are in the group (and not at the ends) by connecting them individually via RS-232 and using “~Bn,t” and “!”. Then, individually connect the pumps that are to be terminated, and enable termination on them using “~Bn,t” and “!”. They can then be connected as planned.

2.6.5 Communication Checks

This section presents some procedures to determine whether a device is communicating with a host controller using RS-232. The checks are predicated upon the use of some form of terminal emulator program running on a PC.

Before using such a program, check the address switch to make sure the pump is set to the desired address. The address switch setting determines the value that must be substituted for the notation <addr>. Each new script must begin with a forward slash ("/") followed by the value of <addr>. Each script must end with a carriage return (the Return or Enter key). See Section 4 for specific command syntax.

After the communications wiring is connected and the PC serial port cable has been connected to the first device on the bus, turn on the pumps. When the power-up valve move is complete, send a query for the module status, as shown below.

For example, enter `/1` to query the status of pump 1. The response is `/0``, which indicates "not busy, no errors".

If a response occurs, communications are operating properly. A valid response from a communicating module always begins with `/0`. This is the default address of the host controller. If there are no errors to report, the next character after the `0` is either an accent grave (``) or `@`. The accent grave signifies that the module is not busy and is ready to accept any commands. The `@` signifies that the module is busy, and therefore, only queries and the terminate (`T`) command are acceptable.

If the query is sent while the power-up initialization sequence is in progress, no response occurs. The pump does not accept commands until the power-up sequence is completed. If either an accent grave (``) or `@` is not returned and a letter is returned in its place, then the module is reporting an internal error condition (see 5.1).

If there is no response, then the pump is not powered up, the communications hardware connection is not properly made, or the communications program is either not properly configured or not operating correctly.

Check the following items:

- > Make sure that the communications connector is inserted properly into the RS-232 connector and not into the RS-485 connector.
- > Try another COM port selection.
- > Using a voltmeter with the communications cable connected to the RS-232 pins on the Primary Connector, measure voltages from the ground pin to the RS-232 RxD pin and the RS-232 TxD pin. Each should measure -6 VDC to -15VDC. If they do not, the following errors may exist:
 - If RS-232 RxD fails the check, the host PC port is not functioning, or the communications cable is either defective, or plugged in backwards.
 - If RS-232 TxD fails the check, either the RS-232 converter board or the communications cable is defective, or the module is not powered.

2.6.6 RS-485 Line Turnaround

The line turnaround configuration command (“~Dn,m”) is used to modify the line turnaround delays for the RS-485 serial port. The first parameter determines the delay for switching from transmit to receive mode after the end of an incoming message. The second parameter sets the delay when switching from receive to transmit mode after the last bit of an outgoing message has completed.

This second parameter is important for compatibility with older V-Series pumps or the IMI Norgren 50500 RS-232 to RS-485 converter board. The RS-485 interface on legacy products uses a 12ms hardware timer to hold the transmission line open after the last bit of a message has been sent. If the Cadent Controller attempts to send a message before the 12ms timer has expired, there will be a collision with two transmitters enabled on the RS-485 bus.

If there are no legacy RS-485 interfaces on the bus, then the receive-to-transmit delay should be slightly longer than the transmit-to-receive delay. The Cadent Controller ensures the last bit is always completed before a line turnaround in either direction.

When transmitting a data packet to the pump on an RS-485 port, the interframe timeout should not exceed a max of 10ms at the slowest baud rate and a minimum of 1.75ms at the fastest baud rate. Or, the interframe timeout should be between 1.75ms and 10ms, depending on baud the rate used.

2.7 Communication Via CAN Bus

The Controller Area Network (CAN) bus is a two-wire multi-drop bus that connects several nodes together. The IMI CAN Cavo protocol is used to send Cavo commands over CAN.

The Cadent 6 is connected through a D-Sub connector when operating from a CAN controller. The Cadent 6 is connected through pins 5 (CAN high) and 6 (CAN low), as shown in Table 2-15. The CAN interface allows a daisy chain connection with one host and multiple slave pumps. There is no peer-to-peer communication.

The Address Switch is set such that the first pump address is set to “1” and the second pump address is set to “2” for up to fifteen devices. Communication termination is set internally and can be configured using Cavo commands, but the defaulted termination is set at address 1 and F.

The protocol uses the 11-bit identifier, as shown in the Figure below.

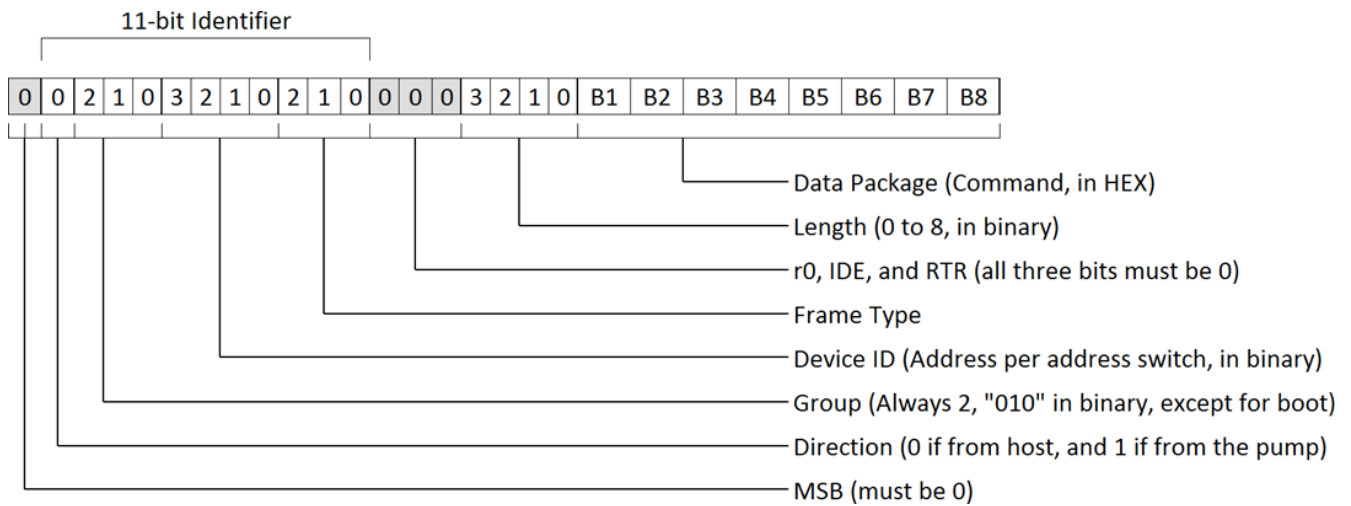


Figure 2-9: CAN Cavo Use of Standard CAN Message 11-Bit Identifier

CAN Message Frame Fields:

- **SOF** – start of frame (denoted as MSB above)
- **Identifier** – standard CAN 11-bit identifier
- **RTR** – remote transmission request
- **IDE** – identifier extension indication (used to indicate 11- or 29-bit identifiers)
- **r0** – reserved
- **DLC** – data length code; indicates the number of data bytes that follow
- **CRC** – cyclic redundancy check
- **ACK** – ack frame where receiving nodes acknowledge receipt of the message

2.7.1 CAN Cavo Communication

CAN Cavo is a protocol used to send Cavo commands to equipment that supports the Cavo command language. The CAN Cavo specification uses a master/slave relationship with one master on the bus. The 11-bit identifier is used to identify the slave node, the direction (to/from master), and the message type. The slave node is identified via a group number and device ID. The message type is indicated by the frame type. Peer-to-peer messaging is not supported.

2.7.1.1 Device

This is the address of the module in the particular group. Each group can have up to 15 devices. The address value can range from 1 to 15 (1 through F in hex).

2.7.1.2 Frame

This lets the device know what type of message is coming.

2.7.1.3 Group Number

This lets the host know which slave node is being communicated with.

2.7.2 CAN Frame Types

The frame types allow each device to know what type of command is coming in and enables faster processing of commands. Pumps respond to the frame types described below. Normal commands, such as action commands, use a frame type 1. Commands with a particular frame type must be completed before the next command using the same frame type can be issued.

2.7.2.1 “On-the-Fly” Commands, Type 0

Frame type 0 must be used when issuing “on-the-fly” commands.

2.7.2.2 Action Frames, Type 1

This frame type is used for action commands, such as initialization commands, movement commands, or valve commands, or to set pump operating parameters. All task-type commands are sent in this type message frame. When multi-frame messages are used to send an action command, this frame is the end message sent to the pump.

The following is the message that would be sent to a pump with address 1 and the action “A100R”.

0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	41	31	30	30	52			
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	--	--	--

Figure 2-10: Action Command Message

2.7.2.3 Common Commands, Type 2

This frame type is used for commands that are common to every device on the bus. The frame type is set to 2, and the command is a single ASCII character in the data block. The single ASCII character is described below.

2.7.2.4 Multi-Frame Start Message, Type 3

This frame type lets the pump know that the message will be longer than the 8-byte maximum for each frame. Subsequent frames will follow to complete the message.

2.7.2.5 Multi-Frame Data, Type 4

This frame type is used to identify a frame in the middle of a multi-frame message. The last frame of a multi-frame message for action commands must be type 1. The last frame of a multi-frame message response from the pump for report commands will be type 6.

A complete multi-frame command consists of up to three frame types: 3 (the start), 4 (the middle, if required), and 1 (the end/standard action). The following is an example that uses all three frame types to send the command “V2000A0go1A6000M2000o2A3000M2000o3A0G3R”:

MSG#1	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	56	32	30	30	30	41	30	67
MSG#2	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	6F	31	41	36	30	30	30	4D
MSG#3	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	32	30	30	30	6F	32	41	33
MSG#4	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	30	30	30	4D	32	30	30	30
MSG#5	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	6F	33	41	30	47	33	52	

Figure 2-11: Multi-Frame Command Message

2.7.2.6 Report/Answer Commands, Type 6

This frame type is used to get information back from the pump. It is similar in operation to the query commands (i.e. [?]) used in the OEM and DT protocols. The report command is one byte long and consists of one or more ASCII characters in the data block.

2.7.3 Handling of Pump Boot Requests

At power-up, the pump notifies the host that it is online by sending a boot request message at 100-millisecond intervals until it receives a boot response from the host. The group number is 1 for the boot request message. The frame type is 2 when the pump sends messages to the host.

The message below should be sent by the host as the boot response if the address is 1. The data package needs to be the address+32 then converted to hex and repeated twice (2F for the address of F).

0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	21	21								
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	----	----	--	--	--	--	--	--	--	--

Figure 2-12: Boot Response Message

3 Programming Techniques

3.1 Script Memory

The Cadent 6 has the ability to store and execute scripts. A script is a group of sequential commands without spaces that form a single line of legal ASCII characters.

For example, the following commands

I Move valve to input position
A0 Move syringe to fully closed position
A3000 Fill syringe
M500 Delay 500 milliseconds
O Move valve to output position
D1500 Dispense half of syringe

can be placed into a script as follows:

IA0A3000M500OD1500

In a script, each command is executed immediately after the preceding command has completed. The script runs in the minimum possible time without the need to query the drive to determine whether it is busy or ready for the next command, which eliminates a lot of communications overhead. Scripts can be executed immediately or stored for later execution. They can be executed from temporary memory (RAM) or from NVM.

3.1.1 Temporary Memory

When a script is sent to the pump, the string is stored in temporary memory (RAM). This memory retains its contents while power is applied to the pump. When power is removed, the contents of RAM are lost. After a script is executed, it can be repeated by sending the “**X**” command.

To execute a script immediately, append an “**R**” (run) command to the string. If no “**R**” command is appended, the script will be executed when a subsequent “**R**” command is sent. If another script is sent after the script and before the “**R**”, or if the “**R**” is appended to another script, the original script will be overwritten by the new script.

For example, the script “**IA0A3000M500OD1500R**” is executed as soon as it is received by the pump. The script “**IA0A3000M500OD1500**” is stored into RAM but is not executed until a separate “**R**” is sent. Only one script is stored in RAM at a time.

3.1.2 Non-Volatile Memory

Non-volatile memory (NVM) retains its contents for at least 20 years without power. Up to 99 scripts can be stored in the NVM. There is no practical limit on the number of times a stored script can be read or executed.

NOTE: The maximum number of saves to the NVM is 100,000. After this, the integrity of a saved script cannot be guaranteed.

3.1.3 Saving and Erasing a Script

A script is saved into the NVM by sending it to the RAM without an “**R**” command appended and then sending the “**En**” command as a separate command. When the “**En**” command is received, the string in RAM is transferred into the NVM and stored as script number *n*, where *n* is a number between 1 and 99. To erase a script in NVM, either overwrite it with another script or send the “**en**” (erase) command.

Because of the limitation on writes, NVM should not be written every time a script is run. Use it to store a script if that script will be long-lived in the application. Short-term scripts or scripts that can change often should be executed from RAM. Some scripts that vary in terms of their numbers but not in their structure can use accumulators (see 4.6.2).

NOTE: A save (“**!**”) command must be sent to ensure settings are retained after a power cycle.

3.1.3.1 Listing a Script

A script saved into NVM can be queried with the “**qn**” (script query) command. When the “**qn**” command is received, the pump responds by sending the complete script, if any, found in the NVM. If no script is found, a period is returned after the status byte.

3.1.3.2 Auto-Starting an NVM Program

The Cadent 6 has the ability to automatically begin executing a script stored in NVM when power is applied. This feature is known as Auto-Start. Auto-Start is useful for those applications that may require rapid and automatic pump initialization or in cases where a script sequence is controlled with user inputs. The Auto-Start feature is enabled by setting the “**~An**” parameter to the script number that the pump should start with. The Auto-Start can be disabled by setting the “**~An**” parameter to 0 with the “**~A0**” command. These commands do not require the “**R**” command. Any one of the 99 stored scripts can be used with Auto-Start.

When writing an Auto-Start script, remember that, if the syringe position has changed since the last time the pump was powered or the backup power has been lost, the syringe cannot be commanded to move until it has first been told to initialize with a “**W4**”, “**Y4**”, or “**Z4**” command. However, all commands other than a syringe move command can be executed before (or without) executing a “**W4**”, “**Y4**”, or “**Z4**”.

NOTE: A save (“**!**”) command must be sent to ensure settings are retained after a power cycle.

3.2 Pump Programming Tips

This section offers techniques for programming the pump using scripts. These techniques extend the usefulness of the pump.

3.2.1 Programming Error Traps

Errors can occur during operation of the pump. Errors can range from incorrect commands to motor overloads. The pump can detect many error conditions. For some systems, the robustness of the design can be enhanced by programming the pump to take corrective action automatically. This is called “trapping”, and the part of the user script designed to handle the error is called the “error handler” or “exception handler”. See 4.9 for more details on error trapping.

3.2.1.1 Error Trapping Example

Trap a syringe overload error. Save the value of the syringe position, initialize the syringe and valve, then return to the stall position and continue the dispense cycle. This assumes the dispense was intended to deliver all the contents of the syringe.

In the main user script, the trap is set by declaring “**x9V**”. If a syringe overload occurs, go to label “**V**”. Error handlers are normally located at the end of the script. The handler might consist of the following:

- :V** Identifies the start of the handler (label used in the trap instruction above).
- k@7** Stores the current syringe position in Accumulator 0 for later use.
- k^1** Exchanges the value of Accumulator 0 with Accumulator 1.
- k@6** Saves the current valve port position in Accumulator 0.
- Y4** Initializes the syringe through the reservoir port. The port is set by the “**~Yn**” command.
- o@5** Moves the valve back to the previous port position.
- k^1** Places the previous syringe position back in Accumulator 0.
- A@5** Moves the syringe back to the stall position.
- A0** Completes the dispense.
- t1** Resumes the script execution with the next instruction.

3.2.1.2 Limitations

There are some limitations on error trapping. The error trapping feature is designed to provide a graceful recovery from errors, but it cannot fix system errors. Any error induced by mechanical or fluidic problems cannot be fixed by a script. Such things must be fixed at the external root cause. To prevent such external problems from causing the pump to enter an infinite loop, an error cycle counter can be used, or the user outputs can be used to send error information to the external system.

Example:

Recover from a valve overload by initializing the valve and then repeating the valve move. An “error cycle counter” is included to prevent a run-away loop. In the main script, zero Accumulator 2 and declare a trap, as outlined below.

- k^2** Exchanges Accumulator 0 with Accumulator 2 (preserves the k value).
- k0** Sets Accumulator 0 to zero.
- k^2** Restores Accumulator 0 and places the zero in Accumulator 2.
- x10p** If a valve overload (error 10) occurs, goes to script label “**p**”.

At the end of the script, where error handlers are normally located, the handler might be as follows:

- :p** Identifies the start of the handler (label used in the trap instruction above).
- o1** Initializes the valve position.
- k^2** Gets Accumulator 2.
- k+1** Increments the error loop counter.
- k>5q** If there are more than 5 valve errors, goes to label “**q**”.
- k^2** If not, restores the previous accumulator value.
- t4** Tries the valve move again.
- :q** Label “**q**” means there are more than 5 valve errors (from “**k>5q**”).
- k^2** Restores the previous accumulator value.
- U3** Signals a terminal error (via User Output 3).
- t3** Does a normal script error exit (stops the script and makes an error message).

In the preceding example, the “t4” exit retries the instruction that had caused the error. If the initialize move works but there are more than 5 retries without success, the handler exits the script after setting an external “error” signal. If the initialize move fails, a script exit occurs. If an error occurs while in the error handler routine, a normal error exit takes place, superseding any user error handler.

NOTE: If an error occurs while an error handler is executing, the script will perform a normal script error exit, regardless of the error handler.

Example:

If any error occurs, set User Output 2 to low, set User Output 1 to high, save the current Analog Input 1 input value, and then exit the script.

Set the error trap in the main script, as follows:

x*s if any error (“*”) occurs, go to script label “s”.

At the end of the script, where error handlers are normally located, the handler might be:

:s Marks the start of the error handler.

U2 Sets User Output 2 to low.

u1 Sets User Output 1 to high.

k@3 Saves the current Analog Input 1 value.

t3 Does a normal error exit.

3.2.2 Setting Syringe Movement Parameters

The syringe uses three speeds and two accelerations that can be set. The speeds are start speed, top speed, and stop speed. The commands “Ln” and “In” set the acceleration and deceleration rates, respectively. Information on the commands for setting each of the movement parameters as well as the limits on their values can be found in 4.2.2.

The syringe motor does not start at a speed of zero and accelerate smoothly to the top speed (at which syringe moves normally occur). Rather, the motor jumps abruptly from zero to the start speed and then accelerates smoothly to the top speed. The move proceeds at the top speed. As the destination is approached, the motor decelerates from the top speed to the stop speed. When the stop speed is reached, the motor performs an abrupt stop at the target position. The speed profile is thus trapezoidal.

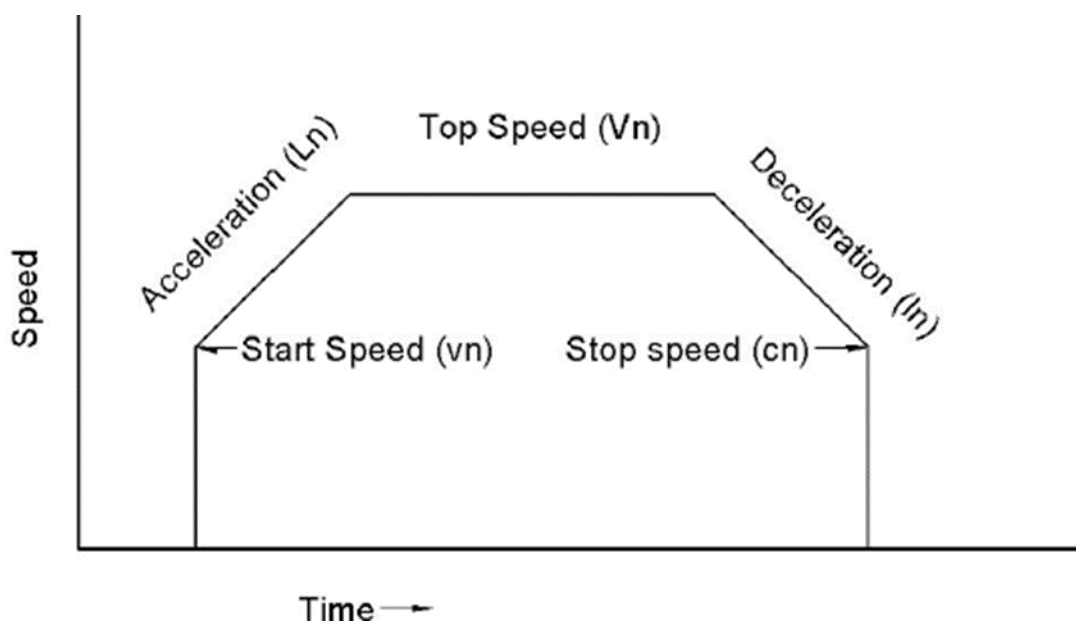


Figure 3-1: Acceleration Example

If a high top speed and low acceleration are combined with a very short move, the syringe speed may not reach the programmed top speed, and the profile of the following figure will result.

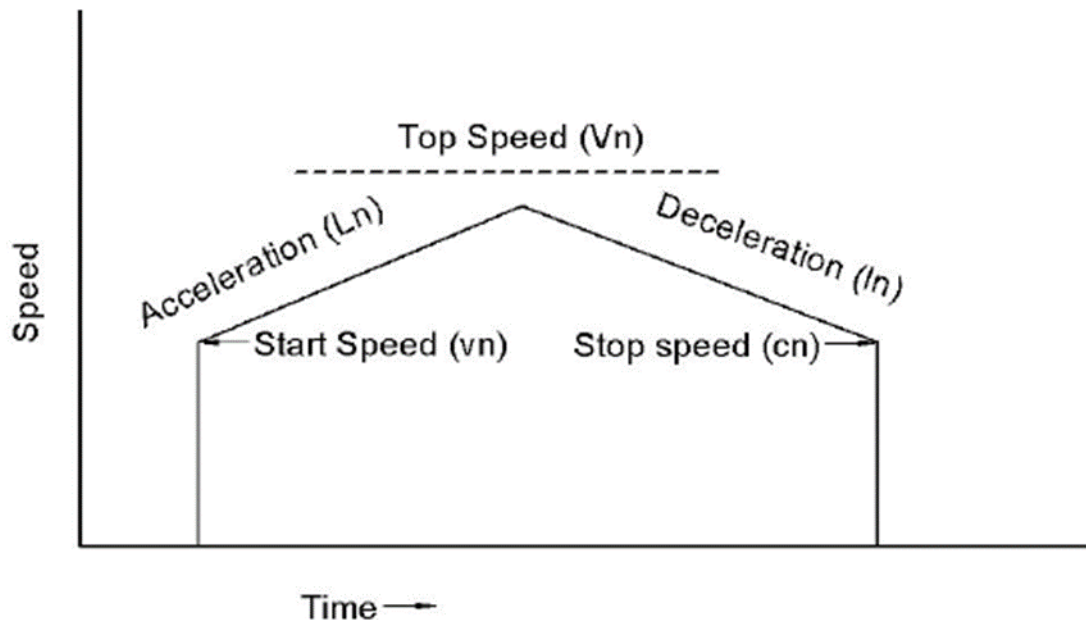


Figure 3-2: Slow Acceleration Example

In practice, a typical move spends nearly all its time at the top speed, and the acceleration and deceleration are very small parts of the total move.

For most applications, the factory default values for syringe accelerations and speeds are adequate. Usually, only the top speed is changed. If the top speed is set lower than the start speed, the pump will begin a move at the top speed. If the top speed is set lower than the stop speed, the move will end at the top speed. For this reason, values of top speed that are set lower than either the start speed or the stop speed do not require any adjustment in start speed or stop speed.

NOTE: If the acceleration setting exceeds the power capability of the syringe motor it may generate a syringe overload error.

3.2.3 Counting Program Cycles

The number of times a given event has occurred can be determined with the Software Counter (also referred to as “Accumulator 0”). In the example below, the number of times a programmed dispensing sequence has occurred is counted so a controller can query the pump to determine the number of dispenses that have occurred since the script was initiated.

Example: Count the number of times a dispense has been made in an automated dispensing cycle.

k0 Sets the counter to zero (start of dispense script).
:B Marks the start of the filling of the syringe.
o-1 Moves the valve to the reservoir port.
A24000 Fills the syringe (24000-step model).
o3 Moves the valve to the dispense port.
:A Marks the start of the dispense loop.
y<1500B If there is not enough left to dispense, refill (goes to label “B”).
D9600 Dispenses 40% of a syringe (24000-step model).
k+1 Increments the software counter.
JA Does another dispense loop.

In the example above, the script begins by setting the counter to zero. The syringe is refilled each time the syringe position is too small to do another dispense. After each dispense, the counter is incremented by 1. The counter can be queried at any time to read how many dispenses have occurred. The complete script is:

k0:Bo-1A24000o3:Ay<1500BD9600k+1JA

3.2.4 Converting Volume to Steps

The conversion of syringe volume to steps (syringe increments) can be easily done using the following proportion:

$$\text{Steps required} / \text{Total steps in full stroke} = \text{Desired volume} / \text{Total volume of full stroke}$$

For example, assume a total syringe volume of 5mL (5,000µL), a desired volume of 250µL, and a 12,000-step resolution pump. The required number of steps can be found by solving:

$$\begin{aligned}\text{Steps required} &= (250\mu\text{L} / 5,000\mu\text{L}) \times 12,000\text{steps} \\ \text{Steps required} &= 600\text{steps}\end{aligned}$$

The preceding proportion can also be used to find speeds if the “steps required” is replaced with “steps per second” and the “desired volume” is replaced with “volume per second”.

For example, assume a total syringe volume of 5mL (5,000µL), a desired .5mL per second (500µL/second) dispense, and a 12,000-step resolution pump. The required speed, in steps per second, can be found by solving:

$$\begin{aligned}\text{Steps per second} &= ((500\mu\text{L}/\text{sec}) / 5,000\mu\text{L}) \times 12,000\text{steps} \\ \text{Steps per second} &= 1,200\text{steps}/\text{sec}\end{aligned}$$

3.3 I/O Interface Programming

User inputs and user outputs (I/O) can be used to perform a variety of interfacing tasks (see Sections 4.4 and 4.10.1). In stand-alone operation, without a serial communications controller, the I/O can be used to trigger script operations and to indicate operational status. In multiple pump operations, the I/O can be used to coordinate and synchronize the operations of the pumps. One special case is the synthesis of continuous fluid flow using two pumps. This section provides some interfacing techniques that can aid in using the pump I/O.

3.3.1 Waiting for an Input

In many applications, it is desirable for a pump to wait for an external input signal to start an operation. Test-and-jump commands are used to sense the state of an input signal at a user input and control the script operation.

There are two basic scenarios:

- ➔ Wait for a high input level
- ➔ Wait for a low input level

Example: Wait for a high input level

```
:A Program label "A"
i2A If User Input 2 is low, goes back to label "A"
```

When User Input 2 becomes high, the "i2A" instruction is false, and the jump back to label "A" is not taken. The next instruction in the line is executed. It should be noted that user inputs have internal pull-up resistors; thus, in the absence of a signal connection, the default input level is high.

Example: Wait for a low input level

```
:r Program label "r"
i3T If input 3 is low, goes to label "T"
Jr Always jumps to label "r"
:T Label "T"
```

While the input is high, the test for User Input 3 to be low fails, and the jump to "T" is not taken. The next command, "Jr", is therefore executed. The "Jr" command sends the script execution back to the "r" label, and the test is repeated. When User Input 3 becomes low, the jump to label "T" is taken. Label "T" then leads to the next instruction in the script.

3.3.2 Analog Input 1 as a Selector Switch

Analog Input 1 can be used as a selector switch. If repeatable voltage levels can be applied to Analog Input 1, a series of tests can be made to determine what the script should do next.

For example, assume six discrete voltage levels at 0, 1, 2, 3, 4, and 5 volts. The test thresholds are at 0.5, 1.5, 2.5, 3.5, and 4.5 volts for a total of five possible selections (and a default state). The actual test numbers are in terms of millivolts.

Selection 1: 0.5 volts = 500

Selection 2: 1.5 volts = 1500

Selection 3: 2.5 volts = 2500

Selection 4: 3.5 volts = 3500

Selection 5: 4.5 volts = 4500

The test sequence uses the numbers calculated above.

:A Labels that mark the start of the test loop.

i>4500B If the selection is 5, goes to label “**B**” (start of selection 5).

i>3500C If the selection is 4, goes to label “**C**” (start of selection 4).

i>2500D If the selection is 3, goes to label “**D**” (start of selection 3).

i>1500E If the selection is 2, goes to label “**E**” (start of selection 2).

i>500F If the selection is 1, goes to label “**F**” (start of selection 1).

JA If there is no selection, tests the loop again.

The command sequence at each selection’s label must end with a jump back to label “**A**” (“**JA**”) so that the selection process continues when each selection is done.

3.3.3 Position Snapshots

The pump provides a means to record the precise location of the syringe at the time of an external event. Although the syringe position can be queried while the syringe is in motion, the communications overhead prevent the exact syringe position from being determined through “on-the-fly” position queries.

The snapshot feature overcomes these limitations and provides exact measurements. The snapshot feature uses the input pin assigned using the “**~Mn,m**” command and is enabled or disabled with flag 14. Each time the designated input transitions from high to low, the current position of the syringe is stored in memory. The snapshot memory retains the four most recently captured positions in a stack. Each time a position is captured, it is placed at the top of the stack. Every time a position is queried, it is removed from the top of the stack. If the stack is full, the oldest position is discarded. The two most recent positions can be queried at any time with the “**?29**” command. The most recent captured position is also available in the variable “**@110**”. When the memory is empty, -1 is reported for the position.

3.3.4 A Binary Input Selector

In some applications, one of several selections must be made via a selector switch or PLC logic lines. The most economical approach is to encode the inputs as binary numbers. This section describes a way to program a binary selection tree. This is a way of making the input bits act as if they have a binary weighting (1, 2, 4, etc.). This illustration uses three inputs as a seven-way selector. Figure 3-3 shows a flow chart of the algorithm.

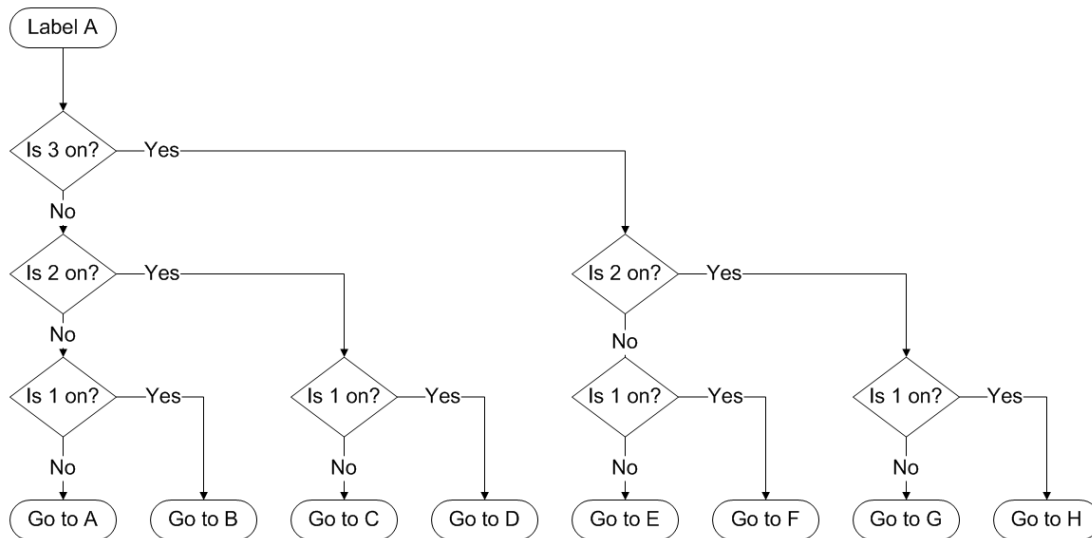


Figure 3-3: Binary Input Selector Example

The corresponding commands are:

:A Label "A" marks the start of the selection tree.
i3K Is User Input 3 On? If yes, goes to label "K".
i2L Is User Input 2 On? If yes, goes to label "L".
i1B Is User Input 1 On? If yes, goes to label "B".
JA Goes to label "A" (no selection).
:L Label "L".
i1D Is User Input 1 On? If yes, goes to Label "D" (selection = D).
JC Goes to label "C" (selection = C).
:K Label "K" (second half of binary tree).
i2M Is User Input 2 On? If yes, goes to label "M".
i1F Is User Input 1 On? If yes, goes to label "F" (selection = F).
JE Goes to label "E" (selection = E).
:M Label "M".
i1H Is User Input 1 On? If yes, goes to label "H" (selection = H).
JG Goes to label "G" (selection = G).

Each label from "B" to "H" is the start of the command sequence that corresponds to the input selection. Each selection command sequence must end with a "JA" (jump to label "A") so that the input selection procedure can continue when a process is done. Note that selection A goes back to the start of the selection tree. This is done to provide a "no selection" position. If some other means of initiating the selection process (other than a valid selection) is used, then label "A" can be used for an eighth selection.

3.3.5 Connecting the User Outputs

User Outputs 1, 2, 3 and 4 on the Secondary Connector are suitable for driving a variety of loads. These open-drain MOSFET outputs can drive up to 300 milliamps (mA) of current and withstand voltages of 48V, with peaks up to 70V.

3.3.5.1 Inductive Loads

Each user output on the Secondary Connector includes integral protection against inductive turn-off transients (see Figure 2-8). The Zener diodes trigger at about 48 volts across an inductive load during the current decay to speed up the inductive load turn-off.

3.3.5.2 Logic Loads

Logic loads, such as CMOS and TTL inputs, require a pull-up resistor to the logic supply voltage, as shown in Figure 3-4. Other loads, such as indicator lights, relays, optoisolators, and solenoids, usually operate from higher voltages. The connection for such loads is also shown in Figure 3-4. Note that the load is connected from the output to the load and from the load to the load supply. The ground connection from the load supply ground to the pump ground is essential.

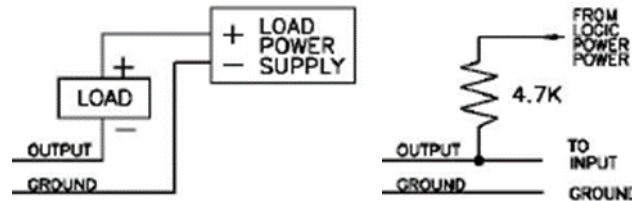


Figure 3-4: Examples of User Output Connections

NOTE: External pull-up resistors are unnecessary when connecting a user output to a user input on another Cadent pump, as all user inputs have internal pull-up resistors (see Figure 2-6).

3.3.5.3 Outputs on The Primary Connector

The user outputs available on the Primary Connector are 3.3V CMOS outputs. They are designed to allow the pump to interface with logic level external circuits. They should not be used to drive inductive loads or logic loads in excess of 20mA. Their On state is 3.3V, and their Off state is 0V. Commands that affect user outputs (see 4.4.1) affect outputs on both connectors; e.g. “U1” sets pin 2 on the Secondary Connector to 0V and pin 13 on the Primary Connector to 3.3V.

NOTE: Because the logic states for user outputs on the Primary Connector are the inverse of those on the Secondary Connector, Primary Connector outputs cannot be used for certain dedicated I/O functions, such as handshaking.

4 Pump Commands

This chapter presents all commands supported by the Cadent 6 pump. When reviewing the commands in the following sections, keep in mind the following:

- Values in parenthesis () indicate the range of values.
- The notation @n signifies that the argument is a variable.

4.1 Foreground vs. Background Commands

All commands are run in either the “foreground” or the “background”. Commands that are executed in the foreground are called “immediate commands”. When immediate commands are sent to the pump, they are executed as soon as they are read.

Background commands are only run when they are told to, either by being stored into a script and then run or being run using “R”. The command “R” will run the last background commands that were sent to the pump as one message. Some background commands (such as “Un”) have modified versions that run in the foreground using the “#” modifier (such as “U#n”). As an example, assume that a pump with the address set to 1 is in the home position. The following example demonstrates how background and foreground commands behave.

```
/1A5000A0 This does not move the syringe at all.
/1R       Executes the last sent background command sequence; thus, the syringe moves to the
          absolute position 5000, then back home.
/1R       “A5000A0” executes as it was the last command entered.
/1A10000 This does not move the syringe at all.
/1A100R   Executes the last sent background command sequence; thus, the syringe moves to absolute
          position 100. It does not go to position 10000 since “A100” was the most recent command
          sequence sent, not “A10000”.
/1U#1U1u#1 Only the foreground commands “U#1” and “u#1” are executed; thus, User Output 1 quickly
          pulses On and then Off.
/1R       The last background command entered, “U1”, is executed; thus, User Output 1 turns On and
          stays On.
```

4.2 Syringe Commands

4.2.1 Position Commands

Position commands cause the syringe to move to a commanded position along its range of motion. Both absolute and relative positions can be used; the difference between them is:

- An **absolute position** is a specific point.
- A **relative position** is a distance offset from the current position.

Position commands come in both synchronous and asynchronous types. Synchronous commands (indicated by an upper-case letter) set the pump status to “busy” and prevent the execution of the next command in a script until they are complete. Asynchronous commands (indicated by a lower-case letter) set the pump status to “ready” and do not block the execution of subsequent commands. The user is responsible for making sure that an asynchronous command is complete before a new position command is executed.

NOTE: When asynchronous position commands are used, the user is responsible for ensuring that the command is complete before another position command is executed. The status of any asynchronous movement can be checked using flag 0 (see 4.10.2).

In each of the commands in the table below, the n value is expressed in steps, where “0” is at the top-of-stroke (volume).

Table 4-1: Position Control Commands

Command	Description
An	Move syringe to absolute position n. Apply backlash if aspirating. If travel limits are enabled (F19 flag set), then position n must be within the travel range. The command waits for the motor to complete the move. (n: lower limit...upper limit, @x)
an	Move syringe to absolute position n. Apply backlash if aspirating. If travel limits are enabled (F19 flag set), then position n must be within the travel range. The command does not wait for the motor to complete the move. The F0 flag can be polled to determine whether the syringe motor is busy. (n: lower limit...upper limit, @x)
Dn	Dispense n steps from the current position. The dispense direction is upward, towards the valve. If travel limits are enabled (F19 flag set), then the ending position must be within the travel range. A value of zero will move the syringe to the full dispense position (0) or until terminated by a “T” command, asserting the lower travel limit trigger, or asserting the Dispense Halt trigger. The command waits for the motor to complete the move. (n: 0...remaining dispense distance, @x)
dn	Dispense n steps from the current position. The dispense direction is upward, towards the valve. If travel limits are enabled (F19 flag set), then position n must be within the travel range. A value of zero will move the syringe to the full dispense position (0) or until terminated by a “T” command, asserting the lower travel limit trigger, or asserting the Dispense Halt trigger. The command does not wait for the motor to complete the move. The F0 flag can be polled to determine whether the syringe motor is busy. (n: 0... remaining dispense distance, @x)
Pn	Aspirate n steps from the current position. The aspirate direction is downward, away from the valve. If travel limits are enabled (F19 flag set), then the ending position must be within the travel range. A value of zero will aspirate to the full stroke unless terminated by a “T” command, triggering the upper travel limit input, or asserting the Aspirate Halt trigger input. (n: 0...remaining aspirate distance, @x)
pn	Aspirate n steps from the current position. The aspirate direction is downward, away from the valve. If travel limits are enabled (F19 flag set), then position n must be within the travel range. A value of zero will aspirate to the full stroke unless terminated by a “T” command, triggering the upper travel limit input, or asserting the Aspirate Halt trigger input. The command does not wait for the motor to complete the move. The F0 flag can be polled to determine whether the syringe motor is busy. (n: 0...remaining aspirate distance, @x)

A relative position is measured from the current position to the target position. Absolute position is always measured from the zero position (top-of-stroke). For example, a move from position 8600 (absolute) to position 20000 (absolute) can be done with either an absolute (go to position) or relative (move downward) aspirate command as follows:

Absolute move: **A20000** (final position measured from zero)
Relative move: **P11400** (final position measured from 8600)

4.2.1.1 Handshake Dispense Commands

Handshaking is a procedure for synchronizing two or more pumps in order to maintain a constant flow. A user output pin is configured to signal when a pump is nearing the end of a dispense cycle, and a corresponding user input pin is configured to trigger the start of a full dispense cycle on another pump, typically after an aspirate cycle has completed. This signal is in the form of a pulse. The output is normally high; it becomes low when the trigger point (“@49”) is passed and remains low until the dispense is completed. Additional information on handshaking can be found in 4.4.5.

Table 4-2: Handshake Dispense Commands

Command	Description
hn	Listen for 1 falling edge on User Input n, then begin a handshake dispense using User Output n. (n: 1...4 = User Inputs 1...4, @x)
hn,m	Listen for m falling edges on User Input n, then begin a handshake dispense using User Output n. (n: 1...4 = User Inputs 1...4 @x) (m: 1...16, @x)
h-n	Begin a handshake dispense immediately using User Output n. (n: 1...4 = User Outputs 1...4 @x)

The value of n in the handshake dispense commands determines which user input and user output will be used for the handshake coordination. For example, for the “h2” and “h-2” commands, the pump will use User Input 2 and User Output 2, respectively.

The “hn” command is used to wait for a trigger on User Input n. When User Input n becomes low, the pump begins a full dispense from its current location. At the trigger point, the pump pulls User Output n low in order to trigger the next pump connected, if connected, as the first pump did. The “hn,m” command is the same, but it needs m transitions from high to low in order to begin a full dispense cycle.

The “h-n” command immediately begins a dispense cycle, triggering User Input n at the end of it.

4.2.2 Movement Parameters

The syringe driver uses the following commands to set the syringe speeds, accelerations, and drive compensation. During the power-up of the pump, default values in the operational memory are recalled from the NVM. The operational values can be set at any time a move is not in progress. Top speed is an exception, as it can be set on the fly. A save command (“!”) is necessary if changes to motion variables are to be carried across resets. A detailed explanation of the speed and acceleration settings and their interactions with each other can be found in 3.2.2.

Except as noted, all commands require an “R” (Run) command to be executed immediately.

Regular Cavro commands that are executed in the background can also be executed in the foreground by using the (“#”) immediate qualifier.

Table 4-3: Movement Parameter Commands

Command	Description																																																																																								
Cn,m C#n,m	Set syringe pump calibration speed in n steps/sec and set calibration force value to m, used in the “Wn”, “Yn”, and “Zn” commands. (n: 5...2000, @x) (m: 0 ... 255)																																																																																								
cn c#n	Set syringe end speed to n steps/sec. (n: 5...10000, @x)																																																																																								
Kn,m K#n,m	Set number of syringe backlash steps to n and headspace steps to m. (n: 0...5% of full stroke, @x) m: 0...5% of full stroke, @y)																																																																																								
Ln L#n	Set syringe acceleration and deceleration in n step rate increase/step. (n: 1...2000, @x)																																																																																								
In I#n	Set syringe deceleration in n step rate decrease/step. (n: 1...2000, @x)																																																																																								
Sn	Set syringe top speed (in steps) from the table (deprecated command). (n: 0...36, @x)																																																																																								
	<table><tr><td>n</td><td>Steps/s</td><td>n</td><td>Steps/s</td><td>n</td><td>Steps/s</td><td>n</td><td>Steps/s</td></tr><tr><td>0</td><td>6400</td><td>10</td><td>1600</td><td>20</td><td>170</td><td>30</td><td>70</td></tr><tr><td>1</td><td>5600</td><td>11</td><td>1400</td><td>21</td><td>160</td><td>31</td><td>60</td></tr><tr><td>2</td><td>5000</td><td>12</td><td>1200</td><td>22</td><td>150</td><td>32</td><td>50</td></tr><tr><td>3</td><td>4400</td><td>13</td><td>1000</td><td>23</td><td>140</td><td>33</td><td>40</td></tr><tr><td>4</td><td>3800</td><td>14</td><td>800</td><td>24</td><td>130</td><td>34</td><td>30</td></tr><tr><td>5</td><td>3200</td><td>15</td><td>600</td><td>25</td><td>120</td><td>35</td><td>20</td></tr><tr><td>6</td><td>2600</td><td>16</td><td>400</td><td>26</td><td>110</td><td>36</td><td>15</td></tr><tr><td>7</td><td>2200</td><td>17</td><td>200</td><td>27</td><td>100</td><td></td><td></td></tr><tr><td>8</td><td>2000</td><td>18</td><td>190</td><td>28</td><td>90</td><td></td><td></td></tr><tr><td>9</td><td>1800</td><td>19</td><td>180</td><td>29</td><td>80</td><td></td><td></td></tr></table>	n	Steps/s	n	Steps/s	n	Steps/s	n	Steps/s	0	6400	10	1600	20	170	30	70	1	5600	11	1400	21	160	31	60	2	5000	12	1200	22	150	32	50	3	4400	13	1000	23	140	33	40	4	3800	14	800	24	130	34	30	5	3200	15	600	25	120	35	20	6	2600	16	400	26	110	36	15	7	2200	17	200	27	100			8	2000	18	190	28	90			9	1800	19	180	29	80		
n	Steps/s	n	Steps/s	n	Steps/s	n	Steps/s																																																																																		
0	6400	10	1600	20	170	30	70																																																																																		
1	5600	11	1400	21	160	31	60																																																																																		
2	5000	12	1200	22	150	32	50																																																																																		
3	4400	13	1000	23	140	33	40																																																																																		
4	3800	14	800	24	130	34	30																																																																																		
5	3200	15	600	25	120	35	20																																																																																		
6	2600	16	400	26	110	36	15																																																																																		
7	2200	17	200	27	100																																																																																				
8	2000	18	190	28	90																																																																																				
9	1800	19	180	29	80																																																																																				
Vn V#n	Set syringe top speed in n steps/sec. An "R" command is not required for this instruction. (n: 5...10000, @x)																																																																																								

Command	Description
V_n	Set syringe top speed in n micro-steps/sec. This is how speeds lower than 5 steps per second are achieved. 0.0625 steps per second equals 1 micro-step/sec, and 5 steps per second equals 80 micro-steps per second. (n: 1...160) Note: $n = (\text{desired steps/sec}) * 16$.
vn v#n	Set syringe start speed in n steps/sec. (n: 1...10000, @x) Note: High start speeds may increase syringe drive stall probability.
!	Save current operational parameters in NVM.

For most applications, only acceleration and top speed are adjusted. The remaining parameters are left at the default settings. If the top speed is set to a value lower than the start speed, the pump begins to move at the top speed. If the top speed is set lower than the stop speed, the move will end at the top speed. For this reason, top speed values that are set lower than either the start speed or the stop speed do not require any adjustments in the start speed or stop speed. For a visualization of this behavior, see Figure 3-1 and Figure 3-2.

NOTE: Changing motion variables from default values may result in undesirable behavior.

4.2.3 Initialization Commands

The syringe position should be initialized (calibrated) after each power-up, reset, or syringe overload condition. The motor controller has non-volatile memory that is used to preserve the status of the pump across power cycles. If the pump motors were not active when the power was shut off, the position of the valve and syringe can usually be recovered without requiring an initialization cycle. However, if this is not the case, the valve and syringe must be initialized.

An initialization command causes the syringe to go to the home or zero position. This is the only absolutely known location on the syringe stroke when position information is lost or corrupted. All other positions can be determined once this position is known.

Initialization uses the “**Wn**”, “**Yn**”, or “**Zn**” commands. Each operates in the same way except for the definition of the valve port positions used during initialization. For all initialization commands, the argument n denotes an initialize move ($n = 4$) or a set home operation ($n = 5$).

The “**W4**” command always initializes the syringe using port A. The “**Y4**” and “**Z4**” commands initialize the syringe using the valve port, which has been set by the “**~Yn**” or “**~Zn**” commands, respectively. This permits three different ports to be used for syringe initialization.

The Home Button on the PCB executes only the “**W4**” command. The initialization commands require an “**R**” command to be executed immediately.

Table 4-4: Initialization Commands

Command	Description
Wn W#n	Initialize syringe and/or valve (n: 0 = equivalent to W4A0 1 = equivalent to Y4A0 2 = equivalent to Z4A0 3 = not used 4 = calibrate valve to port A, then calibrate syringe to Zero position 5 = set lower limit (Zero) to current syringe position 6 = set valve Zero point to current position 7 = calibrate valve, move to port A 8 = force syringe position to nnn with "W8,nnn" command 9 = reset controller 10 = calibrate syringe only)
Yn Y#n	Initialize syringe and select the valve port specified by the "~Y" parameter (n: 4...7, see "Wn")
Zn Z#n	Initialize syringe and select the valve port specified by the "~Z" parameter (n: 4...7, see "Wn")
~Yn	Select the valve position to which the valve will go before initializing the syringe using the "Y4" command. The "~Yn" value is checked for a valid entry before accepting the value of n. This permits a port different from port A to be used for the valve initialization move. (n: 1...number of valve ports, where 1 = port A, 2 = port B, etc.)
~Zn	Select the valve position to which the valve will go before initializing the syringe using the "Z4" command. The "~Zn" value is checked for a valid entry before accepting the value of n. This permits a port different from port A to be used for the valve initialization move. (n: 1...number of valve ports, where 1 = port A, 2 = port B, etc.)

4.2.4 Syringe Queries

Table 4-5: Syringe Query Commands

Command	Description
?	Query syringe position in steps; returns "?" if invalid
?1	Query syringe start speed (steps/sec)
?2	Query syringe top speed (steps/sec)
?3	Query syringe end speed (steps/sec)
?29	Query contents of syringe position snapshot, clear snapshot
?30	Query the syringe ramp up and ramp down
?31	Query the number of syringe backlash steps
~Y	Query the valve port to be used by the "Yn" initialization command (1 = port A, 2 = port B, etc.)
~Z	Query the valve port to be used by the "Zn" initialization command (1 = port A, 2 = port B, etc.)
C?	Query calibration speed and calibration force
c?	Query syringe end speed
K?	Query syringe backlash and headspace
L?	Query syringe acceleration
I?	Query syringe deceleration
V?	Query syringe top speed
v?	Query syringe start speed

4.3 Valve Commands

The valve ports are not inherently directional. The actual direction of fluid flow at any port is determined by the relative motion of the syringe. An aspiration draws fluid into a port, and a dispense ejects fluid from a port.

For non-distribution valves, some valve positions block the syringe port, preventing fluid from entering or leaving the syringe. The pump does not allow syringe moves in those positions.

4.3.1 Valve Type Setting

The Cadent 6 uses a universal valve position encoder that accommodates different valve types. The valve type is selected by sending the “~Vn” valve configuration command. This parameter is not saved to NVM until the save command, “!”, is run. Once saved, do not set it again unless the valve type is changed. The valve configuration command cannot be stored within a script. Whenever the valve type is changed, the valve must be reinitialized using a “Wn”, “Yn”, or “Zn” command. Similarly, the syringe must be reinitialized after the syringe type is changed.

Table 4-6: Valve Type Setting Commands

Command	Description
~Vn ~Vn,m	<p>Set the valve part number to n and the optional syringe part number to m. For units without valves, “20202” is reported. For units without syringes, “30303” is reported. Attempting to set the syringe part number when there is no valve (part number “20202”) will result in an “invalid argument” response. The default valve and syringe part numbers are stored in configuration parameters during assembly.</p> <p>If a valve changes, it must be initialized. If a syringe changes, the syringe pump must be initialized. If the syringe stroke length does not match the pump stroke, an error is reported.</p> <p>A table entry number from below can be entered for a generic valve instead of a specific valve part number.</p> <p>(n: 1 = 3-way non-distribution valve 2 = 3-way distribution valve 3 = 4-way non-distribution valve 4 = 4-way distribution valve 5 = 5-way non-distribution valve 6 = 5-way distribution valve 7 = 6-way non-distribution valve 8 = 6-way distribution valve 9 = 8-way non-distribution valve 10 = 8-way distribution valve 11 = 12-way distribution valve 13 = 2-way distribution valve 17 = 2-way FAS solenoid valve (multichannel only) 20202 = no valve installed nnnnn = IMI Norgren valve part number)</p> <p>(m: 30303 = no syringe installed mmmmm = IMI Norgren syringe part number)</p>

4.3.2 Valve Position Commands

The valve position commands require the “**R**” (Run) command to be appended for immediate execution.

4.3.2.1 Multichannel Solenoid and Three-Way Non-Distribution Commands

The following commands are usable by solenoid and three-way non-distribution valves:

- > **I** – Moves a three-way valve to the input position (port A to syringe)
- > **O** – Moves a three-way valve to the output position (port B to syringe)

It should be noted that solenoid valves also use the “**I**” and “**O**” commands to set the ports marked “NO” and “NC”, respectively.

The “**B**” command can only be used by non-distribution valves.

- > **B** – Moves a three-way valve to the bypass position (port A to port B)

4.3.2.2 Discrete Valve Position Command

The “**on**” command moves the valve to the position selected by n, moving the shortest distance to arrive at the port. This command is the preferred command for all valve moves. The values of n must be consistent with the configured valve type.

(n: 1...number of ports (where 1 = port A, 2 = port B, etc.), @n)

Two modifications of this command exist. One is “**o+n**”, and the other is “**o-n**”. The first forces a turn in the clockwise direction, and the latter forces a turn in the counterclockwise direction.

For example, the “**/1o4R**” command moves the valve on pump 1 (“**/1**”) the shortest distance to port 4 (port D) and does it immediately (“**R**”).

For example, the “**/3o-2R**” command moves the valve on pump 3 (“**/3**”) counterclockwise to port 2 (port B) immediately (“**R**”).

4.3.2.3 Valve Stalls

When a valve fails to turn the commanded amount, a valve stall has occurred. The current script will be halted, and the LED will repeatedly flash two green lights, then a blue light. For more information on LED codes, refer to 5.2.

4.3.3 Valve Queries

Valve queries do not require the “**R**” command in order to be executed. They are executed immediately after they are received by the pump. These commands cannot be stored within a script.

Table 4-7: Valve Query Commands

Command	Description
?8	Query valve position as a port (1 = port A, 2 = port B, etc.); returns “?” if position is invalid
\$	Query number of valve stalls
%	Query number of valve movements, including stalls and calibrations
~V	Query the valve and syringe part numbers

4.4 I/O Commands

4.4.1 Output Commands

Output commands change the state of a user output. These commands require the “**R**” command in order to be executed immediately.

4.4.1.1 The Un Command

The “**Un**” command sets User Output n to On (see Table 2-8 and Table 2-9 for true/On definitions). For example, “**U2**” would turn on User Output 2. The argument n is defined as follows:

(n: 1...4 equals Output 1...4)

4.4.1.2 The un Command

The “**un**” command sets User Output n to Off (see Table 2-8 and Table 2-9 for false/Off definitions), where

(n: 1...4 equals Output 1...4)

NOTE: The foreground modifier (“**#**”) can be used to force the immediate execution of these commands, even while other commands are executing in the background (see 4.1).

4.4.1.3 Output Query Commands

The state of the user outputs can be queried using the commands in Table 4-8. Individual queries (e.g. “?61”) return “1” if the user output is true (On) and “0” if it is false (Off). The state of all user outputs can be queried simultaneously using the User Output Byte (“?60”).

The User Output Byte is a binary value where each bit represents one user output, as shown in Figure 4-1. Each bit will be “1” if its associated user output is true (On) and “0” if it is false (Off).

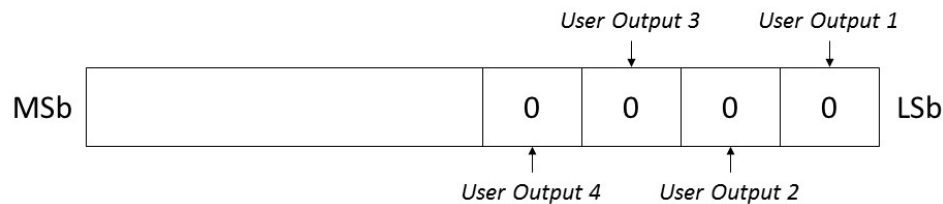


Figure 4-1: User Output Byte

The query “?60” returns the decimal equivalent value of the User Output Byte. For example, if User Outputs 1 and 2 were true while User Outputs 3 and 4 were false, “?60” would return “3” (the decimal equivalent of binary 0011).

Input query commands are sent from a host controller and request a status response from the pump. These commands are executed when they are received by the pump and do not require an “R” command. These commands cannot be stored in script.

Table 4-8: Output Query Commands

Command	Description
?60	Query User Output Byte
?61	Query User Output 1
?62	Query User Output 2
?63	Query User Output 3
?64	Query User Output 4

4.4.2 Input Commands

4.4.2.1 Input Latches

Each user input has an associated input latch. The latch is set (becomes “1”) when a falling edge is detected on the input signal. The latch remains set until it is reset by the user. The user can reset the latch by using input query commands, as described in Table 4-9.

Flag 40 (F40) indicates latch status. This flag is set (“1”) if at least one of the user input latches is set. It is clear (“0”) if no user input latches are set. Manually resetting F40 (“f40-”) resets all user input latches.

Unless otherwise specified, all input-related commands use the input latch rather than the current input state.

4.4.2.2 Input Query Commands

The state of the user inputs can be queried using the commands in Table 4-9. Individual queries exist for returning both the current state of a user input (“1” if true/low, “0” if false) and the state of the associated user input latch (“1” if falling edge has been detected, “0” otherwise). The state of all user input latches can be queried simultaneously using the User Input Byte (“?40”).

The User Input Byte is a binary value where each bit represents one user input latch, as shown in Figure 4-2. Each bit will be “1” if its associated user input has detected a falling edge, and “0” otherwise.

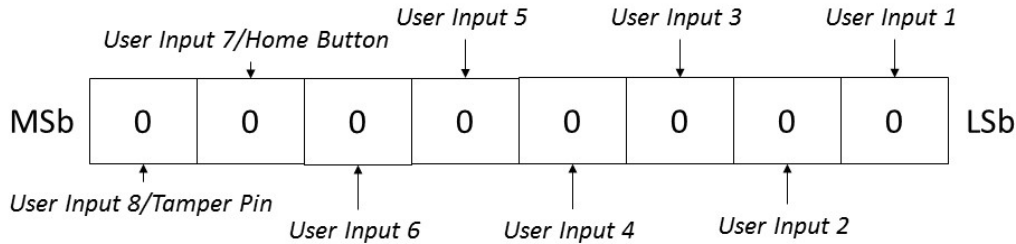


Figure 4-2: User Input Byte

The query “?40” returns the decimal equivalent value of the User Input Byte. For example, if User Inputs 1, 3, and 5 had detected falling edges, “?40” would return “21” (the decimal equivalent of binary 00010101).

Input query commands are sent from a host controller and request a status response from the pump. These commands are executed when they are received by the pump and do not require an “R” command. These commands cannot be stored in a script.

Table 4-9: Input Query Commands

Command	Description
?4	Query User Input 1 state (not latch)
?5	Query User Input 2 state (not latch)
?6	Query User Input 3 state (not latch)
?7	Query User Analog Input 1 voltage (20mV units)
?40	Query the User Input Byte (does not reset latches)
?41 to ?48	Query User Input Latches 1 to 8 (resets latch)
?49	Query User Analog Input 2 (20mV units)

4.4.2.3 Input Test and Jump Commands

The value of an input can be checked, and its status can be used to cause a script to change the path of execution. This is called a conditional jump. The general format is that if the input is true (see Table 2-10 for true/on definitions), then jump to the place marked by the script label.

These commands are used to control the way a script is executed, depending upon the state of an input variable. The commands are intended to be embedded within a script and not to be executed alone.

4.4.2.3.1 The Digital Input Test Command

For the “**inp**” command, if the input level for User Input *n* is true (see Table 2-10 for true/On definitions), then jump to label *p*. There are up to eight user inputs.

(*n*: 1... 8) User Input 1 ... 8

(*p*: a...z, A...Z) Program label

For example, the “**i4b**” command checks whether User Input 4 is true. If so, the command begins to execute the instructions at script label “**b**”. If not, it continues with the next instruction.

4.4.2.3.2 The Analog Input 1 Test Command

For the “**i<np**” and “**i>np**” commands, if the Analog Input 1 value is less than (“<”) or greater than (“>”) *n*, then jump to label *p*. *n* is expressed in mV and must be within the input voltage range of 0 to 5V.

(*n*: 0...5000) 0 = 0V to 5000 = 5V

p: a...z, A...Z) Program label

For example, the “**i<2480s**” command tests Analog Input 1, and, if the voltage is less than 2.48V, it goes to script label “**s**”.

4.4.3 The Home Button

If the Home Button is not used to initialize the pump, the home function can be disabled, and the button can be used as User Input 7. The button can be disabled and used as an input in two ways: first, by setting flag 9 using the “**f9+**” command, and second, by using the “**~H1**” command. After that, the button acts as User Input 7. The location of the Home Button is shown in Figure 2-3.

To reset this, clear the flag using either “**f9-**” or “**~H0**”. You can query the status of the Home Button by sending “**~H**”. A save command (“**!**”) must be used if this setting is to persist across resets.

4.4.4 The Tamper Pin

If the Tamper Pin is not used for tamper protection, the tamper function can be disabled, and the Tamper Pin can be used as User Input 8. The tamper function is disabled by setting flag 32 (“**f32+**”). After that, the Tamper Pin acts as User Input 8. Setting flag 32 also enables timestamping (see 2.4.3.3).

To reset this, clear the flag using “**f32-**”. A save command (“**!**”) must be used if this setting is to persist across resets.

4.4.5 Handshaking Commands

Table 4-10: Handshaking Commands

Command	Description
hn	Listen for 1 falling edge on User Input n, then begin a handshake dispense using User Output n. (n: 1...4 = User Inputs 1...4)
hn,m	Listen for m falling edges on User Input n, then begin a handshake dispense using User Output n. (n: 1...4 = User Inputs 1...4) (m: 1...16, @x)
h-n	Begin a handshake dispense immediately, using User Output n. (n: 1...4 = User Outputs 1...4)

The handshaking commands enable multiple Cadent 6 pumps to perform a full dispense in a specific order. They do this using the user inputs and user outputs on the Secondary Connector. All outputs start at a high logic level (open collector); then, half-second pulses to a low logic level are sent. The number of pulses received is used to determine when to fully dispense. Additional information on handshaking can be found in 4.2.1.1.

To begin a handshake dispense, one pump begins to listen on User Input n using “hn” or “hn,m”. If “hn” is used, the pump will do a full dispense after one received pulse. If “hn,m” is used, the pump will do a full dispense after m pulses are received. A second pump immediately begins a full dispense using the “h-n” command. When the syringe reaches the trigger point (@49), it will send one pulse on User Output n. The first pump will receive the pulse, triggering a full dispense if the correct number of pulses have been received.

To fully understand how this can be used, imagine three pumps chained together with addresses 1, 2, and 3. In order to have the first, then the second, then the third do a full dispense, connect User Output 1 on pump 1 to User Input 1 on pump 2. Then, connect User Output 1 on pump 2 to User Input 1 on pump 3. Then, send the following commands:

/3A5000h1R	Pump 3 is at position 5000 and listening on User Input 1.
/2A5000h1R	Pump 2 is at position 5000 and listening on User Input 1.
/1A5000R	Pump 1 is at position 5000.
/1h-1R	Pump 1 does a full handshake dispense using User Output 1 to trigger a full dispense of pump 2, which, in turn, triggers a full dispense of pump 3.

4.5 User Program Commands

Individual commands and scripts are executed in the pump RAM (temporary memory). The pump can also store scripts in NVM. For details on the pump's internal memory, see Program Memory, 3.1.

There are three types of script commands:

- **User script storage commands** can load, save, run, or erase user scripts in the pump memory.
- **Script execution commands** are used to stop or start scripts.
- **Script control commands** determine the order of execution (flow) of a script.

4.5.1 Program Storage Commands

These commands control the storage, retrieval, and erasure of a user script in the non-volatile user script memory. These commands are executed when received and cannot be stored within a script. The “R” command is not required.

The maximum length per script is 390 characters or 99 commands (whichever limit is reached first).

Table 4-11: Program Storage Commands

Command	Description
En	Save script to directory entry n; delete prior entry if any. (n: 1...99)
en	Erase script at directory entry n. (n: 1...99)
qn	Read user script n into background, query script contents, returns a period after status byte if script is not found. (n: 1...99)
?9	Query the number of unused bytes in user script storage.
?19	Query list of script numbers saved in script storage, only status byte is returned if no scripts are stored.
?33	Query contents of active background script (same as “q”).

4.5.2 Program Execution Commands

Table 4-12: Program Execution Commands

Command	Description
~A	Query the auto-start script number (returns 0 if auto-start is disabled).
~An	Set auto-start script number to n. (n: 1...99 0 = disable [default])
H	Halt background script (breakpoint); use foreground "R" command to resume. The immediate Halt command is used to stop a background script for debugging purposes.
rn	Load and run stored user script n in background. (n: 1...99)
jn	Execute script n, and then return to next instruction in calling script. (n: 1...99, @x)
R	Run background script or resume after script halt.
T	Terminate execution of command or background script.

4.5.3 Program Control Commands

4.5.3.1 Jumps and Labels

A script jump provides a means to change the order of execution of script commands. The point from which a jump occurs is a jump command. Program execution is changed from the location of the jump command to the destination label *p* specified in the jump command. Labels are created in scripts with the “:p” command where *p* can be any upper- or lower-case letter.

A jump can be unconditional, in which case it is executed every time it is encountered, or it can be conditional. Conditional jumps are “if... then” commands. The jump to a label occurs only if the specific test condition in the command is true.

Table 4-13: Jump and Label Commands

Command	Description
:p	Set script label <i>p</i> . (<i>p</i> : a...z, A...Z)
Jp	Unconditional jump to label <i>p</i> . (<i>p</i> : a...z, A...Z)
fnnnp f-nnnp	If flag <i>n</i> is set, then jump to label <i>p</i> . For – <i>n</i> , jump to label <i>p</i> if flag <i>n</i> is not set. (<i>n</i> : flag) (<i>p</i> : a...z, A...Z)
inp	If User Input <i>n</i> is true, then jump to label <i>p</i> . (<i>n</i> : 1...8 = User Inputs 1...8, @x) (<i>p</i> : a...z, A...Z)
i>np	If Analog Input 1 is greater than <i>n</i> (in mV), then jump to label <i>p</i> . (<i>n</i> : 0... 5000, @x) (<i>p</i> : a...z, A...Z)
i<np	If Analog Input 1 is less than <i>n</i> (in mV), then jump to label <i>p</i> . (<i>n</i> : 0... 5000, @x) (<i>p</i> : a...z, A...Z)
k<np	If the software counter is less than <i>n</i> , then jump to label <i>p</i> . (<i>n</i> : 0... (2 ³¹ -1), @x) (<i>p</i> : a...z, A...Z)
k=np	If the software counter is equal to <i>n</i> , then jump to label <i>p</i> . (<i>n</i> : 0... (2 ³¹ -1), @x) (<i>p</i> : a...z, A...Z)
k>np	If the software counter is greater than <i>n</i> , then jump to label <i>p</i> . (<i>n</i> : 0... (2 ³¹ -1), @x) (<i>p</i> : a...z, A...Z)

Command	Description
y<np	If the syringe step position is less than n, then go to label p. (n: lower limit...upper limit) (p=a...z, A...Z)
y=np	If the syringe step position is equal to n, then go to label p. (n: lower limit...upper limit) (p=a...z, A...Z)
y>np	If the syringe step position is greater than n, then go to label p. (n: lower limit...upper limit) (p=a...z, A...Z)

4.5.3.2 Repeat Loops

A script loop causes a group of commands to repeat. A loop can be constructed from a jump command and a label. This type of loop repeats indefinitely unless a conditional jump is included within the loop to cause an exit from the loop. The repeat command offers a better way when the desired number of repetitions is known.

The repeat command causes a group of instructions to repeat a specific number of times. The syntax is “**g...Gn**”. The “**g**” command marks the beginning of the group of commands to be repeated, and the “**Gn**” command marks the end of the group. The value n denotes the number of times the loop is to be repeated.

Table 4-14: Repeat Commands

Command	Description
g...Gn	Repeat commands between “g” and “Gn” for n times, or loop indefinitely if n is 0. (n: 0...(2 ³¹ -1), @x)

For example, for the command “**go1P6000o3A0G10**”,

g indicates the beginning of a loop command.

o1 moves to the port A (1 = A) location.

P6000 aspirates 6000 steps.

o3 moves the valve to port C (3 = C).

A0 dispenses all the contents of the syringe (go to zero).

G10 closes the loop script and repeats the loop 10 times.

The “**o1P6000o3A0**” script between “**g**” and “**G10**” will be repeated ten times.

4.5.3.3 Time Delays

A time delay is a pause in a script. These are useful for timing events such as generating pulses and event synchronization.

The “**Mn**” command delays (pauses) for n milliseconds. Internally, the command uses a fixed, 1-millisecond clock tick to count down the delay. This may cause an inaccuracy for the first millisecond since it may be counted down as a partial clock tick. To guarantee a minimum delay, add one to the count to account for the partial first millisecond. 1000 milliseconds = 1 second.

(n: 1...(2³¹-1))

4.6 Variables

A variable is a command argument that permits a command to use a value that is determined at the time the command is executed within a script rather than being set to a fixed value when the script is written. This permits more general scripts to be written and stored.

All variables use the syntax “**@n**”, where “**@**” denotes a variable and “**n**” denotes the source of the variable.

4.6.1 Setting Variables

Variables can be set using the “**zn=m**” command, where

- z** indicates a variable assignment
- n** 0-8 for the accumulators, 9 for the user timer, or any writable variable.
- m** the number to assign or any variable.

There is also an immediate version, “**z#n=m**”, which will run in the foreground.

4.6.2 Accumulators

There are nine accumulators numbered Accumulator 0 to Accumulator 8 that can be used to store numbers for a Cavo script. Accumulator 1 through Accumulator 8 can be accessed through variables “**@11**” through “**@18**”. Accumulator 0 can be accessed through the variable “**@5**”.

Although Accumulator 0 can be set and accessed in the same way as the other accumulators, it can also be accessed and modified with a special set of commands, as shown in the following table.

Table 4-15: Accumulator 0 Commands

Command	Description
kn k#n	Set Accumulator 0 to n. (n: 0... (2 ³¹ -1), @x)
k+n	Increase Accumulator 0 by n. (n: 0... (2 ³¹ -1), @x)

Command	Description
k-n	Decrease accumulator 0 by n. (n: 0... $(2^{31}-1)$, @x)
k*n	Multiply Accumulator 0 by n. (n: 0... $(2^{31}-1)$, @x)
k/n	Divide Accumulator 0 by n. (n: 1... $(2^{31}-1)$, @x)
k&n	Bitwise AND Accumulator 0 with n. (n: 0... $(2^{31}-1)$, @x)
k n	Bitwise OR Accumulator 0 with n. (n: 0... $(2^{31}-1)$, @x)
k!n	Bitwise XOR Accumulator 0 with n. (n: 0... $(2^{31}-1)$, @x)
k^n	Exchange the contents of Accumulator n with Accumulator 0. (n: 1...9)
k<np	If Accumulator 0 is less than n, go to label p. (n: 0... $(2^{31}-1)$, @x) (p: a...z, A...Z)
k=np	If Accumulator 0 is equal to n, go to label p. (n: 0... $(2^{31}-1)$, @x) (p: a...z, A...Z)
k>np	If Accumulator 0 is greater than n, go to label p. (n: 0... $(2^{31}-1)$, @x) (p: a...z, A...Z)

These commands allow the accumulators to do a variety of arithmetic operations and better control script flow.

NOTE: The user is responsible for ensuring that the result of any mathematical operations performed on accumulators remains within the range $(0... (2^{31}-1))$. Exceeding this range will result in numeric overflow and may cause undesirable behavior.

4.6.3 Scaling Variables

When flag 33 is set, the commands listed in

Table 4-16 will scale the variables used as their arguments. Values that are scaled are used to compute a proportional amount of the argument's range. The proportion is:

$$\text{Argument Value} = (\text{Variable Value} / \text{Maximum Variable}) \times \text{Maximum Argument}$$

Example (not scaled): o@3

Use the number as the value. If the number in “@3” is 6, the command would be “o6”.

Example (scaled): V@3

The value used for the command will be proportional to the maximum value of the number. In this case, if the value of “@3” (Analog Input 1 in mV) were 2500, the actual argument would be 5000. Top syringe step rate (used by “V” command) is 10000, and the maximum analog value (@3) is 5000. This is computed as follows:

$$\text{Value} = (2500/5000) \times 10000 = 5000$$

Table 4-16: Commands that Scale with Flag 33 Set

Command	Description
A@n a@n	Scale the @n variable as a percentage of full syringe stroke.
c@n	Scale the @n variable as a percentage of ending speed.
D@n d@n	Scale the @n variable as a percentage of remaining dispense distance.
K@n	Scale the @n variable as a percentage of backlash.
P@n p@n	Scale the @n variable as a percentage of remaining aspirate distance.
v@n	Scale the @n variable as a percentage of starting speed.
V@n	Scale the @n variable as a percentage of top speed.

A given variable is not restricted to use by one command or to one instance of a command. However, the value of a variable must be compatible with all commands that use it. The commands listed in Table 4-17 can use a variable in place of a fixed value for n. Variables cannot be used for labels.

4.6.4 The User Timer

In addition to the accumulators, there is a timer that can be used. It can be accessed as if it were a 9th accumulator. Once written to, it will count down to zero in millisecond increments.

Example: Timing Program Execution

The script “**k30000z@19=@5W7k-@19R**” can be used to determine the execution time of a valve calibration move:

k30000 sets the value of Accumulator 0 to 30000.

z@19=@5 sets the value of the User Timer (“@19”) to be the same as Accumulator 0 (“@5”) and begins a countdown.

W7 performs a valve calibration move.

k-@19 subtracts the User Timer value from Accumulator 0.

R runs the script.

After this script is run, Accumulator 0 will hold the valve calibration time in milliseconds. It can be queried using the “**k**” or “**?@5**” commands.

4.6.5 Commands Using Variables

Table 4-17: Commands that Use Variables

Command	Description
An, an	Move syringe to absolute position n.
cn	Set syringe end speed.
Dn, dn	Dispense n steps from current position.
g...Gn	Repeat commands between g and Gn n times.
inp	If User Input n is true, go to label p.
i>np	If Analog Input 1 is greater than n, go to label p.
i<np	If Analog Input 1 is less than n, go to label p.
jn	Run script n, then return.
Kn,m	Set number of syringe backlash steps to n, and headspace steps to m.
kn	Set Accumulator 0 to n.
k+n	Increase Accumulator 0 by n.
k-n	Decrease Accumulator 0 by n.

Command	Description
k*n	Multiply Accumulator 0 by n.
k/n	Divide Accumulator 0 by n.
k&n	Bitwise AND Accumulator 0 with n.
k n	Bitwise OR Accumulator 0 with n.
k!n	Bitwise XOR Accumulator 0 with n.
k<np	If Accumulator 0 is less than n, go to label p.
k>np	If Accumulator 0 is greater than n, go to label p.
Ln	Set syringe acceleration and deceleration slopes in n step-rate-increase/step.
In	Set syringe deceleration slope independently in n step-rate-decrease/step.
Mn	Delay n milliseconds.
on	Move valve to port n.
Pn, pn	Aspirate n steps from the current position.
Sn	Set syringe top speed from table (deprecated command).
Vn	Set syringe top speed in n steps/sec.
vn	Set syringe start speed in n steps/sec.
y<np	If syringe step position is less than n, go to label p.
y=np	If syringe step position is equal to n, go to label p.
y>np	If syringe step position is greater than n, go to label p.
~An	Set the auto-start script number to n.
~Bn,t	Set the RS-485 network baud rate and set the bus termination.
~Fn	Set RS-232 communication baud rate.

4.7 Configuration Commands

Configuration commands are used to determine the operating parameters of the pump. All configurations commands begin with a tilde (“~”) and have two forms: the set form and the query form. The set form uses a numerical argument to set the value of a parameter. The query form is the command with no argument attached. The query form reports the current value of the parameter.

Configuration parameters are not automatically saved into the NVM when they are set. Because of this, changed settings will revert to their previous values once power has been lost. To save the changed values, the save command (“!”) should be executed.

Configuration commands are executed when they are received and do not require an “R” command. Also, they cannot be stored within a script. Either upper- or lower-case letters may be used.

NOTE: A save (“!”) command must be sent to ensure changed settings are retained after a power

Table 4-18: Configuration Commands

Command	Description
~A	Query the auto-start script number (returns 0 if auto-start is disabled).
~An	Set the auto-start script number to n. (n: 1...99 0 = disable [default])
~B	Query the RS-485 network baud rate and termination setting.
~Bn,t	Set the RS-485 network baud rate, and set the bus termination on/off. (n: 1 = 38400 2 = 19200 3 = 9600 [default] 4 = 4800 5 = 2400 6 = invalid 7 = invalid 8 = invalid 9 = 57600 10 = 115200 11 = 230400 12 = 460800) (t: 0 = termination off 1 = termination on 2 = termination off, except nodes 1 and 15, [default])
~C	Query the CAN bus bit rate and termination setting.

Command	Description
~Cn,t	Set the CAN bus bit rate, and set the bus termination on/off. (n: 0 = CAN bus disabled 1 = 10 Kbit 2 = 20 Kbit 3 = 125 Kbit 4 = 250 Kbit [default] 5 = 500 Kbit 6 = 1000 Kbit) (t: 0 = termination off 1 = termination on 2 = termination off, except node 1 and 15, [default])
~D	Query RS-485 turnaround delays.
~Dn,m	Set the RS-485 turnaround delays where n is a delay in milliseconds (default 2 ms) for transmit-to-receive, and m is the delay for receive-to-transmit (use 12 ms [default] for V-Series compatibility). (n: 1...12) (m: 1...12)
~En	Erase all user scripts in non-volatile script storage. (n must be set to 12345 to enable erase)
~F	Query the RS-232 console baud rate.
~Fn	Set the RS-232 communications baud rate. (n: 1 = 38400 2 = 19200 3 = 9600 [default] 4 = 4800 5 = 2400 6 = 1200 7 = invalid 8 = invalid 9 = 57600 10 = 115200)
~H	Query the Home Button mode (F9 flag).
~Hn	Set the Home Button mode. When disabled, the Home button functions as User Input 7. (n: 0 = enabled [default] 1 = disabled)
~I	Query the power-up valve move mode (inverse of F35 flag).

Command	Description
~In	Set the power-up valve move mode. (n: 0 = enabled [default] 1 = disabled)
~L	Query the Dispense Halt operating mode (F17 flag).
~Ln	Set the Dispense Halt mode. (n: 0 = Dispense Halt alternate function input disabled 1 = Dispense Halt alternate function input enabled)
~Mn	Query the pin mapped to function n
~Mn,m	Map alternate function n to pin m; pin direction is implicitly determined by alternate function input or output. (n: 1-2 = Reserved 3 = handshake input 4 = position capture input 5 = run/stop input 6 = emergency halt input 7 = dispense halt input 8 = aspirate halt input 9-10 = Reserved 11 = valve active output 12 = syringe active output 13 = handshake output 14 = error output) (m: 0 = no pin, function disabled 1...8 = GPIO pin)
~P	Query the command protocol.
~Pn	Select the command protocol. (n: 1 = DT [default] 2 = OEM)
~Q	Query the pump and OEM part numbers.
~Qn,m	Set the pump part number in n and, optionally, the OEM part number in m. (n: nnnnnn = IMI Norgren pump part number) (m: 0 = no OEM part number [default] mmmmm = OEM part number)
~Rn	Reset the controller. (n: 1 = software reset (warm start) [default] 2 = reset, clear backup SRAM 3 = reset, restore backup copy configuration parameters 4 = reset, load factory defaults)

Command	Description
~V	Query the valve and syringe part numbers.
~Vn ~Vn,m	<p>Set the valve part number to n and the optional syringe part number to m. For units without valves, "20202" is reported. For units without syringes, "30303" is reported. Attempting to set the syringe part number when there is no valve (has part number "20202") will result in an "invalid argument" response. The default valve and syringe part numbers are stored in configuration parameters during assembly.</p> <p>If a valve changes, it must be initialized. If a syringe changes, the syringe pump must be initialized. If the syringe stroke length does not match the pump stroke, an error is reported.</p> <p>A table entry number from below can be entered for a generic valve instead of a specific valve part number.</p> <p>(n: 1 = 3-way non-distribution valve 2 = 3-way distribution valve 3 = 4-way non-distribution valve 4 = 4-way distribution valve 5 = 5-way non-distribution valve 6 = 5-way distribution valve 7 = 6-way non-distribution valve 8 = 6-way distribution valve 9 = 8-way non-distribution valve 10 = 8-way distribution valve 11 = 12-way distribution valve 13 = 2-way distribution valve 17 = 2-way FAS solenoid valve 20202 = no valve installed nnnnn = IMI Norgren valve part number)</p> <p>(m: 30303 = no syringe installed mmmmm = IMI Norgren syringe part number)</p>
~Y	Query the valve port to be used by the "Yn" initialization command (1 = port A, 2 = port B, etc.).
~Yn	<p>Select the valve position to which the valve will go before initializing the syringe using the "Y4" command. The "~Vn" value is checked for a valid entry before accepting the value of n. This permits a port different from port A to be used for the valve initialization move.</p> <p>(n: 1...number of valve ports, where 1=port A, 2=port B, etc.)</p>
~Z	Query the valve port to be used by the "Zn" initialization command (1 = port A, 2 = port B, etc.).
~Zn	<p>Select the valve position to which the valve will go before initializing the syringe using the "Z4" command. The "~Vn" value is checked for a valid entry before accepting the value of n. This permits a port different from port A to be used for the valve initialization move.</p> <p>(n: 1...number of valve ports, where 1=port A, 2=port B, etc.)</p>

4.8 Query Commands

Query commands are executed when they are received. They return a value or set of values to the query. A query command can be sent at any time, even if the pump is busy doing something else.

All query commands are executed when they are received, and they cannot be stored within a script. Query commands do not require an “R” command to be executed immediately. For a list of query commands, see 4.11.6.

4.9 Error Trapping Commands

Errors can occur during pump operation, in the structure of a user script, during communications, or in the way a command is given.

The pump recognizes these errors. Normally, an error causes a script or command to halt and generate an error message to be reported in response to the next received command. This normal response to an error can be redefined by a user script using a trap command.

A trap is a command that directs the pump to go to a label in a script if a particular error occurs. The commands following the label then determine what actions will be taken as a result of the error. An exit command marks the end of the error- handling script (the “handler”) and determines what happens next.

A user error handler is composed of the following three parts:

- The label that marks the beginning of the handler,
- The commands that comprise the body of the handler, and
- The exit command that marks the end of the handler.

4.9.1 Trap Declarations

A trap instruction takes effect when it is declared in the script. It remains in effect as written unless it is changed afterwards. Thus, error traps can be redefined on the fly in a script. The syntax for an error trap is “xnp”, where

- x** denotes an exception (trap) instruction,
- n** denotes the error number to be trapped, and
- p** denotes the script label that starts the error handler routine.

n must be in the range of 1 to 26, where the number corresponds with the “Error Number” in Table 5-1 (e.g. **n** = 3 creates a trap for the “invalid argument” error). The alternative syntax “**x*p**” creates a trap for all errors (i.e. if any error occurs, the script will jump to label **p**). If error **n** occurs after a trap for error **n** is set, the script jumps to the label **p**. By declaring the same trap with a different label, different error handling routines can be used for the same type of error in different parts of a script.

A trap operates any number of times if the error occurs externally to the error handling routine. If the error recurs while executing the error handler (before the error handler exit), the script terminates with a standard error exit. In general, if an error persists in recurring, it cannot be solved with a trap.

NOTE: Traps can provide graceful recovery for controlled exits from occasional error conditions. A trap cannot fix system problems or overcome serious mechanical difficulties.

4.9.2 Trap Exits

The last instruction of an error handler (exception script) must be a trap exit command. Trap exit commands mark the end of an error handler and specify what action the script is to take when exiting the error handler.

The general syntax is “**tn**”, where

- t1** returns script execution to the instruction following the instruction that caused the error,
- t2** restarts the script from the beginning,
- t3** performs a normal error exit with an error message, and
- t4** retries the instruction that caused the error.

If exit type 4 is used, some means must be used to prevent an “endless loop” of error → handler → error → handler.

4.9.3 Error Trap Query

The pump can be queried at any time to report the last error encountered by a trap. The syntax is “**x?**” for the command. This command is executed when received and does not require an “**R**” command. Do not store this command within a script.

4.10 Miscellaneous

4.10.1 Alternate Pin Functions

By default, alternate pin functions are not assigned. They can be assigned and queried using the “**~Mn,m**” and “**~Mn**” commands, respectively. Multiple alternate functions can be assigned to the same pin. All functions can be disabled, even if they are assigned to a pin, by setting the appropriate flags.

Note that when using the “**~Mn,m**” command to assign alternate functions to pins, the pin direction is implied by the alternate function being assigned. For example, in the command “**~M3,2**”, the “2” refers to User Input 2 because alternate function 3 (Handshake Input) is an input function. However, in the command “**~M13,2**”, the “2” refers to User Output 2 because alternate function 13 (Handshake Output) is an output function.

4.10.1.1 Handshake Input

Any user input pin can be used for a handshake input with the handshaking commands. See 4.4.5 for more on dedicated handshaking commands. The handshake input can be manually enabled/disabled using flag 13.

4.10.1.2 Position Capture Input

By assigning a pin as a position capture input, an external signal can be used to tell the pump to store the current syringe position. The position is stored on the falling edge of the signal. See 3.3.3 for more information. This function can be enabled/disabled using flag 14.

4.10.1.3 Run/Stop Input

When the assigned input becomes low, the pump is halted. Any currently executing valve or syringe command will finish. When the input becomes high again, the pump will resume normal operation. This function can be enabled/disabled using flag 15.

4.10.1.4 Emergency Halt Input

When the assigned input becomes low, the current script will terminate immediately without completing the currently executing command, and the syringe and valve motors will stop moving. This function can be enabled/disabled using flag 16.

4.10.1.5 Dispense Halt Input

When the assigned input becomes low, the current dispense cycle will stop. In addition to using flag 17, the command “~Ln” can be used to enable or disable the Dispense Halt command.

4.10.1.6 Aspirate Halt Input

When the assigned input becomes low, any current aspirate cycle will stop. This function can be enabled/disabled using flag 18.

4.10.1.7 Valve Active Output

When the Valve Active output is assigned to an output pin, the output will turn On when the valve is moving. This function can be enabled/disabled using flag 21.

4.10.1.8 Syringe Active Output

When the Syringe Active output is assigned to an output pin, the output will turn On when the syringe is moving. This function can be enabled/disabled using flag 22.

4.10.1.9 Handshake Output

Any user output pin can be used for a handshake output with the handshaking commands. The “~Mn” command will return the pin that was last used for this function. See 4.4.5 for more on dedicated handshaking commands. This function can be manually enabled/disabled using flag 23.

4.10.1.10 Error Output

When the Error Output is assigned to an output pin, the output will turn On when the emergency halt turns on until the pump recovers. The output will also turn On when a motor fault is detected. This function can be enabled/disabled using flag 24.

4.10.2 Flags

Flags are software switches that can be set (turned on), cleared (turned off), and tested. Flags are used to indicate the status of something or to change the way something is done.

There are six general-purpose flags (F1 through F6) and many special-purpose flags that control alternate functions or provide information. The flag instructions require the “R” command in order to be executed immediately. The test-and-jump instructions must be used within a script. See Table 4-20 for a list of all available flags.

Table 4-19: Flag Commands

Command	Description
fnnn+ f#nnn+	Set flag nnn
fnnn- f#nnn-	Clear flag nnn
fnnnp f-nnp	If flag n is set, then jump to label p. For -n, jump to label p if flag n is not set (p: a...z, A...Z)
fnnn?	Query flag nnn status

4.10.3 Motor Power Control

4.10.3.1 Syringe Motor

The syringe motor is normally on. The syringe motor turns on at the beginning of a move and enters a low-power holding state after a short delay at the end of the move. The delay allows the syringe to come fully to rest before moving on to other commands. The low-powered hold allows the syringe to maintain a precise position and resist some backpressure from the external system.

The default delay and holding power are sufficient for most applications, but they can be altered if necessary using the “**Nn,m**” command, where

n is the settling delay in milliseconds (0...65535, @x), and

m is the holding power (0...100, @y).

NOTE: Changing default delay and hold power settings can result in undesirable behavior, even damage to the motor. Consult with manufacturer before making changes.

4.10.3.2 Valve Motor

The valve motor is normally off. The valve motor automatically turns on at the beginning of a valve move and turns off at the end of a valve move. There are no rotary forces acting on a valve, and the combination of the internal friction in a valve and the motor detent torque are sufficient to hold the valve in place between moves.

4.10.4 Repeat Command String

A script can be repeated in its entirety by sending the script repeat command. The syntax is “**X**”. There is no limit to how often this command can be used to repeat the previous script.

This command does not work for queries and configuration commands.

Example:

o2A12000o-1A0R moves the valve to port B, fills the syringe, moves the valve to port A (counterclockwise), and empties the syringe.

X does all of the above commands again in the same way.

X does all the commands listed above again.

4.11 Command Reference

4.11.1 Flags

Table 4-20: Flags

Flag	Description
0	Valve or syringe motor busy status (reset only)
1-6	General purpose
7	Dispense Halt override
8	If set, stop on second handshake event
9	If set disable Home Button, shows as User Input 7 (“~H” command)
11	If set, enable syringe pump lower travel limit switch (default pin is unassigned)
12	If set, enable syringe pump upper travel limit switch (default pin is unassigned)

Flag	Description
13	If set, enable syringe pump Handshake Input trigger (default pin is unassigned)
14	If set, enable Position Capture trigger (default pin unassigned)
15	If set, enable Run/Stop input (default pin unassigned)
16	If set, enable Emergency Halt input (default pin is unassigned)
17	If set, enable Dispense Halt trigger (default pin is unassigned)
18	If set, enable Aspirate Halt trigger (default pin is unassigned)
19	If set, enforce travel limit range checking (except on calibration)
21	If set, enable Valve Busy status (default pin is unassigned)
22	If set, enable Syringe Busy status (default pin is unassigned)
23	If set, enable Handshake Output status (default pin is unassigned)
24	If set, enable Error Out status (default pin is unassigned)
30	If set, Tamper Pin input polarity active high
31	If set, ZERO switch input polarity active high
32	If set, disable Tamper Pin and enable timestamping; shows as User Input 8
33	If set, enable scaling for certain variables
34	If set, check loop nesting levels
35	If cleared, valve does not initialize on reset (“~I” command)
40	User input edge latch status
64	If set, the Tamper Pin detected an event
65	If set, valve initialized
66	If set, syringe initialized
67	If set, backup SRAM was valid after last reset
68	If set, RTC time was valid after last reset
69	If set, a power on reset (POR) occurred
70	If set, node is a network gateway
71	If set, ZERO switch detected a block
72	If set, RTC triggered an alarm
104	Halt on errors
105	Halt on faults

4.11.2 Variables

In the table below, variables that can only be read are marked with “[r]”. Variables that can be read and written are marked with “[rw]”.

Table 4-21: Variables

Variable	Description
@0	User Input Byte, query does not clear edges [r]
@3	User Analog Input 1 (0 to 5000 millivolts) [r]
@4	User Analog Input 1 (ratiometric percentage (0-99) [r]
@5	Accumulator 0 [rw]
@6	Current valve position as port number (1 = A, 2 = B, etc.) [r] If valve position is invalid, returns “0”
@7	Current syringe position in steps [r] If syringe position is invalid, returns “-1”
@8	Current syringe position as a percentage of travel (0-99) [r]
@11	Accumulator 1 [rw]
@12	Accumulator 2 [rw]
@13	Accumulator 3 [rw]
@14	Accumulator 4 [rw]
@15	Accumulator 5 [rw]
@16	Accumulator 6 [rw]
@17	Accumulator 7 [rw]
@18	Accumulator 8 [rw]
@19	User Timer (milliseconds) [rw]
@20	User Output Byte [r]
@21	Controller temperature (Celsius) [r]
@22	Motor power supply (0 to 30000 millivolts) [r]
@23	User Analog Input 2 (0 to 5000 millivolts) [r]
@26	Syringe pump resolution (steps) [rw]
@27	Syringe stroke length (µm) [rw]
@28	Syringe orifice diameter (mil) [rw]
@29	Syringe capacity (µL) [rw]
@30	Current syringe speed in steps/sec (0-idle) [r]
@31	Syringe start speed [rw]
@32	Syringe top speed [rw]
@33	Syringe end speed [rw]
@34	Syringe acceleration [rw]
@35	Syringe deceleration [rw]
@36	Syringe backlash steps [rw]

Variable	Description
@37	Syringe hold delay (ms) [rw]
@38	Syringe lower travel limit [r]
@39	Syringe upper travel limit [r]
@44	Syringe hold power percentage [rw]
@49	Syringe handshake trigger point [r]
@50	Current valve speed in steps/sec (0-idle) [r]
@51	Valve start speed [rw]
@52	Valve top speed [rw]
@53	Valve end speed [rw]
@54	Valve acceleration [rw]
@55	Valve deceleration [rw]
@56	Valve backlash steps [rw]
@57	Valve hold delay (ms) [rw]
@58	Valve lower travel limit [r]
@59	Valve upper travel limit [r]
@64	Valve hold power percentage [rw]
@69	Number of valve ports [r]
@72	Syringe motor minimum step rate [r]
@73	Syringe motor maximum step rate [r]
@74	Valve motor minimum step rate [r]
@75	Valve motor maximum step rate [r]
@76	Syringe headspace steps [r]
@77	Syringe encoder position (steps) [r]
@78	Valve encoder position (steps) [r]
@79	Valve step position (steps) [r]
@80	Most recent background error number (see Table 5-1) [r]
@81	Free-running timer (32-bit unsigned, 1μsec tick) [r]
@82	Free-running timer (32-bit unsigned, 1ms tick) [r]
@83	Time of day (HHMMSS) [rw], leading zero (if any) is suppressed [rw]
@84	Date (YYYYMMDD) [rw]
@85	Seconds since 1/1/70 (32-bit unsigned, 1s tick) [r]
@86	Days since 1/1/70 (32-bit unsigned, 24hour tick) [r]
@87	Valve motor power on time in seconds [r]
@88	Syringe motor power on time in seconds [r]
@89	Controller power on time in seconds [r]
@90	Tamper Pin timestamp in seconds since 1/1/70 (32-bit unsigned, 1s tick) [r]

Variable	Description
@91	32-bit signed random number (from hardware RNG) [r]
@92	Pump address (from hex address switch) [r]
@93	Syringe part number [r]
@94	Valve part number [r]
@95	Pump part number [r]
@96	Firmware build part number [r]
@97	Firmware base part number [r]
@99	Customer/OEM part number [r]
@109	Valve motor shaft angle in degrees (0 or 1 for solenoid) [r]
@110	Most recent syringe position capture [r]
@113	Aspirate cycles (count of aspirate commands) [rw]
@114	Dispense cycles (count of dispense commands) [rw]
@115	Valve moves [rw]
@116	Valve stalls [rw]

4.11.3 Standard Commands

Table 4-22: Standard Commands

Command	Description
An	Move syringe to absolute position n. Apply backlash if aspirating. If travel limits are enabled (F19 flag set), then position n must be within the travel range. The command waits for the motor to complete the move. (n: lower limit...upper limit, @x)
an	Move syringe to absolute position n. Apply backlash if aspirating. If travel limits are enabled (F19 flag set) then position n must be within the travel range. The command does not wait for the motor to complete the move. The F0 flag can be polled to determine whether the syringe motor is busy. (n: lower limit...upper limit, @x)
B	Moves a three-way valve to the bypass position (port A to port B).
Cn,m C#n,m	Set syringe pump calibration speed in n steps/sec, and set the calibration force value to m; used in the "Wn", "Yn", and "Zn" commands. (n: 5...2000, @x) (m: 0 ... 255)
cn c#n	Set syringe end speed to n steps/sec. (n: 5...10000, @x)
Dn	Dispense n steps from the current position. The dispense direction is upward, towards the valve. If travel limits are enabled (F19 flag set), then the ending position must be within the travel range. A value of zero will move the syringe to the full dispense position (0) or until terminated by a "T" command, asserting the lower travel limit trigger, or asserting the Dispense Halt trigger. The command waits for the motor to complete the move. (n: 0...remaining dispense distance, @x)
dn	Dispense n steps from the current position. The dispense direction is upward, towards the valve. If travel limits are enabled (F19 flag set), then position n must be within the travel range. A value of zero will move the syringe to the full dispense position (0) or until terminated by a "T" command, asserting the lower travel limit trigger, or asserting the Dispense Halt trigger. The command does not wait for the motor to complete the move. The F0 flag can be polled to determine whether the syringe motor is busy. (n: 0... remaining dispense distance, @x)
fnnn+ f#nnn+	Set flag nnn. (n: flag)
fnnn- f#nnn-	Clear flag nnn. (n: flag)
fnnnp f-nnp	If flag n is set, then jump to label p. For -n, jump to label p if flag n is not set. (n: flag) (p: a...z, A...Z)
g...Gn	Repeat commands between "g" and "Gn" n times, or loop indefinitely if n is 0. (n: 0...(2 ³¹ -1), @x)

Command	Description
H H#	Halt background script (breakpoint), use foreground "R" command to resume. The immediate Halt command is used to stop a background script for debugging purposes.
hn	Listen for 1 falling edge on User Input n, then begin a handshake dispense using User Output n. (n: 1...4 = User Inputs 1...4, @x)
hn,m	Listen for m falling edges on User Input n, then begin a handshake dispense using User Output n. (n: 1...4 = User Inputs 1...4 @x) (m: 1...16, @x)
h-n	Begin a handshake dispense immediately, using User Output n. (n: 1...4 = User Outputs 1...4 @x)
I	Moves a three-way valve to the input position (port A to syringe), or a solenoid valve to the "NO" position.
inp	If User Input n is true, then jump to label p. (n: 1...8 = User Inputs 1...8, @x) (p: a...z, A...Z)
i>np	If Analog Input 1 is greater than n (in mV), then jump to label p. (n: 0... 5000, @x) (p: a...z, A...Z)
i<np	If Analog Input 1 is less than n (in mV), then jump to label p. (n: 0... 5000, @x) (p: a...z, A...Z)
Jp	Unconditional jump to label p. (p: a...z, A...Z)
jn	Execute script n, then return to the next instruction in the calling script. (n: 1...99, @x)
Kn,m K#n,m	Set the number of syringe backlash steps to n and the number of headspace steps to m. (n: 0...5% of full stroke, @x) (m: 0...5% of full stroke, @y)
kn k#n	Set Accumulator 0 to n. (n: 0... ($2^{31}-1$), @x)
k+n	Increase Accumulator 0 by n. (n: 0... ($2^{31}-1$), @x)

Command	Description
k-n	Decrease Accumulator 0 by n. (n: 0... (2 ³¹ -1), @x)
k*n	Multiply Accumulator 0 by n. (n: 0... (2 ³¹ -1), @x)
k/n	Divide Accumulator 0 by n. (n: 1... (2 ³¹ -1), @x)
k&n	Bitwise AND Accumulator 0 with n. (n: 0... (2 ³¹ -1), @x)
k n	Bitwise OR Accumulator 0 with n. (n: 0... (2 ³¹ -1), @x)
k!n	Bitwise XOR Accumulator 0 with n. (n: 0... (2 ³¹ -1), @x)
k^n	Exchange the contents of Accumulator n with Accumulator 0. (n: 1...9)
k<np	If the software counter is less than n, jump to label p. (n: 0... (2 ³¹ -1), @x) (p: a...z, A...Z)
k=np	If the software counter is equal to n, jump to label p. (n: 0... (2 ³¹ -1), @x) (p: a...z, A...Z)
k>np	If the software counter is greater than n, jump to label p. (n: 0... (2 ³¹ -1), @x) (p: a...z, A...Z)
Ln L#n	Set syringe acceleration and deceleration in n step rate increase/step. (n: 1...2000, @x)
In I#n	Set syringe deceleration in n step rate decrease/step. (n: 1...2000, @x)

Command	Description																																																																																								
Mn	Delay for n milliseconds. (n: 1...(2 ³¹ -1))																																																																																								
Nn,m N#n,m	Set syringe hold time n in ms, and hold power percentage m. (n: 0...65535, @x) (m: 0...100, @y)																																																																																								
nn,m n#n,m	Set valve hold time n in ms, and hold power percentage m. (n: 0...65535, @x) (m: 0...100, @y)																																																																																								
O	Moves a three-way valve to the output position (port B to syringe), or a solenoid to the “NC” position.																																																																																								
on o+n o-n	Moves the valve to the position selected by n, moving the shortest distance to arrive at the port. The “+” modifier forces a clockwise rotation. The “-” modifier forces a counterclockwise rotation. (n: 1...number of ports (where 1= port A, 2= port B, etc.), @n)																																																																																								
Pn	Aspirate n steps from the current position. The aspirate direction is downward, away from the valve. If travel limits are enabled (F19 flag set), then the ending position must be within the travel range. A value of zero will aspirate to the full stroke unless terminated by a “T” command, triggering the upper travel limit input, or asserting the Aspirate Halt trigger input. (n: 0...remaining aspirate distance, @x)																																																																																								
pn	Aspirate n steps from the current position. The aspirate direction is downward, away from the valve. If travel limits are enabled (F19 flag set), then position n must be within the travel range. A value of zero will aspirate to the full stroke unless terminated by a “T” command, triggering the upper travel limit input, or asserting the Aspirate Halt trigger input. The command does not wait for the motor to complete the move. The F0 flag can be polled to determine whether the syringe motor is busy. (n: 0...remaining aspirate distance, @x)																																																																																								
Sn	Set syringe top speed (in steps) from the table below (deprecated command). (n: 0...36, @x) <table><tr><td>n</td><td>Steps/s</td><td>n</td><td>Steps/s</td><td>n</td><td>Steps/s</td><td>n</td><td>Steps/s</td></tr><tr><td>0</td><td>6400</td><td>10</td><td>1600</td><td>20</td><td>170</td><td>30</td><td>70</td></tr><tr><td>1</td><td>5600</td><td>11</td><td>1400</td><td>21</td><td>160</td><td>31</td><td>60</td></tr><tr><td>2</td><td>5000</td><td>12</td><td>1200</td><td>22</td><td>150</td><td>32</td><td>50</td></tr><tr><td>3</td><td>4400</td><td>13</td><td>1000</td><td>23</td><td>140</td><td>33</td><td>40</td></tr><tr><td>4</td><td>3800</td><td>14</td><td>800</td><td>24</td><td>130</td><td>34</td><td>30</td></tr><tr><td>5</td><td>3200</td><td>15</td><td>600</td><td>25</td><td>120</td><td>35</td><td>20</td></tr><tr><td>6</td><td>2600</td><td>16</td><td>400</td><td>26</td><td>110</td><td>36</td><td>15</td></tr><tr><td>7</td><td>2200</td><td>17</td><td>200</td><td>27</td><td>100</td><td></td><td></td></tr><tr><td>8</td><td>2000</td><td>18</td><td>190</td><td>28</td><td>90</td><td></td><td></td></tr><tr><td>9</td><td>1800</td><td>19</td><td>180</td><td>29</td><td>80</td><td></td><td></td></tr></table>	n	Steps/s	n	Steps/s	n	Steps/s	n	Steps/s	0	6400	10	1600	20	170	30	70	1	5600	11	1400	21	160	31	60	2	5000	12	1200	22	150	32	50	3	4400	13	1000	23	140	33	40	4	3800	14	800	24	130	34	30	5	3200	15	600	25	120	35	20	6	2600	16	400	26	110	36	15	7	2200	17	200	27	100			8	2000	18	190	28	90			9	1800	19	180	29	80		
n	Steps/s	n	Steps/s	n	Steps/s	n	Steps/s																																																																																		
0	6400	10	1600	20	170	30	70																																																																																		
1	5600	11	1400	21	160	31	60																																																																																		
2	5000	12	1200	22	150	32	50																																																																																		
3	4400	13	1000	23	140	33	40																																																																																		
4	3800	14	800	24	130	34	30																																																																																		
5	3200	15	600	25	120	35	20																																																																																		
6	2600	16	400	26	110	36	15																																																																																		
7	2200	17	200	27	100																																																																																				
8	2000	18	190	28	90																																																																																				
9	1800	19	180	29	80																																																																																				

Command	Description
tn	Marks the end of an error handler and specifies what action the script is to take when exiting the error handler. (n: 1 = returns script execution to the instruction following the one which caused the error 2 = restarts the script from the beginning 3 = performs a normal error exit with an error message 4 = retries the instruction which caused the error)
Un U#n	Sets User Output n to On. (n: 1...4 equals Output 1...4)
un u#n	Sets User Output n to Off. (n: 1...4 equals Output 1...4)
Vn V#n	Set the syringe top speed in n steps/sec. An "R" command is not required for this instruction. (n: 5...10000, @x)
V_n	Set syringe top speed in n micro-steps/sec. This is how speeds lower than 5 steps per second are achieved. 0.0625 steps per second equals 1 micro-step/sec, and 5 steps per second equals 80 micro-steps per second. (n: 1...160) Note: $n = (\text{desired steps/sec}) * 16$.
vn v#n	Set syringe start speed in n steps/sec. (n: 1...10000, @x) Note: High start speeds may increase the probability of a syringe drive stall.
Wn W#n	Initialize the syringe and/or valve. (n: 0 = equivalent to W4A0 1 = equivalent to Y4A0 2 = equivalent to Z4A0 3 = not used 4 = calibrate the valve to port A, then calibrate the syringe to Zero position 5 = set the lower limit (Zero) to the current syringe position 6 = set the valve Zero point to the current position 7 = calibrate the valve, then move to port A 8 = force the syringe position to nnn with the "W8,nnn" command 9 = reset the controller 10 = calibrate the syringe only)
x*p	Trap all errors using error handler at label p. (p: a...z, A...Z)
xnp	Trap errors of type n using error handler at label p. (n: 1...26, corresponding to "Error Number" in Table 5-1) (p: a...z, A...Z)

Command	Description
Yn Y#n	Initialize the syringe and select the valve port specified by the “~Y” parameter. (n: 4...7, see “Wn”)
y<np	If the syringe step position is less than n, go to label p. (n: lower limit...upper limit) (p=a...z, A...Z)
y=np	If the syringe step position is equal to n, go to label p. (n: lower limit...upper limit) (p=a...z, A...Z)
y>np	If the syringe step position is greater than n, go to label p. (n: lower limit...upper limit) (p=a...z, A...Z)
Zn Z#n	Initialize the syringe and select the valve port specified by the “~Z” parameter. (n: 4...7, see “Wn”)
zn=m z#n=m z@n=m z#@n=m	Assign value m to variable n. (n: 0...8 = Accumulator 0 to 8 9 = User Timer, @x) (m: value, @y)
:p	Set script label p. (p: a...z, A...Z)
“n “n,m	Display parameter n as a number on the console. If parameter m is included, send a column delimiter after the number; otherwise, send end of line. String is sent to console when end-of-line is sent. (n: nnn, any number @nnn, any variable) (m: 1 = comma after number 2 = tab after number 3 = space after number 0 or no parameter = end of line @y, column delimiter in a variable)
!	Save current operational parameters in NVM.

4.11.4 Program Storage and Execution Commands

Table 4-23: Program Storage and Execution Commands

Command	Description
En	Save script to directory entry n, delete prior entry if any. (n: 1...99)
en	Erase script at directory entry n. (n: 1...99)
jn	Execute script n and then return to next instruction in calling script. (n: 1...99, @x)
qn	Read user script n into background and query script contents; returns a period after status byte if script is not found. (n: 1...99)
R	Run background script or resume after script halt.
rn	Load and run stored user script n in background. (n: 1...99)
T	Terminate execution of command or background script.
X	Repeat last script (background only).
!	Save current operational parameters in NVM.

4.11.5 Configuration Commands

Table 4-24: Configuration Commands

Command	Description
~A	Query the auto-start script number (returns 0 if auto-start is disabled).
~An	Set the auto-start script number to n. (n: 1...99 0 = disable [default])
~B	Query the RS-485 network baud rate and termination setting.
~Bn,t	Set the RS-485 network baud rate; set bus termination on/off. (n: 1 = 38400 2 = 19200 3 = 9600 [default] 4 = 4800 5 = 2400 6 = invalid 7 = invalid 8 = invalid 9 = 57600 10 = 115200 11 = 230400 12 = 460800) (t: 0 = termination off 1 = termination on 2 = termination off, except node 1 and 15, [default])
~C	Query the CAN bus bit rate and termination setting.
~Cn,t	Set the CAN bus bit rate; set the bus termination on/off. (n: 0 = CAN bus disabled 1 = 10 Kbit 2 = 20 Kbit 3 = 125 Kbit 4 = 250 Kbit [default] 5 = 500 Kbit 6 = 1000 Kbit) (t: 0 = termination off 1 = termination on 2 = termination off, except node 1 and 15, [default])
~D	Query RS-485 turnaround delays.

Command	Description
~Dn,m	Set the RS-485 turnaround delays, where n is a delay in milliseconds (default 2 ms) for transmit-to-receive, and m is the delay for receive-to-transmit (use 12 ms [default] for V-Series compatibility). (n: 1...12) (m: 1...12)
~En	Erase all user scripts in non-volatile script storage. (n must be set to 12345 to enable erase)
~F	Query the RS-232 console baud rate.
~Fn	Set the RS-232 communications baud rate. (n: 1 = 38400 2 = 19200 3 = 9600 [default] 4 = 4800 5 = 2400 6 = 1200 7 = invalid 8 = invalid 9 = 57600 10 = 115200)
~H	Query the Home Button mode (F9 flag).
~Hn	Set the Home Button mode. When disabled, the Home Button functions as User Input 7. (n: 0 = enabled [default] 1 = disabled)
~I	Query the power-up valve move mode (inverse of F35 flag).
~In	Set the power-up valve move mode. (n: 0 = enabled [default] 1 = disabled)
~L	Query the Dispense Halt operating mode (F17 flag).
~Ln	Set the Dispense Halt mode. (n: 0 = Dispense Halt alternate function input disabled 1 = Dispense Halt alternate function input enabled)
~Mn	Query pin mapped to function n.

Command	Description
~Mn,m	Map alternate function n to pin m; pin direction is implicitly determined by alternate function input or output. (n: 1-2 = Reserved 3 = handshake input 4 = position capture input 5 = run/stop input 6 = emergency halt input 7 = dispense halt input 8 = aspirate halt input 9-10 = Reserved 11 = valve active output 12 = syringe active output 13 = handshake output 14 = error output) (m: 0 = no pin, function disabled 1...8 = GPIO pin)
~P	Query the command protocol.
~Pn	Select the command protocol. (n: 1 = DT [default] 2 = OEM)
~Q	Query the pump and OEM part numbers.
~Qn,m	Set the pump part number to n and, optionally, the OEM part number to m. (n: nnnnnn = IMI Norgren pump part number) (m: 0 = no OEM part number [default] mmmmm = OEM part number)
~Rn	Reset controller. (n: 1 = software reset (warm start) [default] 2 = reset, clear backup SRAM 3 = reset, restore backup copy configuration parameters 4 = reset, load factory defaults)
~V	Query the valve and syringe part numbers.

Command	Description
~Vn ~Vn,m	<p>Set the valve part number to n and the optional syringe part number to m. For units without valves, “20202” is reported. For units without syringes, “30303” is reported. Attempting to set the syringe part number when there is no valve (has part number “20202”) will result in an “invalid argument” response. The default valve and syringe part numbers are stored in configuration parameters during assembly.</p> <p>If a valve changes, it must be initialized. If a syringe changes, the syringe pump must be initialized. If the syringe stroke length does not match the pump stroke, an error is reported.</p> <p>A table entry number from below can be entered for a generic valve instead of a specific valve part number.</p> <p>(n: 1 = 3-way non-distribution valve 2 = 3-way distribution valve 3 = 4-way non-distribution valve 4 = 4-way distribution valve 5 = 5-way non-distribution valve 6 = 5-way distribution valve 7 = 6-way non-distribution valve 8 = 6-way distribution valve 9 = 8-way non-distribution valve 10 = 8-way distribution valve 11 = 12-way distribution valve 13 = 2-way distribution valve 17 = 2-way FAS solenoid valve 20202 = no valve installed nnnnn = IMI Norgren valve part number)</p> <p>(m: 30303 = no syringe installed mmmmm = IMI Norgren syringe part number)</p>
~Y	Query the valve port to be used by the “Yn” initialization command (1 = port A, 2 = port B, etc.).
~Yn	<p>Select the valve position to which the valve will go before initializing the syringe using the “Y4” command. The “~Vn” value is checked for a valid entry before accepting the value of n. This permits a port different from port A to be used for the valve initialization move.</p> <p>(n: 1...number of valve ports, where 1 = port A, 2 = port B, etc.)</p>
~Z	Query the valve port to be used by the “Zn” initialization command (1 = port A, 2 = port B, etc.).
~Zn	<p>Select the valve position to which the valve will go before initializing the syringe using the “Z4” command. The “~Vn” value is checked for a valid entry before accepting the value of n. This permits a port different from port A to be used for the valve initialization move.</p> <p>(n: 1...number of valve ports, where 1 = port A, 2 = port B, etc.)</p>

4.11.6 Query Commands

Table 4-25: Query Commands

Command	Description
C?	Query calibration speed and calibration force
c?	Query syringe end speed
F	Query background script status (return 0 for no background script, or 1 if background script is loaded)
fnnn?	Query flag nnn status
K?	Query syringe backlash and headspace
k	Query Accumulator 0
L?	Query syringe acceleration
l?	Query syringe deceleration
N?	Query syringe hold time and hold power percentage
n?	Query valve hold time and hold power percentage
Q	Query pump status code, return ASCII rendition of binary code
q	Query background script contents; display "." if no script
V?	Query syringe top speed
v?	Query syringe start speed
x?	Query the last trapped exception (background only)
?	Query syringe position in steps; returns "?" if invalid
?1	Query syringe start speed (steps/sec)
?2	Query syringe top speed (steps/sec)
?3	Query syringe end speed (steps/sec)

Command	Description
?4	Query User Input 1 state (not latch)
?5	Query User Input 2 state (not latch)
?6	Query User Input 3 state (not latch)
?7	Query User Analog Input 1 voltage (20mV units)
?8	Query valve position as port (1 = port A, 2 = port B, etc.); returns "?" if position is invalid
?9	Query the number of unused bytes in user script storage
?19	Query list of script numbers saved in script storage; only status byte is returned if no scripts are stored
?29	Query contents of syringe position snapshot, clear snapshot
?30	Query the syringe ramp up and ramp down
?31	Query the number of syringe backlash steps
?32	Query firmware build part number (software revision)
?33	Query contents of active background script (same as "q")
?34	Query current call depth and max call depth for nested scripts
?35	Query current run stack level
?36	Query next background command (only status is returned if no background script is loaded)
?37	Query error history (returns the last two sent status characters as hexadecimal numbers)
?38	Query vendor string
?39	Query product ID string
?40	Query the User Input Byte (does not reset latches)
?41	Query User Input Latch 1 (resets latch)

Command	Description
?42	Query User Input Latch 2 (resets latch)
?43	Query User Input Latch 3 (resets latch)
?44	Query User Input Latch 4 (resets latch)
?45	Query User Input Latch 5 (resets latch)
?46	Query User Input Latch 6 (resets latch)
?47	Query User Input Latch 7 (resets latch)
?48	Query User Input Latch 8 (resets latch)
?49	Query User Analog Input 2 (20mV units)
?60	Query User Output Byte
?61	Query User Output 1
?62	Query User Output 2
?63	Query User Output 3
?64	Query User Output 4
?@xxx	Query variable @xxx
\$	Query number of valve stalls
%	Query number of valve movements, including stalls and calibrations
&	Query firmware base part number
*	Query stepper motor supply voltage (200mV units)

5 Status and Error Messages

5.1 Status and Error Messages

A status/error character is included in every pump response immediately following the host address (“/0”). All possible response characters are listed in Table 5-1. Note that each character has both a “busy” and a “ready” variant to indicate whether the pump is busy executing a task or ready to receive new commands.

Error characters are generally returned in response to the command that follows the occurrence of the error. For example, if the syringe experiences an overload, the next time the user sends a command, the pump response would begin with “/0i” to indicate the error. The exception to this is syntactic errors in immediate commands (e.g. queries, configuration commands, and commands executed with the immediate [“#”] modifier). These errors will be indicated in the response immediately following those commands rather than the following command. Compare and contrast the following examples:

Example: Syntactic Error in a Background Command

Step	Entered Command	Expected Response
1	/1~L7	/0c-invalid argument
2	/1	/0`

Example: Syntactic Error in an Immediate Command

Step	Entered Command	Expected Response
1	/1D50000R	/0@
2	/1	/0z-syringe may go past home
3	/1	/0`

Table 5-1: Status/Error Messages

Error #	ASCII		Decimal		Binary	Status
	Busy	Ready	Busy	Ready		
0	@	`	64	96	01X00000	no error
1	A	a	65	97	01X00001	syringe failed to initialize
2	B	b	66	98	01X00010	invalid command
3	C	c	67	99	01X00011	invalid argument
4	D	d	68	100	01X00100	communication error
5	E	e	69	101	01X00101	invalid "R" command
6	F	f	70	102	01X00110	supply voltage too low
7	G	g	71	103	01X00111	device not initialized
8	H	h	72	104	01X01000	script in progress
9	I	i	73	105	01X01001	syringe overload
10	J	j	74	106	01X01010	valve overload
11	K	k	75	107	01X01011	syringe move not allowed
12	L	l	76	108	01X01100	cannot move against limit
15	O	o	79	111	01X01111	command buffer overflow
16	P	p	80	112	01X10000	use for 3-way valve only
17	Q	q	81	113	01X10001	loops nested too deep
18	R	r	82	114	01X10010	script label not found
19	S	s	83	115	01X10011	end of script not found
20	T	t	84	116	01X10100	out of script space
21	U	u	85	117	01X10101	home not set
22	V	v	86	118	01X10110	too many script calls
23	W	w	87	119	01X10111	script not found
24	X	x	88	120	01X11000	valve position error
25	Y	y	89	121	01X11001	syringe position corrupted
26	Z	z	90	122	01X11010	syringe may go past home

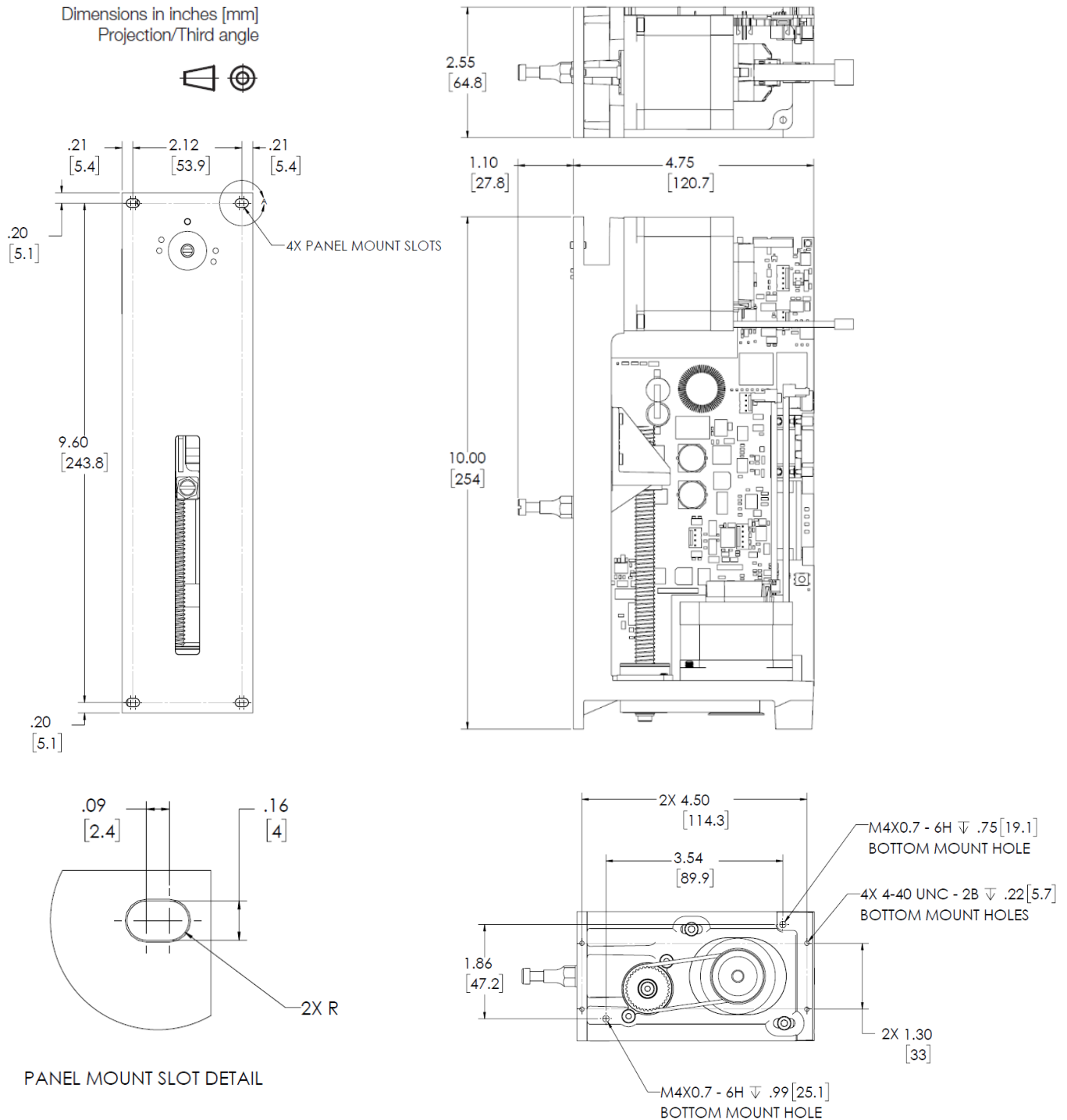
5.2 Status and Error LED Codes

Table 5-2: LED Fault Codes

Fault Name	Number of Flashes			Description
	Red	Green	Blue	
FAIL_NONE	0	0	0	No error; clear LEDs
FAIL_SCRIPT	1	0	1	Script command error
FAIL_MOTOR	1	0	2	Motor task failed
FAIL_APP	1	0	3	Application task failed
FAIL_NET	1	0	4	Network task failed
FAIL_CON	1	0	5	Console task failed
FAIL_LOG	1	0	6	Event logger task failed
FAIL_CLI	1	0	7	CLI task did not start
FAIL_WDG	2	0	1	Watchdog fault
FAIL_SSD	2	0	2	Storage device failed
FAIL_IFM	2	0	3	configuration data flash failed
FAIL_OSC	2	0	4	Configuration or PLL failed to lock; HSE failed
FAIL_STP	2	0	5	Stepper motor failure
FAIL_PWR	2	0	6	Power failure
FAIL_SPI	2	0	7	SPI bus failed
FAIL_UART	3	0	1	USART failed
FAIL_TIM	3	0	2	TIM failed
FAIL_RTC	3	0	3	RTC failed
FAIL_ADC	3	0	4	ADC failed
FAIL_DMA	3	0	5	DMA failed
FAIL_I2C	3	0	6	I2C failed
FAIL_CAN	3	0	7	CAN failure
FAIL_TASK	4	0	1	Task failed
FAIL_ASSERT	4	0	2	Assertion fault
FAIL_STACK	4	0	3	Stack fault
FAIL_HEAP	4	0	4	Memory heap allocation fault
FAIL_TIMER	4	0	5	Soft timer fault
FAIL_RTOS	4	0	6	RTOS start failure
FAIL_STATUS	4	0	7	Status task failed
FAIL_TASKWD	4	0	8	Fail task watchdog
FAIL_HARD	5	0	1	Hard fault
FAIL_USAGE	5	0	2	Usage fault
FAIL_BUS	5	0	3	Bus fault

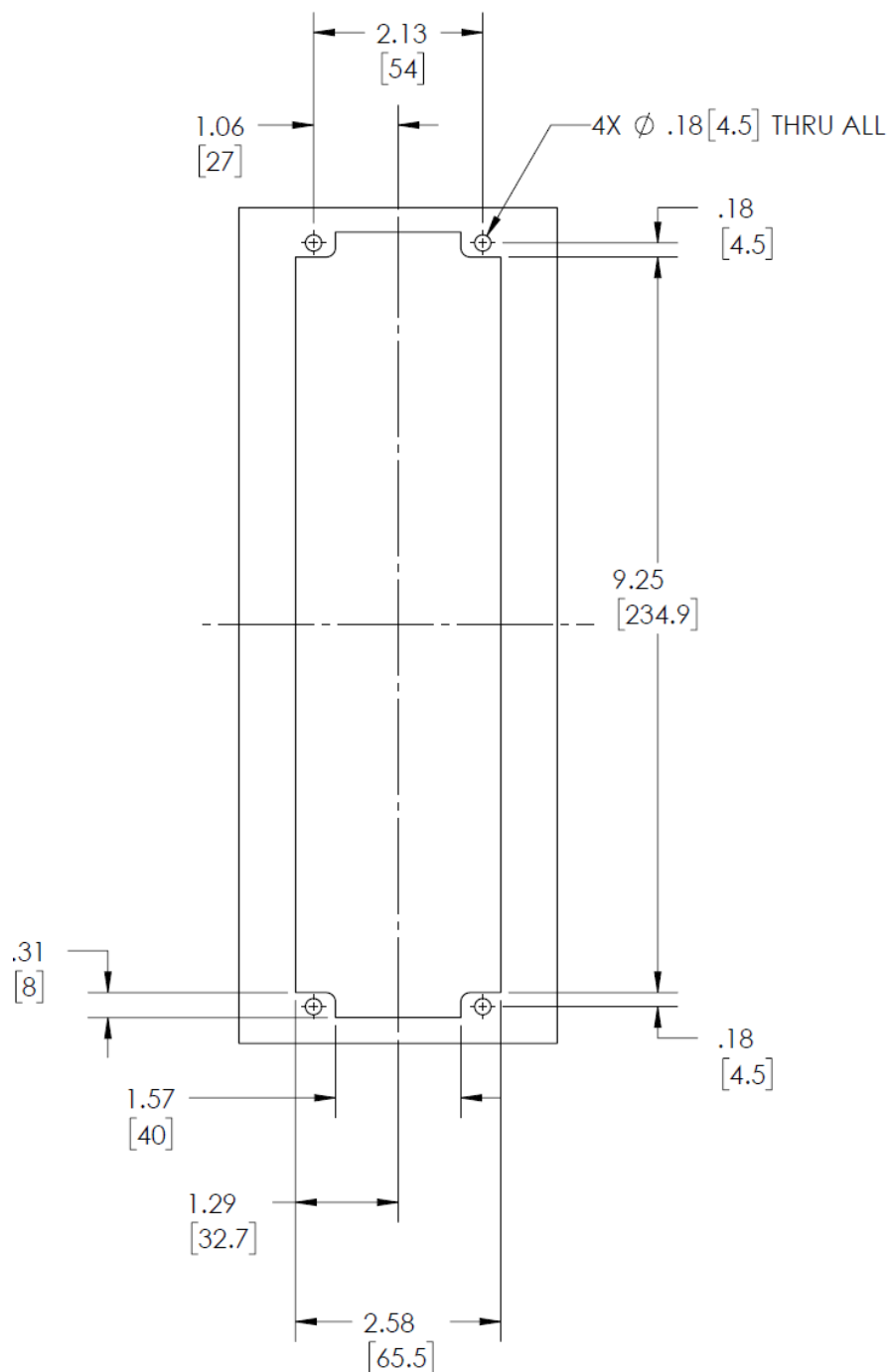
Fault Name	Number of Flashes			Description
	Red	Green	Blue	
FAIL_MMU	5	0	4	Memory protection fault
FAIL_IRQ	5	0	5	Unexpected interrupt fault
FAIL_ITM	5	0	6	Trace and profile failed
FAIL_TRAP	5	0	7	Fault handler failed
FAIL_CRC	6	0	1	CRC hardware failed
FAIL_PCB	6	0	2	Circuit board part number mismatch
FAIL_FIRM	6	0	3	Firmware part number mismatch
FAIL_BUILD	6	0	4	Build part number mismatch
FAIL_OTP	6	0	5	Could not read OTP history
FAIL_FPU	6	0	6	Floating point fault
FAIL_DFU	6	0	7	Field update fault
FAIL_CPU	7	0	1	CPU resource failed
FAIL_LCD	7	0	2	LCD or FSMC bus failed
FAIL_TEST	7	0	3	Testing fault
FAIL_DAC	7	0	4	DAC failed
FAIL_ENET	7	0	5	Ethernet failure
FAIL_FSMC	7	0	6	FSMC external bus failure
FAIL_SDIO	7	0	7	SDIO socket failure
FAIL_IDLE	0	0	1	Idle; no error
FAIL_COLD	0	0	2	Cold start; no error
FAIL_WARM	0	0	3	Warm start; no error
FAIL_MOVE	0	1	0	Motor moving; no error
FAIL_HOME	0	2	0	Motor calibrating; no error
FAIL_STALL	0	2	1	Motor stalled
FAIL_REVERSE	0	2	2	Motor reversed
FAIL_LIMIT	0	2	3	Motor at travel limit
FAIL_MPWR	0	2	4	Motor power warning
FAIL_BAL	0	2	5	Motor phase imbalance
FAIL_TEMP	0	2	6	CPU temperature warning
FAIL_LOGIC	0	2	7	ADC Vref (bandgap) warning
FAIL_TAMPER	0	3	1	TAMPER warning
FAIL_PANEL	0	3	2	HOME pushbutton
FAIL_ERASE	0	3	3	Flash erase; no error
FAIL_STOP	0	3	4	Motor stopped prematurely
FAIL_ASSET	0	4	14	Asset ID; special sequence R-G-B
FAIL_DISPLAY	0	4	15	No fault; used to time LED display

Appendix A: Cadent 6 Syringe Pump Dimensions

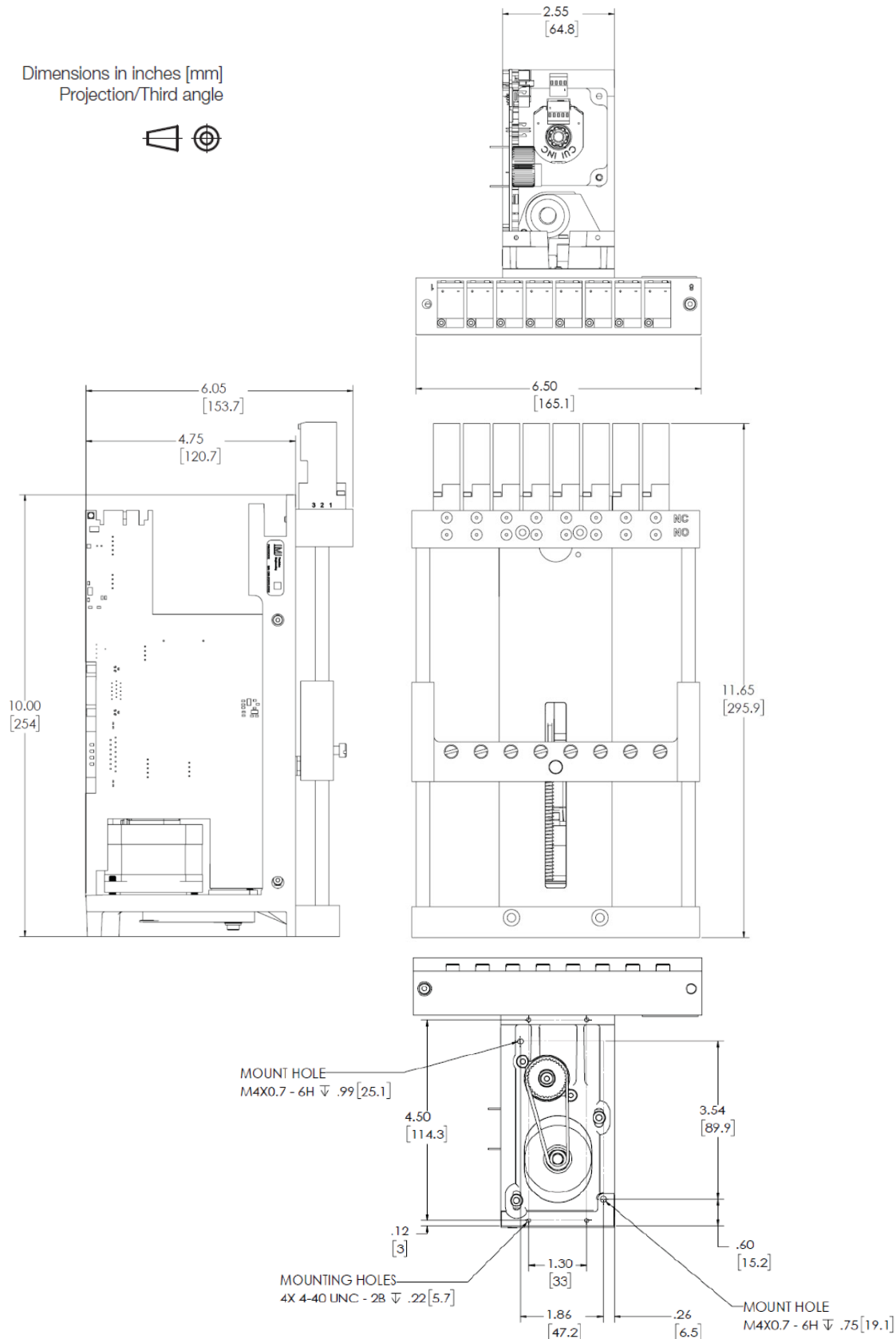


Appendix B: Recommended Panel Clearance for Panel Mounting

PANEL MOUNT RECOMMENDATIONS



Appendix C: Cadent 6M Syringe Pump Dimensions



Appendix D: Errata

1. The “inp” command uses the input latch value rather than the current pin status.

The “inp” command is implemented using the input latch value rather than the instantaneous voltage on the pin. This invalidates several examples used in the manual, namely:

“Wait for a High Input Level” in 3.3.1,
“Binary Input Selector” in 3.3.4, and
“Digital Input Test Command” in 4.4.2.3.1.

Workaround: adjust polarity of external circuitry as necessary. make use of flag 40 (F40) to clear the input latches as often as necessary.

2. Default voltage on User Inputs 1 and 2 may not be considered a logical “false”.

Because User Inputs 1 and 2 share both Analog and Digital Input circuits, a voltage divider is formed, making the default voltage on these pins ~1.8V (as opposed to 3.3V for other user input pins). This voltage may be insufficient to be considered a logical false and can result in unexpected behavior when making use of these user inputs.

Workaround: Include an external pull-up resistor to 5V.

3. Foreground/immediate commands (and “V” command) give two error responses.

Some commands (particularly foreground/immediate commands and the “V” command) give two error responses for some errors—one as an immediate response to the command, and the other as a response to the following command.

Workaround: Clear the error buffer each time an error is received by sending status pings (“/1” for a pump with address “1”) until the “/0” response is received before sending other commands.

4. Limits on “Kn,m” command arguments are fixed.

The upper limit for the n argument of the “Kn,m” command is fixed at 300 rather than being a percentage of the full stroke length. Similarly, the upper limit of the m argument is fixed at 1500.

5. The “L” command sets syringe acceleration only (not deceleration).

The “L” command sets only the syringe acceleration rather than both the acceleration and deceleration, as described in the manual.

Workaround: Use the “I” command to separately set deceleration.

6. The “%” command does not count calibration moves or stalls during calibration moves.

The “%” command does not count calibration moves or stalls during calibration moves.

7. The “\$” command does not count stalls during calibration moves.

The “\$” command does not count stalls during calibration moves.

8. The Halt/Resume functionality does not currently work.

The Halt command (“H”) is not currently implemented.

9. The “g...G0” command does not always result in an infinite loop.

Infinite command loops created using the “g...G0” command sometimes stop prematurely.

Workaround: Use a label paired with an unconditional jump, “Jp”, to achieve an infinite loop (e.g. “:p...Jp”).

10. Some query commands return extraneous values.

The following query commands return a second value that is meaningless to the user:

“c”, “~F”, “L”, “I”, “V”, and “v”

In addition, the “~C” command returns three values. The first and third are those described in the manual. The second is meaningless to the user.

Workaround: ignore the extraneous values as described above.

11. The “x?” command is currently unimplemented.

The “x?” command is currently unimplemented.

Workaround: Make use of the “@80” variable instead.

12. The “?30” command returns an incorrect delimiter between values.

The “?30” command returns a space-comma between its values rather than a comma-space (e.g. “20 ,50” instead of “20, 50”).

13. The “Y7” and “Z7” commands behave exactly like the “W7” command.

The manual describes the usage of the “Y” and “Z” commands for arguments in the range of 4 to 7, but the alternate port designated by the “~Y” or “~Z” command is not used in the case of “Y7” or “Z7”.

Workaround: The “Y7” or “Z7” command can be replaced with “W7o2” where “2” is the desired port number at which to end the calibration move. If a more general solution is desired, “2” can be replaced with an accumulator, such as “@11”.

14. The “~Mn” command sometimes returns values outside the range of 1 to 8.

The “~Mn” command sometimes returns values outside the range of 1 to 8.

Workaround: Treat responses greater than 8 in the same way that a response of 0 is treated.

15. The “@21” variable is currently unimplemented.

The “@21” variable is currently unimplemented.

16. The “@30” and “@50” do not return “0” when the pump is idle.

Instead of returning 0 when the pump is idle, the “@30” and “@50” variables return the ending speed of the previous syringe and valve move respectively.

Workaround: Make use of other indicators of whether the pump is idle, such as flag 0 (F0).