

OEM Pipettor User Manual & Protocol Command Set



Manual version: 2023-12-14
Firmware version: 2023-12-19



Contents

Overview	5
Safety Warnings.....	7
Hardware Information	8
Internal Schematic	8
Input/Output (IO)	9
Power & Communication (4-pin MTA-100).....	9
DIP Switches	10
Auxiliary 1 (5-pin PicoBlade).....	12
Auxiliary 2 (2-pin PicoBlade)	13
Buzzer	13
LEDs.....	13
Air Port	13
Tip Ejector	13
Overall Device Dimensions	14
Protocol.....	15
Serial Port Settings	15
Command Structure	15
Addressing	16
Device Response	17
Alternate OEM (Checksum) Protocol	19
Software	20
Command Set	21
Basic Operations Overview	21
Basic Operations.....	23
“m” (sets pump target power)	23
“p” (sets pump target pressure).....	23
“d” (sets pump valves to +/- pressure mode).....	23
“B” (enables/disables the pump).....	24
“l” (sets isolation valve state).....	24



"P" (pulses isolation valve open)	24
"E" (ejects tip)	25
"b" (turns on buzzer alarm)	25
"Q" (queries current status).....	25
"Z" (initializes device)	25
Waiting and Conditionals	26
"M" (delays for given milliseconds).....	26
"Me" (delays for given milliseconds since last M0 command)	26
"p<", "p>" (waits until pressure falls below or rises above value)	26
Loop / Branch / Execution Commands.....	27
"g" and "G" (loops execution)	27
"R" (begins execution).....	27
"T" (terminates currently running command)	28
"s" (stores a program string)	28
"e" (executes stored program)	28
"?s{id}" (gets stored program string)	29
Status / Query	29
"&" (get firmware revision and date)	29
"?m" (gets pump power target).....	29
"?p" (gets pump pressure target)	29
"?z" (gets valve/pump high level state).....	30
"?4" (reads inputs).....	30
"?20" (gets command run time).....	30
"?U500" (gets device serial number).....	31
Logging	31
"?" (queries multiple status variables).....	31
"L" (begins/ends pressure logging)	32
"?L" (gets pressure log data).....	32
PID Configuration.....	33
"?10" / "U10" (gets/sets power limit)	34
"?11" / "U11" (gets/sets PID proportional constant)	34



"?12" / "U12" (gets/sets PID integral constant)	34
"?13" / "U13" (gets/sets PID derivative constant)	35
"?14" / "U14" (gets/sets PID integral limit)	35
"?19" / "U19" (gets/sets PID polarity)	35
Configuration	35
"U41" "U47" "U48" (sets baud rate)	35
"?40" / "U40" (gets/sets DIP switch mode)	36
"?100" (gets baud rate setting in flash)	36
System	36
"Q*" (reset board)	36
Low-Level Operations	36
"J" (turns valve driver on/off)	37
"?J" (gets valve state)	37
Miscellaneous Topics	38
Multiple Devices/Axes	38
Firmware Burning	39
Baud Rate	39
Running without a PC or RS485 Communication	39



Overview

This document describes the communications protocol and command set for the IMI Adapta OEM Single Channel Pipettor, PN: INFP110UNS000.

Main features:

- Self-contained pipetting module
- No external pressure source required
- FAS CHIPSOL solenoid valves and piezoelectric air pump
- Broad dynamic dispense range; μL to mL, handles multiple tip sizes
- Closed loop pressure control (tuneable PID) enables liquid level, tip, clog and bubble detection
- Multiple pipettors can be mounted with tips on 9 mm centers
- Silent, compact and lightweight

Specifications:

- Dimensions: 97.2 mm x 16.9 mm x 72.1 mm (3.83 in x 0.67 in x 2.84 in), Width at tip mandrel: 8.95 mm (0.35 in)
- Weight: 100 g (3.5 oz)
- Pressure Range: -200 mbar [-2.9 psig] to 300 mbar [4.4 psig]. Resolution 0.2 mbar.
- Time:
 - o Minimum setpoint time: 5 ms
 - o Resolution: 0.2 ms
- Precision:
 - o 200 to 1000 μL : < 1% CV
 - o 50 to 199 μL : < 2% CV
 - o 10 to 49 μL : < 3% CV
 - o 5 to 9 μL : < 5% CV
 - o 1 to 4 μL : < 10% CV
- Power: 24 VDC with 0.5 A peak current (without tip ejector)
- Interface: Type: RS-485; Baud Rate: 9600, 38400, or 115200; Addressing: 16 addresses configurable through DIP switches or flash.



- Communications: Protocol: ASCII based on Cavro®/data terminal.
- Auxiliary I/O: One analog IN, one solenoid OUT, one I2C (flow sensor support).
- Electrical Connection: 4-pin MTA-100
- Optional: tip eject



Safety Warnings

Please note these hazards.

CAUTION: Do not aspirate liquid into the device. A filter element is provided (at the insertion of the mandrel into the manifold) for limited protection from short bursts, but the internal air pump will be permanently damaged by liquid penetration.

CAUTION: Flammable, explosive, corrosive, or toxic liquids should be avoided.

CAUTION: Electrostatic discharge (ESD) precautions should be taken to avoid damaging the circuit board.

CAUTION: Turn off the pump and valves when not in use for best longevity (d0B0 commands); the pump need be on or at full power only intermittently in many applications.

Hardware Information

Internal Schematic

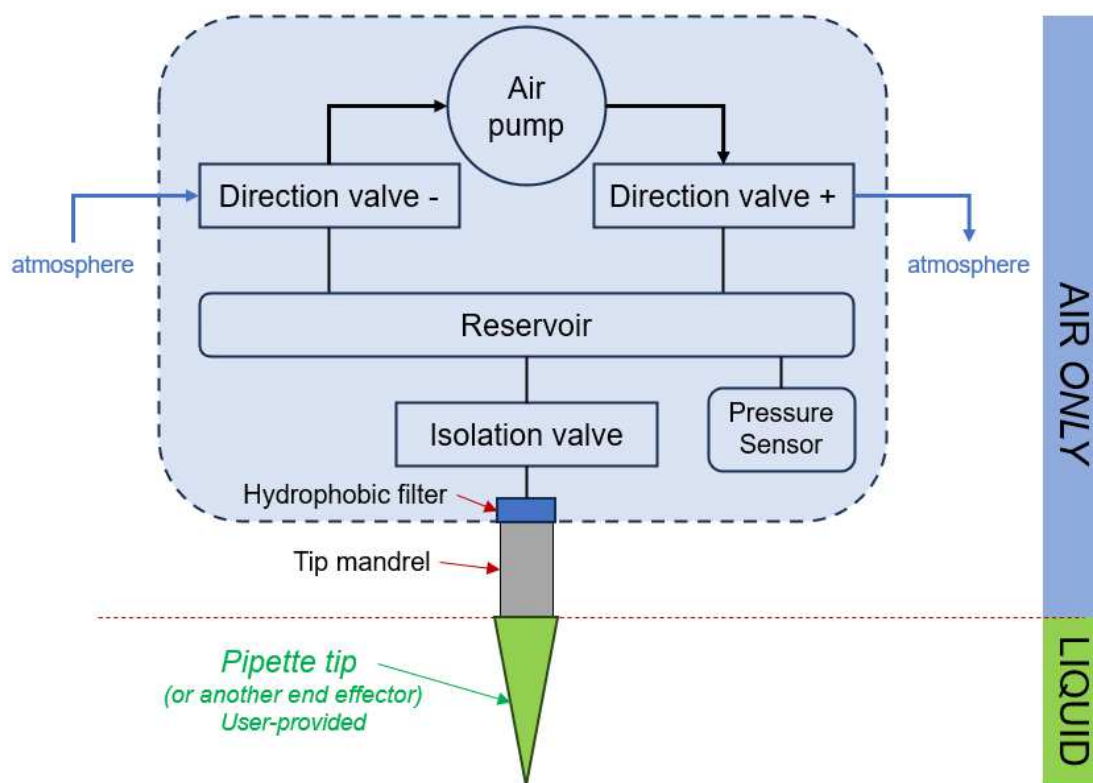


Figure 1: Fluidic schematic of OEM pipetting module.

The main capabilities of the pipettor are to pressurize the reservoir, hold pressure, release pressure through the isolation valve with very precise timing, and log pressure through these operations (see Figure 1). To generate pressure in the reservoir, close the isolation valve, turn on the pump at a certain power level (milliwatts), and set the directional valves to positive or negative pressure mode. The power can be controlled in open loop fashion or closed loop (PID) which uses a pressure sensor to automatically vary power to obtain a target pressure (millibar) in the reservoir. There other capabilities such as holding the reservoir pressure statically (blower off) by closing off all three valves; another is leaving the pump on with the isolation valve still open (e.g. longer blowout of pipette tip). The isolation valve response time can be as low as 2 ms, and the pump can turn on/off in 1 ms. The pressure can be polled or logged at a high rate (up to 1000 Hz) to the on-device memory buffer.

Input/Output (IO)

The device input and output pinouts are shown and described below (Figure 2).

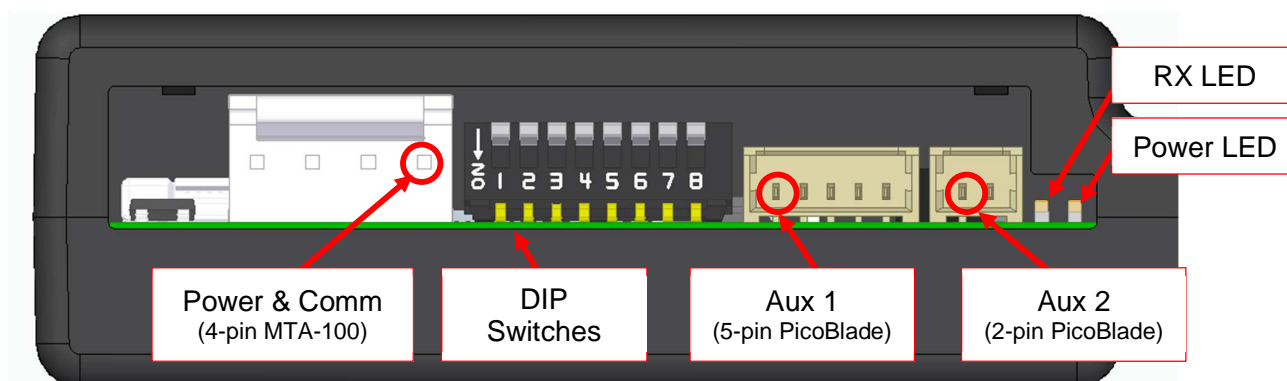


Figure 2: I/O on device (top view). Pin #1 is circled in red for each connector.

Power & Communication (4-pin MTA-100)

- 1: +24 VDC power input. 0.14 A max (full power without tip ejector).
- 2: GND
- 3: RS485 B (-)
- 4: RS485 A (+)

The connector is a 4-pin MTA-100 (0.1" pitch). Pin #1 is closest to the DIP switches. TE # 3-643814-4 (MTA-100, 24 AWG white, locking/tab, closed-end, tin) is suggested, but various MTA-100 connectors are available. Different wire gauges are indicated by color: 22 AWG (red), 24 AWG (white), 26 AWG (blue), and 28 AWG (green). Ideally, the connector should have the locking ramp and tabs, and pins should be tin (not gold) to match the board. Both closed-end and feed-thru (e.g. TE # 3-644563-4, for daisy chaining multiple devices) are available. See Figure 3 for example.

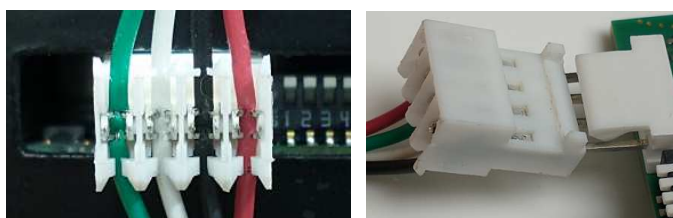


Figure 3: (left) Power & comm, shown with MTA-100 feed-thru style connector. (right) MTA connector (with polarizing tabs) properly inserted.

WARNING: Avoid making the MTA-100 connection when under power.

WARNING: Avoid plugging the MTA-100 connection in reverse as this would damage the device (see Figure 3b).

The RS485 lines should be terminated with 130 ohm resistors between A and B line on both ends of the cable, though omitting them can still work for shorter runs of cable (e.g. <1 m). The device has an on-board RS485 termination resistor, which can be enabled by turning ON the DIP switches 1 & 2 (see below), which should only be used for a device at the end of RS485 lines as may occur if daisy chaining multiple devices. The available cable kit with a built-in USB-RS485 converter (PN: INFP11000SCAB) has such resistors included.

DIP Switches

There are eight DIP switches, which by default are OFF as shown in Figure 4. Four of these (1-4) are hard-wired to resistors for communication termination. Four (5-8) are digital inputs that can be programmed.

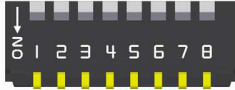
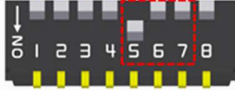


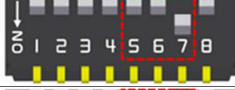


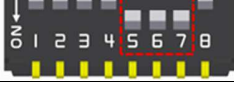


Figure 4: DIP switches in default position (OFF).

1 & 2: RS485 termination resistors; typically OFF. Turn both ON for the last device in an RS485 daisy-chain lacking external termination resistors. Termination may be omitted for short cables or lower baud rates but is recommended for best practices, especially for longer cable runs of several meters, to avoid communication errors.

3 & 4: reserved for CANbus resistors. Keep OFF typically.

5, 6, and 7: digital inputs 1-3 respectively. Used for RS485 address (effective on next restart). DIP7 can be reconfigured using the U40 command if not needed for addressing.

RS485 address	Command Prefix	DIP5	DIP6	DIP7	Picture
1	/1	OFF	OFF	OFF	
2	/2	ON	OFF	OFF	
3	/3	OFF	ON	OFF	
4	/4	ON	ON	OFF	
5	/5	OFF	OFF	ON	
6	/6	ON	OFF	ON	
7	/7	OFF	ON	ON	
8	/8	ON	ON	ON	

(*) **Note:** more than eight addresses are available through firmware/flash setting and/or reconfiguring DIP8 for addressing (U40 command). The higher addresses (15 and 16) are not recommended since they can be repurposed for special operations, including 16 for forcing 9600 baud and address #1 (recovery mode). The protocol itself supports 16 addresses unless protocol extensions are used.

RS485 address	Command Prefix	DIP5	DIP6	DIP7* (special)	DIP8* (special)
1	/1	OFF	OFF	OFF	OFF
2	/2	ON	OFF	OFF	OFF
3	/3	OFF	ON	OFF	OFF
4	/4	ON	ON	OFF	OFF
5	/5	OFF	OFF	ON	OFF
6	/6	ON	OFF	ON	OFF
7	/7	OFF	ON	ON	OFF
8	/8	ON	ON	ON	OFF

9	/9	OFF	OFF	OFF	ON
10 (A)	/:	ON	OFF	OFF	ON
11 (B)	/;	OFF	ON	OFF	ON
12 (C)	/<	ON	ON	OFF	ON
13 (D)	/=	OFF	OFF	ON	ON
14 (E)	/>	ON	OFF	ON	ON
15 (F)	/?	OFF	ON	ON	ON
16 (0)	/@	ON	ON	ON	ON

8: digital input 4. OFF indicates 115200 baud default or flash setting. ON indicates 9600 baud and overrides flash. Any change is effective on next restart. This may be repurposed for extended addressing (U40).



Default 115200 baud or flash setting



Force 9600 baud

Figure 5: DIP switch positions for default (115200 or flash) and forced (9600) baud rates.

Note: DIP #1 is closest to the 4-pin white connector.

Auxiliary 1 (5-pin PicoBlade)

This is normally unconnected. It can be purposed to connect a low-current 5 V device with 3.3 V digital I/O communication (including I2C or PWM), such as a liquid flow meter or small actuator.

- 1: i2c slave (SDA), 3.3 V
- 2: i2c slave (SCL), 3.3 V
- 3: 5V power output (< 3 A shared with device pump & valves)
- 4: GND
- 5: digital I/O (3.3V, PWM capable). By default this is a digital input, pulled high (> 10 Kohm). Command J6 will switch it to a PWM output.

Auxiliary 2 (2-pin PicoBlade)

Usable for an extra solenoid valve, actuator, or digital I/O. It is solenoid compatible, 1.3A continuous (< 3 A shared with device pump & valves).

- 1: spare solenoid #4 power, 5 VDC (+)
- 2: spare solenoid #4 power, 5 VDC (-)(switched)

Buzzer

An Internal buzzer can provide a programmable beep.

LEDs

LD1 – programmable indicator (GREEN). Blips ON when serial command received. Remains ON when busy running a command.

LD2 – power indicator (GREEN). ON when 24VDC power applied.

Air Port

This is the functional end-effector of the device, providing air flow in/out. The default device has a universal-fit disposable pipette tip mandrel.

Tip Ejector

An optional tip ejector accessory is available.

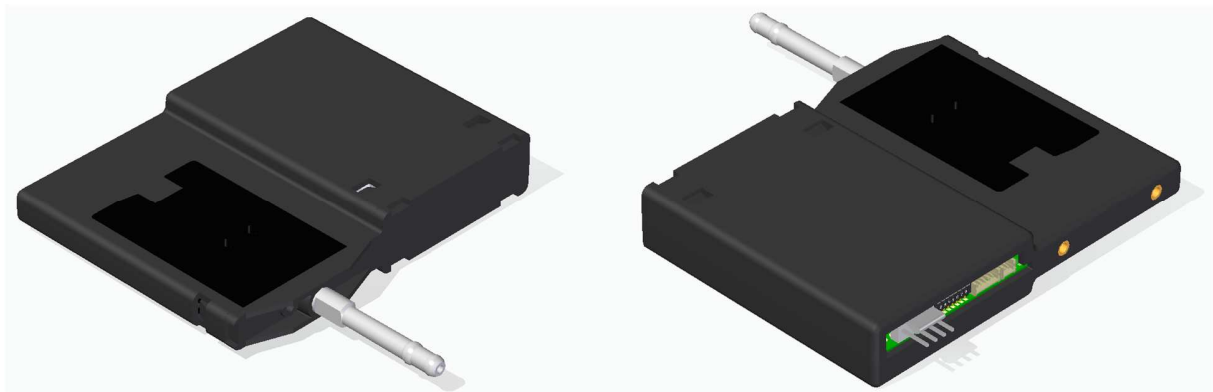


Figure 6: Model of the OEM Pipettor showing front-bottom and back-top views. Features include black plastic case, metal black manifold, and an air IO port with aluminium tip mandrel.

Overall Device Dimensions

Dimensions are shown in “mm (in).”

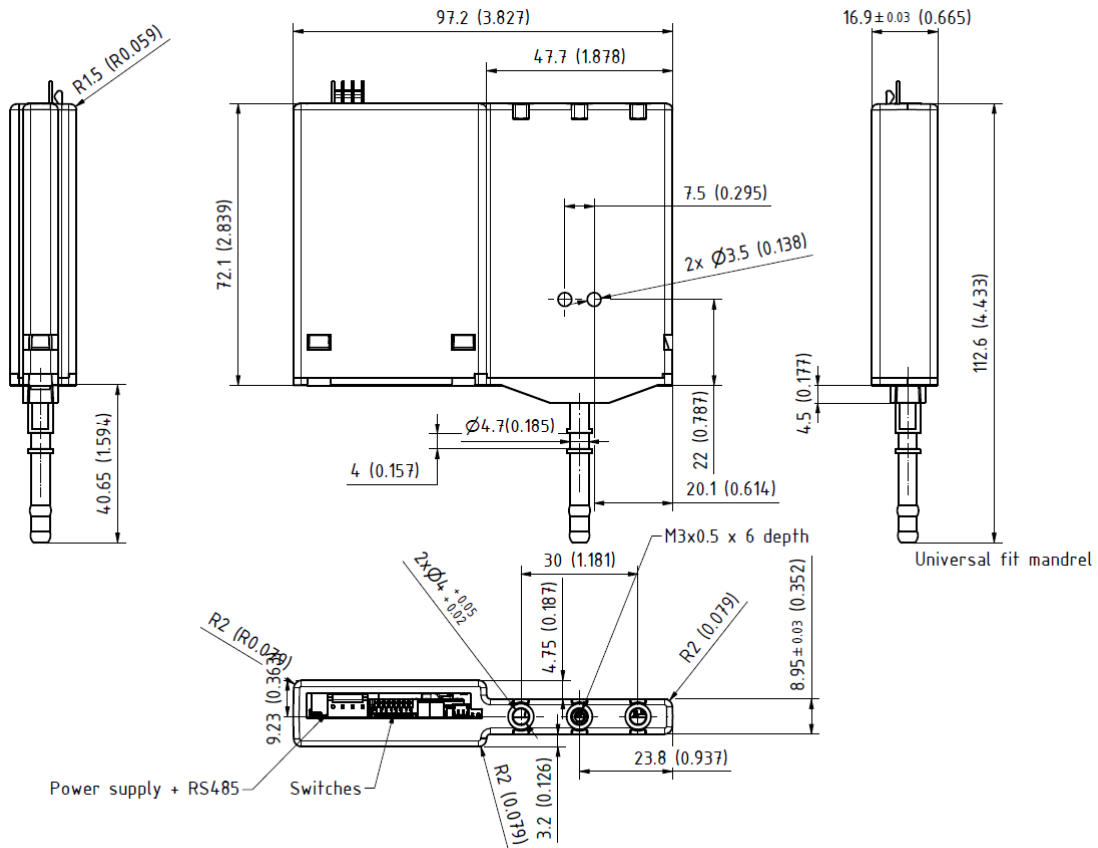


Figure 7: INFP110UNS000 Schematic



Protocol

The device normally communicates via RS-485 as a slave, which can also be converted to other protocols (e.g. RS232/UART/USB) with an external converter. It can also be programmed to operate standalone (without RS485). An RS485-to-USB serial converter (as included in the starter kit cable) is typically used for interfacing to a PC.

Serial Port Settings

Baud rate: 9600, 38400, or 115200 (default)
 Data bits: 8
 Stop bits: 1
 Parity: None
 Handshake: None

The baud rate can be switched in flash memory using the **U41/U47/U48** commands. DIP switch # 8 if ON will override the flash setting to 9600 baud.

Command Structure

The protocol is based on a popular ASCII data-terminal "DT" (Cavro-style) serial communications protocol.

The command structure is shown below. The first character is always "/" followed by the number of the driver board (address). This number is set on the DIP switches on the board (DIP5-DIP8), which supports up to eight addresses (15 with **U40** command). The device ships with position set to "1" by default.

Start Character	/
Address	(See Addressing Below)
Commands	<Command string>
End of string	<CR>

where <CR> indicates 0x0D in ASCII hex.

The command (start to end char) has a maximum length of 255 characters.

The command set for the driver and controller are single alpha characters often followed by a numeric value. The case sensitive alpha character represents "what to do" and the numeric value represents "how much to do it" (e.g. **m100B1** sets the pump power to 100 mW and turns on the pump).

There is also a similar "OEM" protocol with a checksum (described later).

Addressing

The address character is normally "1" but can take other values below and must be unique if multiple devices are on the same RS485 bus:

Numeric Address (Hex)	Address character in command
1 – 9	"1" - "9"
A	":" (colon)
B	";" (semicolon)
C	"<" (less than)
D	"=" (equals)
E	">" (greater than)
F	"?" (question mark)
10 (reserved)	"@" (at sign)

If two or more devices are connected, commands may be sent simultaneously to multiple devices:

1 and 2	"A"
3 and 4	"C"
5 and 6	"E"
7 and 8	"G"
9 and A	"I"
B and C	"K"

D and E	"M"
F and 0	"O"
1, 2, 3 and 4	"Q"
5, 6, 7 and 8	"U"
9, A B and C	"Y"
D, E, F, and 0	"]" close bracket
All Devices	"_" (underscore)

Commands issued to multiple addresses do not return a response, which avoids two devices using the bus at the same time. Commands may be issued to each device without the R and then issue the R command to both axes to run them, as shown below.

```
/1P1000
/2P500
/AR
```

Device Response

The device responds to command inquiries by sending back reply messages addressed to a master device. The master device has an address 0 (zero). The master device should parse the communications on the bus continuously for responses starting with /0 . NOTE: Do not look for the next character coming back after issuing a command which may be a false character due to the reversal of the RS485 bus.

Zero or more optional turn-around characters, ignored	e.g. <0xff>
Start Character	/
Address	0
Status Character	(see below)
Commands	<i>(zero or more bytes of data)</i>
End of string	<ETX><CR><NL>

where <ETX> indicates 0x03, <CR> = 0x0D, and <NL> = 0x0A in ASCII hex.

After the /0, the next character is the “Status Character” which is a collection of 8 bits. These bits are:

- Bit7 - Reserved (= 0)
- Bit6 - Always Set
- Bit5 - Ready Bit. This is set (= 1) when ready to accept a command or unset (= 0) when busy doing an operation
- Bit4 - Reserved (= 0)
- Bits 3 thru 0 - Error code from 0-15 as follows:

Code	Letter (not ready)	Letter (ready)	Descriptor	Note
0	@	`	No Error	
1	A	A	N/A	
2	B	b	Bad Command	
3	C	c	Bad Parameter	Out of range value (reserved for future use)
4	D	d	N/A	
5	E	e	N/A	
6	F	f	N/A	
7	G	g	Device not initialized	Unused but reserved for future use.
8	H	h	N/A	
9	I	i	N/A	Pump failure. This error is sticky, i.e. persists until a reset (including Q* command).
10	J	j	N/A	
11	K	k	N/A	
12	L	l	N/A	
13	M	m	Time limit exceeded	Unused but reserved for future use.



14	N	n	Execution error	Unused but reserved for future use.
15	O	o	N/A	

Examples:

`/1d+m150B1R` Switch to positive pressure, set pump to 150 milliwatt, and enable pump.

`/1P100R` Pulse isolation valve open for 100 milliseconds.

The response to command `/1?4` can be e.g. `/0`1110` (followed by `<ETX><CR><LF>`). For brevity we will usually omit the last three response characters in this manual since they are always the same.

The last example `/1P100R` will put the device in a busy (not ready state) for 100 ms. You can poll the device, e.g. `/1Q`, to check for completion.

Alternate OEM (Checksum) Protocol

An alternate protocol is available which includes a checksum in the communication and support retransmission. You may skip this section if using the regular protocol.

The command send consists of these bytes sent:

- 02 hex (`<STX>` character).
- Pump address, such as ASCII "1" (31 hex) or any other address noted above.
- Sequence number & repeat flag byte. It's a binary string 0011XYYY (MSB) where X is the repeat flag (1=repeat, 0=no repeat) and YYY is the sequence number (0-7), as described below. This takes the form of ASCII "0" (30 hex or 00110000 binary) – "?" (3F hex or 00111111 binary).
- Command (1-250 bytes, e.g. "P100R")
- 03 hex (`<ETX>` character)
- Checksum. This is a XOR of bytes from the STX to ETX inclusive.

These bytes are received back:

- 02 hex (start of transmission, <STX>, character)
- 30 hex (ASCII "0"), which is the address of the RS485 master.
- Status code byte (see above).
- Command data (0-250 bytes)
- 03 hex (end of transmission, <ETX>, character)
- Checksum. This is a XOR of bytes from the <STX> to <ETX> inclusive.

Any other characters outside the <STX> and checksum after <ETX> are ignored.

If the RS485 master believes its command was not received, it may resend the command with the repeat flag = 1. The device will perform no action except reply if the repeat flag is set and the sequence number is unchanged from the last command it saw (already performed). The sequence number is ignored if the repeat flag is not set. The sequence numbers can be used in any order, but cycling between only '0' and '1' would be sufficient.

This implementation allows switching between OEM and DT protocols without power cycle, unlike other implementations.

This OEM example is equivalent to the DT example `/1P100R → /0@`. Hex (and ASCII) values are shown for clarity.

- 02(<STX>) 31('1') 30('0') 50('P') 31('1') 30('0') 30('0') 52('R') 03(<ETX>) 33('3') →
- 02(<STX>) 30('0') 40('@') 03(<ETX>) 71('q')

Software

You can write your own software to control the device or for quick startup send commands to the device via

- The IMI Adaptas starter software can control a wide variety of Adaptas and non-Adaptas instruments, via GUI controls, direct serial commands, and user program scripting (Lua code). A module for the OEM Pipettor is included. This software is not required except for firmware updates.
- Python language, as seen in the Python example code `"scripts\inf-example.py"` in the starter software.
- Terminal programs like `putty`.

Command Set

Basic Operations Overview

Commands in the following section are the most common operations. For example, `/1Z1I0d+p100B1M500B0R` will charge the reservoir to +100 mbar, as follows:

- `Z1` initializes the device (should be done once after power-up).
- `I0` closes the isolation valve.
- `d+` puts the directional valves on the pump for positive pressure.
- `p100` sets the target pressure to +100 mbar.
- `B1` turns on the pump, blowing air and using the pressure target previously specified.
- `M500` waits for 500 ms, sufficient time to stabilize to most pressures.
- `B0` turns off the pump.

Some variations of this are possible, but in general, before turning on the pump (`B1`):

- Close (`I0`) the isolation valve to prevent the reservoir from leaking while you pressurize it.
- Set the pump direction valve to positive `d+` or negative `d-` pressure mode.
- Set either the target power (`m`) for open-loop or target pressure (`p`) for closed-loop operation. E.g. `d+p100` is +100 mbar, but `d-p-100` would be -100 mbar, and `d+m200` would be a 200 mW power target. Setting a target pressure (`p`) rather than target power (`m`) will automatically control the power to reach the stated pressure (in *closed loop* fashion), as measured with the *calibrated* pressure sensor.

The pressure can now be utilized:

- To vent through the isolation valve, issue an `I1`, or momentarily pulse it open for say 50 ms with `P50` (which is short-hand for `I1M50I0`). About 2 ms minimum time is required to physically actuate the valves.
- You may pulse open the isolation valve before or after the `B0` turning off the pump (after if you don't want the pump filling the reservoir while the reservoir is venting). E.g. `/1I0d+p100B1M500d0B0M200P50R` will release pressure from the reservoir through the isolation valve alone for 50 ms (very controlled), but `/1I0d+p100B1M500M200P50B0R` will continue to pump into the reservoir when venting through the isolation valve.



- You can hold a pressure by keeping the pump on (B1). Alternately, turn off the pump (B0) and close all directional valves to the pump (d0) to prevent leaking through the pump, and it will hold for some seconds or minutes with minimal leakage—e.g. B1M500d0B0R.

Some other details:

- Z1 will initialize the device. It is best practice, for most repeatable results, to call this on power-up, before the first time you do a blower or valve operation; it need not be called subsequent times.
- R at the end of the command will immediately run (not queue) the command. Most commands should be terminated by R, apart from queries (e.g. /1Q) and in rare needs of queuing commands for synchronous execution on multiple devices.

Use Q to poll for completion of any long running commands. E.g. /1Q will reply with a busy (/0@) or not busy (/0`) status character. You may not issue another operation until the current operation is busy, but you can query status.

Avoid applying high power to the pump for extended periods of time to avoid stressing the pump. Turn off the pump (B0) when not in use for best life. Be aware closed loop operation will apply maximum power to the pump if the set-point is above when it can reach physically, such as with the isolation valve open.

Other features are detailed in subsequent sections. E.g.

- You can poll pressure (/1??p) or for higher rates and more precisely timed acquisition (up to 1000 samples/s or 1 ms period) log to on-device memory (L) for subsequent download (?L). Another feature is to wait for a pressure change without the overhead of downloading (p< / p>).
- It may be warranted to tune the PID constants in closed loop control mode for greater stability.
- You can store short programs on the device flash memory (s0) for running without RS485 connected (untethered).

Basic Operations

"m" (sets pump target power)

Sets the pump target power in milliwatts. This can be set before or after the pump is turned on (B1). This defaults to 200 mW if unset (but should not be relied on).

Parameter: 0 to 1250, the power in mW, but the upper limit can be changed via the U10 command.

Example: /1m100B1M1000B0R

This mode is referred to as *open loop*, for it directly sets the power. See also the alternative p command for controlling power automatically to reach set-point pressure (*closed loop*). Setting p or m will override the other.

"p" (sets pump target pressure)

Sets the pump target pressure in millibar. When the pump is turned on (B1), the device will regulate pump power so that it holds the given target pressure in mbar. A PID loop (closed loop) is used for this. This pressure can be set before or after the pump is turned on (B1).

Parameter: -1000 to 1000, the target pressure in millibar, though in practice this should not exceed ± 430 mbar, which is the limit on the pressure sensor and the ability of the pump (about -300 to +400 mbar depending on conditions).

Example: /1p100B1M1000B0R

The PID constants (U10-U14) can tune the control behavior.

If the pump fails to reach target pressure, it will max out power in trying to reach that, which should be avoided over long periods of time.

See also the m command which sets the pump power directly. Setting p or m will override the other.

"d" (sets pump valves to +/- pressure mode)

Sets pump valves for positive or negative pressure pumping. Valves will switch, and any pressure PID control will be adjusted accordingly. On power-up the valve state is d0.

Parameter: + for positive pressure, - for negative pressure, 0 for valves powered off (reservoir isolated), 1 for both valves on (reservoir isolated but connected to pump, rarely used).

Examples:

```
/1d+R  
/1d-R  
/1d0R
```

Presently this command is equivalent to the following low-level commands:

d+ is U19, 1J2+J3c (Note: U19 sets the PID polarity.)

d- is U19, 0J2cJ3+

d0 is J2cJ3c

d1 is J2+J3+

"B" (enables/disables the pump)

Turns pump on or off (blowing air).

Parameter: 0=off, 1=on

Examples:

```
/1B1R  
/1B0R
```

The pump will work to maintain the power (m) or pressure (p) target specified.

"I" (sets isolation valve state)

Sets isolation valve state.

Parameter: 0=close, 1=open

Examples:

```
/1I1R  
/1I0R
```

See also the **P** command.

Presently this is equivalent to the low-level commands J1+ (open) or J1c (close). On power-up the valve state is I0.

"P" (pulses isolation valve open)

Pulses open isolation valve for the given number of milliseconds and then closes it.

Parameter: 0-10000, in milliseconds

Example: `/1P100R`

Presently this is equivalent to `I1M{milliseconds}I0` for given value of *{milliseconds}*.

"E" (ejects tip)

Extends or retracts tip ejector, if a tip ejector is connected.

Parameter: ejector position, either extend (1) or retract (0).

This command can be used for tip ejection by combining it with a delay (M) as follows:

```
/1E1M500E0R
```

"b" (turns on buzzer alarm)

Turns on buzzer alarm at the given frequency or turns it off. This can be used as an audible alert.

Parameter: 0 – 16666, the frequency in Hz. 0 is off.

Example: `/1b500M1000b0R`

"Q" (queries current status)

Queries current status. This returns nothing, apart from the status character.

Examples:

```
/1Q → /0@ (binary 0100 0000, i.e. not ready)  
/1Q → /0` (binary 0110 0000, i.e. ready)
```

Some common status characters (see Protocol section for more details):

- `@` (binary 01000000) means not ready (e.g. running commands) and no error
- ``` (binary 01100000) means ready (done) and no error
- `b` or `B` (binary 01X00010, code=2) means bad command.

This command is often used to poll the device for the completion (not busy state) of a long running operation.

"Z" (initializes device)

Initializes the device, including the pump.

Parameter: 1 (always), the initialization mode.

Example:

```
/1Z1R → /0@
```

This may do various initializations and checks but presently only prepares the pump.

It is best practice to call Z1 after power-up and before any other blower (B) or valve operation (I/d), for most reproducible results. Presently you can omit Z1, but future versions can require it. If you omit Z1, then the first time you turn on the blower (B1) it may not act in the usual way in the first 100 ms or so due to needing to initialize.

Waiting and Conditionals

"M" (delays for given milliseconds)

Waits for given duration in milliseconds.

Parameter: 0 .. 600000, in milliseconds, with up to three decimal places.

Example:

```
/1P200M1000D200R
/1I1M1.523I0R
```

Longer waits could be done with a loop (e.g. gM1000G300).

"Me" (delays for given milliseconds since last M0 command)

Waits until the given milliseconds after the last executed M0 (zero delay) call, similar to M.

Parameter: 0 .. 600000 milliseconds, with up to three decimal places.

This is useful to ensure a fixed time after a variable time operation. Example:

```
/1M0m200B1p>100M1000B0MB1000b500M100b0R
```

This marks the current time (M0), turns on the pump at 200 mW (m200B1), waits up to 1000 ms or until it reaches 100 mW (p>100M1000), whichever first, turns off pump (B0), waits for the remainder of the 1000 ms (Me1000) since the M0, and beeps at 500 Hz for 100 ms (b500M100b0).

"p<", "p>" (waits until pressure falls below or rises above value)

Causes a subsequent delay (M or Me command) to prematurely stop if the pressure falls below (<) or rises above (>) the given value in mbar.

Parameter: -999.9 to 999.9 in mbar

Examples:

```
/1p<10.5M5000R (waits until falls below 10.5 mbar)
/1p>50M5000R (waits until rises above 50 mbar)
```

Loop / Branch / Execution Commands

"g" and "G" (loops execution)

Repeats executing commands in a loop.

Parameter: 0 .. $2^{32}-1$, the number of iterations. 0 is infinite loop.

Example:

```
/1gP100M1000G3R (pulses isolation valve open three times)
/1gP100M1000G3R (pulses isolation valve open infinitely)
```

Usage caveats:

Loops can be nested up to five levels.

g is required to be used with this. So /1z0P1G3R is invalid.

Loops being run can be stopped using the T command.

Use G0 not G (deprecated) as in some other implementations, to avoid ambiguity.

"R" (begins execution)

Nothing executes until R is provided at the end of the command string.

```
/1p100B1M500P100B0R
```

R is not used on immediate ? prefixed status commands, Q, &, and T (terminate).

R is omitted mainly when precisely coordinating multiple RS485 devices (see Multiple Devices/Axes).

```
/1P500M100P500
/2P500M100P500
/AR
```

You can queue a command by omitting the R and later execute it one or many times by sending just R.

```
/1P50M100P50
/1R
/1R
```

After doing `R`, most further commands will fail until the device becomes non-busy.

"T" (terminates currently running command)

Terminates any running command. The device will be non-busy after this.

Examples:

```
/1M10000P10R → /0@ (busy)
/1T → /0`
/1Q → /0`
```

"s" (stores a program string).

Stores a program on the (non-volatile) flash memory.

Parameter: 0-15, program string number

Parameter: the command string. This must be terminated by `R`. If empty it deletes the command string.

Examples:

```
/1s0b500M1000b0R (store program 0, which beeps twice)
/1e1M1000e1R (execute program 0 twice, with delay)
/1s0R (clear program 0)
```

Note: When you re-flash, the non-volatile memory contents remain intact (unchanged).

This command will fail if the total program memory exceeds 2044 bytes.

This will fail (bad command/busy) if a command is currently running. Issue `/1T` to terminate any currently running program before issuing the `s0` command.

See also the `?s{id}` command to read a stored program.

"e" (executes stored program)

Executes stored program string.

Parameter: 0-14, the program string number

Examples:

```
/1s0b500M1000b0R    (store program 0)
/1e1M1000e1R        (execute program 0 twice, with delay)
```

Calls cannot be nested (e.g. `e` calling a program string that itself contains `e`), at least at present.

"?s{id}" (gets stored program string)

Retrieves stored program string in flash, as set by the `s{id}` command.

Example:

```
/1s2,b500M100b0R → /0`
/1?s2           → /0`b500M100b0R
```

Status / Query

"&" (get firmware revision and date)

Gets firmware type and version string. It has the format "IMI Adaptas - INF:v{A} .{BB} {YYYYMMDD}" where A.BB is the version number and YYYYMMDD is the date. Additional characters after the date may exist but should be ignored.

Examples:

```
/1&           → /0`IMI Adaptas - INF:v1.09 20231128
```

"?m" (gets pump power target)

Gets pump power setpoint (mW) set by the `m` command. Returns nothing if `p` (or `f`) command is used.

Examples:

```
/1m100R      → /0`
/1?m         → /0`100
/1p200R      → /0`
/1?m         → /0` (if pressure target set instead of power level)
```

See `??P` for actual power.

"?p" (gets pump pressure target)

Gets pump pressure target (mbar), set by the `p` command. Returns nothing if `m` (or `f`) command is used.

Examples:

```
/lp200R → /0`
/l?p     → /0`200
/lm100R  → /0`
/l?p     → /0` (if power level set instead of pressure target)
```

See `??p` for actual pressure.

"?z" (gets valve/pump high level state)

Gets the valve and pump state as a string of three characters:

- 1: pump state: 0 (off), 1 (on). See the `B` command.
- 2: pump valve direction: + (positive pressure), - (negative pressure), 0 (both valves off), 1 (both valves on). See the `d` command.
- 3: isolation valve state: 0 (closed), 1 (open). See the `I` command.

On first powering on the device, this returns `0??` but will return `000` in the next update.

```
/l?z → /0`0+1
```

"?4" (reads inputs)

Gets the state of the digital inputs as a string of binary digits, starting with bit 0:

- Bit 0 = input 1 (DIP 5)
- Bit 1 = input 2 (DIP 6)
- Bit 2 = input 3 (DIP 7)
- Bit 3 = input 4 (DIP 8)
- Bit 4 = accessory I/O pin 5, if configured as an input.
- Bits 5-7 = reserved (unused), default to 1.

0 is OFF. 1 is ON.

Example:

```
/l?4 → /0`00011111 (if DIP8 is on).
```

Warning: a future firmware will only return the first four bits in this command (`/0`0001`) so expect either four or eight digits.

"?20" (gets command run time)

Gets the time in milliseconds to run the last command.

Example:

```
/1M1000R
...
/1Q           (repeat until not busy)
/1?20 → /1`1000 (approximately)
```

"?U500" (gets device serial number)

Gets device serial number stored in flash memory as a 32-bit integer.

Example:

```
/1?U500 → /1`1000 Logging
```

Pressure, power, and other variables can be logged in real-time to the on-device memory buffer. This is an alternative to polling from the serial line (e.g. using `??p`) and uses no communication during the logging but rather the log is downloaded at leisure after the operation being logged is complete or the log is analysed on the device without downloading. This can achieve higher and more repeatable sampling rates (up to 1000 samples/second) and less or deferred communication. The download can take on the order of second depending on sampling rate, sampling duration, and baud rate.

Any operation sequence can be logged on the device by surrounding the operations with `L1` and `L0` commands. The log may be subsequently downloaded by polling `?L`.

"??" (queries multiple status variables)

Queries multiple status variables at once. Required variables are indicated by single letters:

`p` – pressure in mbar (up to tenth of a decimal place), -430.9 to 430.9 mbar (+/- 6.25 psi), though the compensated range is +/- 5 psi, and min/max values can indicate saturation (pressures outside range of sensor). -10000 is an error.

`F` – pump frequency in Hz (to nearest integer)

`I` – pump current in mA (up to tenth of a decimal place)

`P` – pump power in mW (up to tenth of a decimal place)

`V` – pump voltage in Volts (up to tenth of a decimal place)

Returned values are comma delimited.

```
/1??p → /0`-0.6
```

```
/1??pPV → /0`-0.6,0.0,17.2
/1?? → /0`
```

“L” (begins/ends pressure logging)

This enables recording of pressure reading to a buffer in memory which can later be retrieved. Data can be recorded at high frequency and time precision, unlike polling (?? command). When enabled (L1), any previously logged data is cleared, but L2 will append to previous data. The buffer has finite size (presently 10000 records) and will stop appending if full.

Parameter: 0 (disable logging), 1 (enable logging), 2 (resumes logging after disabled)

Parameter: 1-65535 (default 1), the milliseconds between points.

Parameter: 1-2, the format:

1 (default) is pressure in mbar.

2 is pressure in mbar, power in mW, and isolation valve state (0=closed, 1=open, -1=unknown).

Examples:

```
/1L1R → /0`
/1L0R → /0`

/1L1,10R → /0` (logs only every 10th point)
/1L0R → /0`

/1L1,10,2R → /0` (logs only every 10th point, using format 2)
/1L0R → /0`
```

See also the “?L” command.

“?L” (gets pressure log data)

Retrieves records of pressure data saved by the last pressure logging (L command). The first call returns the format (1 or 2). Each additional call returns semicolon delimited records of comma-delimited values. Calls return complete records (records do not span two calls).

If format is 2, records contain

- time (milliseconds). Times are 32-bit unsigned integers.
- pressure (mbar)

If format is 2, there are additional values in the record:

- Power (mW)

- Isolation valve state: 0 (closed), 1 (open), -1 (unknown)

When there is no more data, an empty string is returned. This is designed to be called when pressure logging is finished (L0). Once data is retrieved it cannot be downloaded again.

```

/1p100L1B1M1000B0L0R
/1?L → /0`1
/1?L → /0`0,-0.7;1,-0.7;2,-0.7;3,-0.7;4,-0.7;5,-0.8;6,-0.7;7,-
0.8;8,-0.8;9,-0.8
/1?L → /0`10,-0.8;11,-0.7;12,-0.7;13,-0.7;14,-0.7;15,-0.7;16,-
0.8;17,-0.7;18,-0.8;19,-0.7
...
/1?L → /0`990,-0.9;991,-0.8;992,-0.8;993,-0.8;994,-0.8;995,-
0.8;996,-0.9;997,-0.8;998,-0.9;999,-0.7
/1?L → /0`

/1p100L1,10,2B1M1000B0L0R
/1?L → /0`2
...
/1?L → /0`

```

PID Configuration

Closed loop operation uses PID to control the power $m(t)$ over time to achieve a target pressure defined by the `p` command. The PID constants — proportional K_p , integral K_i , and derivative K_d — and power limit can be read using `?11` - `?14` or adjusted using `U11` - `U14` commands. Formally, given the error difference between set-point and actual pressure, $e(t)$, the PID constants are defined as

$$m(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d * \frac{de(t)}{dt}$$

Concerning these four constants:

- Most typically, you want to **adjust the P (proportional constant)** to make it more or less responsive. That might be the only parameter you need to adjust.

- Sometimes **I (integral constant)** might be adjusted, if there is a long term offset between target and set points (overshoot or undershoot).
- The **D (derivative constant)** should normally be left at zero and is normally not appropriate. It is currently computed on a time scale of 1 ms with no noise elimination.
- The **integral limit** might also be useful cap the milliwatts current when the pump first turns on, especially if the volume to fill is large and takes some time to equilibrate. This **can avoid hitting max power on turn on**.
- The constants can be adjusted by trial-an-error or other methods like Ziegler-Nichols.

"?10" / "U10" (gets/sets power limit)

Gets (?) or sets (U) the power limit in mW for the pump. This does not persist across reboots. Default: 1250 mW. This can be increased beyond this but is not recommended and voids any warranty. You may reduce this if you do not need higher pressures.

Example:

```
/1U10,1000R  
/1?10 → 1000
```

"?11" / "U11" (gets/sets PID proportional constant)

Gets (?) or sets (U) the PID proportional constant K_p for the pump. This constant is in units of mW/mbar for pressure control. This does not persist across reboots. Default: 15.

Example:

```
/1U11,20R  
/1?11 → 20.000000
```

"?12" / "U12" (gets/sets PID integral constant)

Gets (?) or sets (U) the PID integral constant K_i for the pump. This constant is in units of mW/(mbar*seconds) for pressure control. This does not persist across reboots. Default: 50.

Example:

```
/1U12,60R  
/1?12 → 60.000000
```

"?13" / "U13" (gets/sets PID derivative constant)

Gets (?) or sets (U) the PID derivative constant K_d for the pump. This constant is in units of mW/(mbar/seconds) for pressure control. This does not persist across reboots. Default: 0. This should be zero nearly always and is rarely helpful.

Example:

```
/1U13,1R  
/1?13 → 1.000000
```

"?14" / "U14" (gets/sets PID integral limit)

Gets (?) or sets (U) the PID integral maximum value. The integral will not accumulate beyond this value. This does not persist across reboots. Default: 1400.

Example:

```
/1U14,1000R  
/1?14 → 1000R
```

"?19" / "U19" (gets/sets PID polarity)

Gets (?) or sets (U) the PID polarity, either positive (1) or negative (0). In positive mode, applying pump power is assumed to increase pressure; in negative mode, power is assumed to reduce pressure. Typically you do not use this, for d+ or d- sets it automatically.

```
/1U19,1R  
/1?19 → 1R
```

Configuration

"U41" "U47" "U48" (sets baud rate)

Changes the baud rate. The baud rate is stored in flash and will become effective upon reset, such as using the Q* command or reset/power-up. The default is 115200 baud. DIP switch DIP8 if ON will override this to 9600 baud.

Examples:

```
/1U41R (9600 baud)  
/1U47R (39400 baud)
```

```
/1U48R (115200 baud)
/1Q* (reset)
```

"?40" / "U40" (gets/sets DIP switch mode)

Gets or sets the DIP switch interpretation mode:

- 0 – DIP5 and DIP6 are used for address (4 addresses)
- 1 – DIP5, DIP6, and DIP7 are used for address (8 addresses) – default
- 2 – DIP5, DIP6, DIP7, and DIP8 are used for address (16 addresses)

Examples:

```
/1U40,1R → /0`
/1?40 → /0`1
```

This persists in flash memory.

"?100" (gets baud rate setting in flash)

Gets the baud rate setting in flash (set by U41/U47/U48). The actual rate may be overridden by the DIP switch though.

Examples:

```
/1U41R
/1?100 → 9600
```

System

You can reset the board with Q* .

"Q*" (reset board)

Resets the board (soft reset, similar to repowering the device).

Example:

```
/1Q* → (no reply)
```

No reply presently returned, but a future version may return a busy /0@ reply, so assume both.

This takes well under 1/2 second if no stored startup program (s0) is defined.

Low-Level Operations

These operations can be used to control the device more directly than the Basic Operations.

"J" (turns valve driver on/off)

Turns a valve on or off.

Parameter: valve (1-4) and direction (+ on with positive polarity or c for off) for the output driver.

1+ POSITIVE signal to valve 1 (isolation valve)

1c NEUTRAL (off/coast) signal to valve 1

2+ POSITIVE signal to valve 2 (positive pressure supply)

2c NEUTRAL (off/coast) signal to valve 2

3+ POSITIVE signal to valve 3 (negative pressure supply)

3c NEUTRAL (off/coast) signal to valve 3

4+ POSITIVE signal to valve 4 (accessory valve or motor)

4c NEUTRAL (off/coast) signal to valve 4

5, {ratio} Sets tip ejector PWM output to 50 Hz with duty cycle ratio 0 to 1. Full range is 0.0475 to 0.075 (e.g. J5, 0.075). WARNING: going outside of this range can damage the system. The 'E' command is recommended instead.

6, {ratio} Sets Aux #1 port pin 5 PWM output to 50 Hz with duty cycle ratio 0 to 1 (e.g. J6, 0.05).

Example:

```
/1J1+M100J1cR (pulse isolation valve ON for 100 ms)
```

The high-level P and d commands are normally recommended instead.

"?J" (gets valve state)

Returns four characters in set {+, c}, one for each valve coil (J1, J2, J3, J4), representing its state.

Examples:

```
/1J1+J2+J3-J4cR → /0`
```

```
/1?J → /0`+++c
```

Miscellaneous Topics

Multiple Devices/Axes

The device can be paired with other devices, such as additional pipettors, Adaptas Z or XYZ axes, or an I/O expansion board. It is generally possible to daisy chain the devices onto the same RS485 bus, which can be connected to a single USB serial port, provided the following conditions are met on all devices:

- RS485 compatible with data terminal/DT (Cavro-style) communication protocol.
- RS485 addresses unique on the bus (set via DIP switches on the device or rotatory switch on other devices).
- Same baud rate (set via DIP switches or serial command to flash memory).
- 24VDC power optional but allows using the same 4-wire power & communications cables.

The device by default runs at 115,200 baud. It can run just as well at 9600 baud, except transfer to pressure curve logs ($\underline{L}/\underline{?L}$) can be much slower at 9600 baud. So, 115,200 is recommended for all devices especially if this logging feature is used. See Baud Rate Changing.

Commands sent to two devices normally will be sent sequentially.

```
/1P100R  
/2P100R
```

There will be a small delay between these commands, on the order of 10-100 milliseconds, depending on the baud rate and PC load. There is, however, a way to execute commands simultaneously to multiple devices if timing is more critical. A command set to RS485 address “A” is received by both axes 1 & 2 but does not return a status (address “Q” works on axis 1-4).

```
/AP100R
```

You must query status of axes individually for completion:

```
/1Q  
...  
/1Q  
/2Q  
...  
/2Q
```

To send different commands to different devices and have them start at exactly the same time, send each device with a command but without the final `R`. Then send `/AR` to start both axes at the same time.

```
/1P100
/2P100
/AR
/1Q (repeat until not busy)
/2Q (repeat until not busy)
```

The `Q` address will issue to devices 1-4, but if some of those devices are connected but should do nothing, you can queue a dummy command like a short delay (`M`) to them.

```
/1P100
/2M1
/3D100
/4M1
/QR (issues to devices 1-4)
```

Firmware Burning

New firmware can be burned to the device using the starter software (“Firmware” tab on the pipettor module).

Baud Rate

The default baud rate is 115,200 baud, but the device natively supports 9,600, 38,400, and 115,200 baud rates. The baud rate can be configured via serial commands (`U41/U47/U48`) which write to flash, but this can be overridden by a DIP switch (DIP #8) which forces 9,600 baud. The DIP switch is convenient for switching to 9,600 baud or forcing the device to a known baud rate if the baud rate in flash is not set. The device will need to be repowered for the change to become effective.

Your software will need to connect at this same baud rate. E.g. if using the starter software, set the `SERIAL_PORT_BAUD` variable in the `devices\adaptas-inf.lua` file.

Running without a PC or RS485 Communication

The `s0` command can store a command string in flash memory that gets run on power-up. This allows using the device without a PC or any RS485 communication.

```
/1s0gI0M1000I1M1000G0R (toggle isolation valve continually in loop)
/1Q* (restart)
```



Delays (`M`) and infinite loops (`g/G0`) may be helpful in startup programs.

You must terminate any running program (`/1T`) before attempting to re-program a new string.