TG003: PCB SERIAL COMMUNICATIONS TECHNICAL GUIDE

1. DISCLAIMER

This resource is provided "as is" and without any warranty of any kind, and its use is at your own risk. The Lee Company does not warrant the performance or results that you may obtain by using this resource. The Lee Company makes no warranties regarding this resource, express or implied, including as to non-infringement, merchantability, or fitness for any particular purpose. To the maximum extent permitted by law The Lee Company disclaims liability for any loss or damage resulting from use of this resource, whether arising under contract, tort (including negligence), strict liability, or otherwise, and whether direct, consequential, indirect, or otherwise, even if The Lee Company has been advised of the possibility of such damages, or for any claim from any third party.

2. INTRODUCTION

2.1.Piezoelectric Disc Pumps

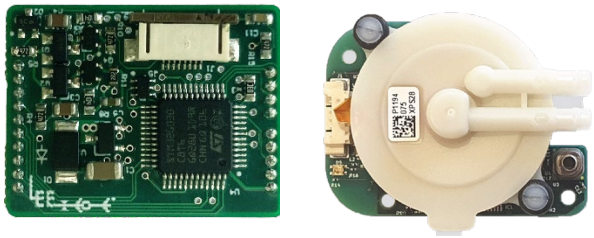
The Lee Company's piezoelectric disc pumps are silent, high-performance gas micropumps.

The disc pumps are designed to provide highly accurate, ultra-smooth pressure and airflow control of gases. Owing to its operating mechanism, the disc pumps can be controlled with unmatched precision, yet at the same time respond to full-scale set point changes in a matter of a few milliseconds. The compact form factor means it can be tightly integrated into products, increasing portability.

Figure 1. A piezoelectric disc pump

2.2.Driver communication

The Lee Company provides a range of PCB designs for driving the disc pumps. All designs share a common register-based control interface, which is accessed either via a UART or I2C protocol, depending on the driver. This Technical Guide provides details of the commands recognised by the drivers.



	General Purpose Driver	Smart Pump Module
UART	✓	✓
I2C	✗	✓

Figure 2. Piezoelectric disc pump drive systems and communication protocol support

2.3.Development Kit Motherboard

Typically, customers' first use of the disc pumps is with The Lee Company's Piezoelectric Disc Pump Development Kit. The Development Kit contains a General Purpose driver mounted on a motherboard. The motherboard acts as a breakout board for much of the driver's functionality, as well as adding useful peripherals, such as a dial control and pressure sensor. The motherboard can also be setup to communicate with a Smart Pump Module.

The motherboard allows UART communication between a host PC, and the General Purpose driver or Smart Pump Module, using a USB-to-UART converter, which appears on the PC as a virtual COM port.

It also allows I2C communication between a host PC, and multiple Smart Pump Modules, using a USB-to-I2C converter, which appears as a Human Input Device (HID) device. The I2C communication can be used through the Disc Pump Control application or third party tools such as Python and will not be covered in

this guide. The Lee Company provides a repository of code snippets to assist with commands on GitHub (https://github.com/The-Lee-Company).

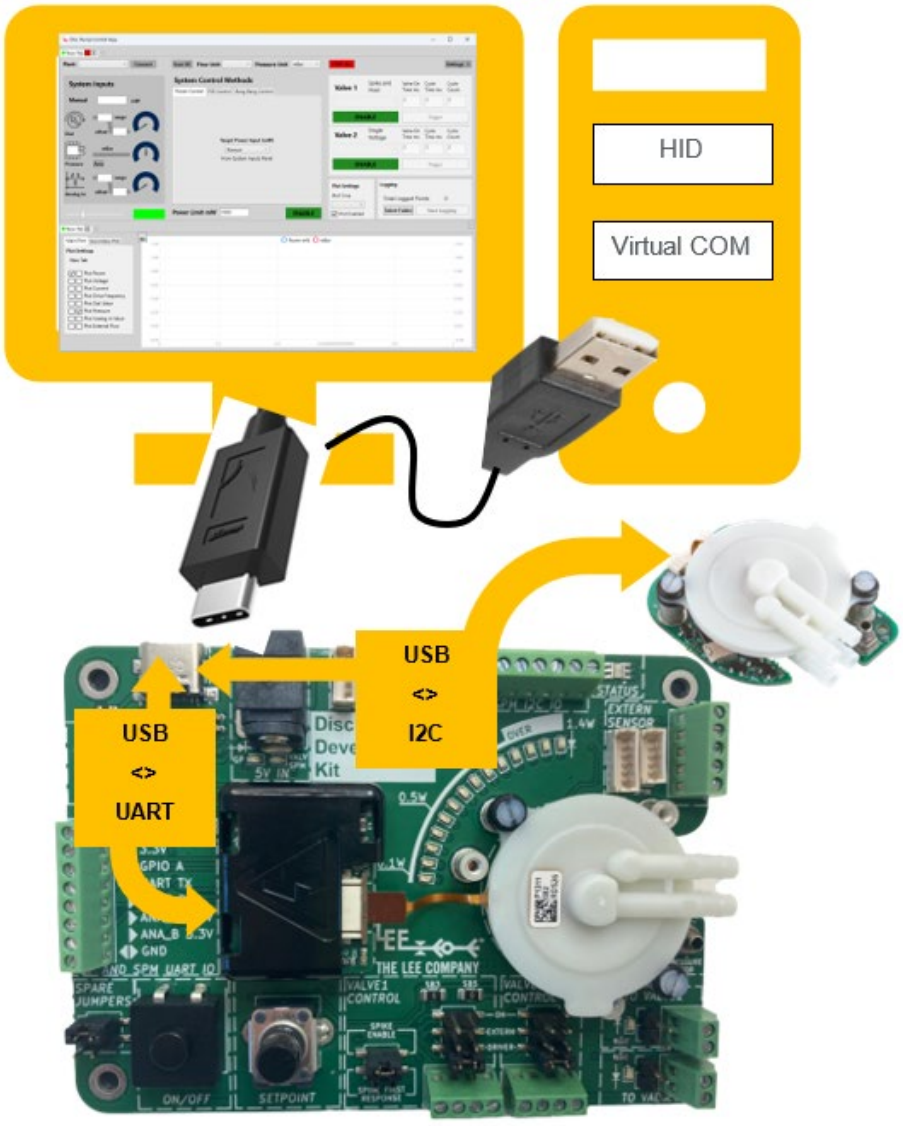


Figure 3. Connecting the Development Kit Motherboard to a host PC.

2.3.1. Using the Development Kit virtual COM Port

The Development Kit motherboard implements a serial-over-USB interface, accessible via a USB C receptacle.

2.3.2. Attaching the PCB to your computer

The PCB is attached to the host computer via a USB A to USB C lead. A suitable lead is supplied with the Development Kit. Please note that similar leads supplied (e.g. with bicycle lights) for battery charging tend not to have the data lines connected and are therefore unsuitable.

Upon first connecting the PCB to your computer, the operating system software should install the necessary driver to create a virtual COM port. If this does not happen automatically, please download and install the appropriate device driver from the Microchip website: https://www.microchip.com/en-us/product/mcp2200

Once the COM port is installed, make a note of the port number. In Windows this can be identified by looking in the “Ports” section of the “Device Manager” window. With the Ports section open, unplug the USB cable connecting your computer to the PCB and take note of which COM port disappears. Reconnect the USB cable once the port is identified.

2.3.3. Opening the COM port

The PCB’s COM port runs at a baud rate of 115,200, 8 bits, no parity, and one stop bit.

Please refer to the documentation supplied with your chosen software development environment for details of how to open a serial port.

2.4.Evaluation Kit Motherboard

The Evaluation kit was the previous development platform for The Lee Company's Piezoelectric Disc Pump. The Evaluation Kit contains a General Purpose driver mounted on a motherboard. The motherboard acts as a breakout board for much of the driver's functionality, as well as adding useful peripherals, such as a dial control and pressure sensor.

The motherboard allows UART communication between a host PC, and the driver, using a USB-to-UART converter, which appears on the PC as a virtual COM port. All communication between the PC and driver uses the register-based UART interface described herein.

Figure 4. Connecting the Evaluation Kit Motherboard to a host PC.

2.4.1. Using the Evaluation Kit virtual COM Port

The Evaluation Kit motherboard implements a serial-over-USB interface, accessible via a USB mini-B receptacle.

2.4.2. Attaching the PCB to your computer

The PCB is attached to the host computer via a USB A to USB mini B lead. A suitable lead is supplied with the Evaluation Kit. Please note that similar leads supplied (e.g. with bicycle lights) for battery charging tend not to have the data lines connected and are therefore unsuitable.

Upon first connecting the PCB to your computer, the operating system software should install the necessary driver to create a virtual COM port. If this does not happen automatically, please download and install the appropriate device driver from the FTDI website: https://www.ftdichip.com/FTDrivers.htm

Once the COM port is installed, make a note of the port number. In Windows this can be identified by looking in the “Ports” section of the “Device Manager” window. With the Ports section open, unplug the USB cable connecting your computer to the PCB and take note of which COM port disappears. Reconnect the USB cable once the port is identified.

2.4.3. Opening the COM port

The PCB’s COM port runs at a baud rate of 115,200, 8 bits, no parity, and one stop bit.

Please refer to the documentation supplied with your chosen software development environment for details of how to open a serial port.

2.5.Units

The following units are used:

Quantity	Unit
Flow	mL/min
Voltage	V
Current	mA
Power	mW
Frequency	Hz
Pressure	For the Development kit, Evaluation kit and drive PCBs, the following pressure units can be selected in the Disc Pump Control App: • mBar (default unit) • mmHg • PSI • kPa • inHg • inH20 Pressure readings are reported as gauge (rather than absolute) values.

3. UART INTERFACE

3.1.Overview

PCB operation is controlled by a number of registers, which offer either “read” or “read/write” access. Commands take the form of a string of ASCII characters terminated with a new-line character. Commands are sent to the driver, following which the driver responds to acknowledge the command and, in the case of a read request, returns the requested value. The UART operates at 3.3V for the Smart Pump Modules and General Purpose Driver, however the modules are 5V tolerant. For the older Fast Response PCB (obsolete) the UART operates at 2.5V.

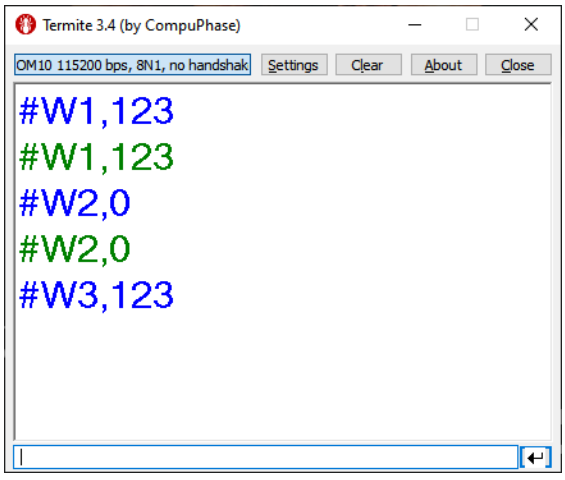
3.1.1. Writing

To write to a register, send the write command, which takes the form:

#W<REGISTER_NUMBER>,<VALUE>\n

(Note that the terminating ‘\n’ represents the ascii new-line character, i.e., ‘\n’ shows the byte 0x12 being sent, rather than the individual characters ‘\’ and ‘n’)

The PCB responds to “write” commands by echoing the command back. This response should be read and checked by the controlling software to confirm that the command has been received correctly. If the command causes an error, or is not received at all, the PCB does not respond.



Example 1. Example UART interaction, writing registers 1 and 2. Writing to register 3 fails, as this register is read only. Blue: Command sent Green: Driver response

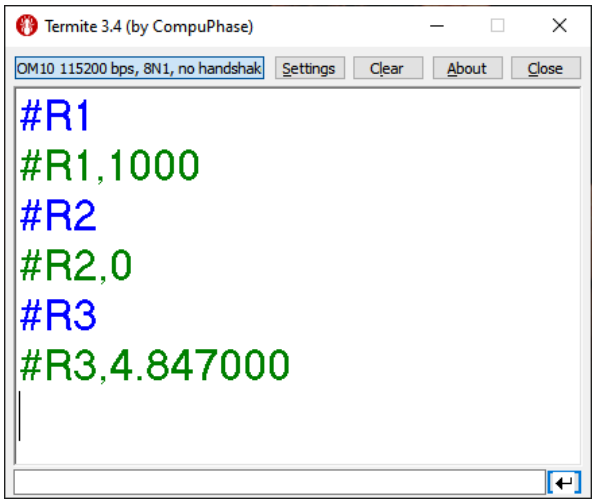
3.1.2. Reading

To read from a register, first send the read register command, which takes the following form:

#R<REGISTER_NUMBER>\n

(Note that the terminating '\n' represents the ascii new-line character, i.e., '\n' shows the byte 0x12 being sent, rather than the individual characters '\' and 'n')

The driver then responds by echoing the command back, followed by a comma and the register value.



Example UART interaction, reading registers 1, 2, and 3 in turn

Blue: Command sent Green: Driver response

3.2.Stream mode (UART)

When communicating over UART a streaming mode is available, in which a comma separate list of useful variables is sent periodically (at approximately 60Hz). This mode can be activated by setting the Enable stream mode register, see Section 5.3. Streamed variables take the form:

#S<PUMP_ENABLED>,<VOLTAGE>,<CURRENT>,<FREQUENCY>,<ANA1>,<ANA2>,<ANA3>,<FLOW>,<CHK>\n

Streaming format for drivers

#S<PUMP_ENABLED>,<VOLTAGE>,<CURRENT>,<FREQUENCY>,0,<DIGITAL_PRESSURE>,<ANA3>,0,<CHK>\n

Streaming format for modules

The <CHK> field contains a simple 1-byte checksum, used to validate the rest of the streamed message. The checksum is computed by taking the ascii value of each character in the line, before <CHK> appears, and adding them together. This sum is then limited to 0-255 by taking the modulo of the sum to 255:

SUM % 256 = expected checksum value

The normal command-response protocol can still be used with streaming mode activated. The driver will intersperse responses to read and write commands with the streaming output as necessary.

3.3.Number formatting

Numerical data is encoded in ASCII format. Floating point numbers should use the format *12.345*. Other number formats (e.g. scientific notation) are not supported.

4. I2C INTERFACE (FOR SPM ONLY)

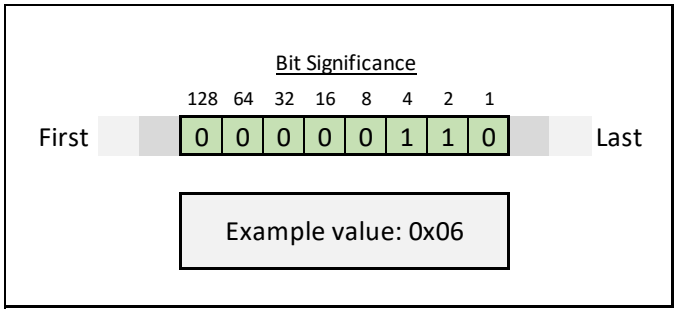
4.1.Overview

The I2C bus is a synchronous, serial, 2-wire, half-duplex, multi-drop protocol that allows communication with several devices using a shared electrical bus. The bus typically operates at 3.3V however it is 5V tolerant.

Specification	Min	Typical	Max	Unit
I2C bus speed	100	400	400	kHz
Output level low		0	0.9	V
Output level high	2.4	3.3	5	V
Pull-up resistance on SDA & SCL *	0.2		10	kOhm

Table 1. I2C Speed and Level Parameters for Ventus Drivers / Modules

* Note that if multiple I2C devices (Smart Pump Modules or other) are connected to the same I2C bus, it is recommended to use lower value pull-up resistor values to compensate for the increased bus capacitance.



Individual bytes are sent over the bus with the most significant bit first

Driver registers have two types:

int16_t	A signed, 16-bit integer
float	A 32-bit floating point value (typical IEEE 754 format)

When reading or writing to a register over I2C, the format of the register must be known up front, and the correct number of bytes read / written.

Note: *The I2C Master must support clock stretching.*

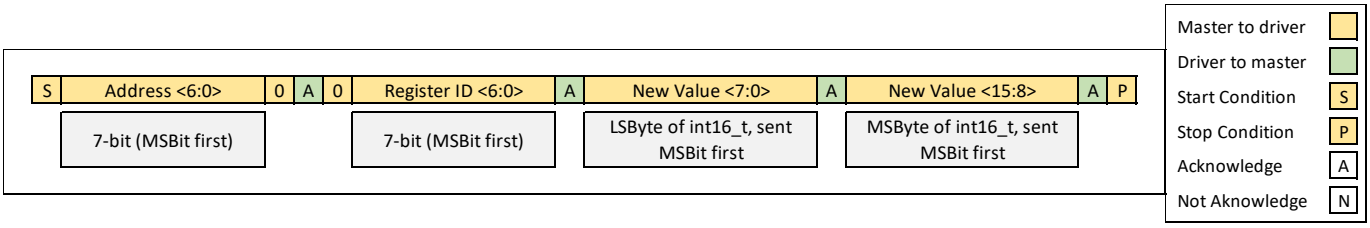
4.2.Address

The default device address is 37. See register “I2C address” for changing the I2C address of the board.

4.3.Writing to register with the int16_t type

To write to an int16_t register, the master:

- Initiates a write transfer with the driver, by sending the driver’s I2C address, followed by the read/write flag, which is set to “0”
- Sends a further byte to the driver, where the 7 LSBits (Least Significant Bits) indicate the register ID to be written, and the MSBit (Most Significant Bit) indicates a register write (set to “0”)
- Sends the int16_t value as two bytes, with the least significant byte first



Writing to an int16_t Register

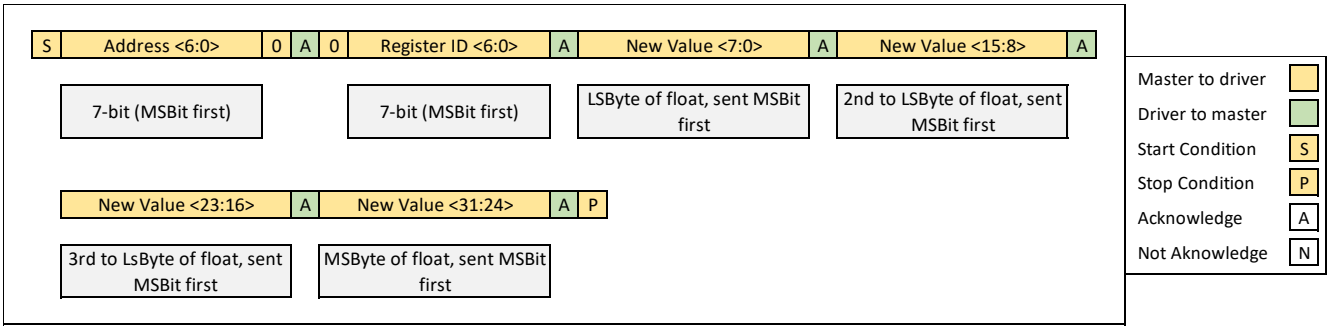
4.4.Writing to register with the float type

To write to a float register, the master:

- Initiates a write transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to "0".

- Sends a further byte to the driver, where the 7 LSBits indicate the register ID to be written, and the MSBit indicates a register write (set to "0")

- Sends the float value as four bytes, with the least significant byte first



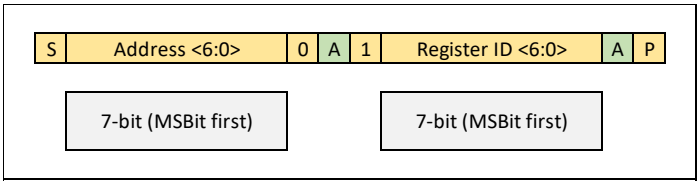
Writing to a float Register

4.5.Reading a register with the int16_t type

To read to a register with the int16_t type, the master first:

- Initiates a write transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to "0"

- Sends a further byte to the driver, where the 7 LSBits indicate the register ID to be written, and the MSBit indicates a register read (set to "1")

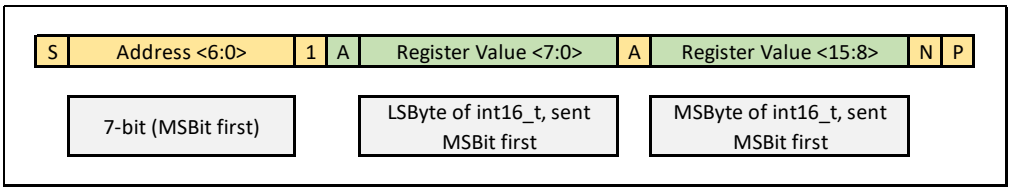


Master selects a register to read

Second, the master:

- Initiates a read transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to "1"

- Reads two bytes back from the driver, where the first byte is the least significant byte of the int16_t



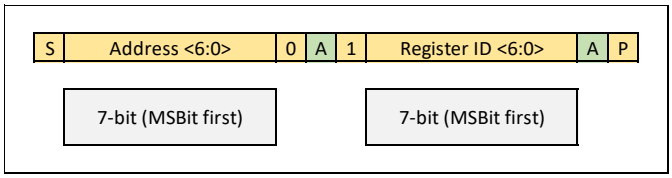
Master reads back the two bytes from the selected int16_t register

4.6.Reading a register with the float type

To read to a register with the float type, the master first:

- Initiates a write transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to "0"

- Sends a further byte to the driver, where the 7 LSBits indicate the register ID to be written, and the MSBit indicates a register read (set to "1")

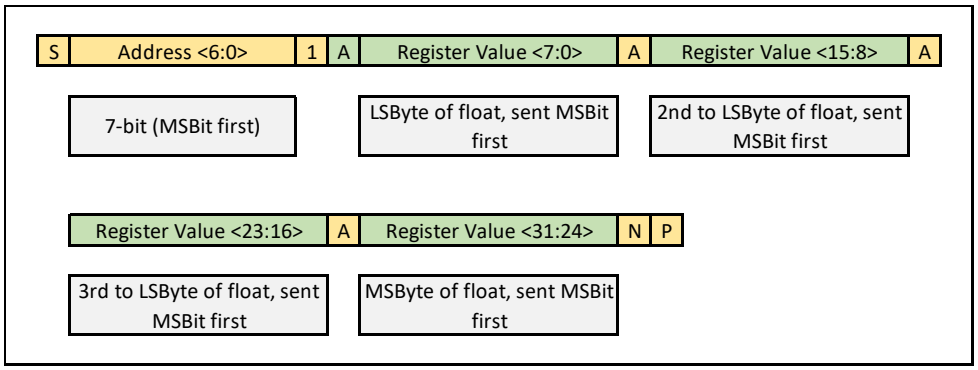


Master selects a register to read

Second, the master:

- Initiates a read transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to "1"

- Reads four bytes back from the driver, where the bytes go least to most significant



Master reads back the four bytes from the selected float register

4.7.I2C stream mode

Typically, when the module receives and unexpected I2C read transfer (an unexpected read transfer is one that was not preceded by a write transfer indicating which register is to be read) it will respond with a singular 0. When I2C stream mode is enabled (though the Enable stream mode register, see Section 5.3), this default behaviour is altered to return a list of useful variables. This direct response method has a much higher data transfer rate as compared to the master requesting each variable separately.

In I2C stream mode when the module receives an unexpected I2C read transfer (an unexpected read transfer is one that was not preceded by a write transfer indicating which register is to be read) it will respond with a list of variables that is the same as the UART stream mode. The total number of bytes sent by the module is 29 and they carry the following information:

[0:1]<PUMP_ENABLED>, [2:5]<VOLTAGE>, [6:9]<CURRENT>, [10:11]<FREQUENCY>, [12:15]<"0">, [16:19]<DIGITAL_PRESSURE>, [20:23]<ANA3>, [24:27]<"0">, [28]<CHK>\n

Streaming format for modules and which bytes (inclusive) each value is. 4 bytes is a float value, and 2 bytes is a int_16 value.

The <CHK> field contains a simple 1-byte checksum, used to validate the rest of the streamed message. The checksum is computed by taking each byte, before <CHK> appears, and adding them together. This sum is then limited to 0-255 by taking the modulo of the sum to 255:

SUM % 256 = expected checksum value

The normal command-response I2C protocol can still be used with streaming mode activated. Caution should be taken to make sure that a streaming mode response is not mistaken for reading a register and vice versa.

Note: Due to the large amount of data being sent at once with I2C streaming, some software implementations of the master device may require additional delay between starting the read transfer and getting all the data back. This is the case with many Python or C# implementations of the MCP2221 USB-I2C chip used on the Development motherboard. For the MCP2221 chip, if the host PC tries to read the data before the read transfer is complete (i.e. if no additional delay is implemented), an empty set of data or an error state may be returned instead.

5. COMMANDS

5.1.Pump enable

This register controls whether the pump is enabled or not and overrides all other register settings.

ID	R/W	Name	Values	Type
0	R/W	Pump enabled	0 = disabled 1 = enabled	int16_t

Example:

#W0,1\n Enables the pump

5.2.Power Limit

The maximum power to the pump under any circumstance is limited with this register value, which overrides all the control modes.

ID	R/W	Name	Values	Type
1	R/W	Power limit	0 to 1400 milliwatts	int16_t

Example:

#W1,1000\n Limits the pump input power to 1000mW

5.3.Stream mode

Enables/disables the streaming mode.

ID	R/W	Name	Values	Type
2	R/W	Enable stream mode	0 = disabled 1 = UART stream mode enabled 2* = I2C stream mode enabled	int16_t

* Smart Pump Module only

Example:

#W2,1\n Enables the streaming mode

5.4.Measurements

Several read only registers for measurements are provided.

ID	R/W	Name	Values	Type
3	R	Drive Voltage	0 to 60 volts	float
4	R	Drive Current	0 to 150 milliamps	float
5	R	Drive Power	0 to 2000 milliwatts	float
6	R	Drive Frequency	20000 to 23000 Hertz	int16_t
7	R	Analog A (dial on Evaluation Kit and Development Kit)	Value range dependent on channel gain and offset	float
8	R	Analog B (pr. sense on Evaluation Kit Evaluation Kit)	Value range dependent on channel gain and offset	float

9	R	Analog C (analog in on Evaluation Kit and Development Kit)	Value range dependent on channel gain and offset	float
32	R	Flow (optional Evaluation Kit and Development Kit flow sensor)	L/min, mL/min, uL/min, nL/min - – range dependent on register for flow measurement unit	float
39	R	Digital Pressure Sensor (for the Smart Pump Module and Development Kit)	mbar, mmHg, PSI, inHg, inH2O, cmH2O – range dependent on register for digital pressure measurement unit	float
41	R	Reserved for future use	Reserved for future use	float

Example:

#R3\n	Requests the current drive voltage
#R3,25.123\n	Response from the PCB, for a drive voltage of 25.123 volts

5.5.Control Mode

The PCB offers three control modes:

- Manual mode, where the drive power can be set directly.
- PID mode, where the output of a PID controller sets the pumps driving voltage. Can be used for closed loop control of a parameter, such as pressure.
- Bang Bang mode, where a measured value (e.g. pressure) is controlled between two limits by enabling/disabling the pump (sometimes called ‘hysteresis control’).

ID	R/W	Name	Values	Type
10	R/W	Control mode	0 = Manual 1 = PID 2 = Bang Bang	int16_t

Example:

#W10,1\n Set the PCB to use the PID control mode

5.6.Manual Mode Settings

One of four system inputs can be used to set the target drive power for the pump, in milliwatts.

ID	R/W	Name	Values	Type
11	R/W	Manual mode source	The source of the target power (in milliwatts) when in manual mode: 0 = Set val (register 23) 1 = Analog A† (dial on Evaluation Kit and Development Kit) 2 = Analog B† (pr. sense on Evaluation Kit) 3 = Analog C (main analog input)	int16_t

† Not available with Smart Pump Module

Example:

#W11,2\n Use the Analog 2 value (after gain and offset) as the target power

5.7.PID Mode Settings

Several registers are available to configure the PID controller. The PID controller controls the power used to drive the pump in milliwatts.

Example: pressure target is 200mBar and the actual target is 150mBar. If the proportional coefficient is 10, the output pump power would be (200-150) * 10 = 500mW and similarly for the other coefficients.

ID	R/W	Name	Values	Type
12	R/W	PID setpoint source	0 = Set val (register 23) 1 = Analog A† (dial on Evaluation Kit and Development Kit) 2 = Analog B† (pr. sense on Evaluation Kit) 3 = Analog C (main analog input)	int16_t
13	R/W	PID input source	0 = Set val (register 23) 1 = Analog A† (dial on Evaluation Kit) 2 = Analog B† (pr. sense on Evaluation Kit) 3 = Analog C (main analog input) 4 = External Flow Sensor* 5 = Digital pressure Sensor**	int16_t
14	R/W	PID proportional coeff.	Unbounded. Values within -2000 to 2000 are recommended	float
15	R/W	PID integral coeff.	Unbounded. Values within -100 to 100 are recommended	float

16	R/W	PID integral limit coeff.	The PID mode output controls the power used to drive the pump in milliwatts. Therefore, setting this value to the peak drive power the pump might use is recommended. Typically, this can be left at 1,400.	float
17	R/W	PID differential coeff.	Unbounded, but rarely useful in practise. Leaving this at 0 is recommended.	float
33	R/W	Reset PID on turn on	0 = No reset 1 = PID loop reset when pump enabled	int16_t

† Not available with Smart Pump Module

* Evaluation Kit and Development kit only

** Smart Pump Module and Development Kit only

Example:

#W12,0\n	Set the manual set value as the PID setpoint source
#W13,2\n	Use Analog B as the PID input source
#W14,100\n	Use a proportional coefficient of 100
#W15,10\n	Use an integral coefficient of 10

5.8.Bang Bang Mode Settings

Several registers are available to configure the Bang Bang controller.

ID	R/W	Name	Values	Type
18	R/W	Bang Bang input source	0 = Set val (register 23) 1 = Analog A† (dial on Evaluation Kit and Development Kit) 2 = Analog B† (pr. sense on Evaluation Kit) 3 = Analog C (main analog input) 4 = External Flow Sensor* 5 = Digital pressure Sensor**	int16_t
19	R/W	Bang Bang lower threshold	Unbounded	float
20	R/W	Bang Bang upper threshold	Unbounded	float
21	R/W	Bang Bang lower power mW	The drive power in milliwatts when the lower threshold is reached: 0 to 1400 milliwatts	float
22	R/W	Bang Bang upper power mW	The drive power in milliwatts when the upper threshold is reached: 0 to 1400 milliwatts	float

† Not available with Smart Pump Module

* Evaluation Kit and Development Kit only

** Smart Pump Module and Development Kit only

Example:

#W18,2\n	Use analog B as the input to the bang bang controller
#W19,10\n	Set the lower threshold to 10
#W20,100\n	Set the upper threshold to 100
#W21,1000\n	Set the power at the lower threshold to 1 watt
#W22,0\n	Turn the pump off when the upper threshold is reached

5.9.Measurement Settings

Several registers are available to configure the system inputs. Each input can be routed to a control mode via the registers for that mode.

The three analog inputs provide a raw value between 0 and 1. A gain and offset is applied to each analog input, before the value is routed to the areas where it is used. For example, a gain of 500, and offset of 250, could be applied to analog C, and then analog C used as the input to the manual mode. This would allow the analog C input to control the drive power between 250mW and 750mW.

ID	R/W	Name	Values	Type
23	R/W	Set Value	Unbounded. This value can be used as the input for the different modes.	float
24	R/W	Analog A offset†	Offset applied to the analog A input after gain is applied. -99,999 to 99,999	float
25	R/W	Analog A gain†	Gain applied to the raw analog A input, which is between 0 and 1. -99,999 to 99,999	float
26	R/W	Analog B offset†	Offset applied to the analog B input after gain is applied. -99,999 to 99,999	float
27	R/W	Analog B gain†	Gain applied to the raw analog B input, which is between 0 and 1. -99,999 to 99,999	float

28	R/W	Analog C offset	Offset applied to the analog C input after gain is applied. -99,999 to 99,999	float
29	R/W	Analog C gain	Gain applied to the raw analog C input, which is between 0 and 1. -99,999 to 99,999	float
40	R/W	Digital pressure offset**	-100 to 100 Unit dependent on Digital pressure measurement unit	float
58	R/W	Digital pressure measurement unit**	0 = mbar 1 = mmHg 2 = PSI 3 = kPa 4 = inHg 5 = inH20 6 = cmH20 Note that the digital pressure might need re-zeroing (through the Digital pressure offset register) after changing the measurement unit.	int16_t
59	R/W	Flow measurement unit*	0 = L / min 1 = mL / min 2 = uL / min 3 = nL / min	int16_t

† Not available with Smart Pump Module

* Evaluation Kit and Development Kit only

** Smart Pump Module and Development Kit only

Example:

#W23,500\n	Set the “Set Value” to 500
#W28,250\n	Set the analog C offset to 250
#W29,500\n	Set the analog C gain to 500

5.10. General Purpose Input Output (GPIO) pins

ID	R/W	Name	Values	Type
44	R/W	GPIO A pin mode***	2 = Output, slow mode, zero state is HIGH**** 3 = Output, slow mode, zero state is LOW**** 4 = Input for toggling pump, no pull-up or pull-down 5 = Input for toggling pump, pull-up 6 = Input for toggling pump, pull-down 7 = Disabled	int16_t
45	R/W	GPIO A state***	-1 to 250 If the GPIO is configured as input, this register reads the state of the pin (0 = LOW, 1 = HIGH). If the GPIO is configured as an output, this register sets the number of pulses that the pin generates. Setting the register to 0 will force the pin to the zero state. Setting the register to -1 will force the pin to the non-zero state.	int16_t
46	R/W	GPIO A pulse duration***	0 to 30000 Sets the pulse duration. In slow mode duration value of 1 corresponds to 1ms.	int16_t

47	R/W	GPIO A pulse period***	0 to 30000 Sets the pulse period. In slow mode duration value of 1 corresponds to 1ms.	int16_t
48	R/W	GPIO B pin mode***	0 = Output, fast mode, zero state HIGH 1 = Output, fast mode, zero state is LOW 2 = Output, slow mode, zero state is HIGH**** 3 = Output, slow mode, zero state is LOW**** 4 = Input for toggling pump, no pull-up or pull-down 5 = Input for toggling pump, pull-up 6 = Input for toggling pump, pull-down 7 = Disabled	int16_t
49	R/W	GPIO B state***	-1 to 250 If the GPIO is configured as input, this register reads the state of the pin (0 = LOW, 1 = HIGH). If the GPIO is configured as an output, this register sets the number of pulses that the pin generates. Setting the register to 0 will force the pin to the zero state. Setting the register to -1 will force the pin to the non-zero state.	int16_t

50	R/W	GPIO B pulse duration***	0 to 30000 Sets the pulse duration. In fast mode pin mode duration value of 1 corresponds to 10us (thus the pin can generate pulses between 10us and 300ms with resolution of 10us) In slow mode duration value of 1 corresponds to 1ms.	int16_t
51	R/W	GPIO B pulse period***	0 to 30000 Sets the pulse period. In fast mode pin mode duration value of 1 corresponds to 10us (thus the pin can generate pulses between 10us and 300ms with resolution of 10us) In slow mode duration value of 1 corresponds to 1ms.	int16_t
52	R/W	GPIO C pin mode***	2 = Output, slow mode, zero state is HIGH**** 3 = Output, slow mode, zero state is LOW**** 4 = Input for toggling pump, no pull-up or pull-down 5 = Input for toggling pump, pull-up 6 = Input for toggling pump, pull-down 7 = Disabled	int16_t
53	R/W	GPIO C state***	-1 to 250	int16_t

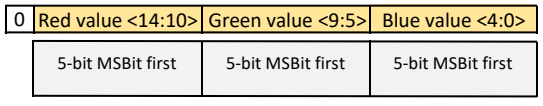
			If the GPIO is configured as input, this register reads the state of the pin (0 = LOW, 1 = HIGH). If the GPIO is configured as an output, this register sets the number of pulses that the pin generates. Setting the register to 0 will force the pin to the zero state. Setting the register to -1 will force the pin to the non-zero state.	
54	R/W	GPIO C pulse duration***	0 to 30000 Sets the pulse duration. In slow mode duration value of 1 corresponds to 1ms.	int16_t
55	R/W	GPIO C pulse period***	0 to 30000 Sets the pulse period. In slow mode duration value of 1 corresponds to 1ms.	int16_t
56	R	GPIO D state***	0 to 1 As the GPIO is configured as input with pull-up, this register reads the state of the pin (0 = LOW, 1 = HIGH).	int16_t

*** Development Kit only

**** Slow mode pulses are expected to be less accurate if UART streaming mode is enabled due to the extra CPU load that the streaming mode creates. Typical error with UART streaming mode is around ±1ms for 10ms long pulses and this error is expected to be independent of pulse length.

5.11. Miscellaneous Settings

ID	R/W	Name	Values	Type
30	R/W	Store current settings	Writing a 1 to this register, causes the current settings to be stored in flash. These are then retrieved when the board powers up. Allow about 1s for the settings to be stored before turning off the board. Reverts to 0 once settings are stored.	int16_t
31	R	Error code	0 = No error 1 = Error: short circuit 2 = Error: over frequency 3 = Error: under frequency	int16_t
34	R/W	Use frequency tracking	0 = Tracking off 1 = Tracking on	int16_t
35	R/W	Manual drive frequency	20000 to 23000 Hz Only used when Frequency tracking is off	int16_t
36	R	Major firmware version		int16_t
37	R	Firmware / Device type	1 = Fast Response Driver (obsolete) 2 = General Purpose Driver (synonymous with Cost Optimised) 3 = Smart Pump Module	int16_t

<table>
<tr><td>38</td><td>R</td><td>Minor firmware version</td><td></td><td>int16_t</td></tr>
<tr><td rowspan="2">42</td><td colspan="4">Warning
The Driver does not support communication with an I2C address in the range 0-7. If you set the I2C address in this range, you will no longer be able to communicate with your device. ***</td></tr>
<tr><td>R/W</td><td>I2C address * ***</td><td>8 to 127
Write a 1 to register 30 to store the new address. Takes effect after a power cycle.</td><td>int16_t</td></tr>
<tr><td rowspan="2">43</td><td colspan="4">Warning
By default, I2C/UART communications are autodetected. Changing register 43 will lock the protocol. Once the protocol is set, you may need to change your setup to communicate with the appropriate protocol. **</td></tr>
<tr><td>R/W</td><td>I2C / UART communication select ***</td><td>1849 = Autodetect I2C/ UART communication at start-up
1892 = UART only
1935 = I2C only
Write a 1 to register 30 to store the communication mode. Takes effect after a power cycle.</td><td>int16_t</td></tr>
<tr><td>57</td><td>R/W</td><td>Status LED colour**</td><td>5-5-5-bit RGB value
 

RGB value = 32 * 32 * R + 32 * G + B
where R, G and B are between 0 and 31</td><td>int16_t</td></tr>
</table>

* Smart Pump Module only

** Smart Pump Module and Development Kit only

*** If you have accidentally set the wrong value to this register and are not able to communicate with the board, please contact your Lee Sales Engineer.

Example:

#W34,0\n	Disable frequency tracking
#W35,21000\n	Drive at 21kHz

5.12. Default values

ID	Name	General purpose board with Evaluation Kit default values	General purpose board with Development Kit/ standalone default values	Smart Pump Module default values
0	Pump enabled	1 = enabled	1 = enabled	1 = enabled
1	Power limit	1000 milliwatts	1000 milliwatts	1000 milliwatts
2	Enable stream mode	0 = disabled	0 = disabled	0 = disabled
10	Control mode	0 = manual	0 = manual	0 = manual
11	Manual mode source	1 = Analog A (dial on Evaluation Kit and Development Kit)	1 = Analog A (dial on Evaluation Kit and Development Kit)	3 = Analog C (main analog input)
12	PID setpoint source	1 = Analog A (dial on Evaluation Kit and Development Kit)	1 = Analog A (dial on Evaluation Kit and Development Kit)	3 = Analog C (main analog input)
13	PID input source	2 = Analog B (pr. sense on Evaluation Kit)	5 = Digital pressure Sensor	5 = Digital pressure Sensor
14	PID proportional coeff.	5	5	5
15	PID integral coeff.	10	10	10
16	PID integral limit coeff.	1400	1400	1400
17	PID differential coeff.	0	0	0

33	Reset PID on turn on	1 = PID loop reset when pump enabled	1 = PID loop reset when pump enabled	1 = PID loop reset when pump enabled
18	Bang Bang input source	2 = Analog B (pr. sense on Evaluation Kit)	5 = Digital pressure Sensor	5 = Digital pressure Sensor
19	Bang Bang lower threshold	10	10	10
20	Bang Bang upper threshold	50	50	50
21	Bang Bang lower power mW	1000 milliwatts	1000 milliwatts	1000 milliwatts
22	Bang Bang upper power mW	0 milliwatts	0 milliwatts	0 milliwatts
23	Set Value	250	250	250
24	Analog A Offset	0	0	N/A
25	Analog A Gain	1000	- 1000 for standalone GP driver - Dependent on factory calibration for GP driver with Development kit.	N/A
26	Analog B Offset	Dependent on factory calibration.	-821	N/A
27	Analog B Gain	Dependent on factory calibration.	2130	N/A
28	Analog C Offset	0	0	0

29	Analog C Gain	1000	1000	1000
40	Digital Pressure Offset	N/A	Dependent on factory calibration.	Dependent on factory calibration.
34	Use frequency tracking	1 = Tracking on	1 = Tracking on	1 = Tracking on
42	I2C address	N/A	N/A	37
43	I2C / UART communication select	N/A	N/A	1849 = Autodetect I2C/ UART communication at start-up
44	GPIO A pin mode	N/A	5 = Input for toggling pump, pull-up	N/A
45	GPIO A state	N/A	Depending on GPIO state (pin input).	N/A
46	GPIO A pulse duration	N/A	0	N/A
47	GPIO A pulse period	N/A	0	N/A
48	GPIO B pin mode	N/A	1 = Output, fast mode, zero state is LOW	N/A
49	GPIO B state	N/A	0	N/A
50	GPIO B pulse duration	N/A	0	N/A

51	GPIO B pulse period	N/A	0	N/A
52	GPIO C pin mode	N/A	3 = Output, slow mode, zero state is LOW	N/A
53	GPIO C state	N/A	0	N/A
54	GPIO C pulse duration	N/A	0	N/A
55	GPIO C pulse period	N/A	0	N/A
57	Status LED colour	N/A	0b000001111100000 = Green	0b000001111100000 = Green
58	Digital pressure measurement unit	N/A	0 = mbar	0 = mbar
59	Flow measurement unit	N/A	1 = mL / min	N/A

6. FURTHER SUPPORT

6.1.Code Snippet Library

The Lee Company code snippet library, hosted on GitHub (https://github.com/The-lee-company), provides serial communication and control examples in Python for common functions, including turning the pump on and off, setting drive power, closed loop control of pressure and reading back and plotting data. The code snippet library implements the aspects of the communication protocol set out in this Application Note and is intended to support customers after their initial evaluation of our pump technology, as they move on to developing prototypes and products.

6.2.Additional Support

The Lee Company website provides advice on:

- Getting Started
- Applications
- Development Process
- Downloads (including datasheets, application notes, case studies and 3D models)
- Reliability testing

The Lee Company is happy to discuss next steps beyond prototyping, including system design. If you would like to discuss this with us, or for any other additional support, please contact your Lee Sales Engineer.

THE LEE COMPANY

7. REVISION HISTORY

Date	Revision	Change
Jan 2026	R260112	Added warning to register 42 (I2C address) to prevent setting the I2C address to values 0-7. Updated the RGB register comment to reflect that there are only 5 bits per colour.
Feb 2024	R230912	Added registers 44 to 59. Added Development kit section. Added I2C stream mode section. These changes were made with the release of General purpose driver firmware version 15.11 and Smart Pump Module firmware version 6.16 Updated I2C Output level high to be 5V instead of empty to signify the maximum operating voltage of the I2C bus. Also added a note that UART bus is 3.3V typical and 5V tolerant.
June 2023	R230621	Rebranded – name changes & link updates.
30th Jan 2023	r230130	Added details about register 43. This register is active from SPM Firmware version 5.6. Changed maximum I2C frequency from 120kHz to 400kHz and minimum frequency from 10 to 100kHz Added range of values for registers 21, 22, 24-29 and 40. Changed register 16 PID integral limit coeff. Description to reflect how the register is really implemented. Updated the description of register 41.
1st Dec 2022	r221201	Change pull-up resistance on SDA & SCL from 50 to 10kOhms
19th Aug 2022	r220819	SUM % 255 changed to SUM % 256

13 May 2022	r130522	Added note that the I2C master must support clock stretching.
05 May 2022	r050522	Correct 5.1. Added details about register 42
21 March 2022	r220321	Remove FR driver physical board
04 March 2022	r220304	Updates to include I2C and UART protocols and Smart Pump Module.
03 August 2021	r210803	Update to TN and new document format.
23 April 2021	r210423	Add setting 4 to Register 13 and Code Snippet Library to Further Support.
19 June 2020	r200619	Corrected typing errors.
28 May 2020	r200528	Added <FLOW> to streamed values in 3.4.
29 January 2020	r200129	Correct wrong USB receptable specification in 3.0.
28 May 2019	r190528	Reissue as AN003.
31 January 2019	r190131	Updated the documentation to match the new evaluation kit commands.
28 September 2018	r180928	Initial revision.