Technical Note TN003: Serial Communications Guide

1 DISCLAIMER

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

2 INTRODUCTION

2.1 Disc Pump

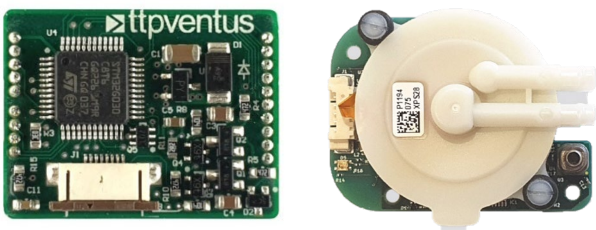
Disc Pump is a silent, high-performance piezoelectric micropump.

Owing to its operating mechanism, Disc Pump 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. XP Series Disc Pumps

2.2 Driver Communication

LEE Ventus provide a range of PCB designs for driving Disc Pump. All designs share a common register-based control interface, which is accessed either via a UART or I2C protocol, depending on the driver. This Application Note provides details of the commands recognised by the drivers.



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

Figure 2: Disc Pump drive systems and communication protocol support

2.3 Evaluation Kit Motherboard

Typically, customers' first use of Disc Pump is with the LEE Ventus Disc Pump Evaluation Kit. 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.

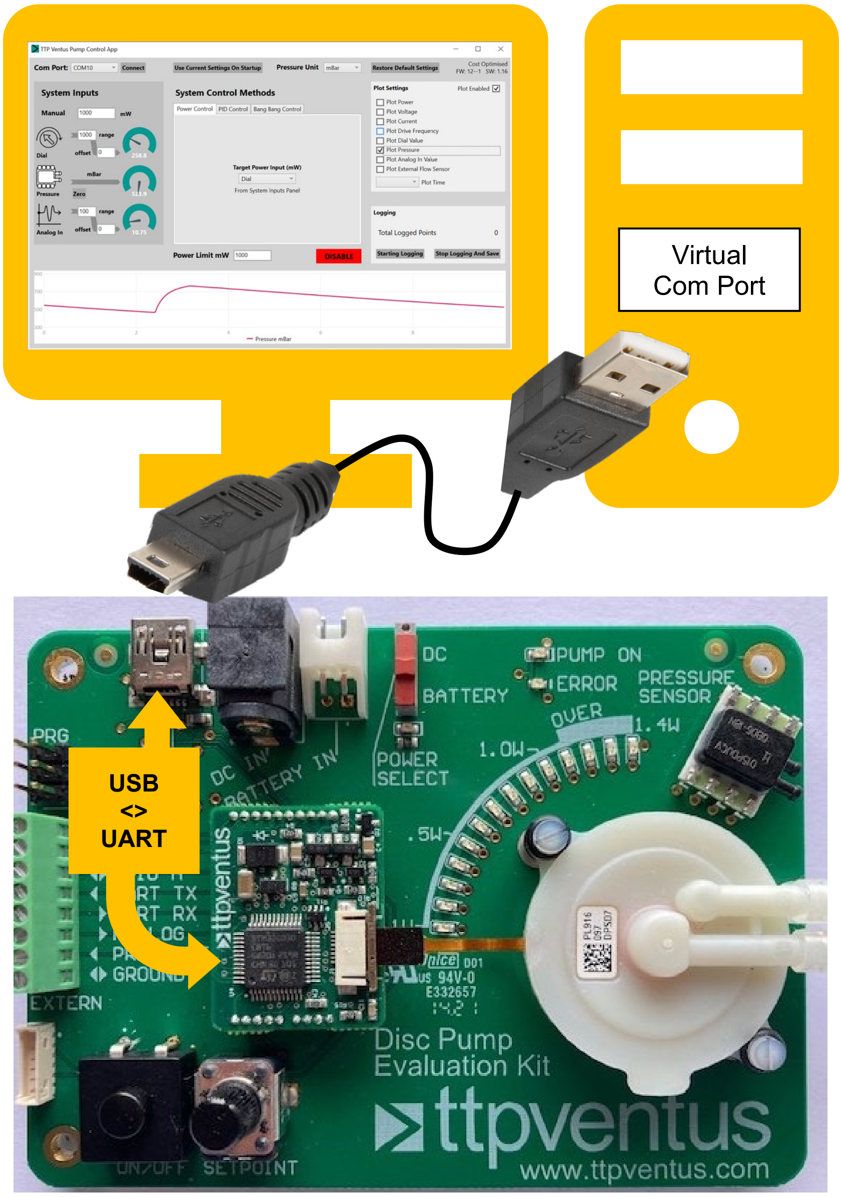


Figure 3: Connecting the Evaluation Kit Motherboard to a host PC

2.3.1 Using the Evaluation Kit Virtual COM Port

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

2.3.2 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.

2.3.3 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 Units

The following units are used:

Quantity	Unit
Flow	mL/min
Voltage	V
Current	mA
Power	mW
Frequency	Hz
Pressure	For the evaluation kit and drive PCBs, the following pressure units can be selected in the LEE Ventus Pump Control App: mBar (default unit) mmHg PSI kPa inHg inH20 For the Smart Pump Module, the pressure unit is mBar.

	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. For the older Fast Response PCB (obsolete) the UART operates at 2.5V.

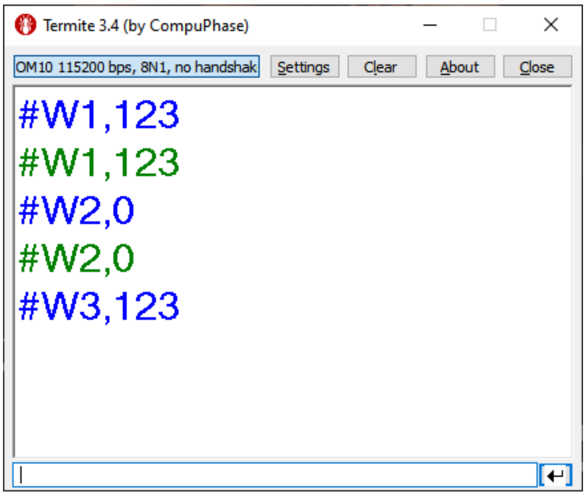
3.1.1 Writing

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

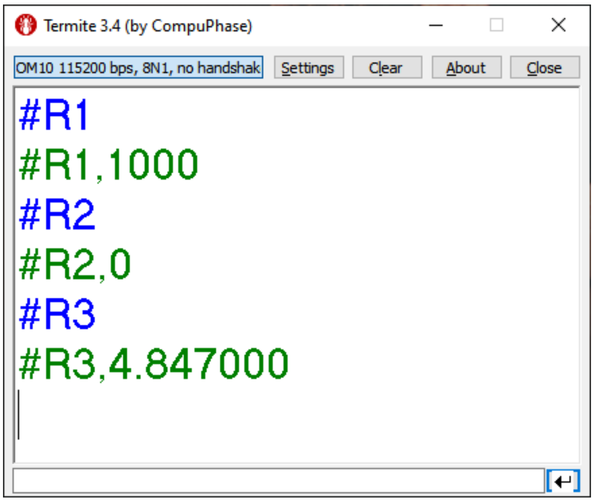
3.1.2 Reading

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

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 with register 2. 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.

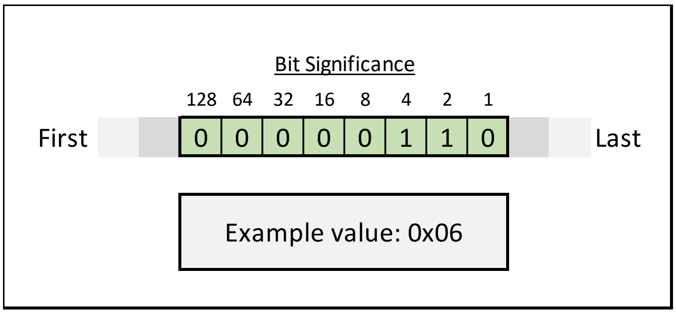
4 I2C INTERFACE (FOR SPM ONLY)

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.

Specification	Min	Typical	Max	Unit
I2C bus speed	10	100	120	kHz
Output level low			0.9	V
Output level high	2.4			V
Pull-up resistance on SDA & SCL	0.2		50	kOhm

I2C Speed and Level Parameters for Ventus Drivers / Modules



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

Driver registers have two types:

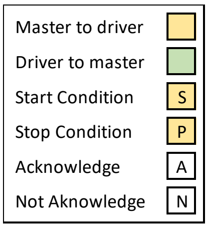
int16_t	A signed, 16-bit integer
float	A 32-bit floating point value (typical IEEE 754 format)

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

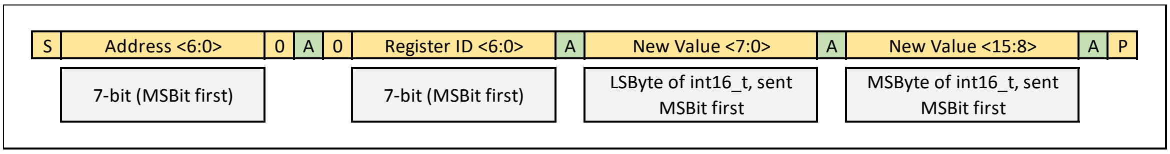
The default device address is 37.



4.3 Writing to a 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 zero
- 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 zero)
- Sends the int16_t value as two bytes, with the least significant byte first

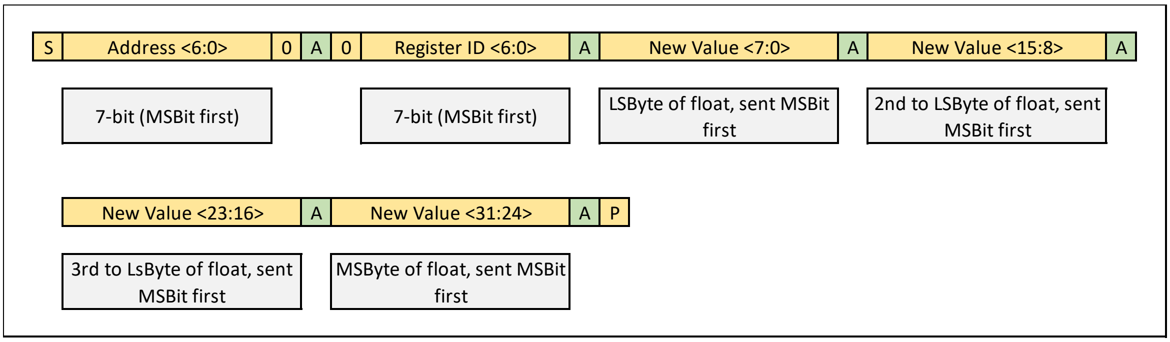


Writing to an int16_t Register

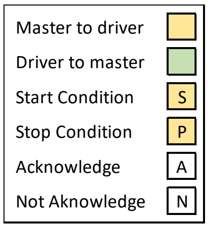
4.4 Writing to a register with the float type

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



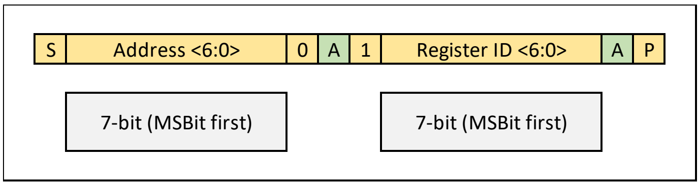
Writing to a float Register



4.5 Reading a register with the int16_t type

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

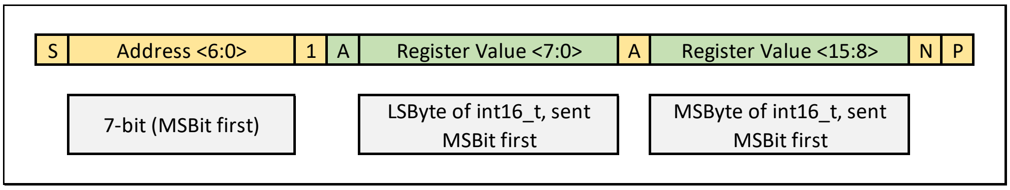
- Initiates a write transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to zero
- Sends a further byte to the driver, where the 7 LSBits indicate the register ID to be written, and the MSBit indicates a register read (set to one)



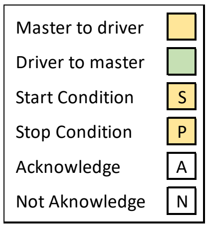
Master selects a register to read

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



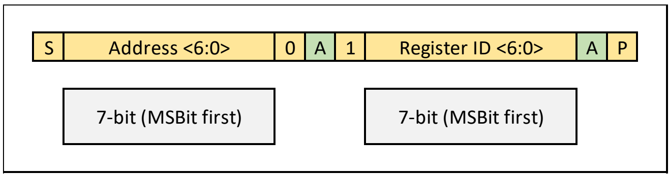
Master reads back the two bytes from the selected int16_t register



4.6 Reading a register with the float type

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

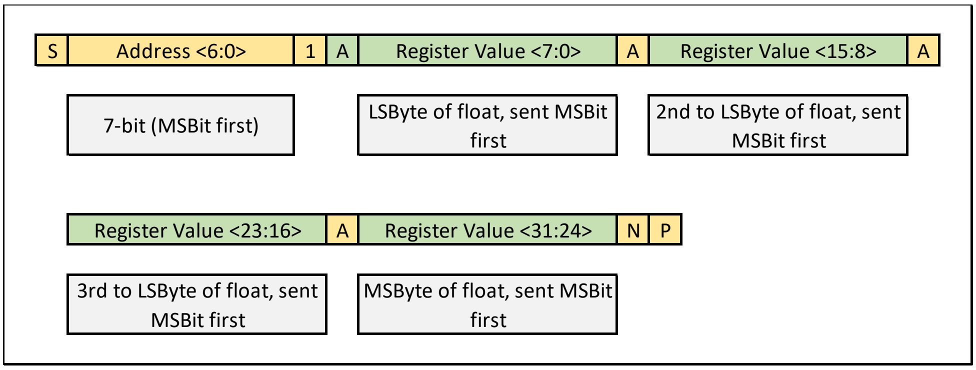
- Initiates a write transfer with the driver, by sending the driver's I2C address, followed by the read/write flag, which is set to zero
- Sends a further byte to the driver, where the 7 LSBits indicate the register ID to be written, and the MSBit indicates a register read (set to one)



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 one
- 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

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=enabled	int16_t

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 1 (dial on eval kit)	Value range dependent on channel gain and offset	float
8	R	Analog 2 (pressure on eval kit)	Value range dependent on channel gain and offset	float
9	R	Analog 3 (analog in on eval kit)	Value range dependent on channel gain and offset	float
32	R	Flow (optional eval kit flow sensor)	mL/min - range dependent on the connected sensor.	float
39	R	Digital Pressure Sensor (for the Single Pump Module)	mBar	float
41	R	Drive Phase (reserved for future use)	-180 to 180 degrees	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 1† (dial on evak kit) 2=Analog 2† (pr. sense on eval kit) 3=Analog 3 (main analog input)	int16_t

† Not available with Smart Pump Module

Examples:

#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.

ID	R/W	Name	Values	Type
12	R/W	PID setpoint source	0=Set val (register 23) 1=Analog 1† (dial on evak kit) 2=Analog 2† (pr. sense on eval kit) 3=Analog 3 (main analog input)	int16_t
13	R/W	PID input source	0=Set val (register 23) 1=Analog 1† (dial on evak kit) 2=Analog 2† (pr. sense on eval kit) 3=Analog 3 (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 voltage used to drive the pump in millivolts. Therefore, setting this value to the peak drive voltage the pump might use is recommended. Typically, this can be left at 55,000.	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 only
** Smart Pump Module only

Example:

```
#W12,0\n      Set the manual set value as the PID setpoint source
#W13,2\n      Use Analog 2 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 1† (dial on evak kit) 2=Analog 2† (pr. sense on eval kit) 3=Analog 3 (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	float
22	R/W	Bang Bang upper power mW	The drive power in milliwatts when the upper threshold is reached	float

† Not available with Smart Pump Module
* Evaluation Kit only
** Smart Pump Module only

Example:

#W18,2\n	Use analog 2 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 3, and then analog 3 used as the input to the manual mode. This would allow the analog 3 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 1 Offset	Offset applied to the analog 1 input after gain is applied.	float
25	R/W	Analog 1 Gain	Gain applied to the raw analog 1 input, which is between 0 and 1.	float
26	R/W	Analog 2 Offset	Offset applied to the analog 2 input after gain is applied.	float
27	R/W	Analog 2 Gain	Gain applied to the raw analog 2 input, which is between 0 and 1.	float
28	R/W	Analog 3 Offset	Offset applied to the analog 3 input after gain is applied.	float
29	R/W	Analog 3 Gain	Gain applied to the raw analog 3 input, which is between 0 and 1.	float
40	R/W	Digital Pressure Offset (for Single Pump Module)	mBar	float

Example:

#W23,500\n	Set the “Set Value” to 500
#W28,250\n	Set the analog 3 offset to 250
#W29,500\n	Set the analog 3 gain to 500

5.10 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. 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 (when register 34 is set to 0)	20000 to 23000 Hz	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) Single Pump Module 4) Soft Driver	int16_t
38	R	Minor firmware version		int16_t
42	R/W	I2C address	0 to 127 Write a 1 to register 30 to store the new address. Takes affect after a power cycle	int16_t

Example:

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

ttpventus
PART OF THE LEE COMPANY

6 FURTHER SUPPORT

6.1 Code snippet library

The LEE Ventus code snippet library, hosted on GitHub (https://github.com/TTP-Ventus/), 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 support section of LEE Ventus website (https://www.ttpventus.com/support) provides advice on:

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

LEE Ventus 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 us at support@ttpventus.com.

7 REVISION HISTORY

Date	Revision	Change
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.