LadyBug Technologies, LLC
9290 W. Barnes Dr.
Boise, ID 83709
World Wide: 707-546-1050 USA: 866-789-7111
www.ladybug-tech.com

LB5900 Series Power Sensor SPI & I2C Interface Guide

TABLE OF CONTENTS

LIST OF FIGURES AND DIAGRAMS

Important Notice:

LadyBug LB5900 Devices operating in SPI and I2C modes differ in several ways from traditional SPI & I2C devices. It is important to understand some of these differences to have a successful implementation of the system.

Due to familiarity to memory and other chip based devices, the user may expect the response time to be the same after each request. The LB5900 Sensor's SPI & I2C engine is dependent on the sensor's primary processor. The time required for each request will vary significantly with the request. Further, the time required to make measurements may depend upon the power level and other factors.

Sensor Firmware updates. When the sensor is updated, the time to process commands may change. Additional features may cause certain commands to require more time for completion, this includes existing commands that may appear to be unrelated.

Adherence to the following list of recommendations will assure success.

- ● Connect Pin 1, SPI/I2C Select, directly to power or ground. Do not connect it to a control line. This line is tested at power up only; and any startup glitches on the controller may cause erroneous selection that can only be cleared by power down.

- Prior to sending commands or queries, test the sensor to make sure it is ready.

- Adhere to the timing recommendations and do not clock data faster than the recommended rates, even though the sensor may perform well at faster rates and reduced setup times. This will assure that future firmware versions will operate properly.

- Do not test the sensor to determine if it is ready for a command or query faster than the recommended minimum time (1ms). Bear in mind that each request interrupts the processor to fulfil the request.

- Avoid unnecessary requests after sending measurement commands. For example if a measurement is expected to take 100ms, request readiness after 90ms, or at 5 or 10ms rather than 1ms.

- If a measurement is in process, do not send the sensor anything other than a test to see if the measurement is complete or to send ●a DCL command.

- Read the REVISIONS & UPDATES Section each time this manual is updated. Important changes are marked with ● or ●

GENERAL

When using Power Sensors in ATE (Automated Test Equipment) systems, it is often desirable to employ a direct connection to a microcontroller or other processor. Using USB or other driver intense connectivity platform can interfere with timing and create startup issues.

LadyBug LB5900 series sensors offer option SPI; the option enables sensor operation through SPI (Serial Peripheral Interface) or I2C (Inter-Integrated Circuit).

Each interface technology has its own advantages, LadyBug recommends that the user research these options prior to selecting the interface. LB5900 sensors act as slave devices in both interface technologies.

Data can be collected rapidly using either interface, please review the sensor specification sheet for measurement rates. Maximum sensor clock rates for SPI is 1 MHz; I2C is 400 kHz.

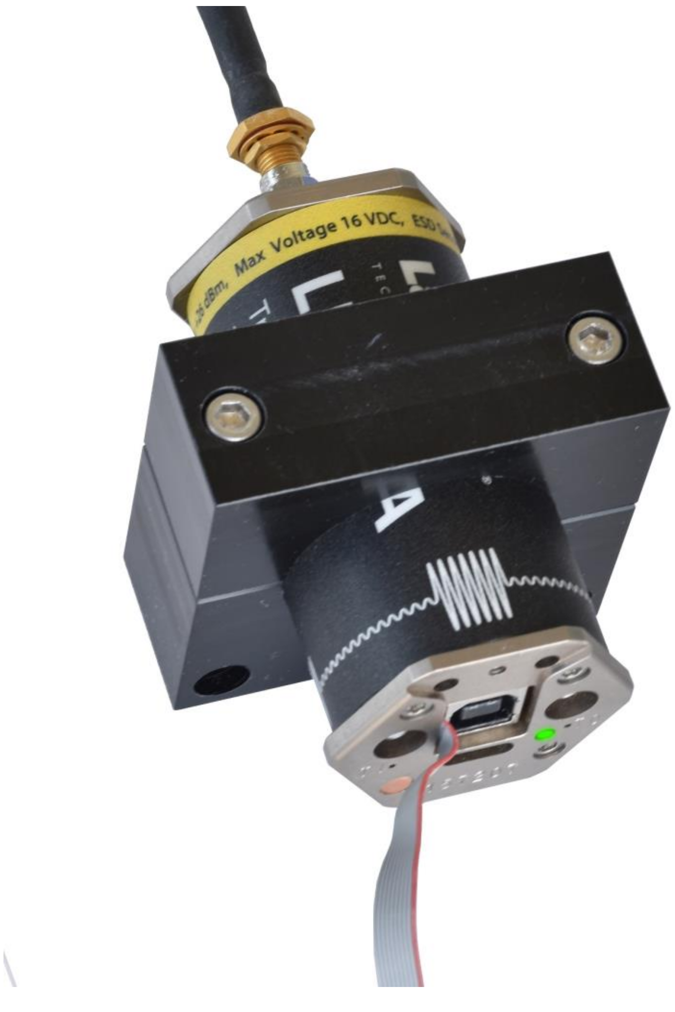


Figure 1 – Sensor with Option SPI, shown with LB956A Mounting Bracket (not included)

Sensor Power

The sensor can be powered using the SPI/I2C interface cable, method is recommended when using the interface; however power can be applied using a USB power only cable. To select SPI/I2C as the power source, connect Pin 4 (Select SPI/I2C Power) to Pin 6 (Power). Pin 4 may also be connected as a control line and will function at the same voltage levels as the data lines.

Unless it is indicated otherwise in the specific sensor data sheet, the recommended supply voltage is 4.65 to 5.35 volts. Maximum supply voltage is 5.5 volts, damage level is 5.95 volts. Additional specifications can be found in the individual sensor's data sheet.

Data Line Electrical Specifications

The data, clock and control lines on LB5900 sensors use 3.3 volt logic levels, however as inputs, they are 5 volt tolerant. This means that the sensor can also be used with control systems running at 5 volts if the controller inputs are compatible with 3.3 volt logic levels. Other specific compatibilities include LVCMOS 3.0V to 3.6V; TTL.

Commands, Data Transmission & Reception

Messages are transmitted to and received from the sensor using standard textual SCPI instructions and queries. Please refer to the *Programming Guide For LadyBug True RMS Generation II Sensors* for command details. Data is transmitted and received in full bytes (8-bits) using standard ASCII codes; however header information described later may be in binary form.

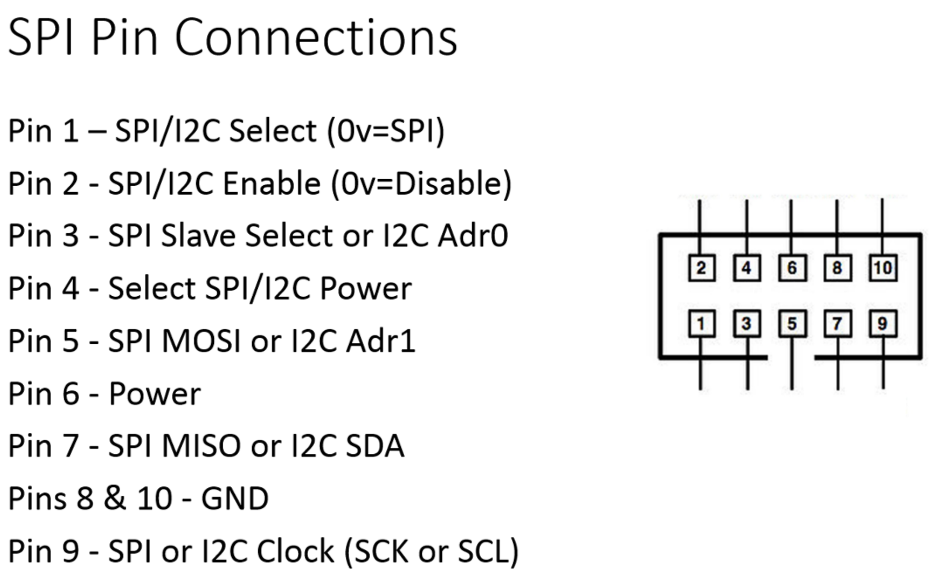
In general, SCPI commands sent USB, I2C or SPI are identical. There are however some exclusions and limitations. The SPI and I2C interfaces do not accept concatenated commands. The SPI and I2C interfaces limit the length of strings to 4095 bytes, this places limits on certain commands requiring large amounts of data such as reading stored unattended memory (Sensors with Option UOP).

By IVI- SCPI definition, data length (number of bytes) can vary on messages returned from the sensor. To accommodate this specification and provide status and command information, a 4-byte binary header is sent at the beginning of messages. Messages may have additional header information (in full byte multiples) to provide information required for I2C or SPI purposes. Please refer to the header information in the SPI or I2C sections.

Sensor Reset and Communication Port Reset

● While in the I2C mode, the communication port can be reset by pulsing the SPI/I2C enable line (Pin2). This connection must be active (high) when power is applied to use I2C or SPI. Pulsing the line low for 100ms or longer, then returning it high, will cause the I2C module to be cleared and reset. This Pin should be tied to Sensor power in SPI mode.

In SPI Mode, while the clock line remains in the inactive state, pulsing the Slave Select line low twice for 1ms resets the Sensor's SPI module and issues a sensor DCL command. It is recommended that this be done at power up to eliminate erroneous data. The module can also be reset by holding the Slave Select line low for greater than one second while the clock line remains in the inactive state.



SPI Pin Connections

Pin 1 – SPI/I2C Select (0v=SPI)
Pin 2 - SPI/I2C Enable (0v=Disable)
Pin 3 - SPI Slave Select or I2C Adr0
Pin 4 - Select SPI/I2C Power
Pin 5 - SPI MOSI or I2C Adr1
Pin 6 - Power
Pin 7 - SPI MISO or I2C SDA
Pins 8 & 10 - GND
Pin 9 - SPI or I2C Clock (SCK or SCL)

Figure 2 User Socket Connector Pin Out

The sensor's cable is installed at the factory and is not removable or replaceable by the user. The cable extends 8" outside of the case, contact LadyBug for other options. The 10 wire ribbon cable exits the housing adjacent to the USB connector. With option SPI, a standard USB cable can also be connected and the sensor will function as a USB sensor when SPI is not enabled.

The SPI/I2C connector end is a Samtec FFSD type (or equivalent) keyed 0.05" pitch two row connector. Recommend 0.05" pitch IDC mating connector:

Samtec ESHF-105-01-L-D-TH (Vertical through hole);
Samtec ESHF-105-01-L-D-SM (Vertical surface mount);
Samtec ESHF-105-01-L-D-RA (Right Angle);

www.samtec.com

Other equivalent connectors are available. Note that the sensor's cable connector has a strain relief and requires a socket accordingly.

Electrical Data Connection and Addressing

Enabling the SPI/I2C interface is controlled by Pin 2 (SPI/I2C Enable). Applying power to Pin 2 (from pin 6) disables USB connectivity and enables the SPI/I2C interface. Choosing SPI or I2C for a LB5900 sensor is done using Pin 1 of the interface cable. Applying power (from Pin 6), selects I2C and disables SPI, grounding the pin will enable SPI and disable I2C.

When using I2C, external pull-up resistors are required, refer to Figure 2. LadyBug's nominal recommended value is 2.2K ohms. The pull-up resistors should be connected to the microcontroller's logic power supply (3.3-5.0 volts) near the microcontroller. High speed applications may benefit from a capacitor to ground on I2C data and clock lines. Recommended starting value is 27pf. The resistor and capacitor values can be optimized to reduce overshoot.

Up to four sensors may be addressed on a single I2C connection. Each sensor's address is set by hardware. When I2C is selected the device address is as shown in Table 1.

Address	Connector Pin 5 SPI MOSI or I2C Adr1	Connector Pin 3 SPI SS or I2C Adr0
1001100	0	0
1001101	0	1
1001110	1	0
1001111	1	1

Figure 3 - I2C Device Address

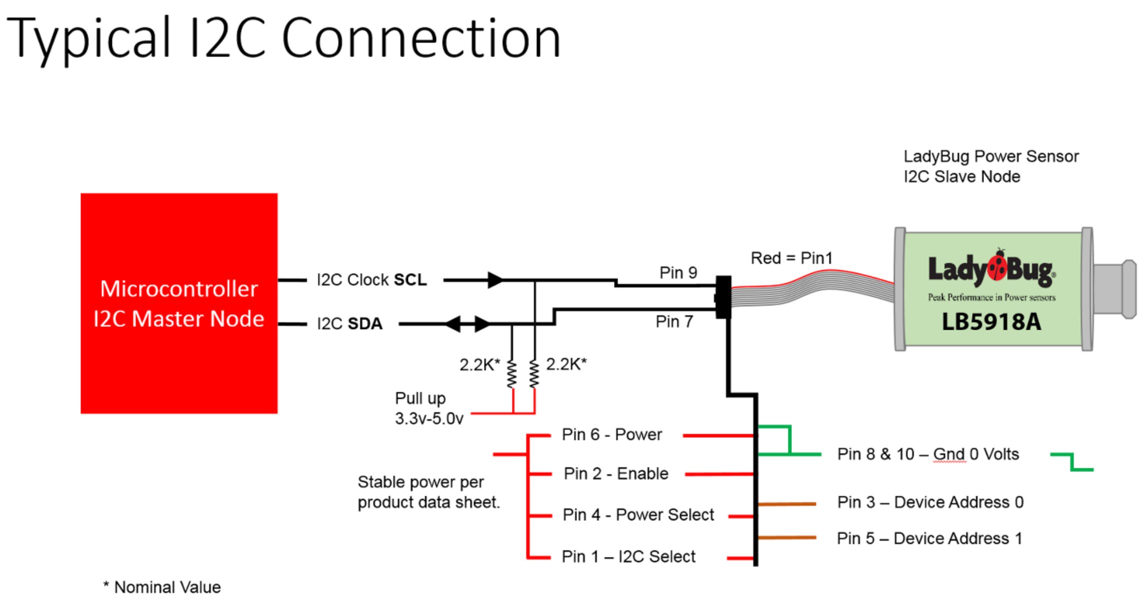


Figure 4 - Typical I2C Connection

I2C Message Overview

Commands with no required response can be sent with a single communication if the sensor is not busy. Queries require at least two communications, one to start a function, such as a measurement, then a second to read back the resultant data. After the request is made to the sensor, the master (user controller) is required to allow sufficient time for the sensor to process the function prior to requesting data. During the time the sensor is processing data, the master can address other sensors. For example, a routine could be created to start measurements on two sensors, then go back and read each one after a period of time has elapsed. NACK, as explained later in this document, can be used to determine if the sensor has completed the function and is no longer busy.

● Once a command is received, **Processing** begins when the *stop condition,* detailed below, is sent by the master (user controller). For example, when issuing a READ?, the process of collecting the averaged measurement begins when the *stop condition* is issued by the master (user controller).

Processing Time Delay

● In order for the sensor to process data and to prepare additional data, a minimum of 1ms is required after any command is sent. The master should not send any requests during this time.

Clock, Start, Stop, Direction

The ***Clock*** (SCL) is always generated by the master (user controller). All messages begin with a *start condition* and are terminated by a *stop condition*. Start and stop conditions are always generated by the master (user controller). After a *start condition* is issued, the bus is considered to be busy and cannot be used for any other transactions until a *stop condition* occurs. A period of time (t_{buff}) is required after the *stop condition*. Data bytes are transferred with the MSB (Most Significant Bit) first out. Each byte must be *acknowledged* by the receiver.

When the master is transmitting data, it sets ***direction*** ***(**$R/\overline{W}$**)*** to zero, when requesting data from a sensor, the master sets the bit to a one.

A ***start condition*** occurs when the master lowers the SDA line while the SCL line is high.

A ***stop condition*** occurs when the master raises the SDA line while the SCL line is high.

Acknowledge & NACK

I2C is a based on standard 8 bit (byte) communications, however there is an extra 9’th (ACK/NACK) bit that is used to acknowledge transfers and is not part of the data being sent. So that the slave (sensor) can acknowledge receipt of transmissions, the master (user controller) releases the SDA line after the 8’th bit of each byte. This allows the sensor to control the line and place it in a low state to acknowledge the transfer on the 9’th clock cycle. If the line remains high during the acknowledge cycle, the sensor is sending a *NACK* (Not Acknowledge) indicating that an error in the message has occurred or that it is not ready. **NOTE: *NACK* is not used to identify SCPI command errors.** The status byte should be read to check for the presence errors. The master (user controller) must manage this condition.

NACK issued by the master (user controller), (master NACK) when receiving data from the sensor may be used to indicate that the master does not require further data and considers the transaction complete, a *stop condition* follows the *NACK* and completes the transaction. This resets the sensor's I2C system preparing the sensor for a new I2C request.

Sensor Busy & NACK

The sensor may not be able to answer I2C requests when calculating measurements, processing non-volatile memory and possibly other functions. If requests to the sensor are made when it is busy, the input buffer will overflow and an error will occur. To prevent this potential error, the sensor uses NACK functionality. NACK is generated automatically when a request is made to the sensor and it busy and can not answer. Prior to making requests to the sensor, it is recommended that the master (user controller) use the sensors NACK feature to determine if the sensor is busy. If the sensor is addressed and responds with ACK, it is ready for a request; if NACK is returned, the master should wait. A loop can be setup and continuous tests made until ACK is returned.

I2C Timing

I2C timing conditions are shown below. Timing listed in the table are preliminary.

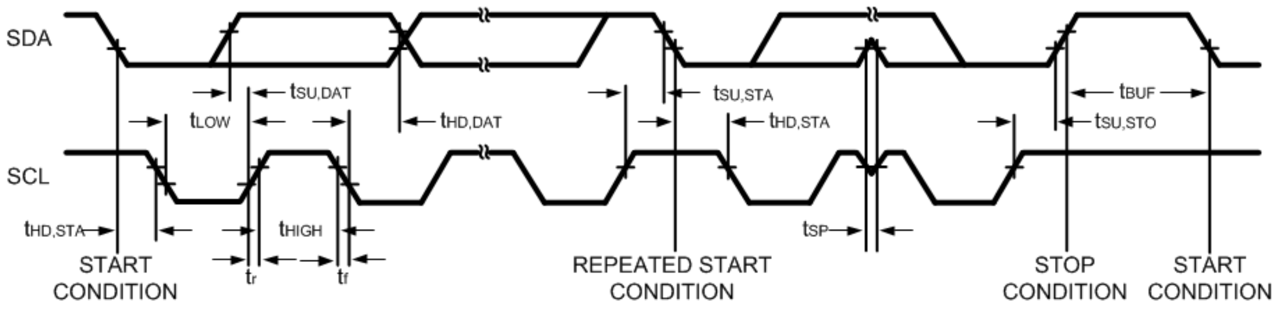


Figure 5 - Timing Diagram

Symbol	Parameter	Value
f_{SCL}	Clock frequency (max)	100kHz
t_{BUFF}	Stop bit to buss free time	5µs
$t_{HD.STA}$	Hold time after start condition	5µs
$t_{SU.DAT}$	Data setup time	5 µs
t_{LOW}	Low Ppriod of the clock (min)	10µs
t_{HIGH}	High period of the clock (min)	10µs
$t_{HD.DAT}$	Data hold time	5µs
$t_{SU.STA}$	Setup time for repeated start condition	5µs
$t_{HD.STA}$	Hold time for repeated clock	5µs
t_r	Rise time (max)	1µs
t_f	Fall time	300ns
t_{SP}	Pulse width of spikes that must be suppressed	5µs
$t_{SU.STO}$	Setup time for stop condition	5µs

Figure 6 - Timing Values & Recommended Conditions

In addition to I2C timing specifications, the sensor requires time to process instructions. To accommodate this, the user must design in the required time. These are discussed later in the document.

Data Format and Communications

LadyBug sensors employ a standard I2C 7-bit addressing scheme with bits 0-6 as address and bit-7, the eighth bit as a direction bit *(R/$\overline{W}$)*. All messages begin by the user (master) issuing a standard I2C *start condition* and terminate with a standard I2C *stop condition*. Each byte must be *acknowledged* by the receiver of the data. The figure below details an I2C message with address information followed by a byte of data, which would normally contain header or instruction information as detailed later in this document.

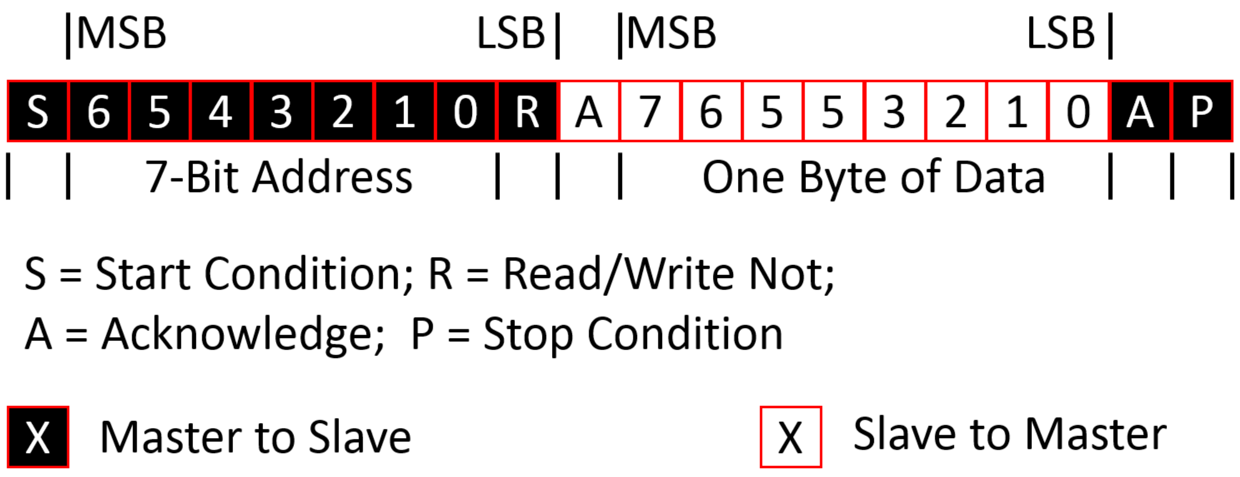
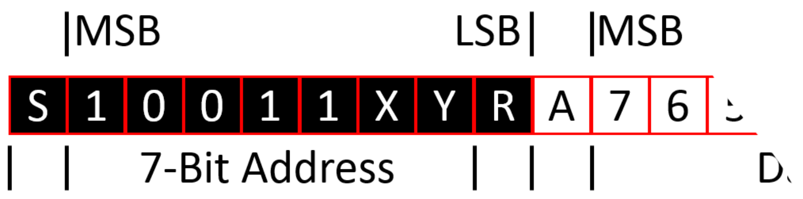


Figure 7 - I2C Address & Data

The Address byte

All instructions and queries from the user controller begin with an address byte which determines the sensor to be accessed. The address byte is the first byte sent after a *start condition* is issued and is read by all connected sensors. Message content after the address is ignored by sensors without matching addresses. LadyBug utilizes I2C standard 7-bit addressing with bits 7-3 set as '10011' followed by two bits for sensor addressing then the direction bit as detailed below. Like all transferred bytes to the sensor, the address byte will be acknowledged.



S = Start Condition;
R = Read/Write Not;
A = Acknowledge;
XY = Sensor Address (0-3) X=Adr1, Y= Adr2.
LadyBug 7-bit addresses begin with 10011

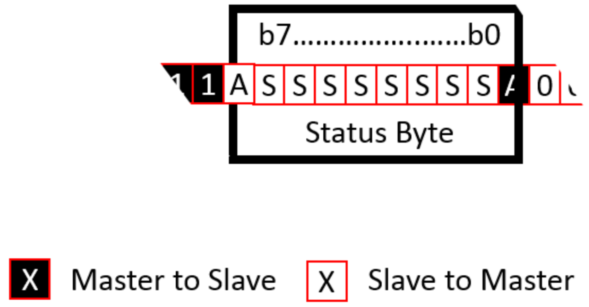
■ X Master to Slave   □ X Slave to Master

Figure 8 - Sensor Addressing

For hardware selection of Adr1 & Adr2, review the *I2C Device Address* table in the *Electrical Connection and Addressing* section. A Command Header and three data length bytes, follow the address byte on outbound communications.

The Status byte

●The status byte is used to determine if information such as a measurement is available/ready and possibly for other purposes. The status byte is returned as indicated in the command details. The primary bit used is bit#4 which indicates that a message is available. The message can be the measurement, an error message, or other information.



Status byte.
Bit 0 - Not used
Bit 1 - Not used
Bit 2 - See Programming Manual
Bit 3 - See Programming Manual
Bit 4 - **Message available (used in I2C)**
Bit 5 - Event Status bit
Bit 6 - Not used
Bit 7 - See Programming Manual

Refer to *STB in programming guide for additional details.

Figure 9 - Status Byte Details

It is important to note that the specific bit should be tested, do not test the entire word against an expected response because additional bits may be set or cleared depending upon conditions. Further firmware updates may enable the use of bits that are currently not in use.

Command Header

●Messages sent to the sensor contain a binary-formatted header containing four bytes that are sent immediately following the address byte. The first byte (Command Header) indicates how the message and any following I2C reads will be handled. For example, if the master (user controller) needs to read sensor status, the Command Header value of 06h should be written to the sensor; the following read will return the status plus the length of any available data. Command Header values are detailed in the table below. After the communication, a *stop condition* initiates the command and causes the sensor to prepare as indicated. The next command can read back the data, status, or execute the indicated command.

Command Header Description	**Header Value**
●Send (write to sensor) Command plus Prepare Status & Length for the next read back. Sends the SCPI command that follows the header and data length and causes the sensor to prepare the status byte and the number of bytes that are ready to be returned. If no SCPI command is included, causes the sensor to prepare the status byte & number of bytes that are ready to be returned by the sensor. **Direction *(address byte R/$\overline{W}$ bit)*: 0 - Write to sensor**	06h
●Send (write to sensor) Next Read is Complete Output Buffer (Note: this command does not read). Causes the sensor to prepare to send the entire buffer on the next read. The buffer will be purged upon completion. **Direction *(address byte R/$\overline{W}$ bit)*: 0 – Write to sensor**	0Ch

Figure 10 – I2C Command Header Table

Note that reading data back from the sensor is done following one of the above commands and is detailed below with each command header explanation.

Using NACK & ACK

Testing the sensor for an ACK should be done prior to sending any commands or queries. It is only necessary to send the address byte with the associated sensor number. A write with no data should be used, if a read is utilized and the sensor ACK's indicating that it is ready, it will expect further data causing an error when it is not sent.

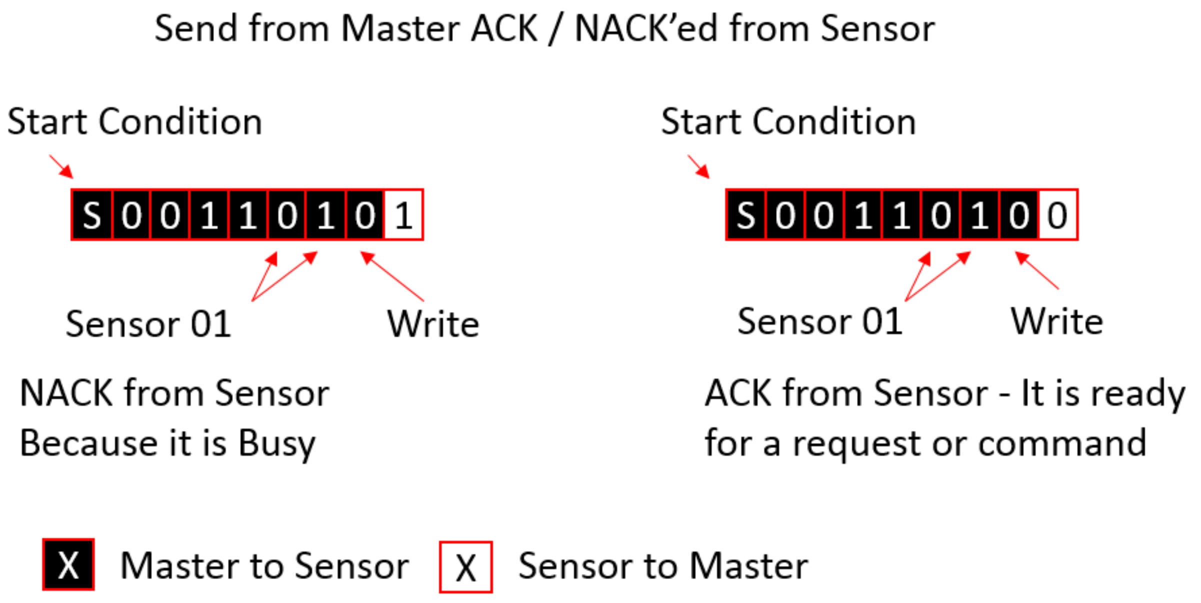

Send from Master ACK / NACK'ed from Sensor
Start Condition
S 0 0 1 1 0 1 0 1
Sensor 01
Write
NACK from Sensor Because it is Busy
Start Condition
S 0 0 1 1 0 1 0 0
Sensor 01
Write
ACK from Sensor - It is ready for a request or command
X Master to Sensor
X Sensor to Master

Figure 11 - ACK & NACK Tests

●The list below details how to use ACK & NACK for testing. At the end of this section is a complete measurement example. ACK.NACK should be utilized prior to beginning any commands or queries to assure readiness. For example after sending a measurement command use ACK/NACK prior to any further communication.

1. Write sensor address only (one byte). Bit 0 = 0 (write) - do not use read (bit 0=1)
2. Test for ACK if NACK
    a. If NACK Wait 1ms minimum or expected cmd execution time – then retry
        i. Optional time out testing
    b. If ACK proceed

06h Send Command plus Prepare Status & Length

● This Command Header, 06h, sends the SCPI command that is included following the header. The command also causes the sensor to prepare the sensor's status byte and the length of the data in the sensor's buffer so that they can be returned. The latter is done after completion of the SCPI command that was included. The header must include the total number of bytes transmitted (excluding the Sensor Number) but including the terminator. Note the Sensor Number is not actually part of the command, it is used by hardware to determine the addressed sensor. If only requesting the sensor status and the length of any available data, no SCPI command or terminator is sent.

Direction *(address byte R/$\overline{W}$ bit)*: 0; Output to sensor.

Output to sensor:
Start Condition, Address byte, 06h, 0, #, #, [SCPI command, terminator], Stop Condition
[] Commands / Queries inside are optional.

<u>Example with command:</u>

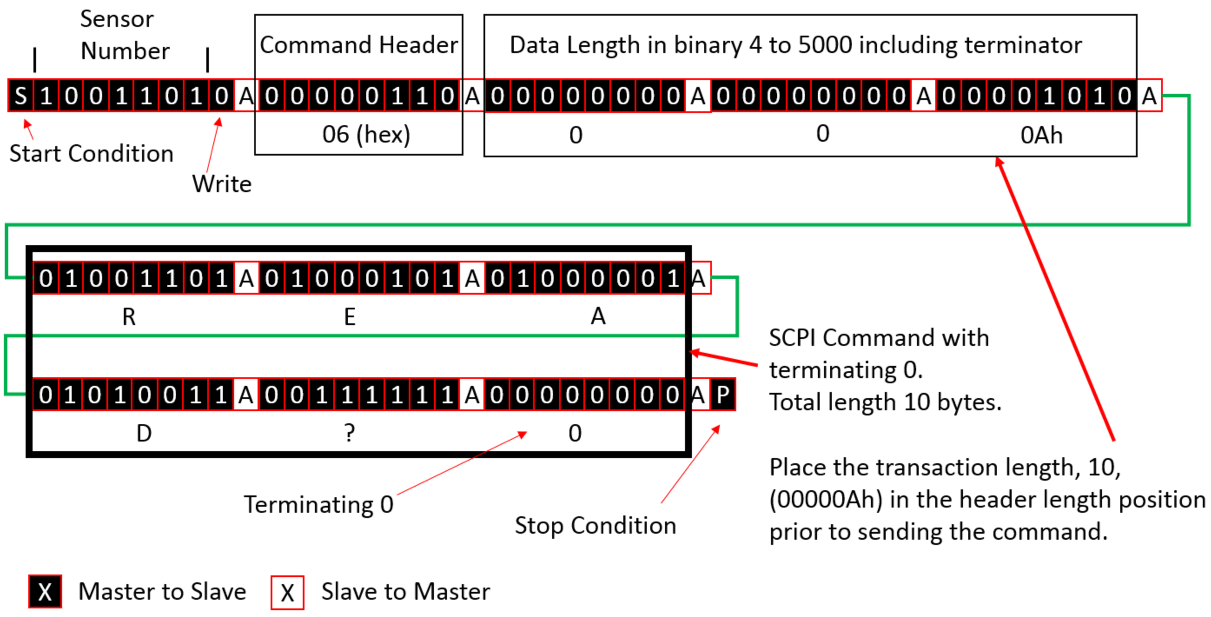

Sensor Number
Command Header
Data Length in binary 4 to 5000 including terminator
S 1 0 0 1 1 0 1 0 A 0 0 0 0 0 1 1 0 A 0 0 0 0 0 0 0 0 A 0 0 0 0 0 0 0 0 A 0 0 0 0 1 0 1 0 A
Start Condition
Write
06 (hex)
0
0
0Ah
0 1 0 0 1 1 0 1 A 0 1 0 0 0 1 0 1 A 0 1 0 0 0 0 0 1 A
R
E
A
0 1 0 1 0 0 1 1 A 0 0 1 1 1 1 1 1 A 0 0 0 0 0 0 0 0 A P
D
?
0
SCPI Command with terminating 0. Total length 10 bytes.
Place the transaction length, 10, (00000Ah) in the header length position prior to sending the command.
Terminating 0
Stop Condition
X Master to Slave
X Slave to Master

Figure 12 – Header 06h with Command Example

Following the command or query, and after sufficient processing time is allowed, ACK should be used to confirm the sensor is ready; then the user controller can read back the four byte return containing status and the length of any available data. This length can then be used to read the data in using header 0Ch. The example and Figure 13 below, details reading back the Status & Length.

Example with no command (Prepare Status & Length):

The controller may request the Status & Length of any available data anytime the sensor is in a ACK condition. This will indicate the sensor's readiness and the number of bytes to be read in. Once the quantity of data is known, Header 0Ch *Read Complete Output Buffer* can be used to read in the data. Using the Prepare Status & Length alone (without a command) is normally not required since the number of bytes available is requested when the command is sent using header 06h. However the status byte can be useful in some instances.

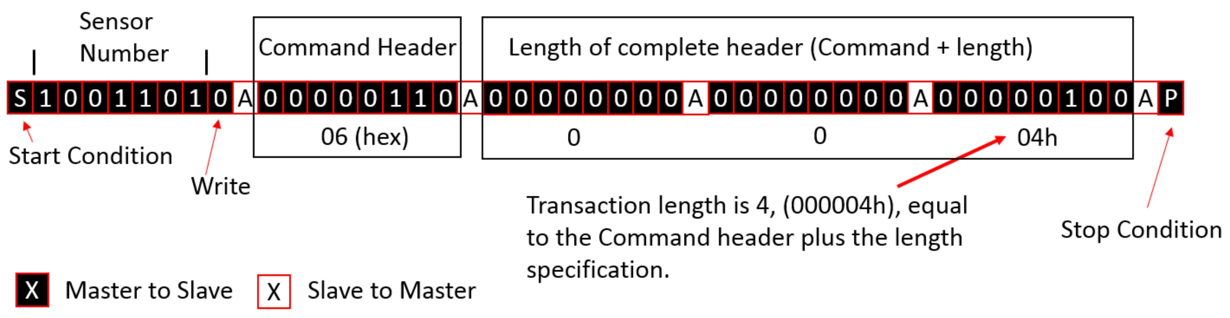

Sensor Number
Command Header
Length of complete header (Command + length)
S 1 0 0 1 1 0 1 0 A 0 0 0 0 0 1 1 0 A 0 0 0 0 0 0 0 0 A 0 0 0 0 0 0 0 0 A 0 0 0 0 0 1 0 0 A P
Start Condition
Write
06 (hex)
0
0
04h
Transaction length is 4, (000004h), equal to the Command header plus the length specification.
Stop Condition
X Master to Slave
X Slave to Master

Figure 13 – Header 06h used to Prepare Status & Length

Following this command, the controller should test using ACK/NACK then read in the 4 bytes. Prepare Status & Length will always return 4 Bytes.

Example of Status and Length Read Back:

● Reading back the status byte and length (either with or without a command) can be done after sending Header 06H and receiving ACK. Normally if sent with no command the length of available data will be 0, and only the status byte is useful.

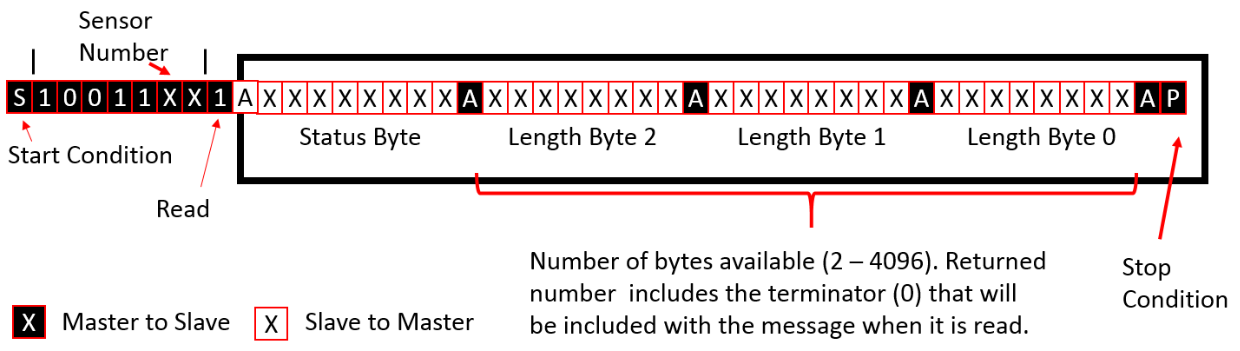

Sensor Number
S 1 0 0 1 1 X X 1 A X X X X X X X X A X X X X X X X X A X X X X X X X X A X X X X X X X X A P
Start Condition
Read
Status Byte
Length Byte 2
Length Byte 1
Length Byte 0
Number of bytes available (2 – 4096). Returned number includes the terminator (0) that will be included with the message when it is read.
Stop Condition
X Master to Slave
X Slave to Master

Figure 14 - Read the Status and Length

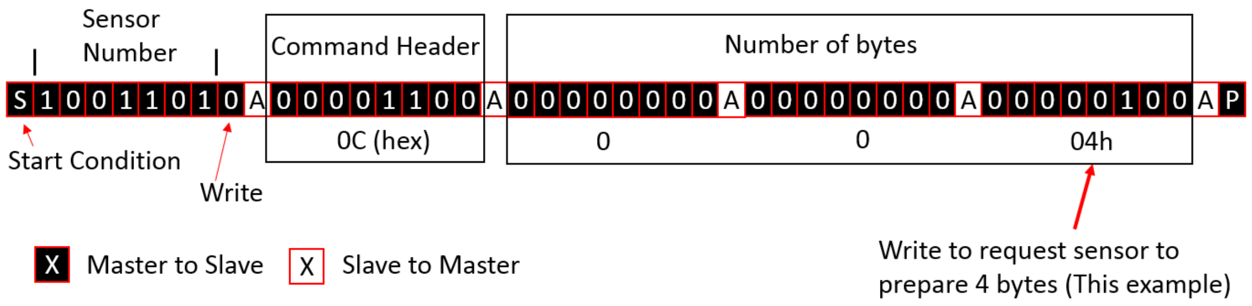
The Status Byte can be used to determine various sensor conditions and the Length is used with the 0Ch command to specify the amount of data to be read in.

0Ch Prepare Complete Output Buffer

● This Command Header, 0Ch, causes the sensor to prepare to return the available data. The number of bytes requested must be the same as the number of bytes indicated in the preceding *Read Status and Length* sequence. The data is clocked in immediately following the complete 0Ch Command as indicated below in the *Read Back* section.

Direction *(address byte R/$\overline{W}$ bit)*: 0; Output to sensor.

Output to sensor: Start condition, Address byte, 0Ch, #, #, #, Stop Condition



Sensor Number
Command Header
Number of bytes
S 1 0 0 1 1 0 1 0 A 0 0 0 0 1 1 0 0 A 0 0 0 0 0 0 0 0 A 0 0 0 0 0 0 0 0 A 0 0 0 0 0 1 0 0 A P
0C (hex)
0
0
04h
Start Condition
Write
X Master to Slave
X Slave to Master
Write to request sensor to prepare 4 bytes (This example)

Figure 15 – Command Header 0Ch Example

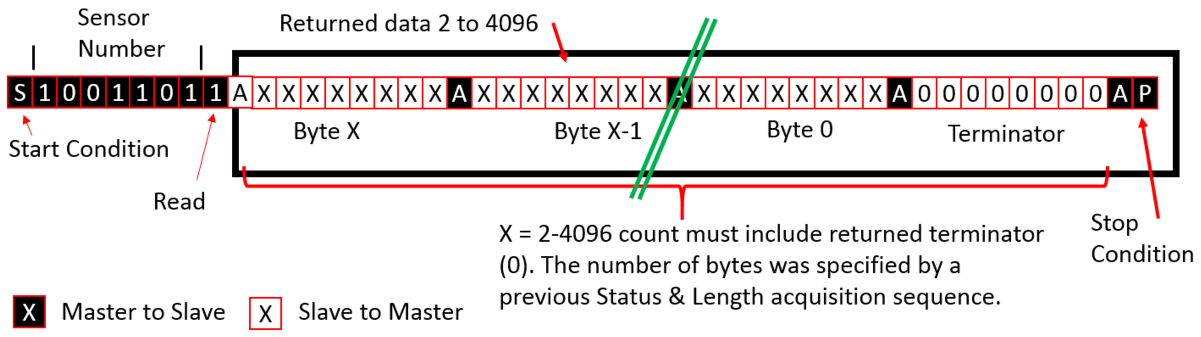
Read Complete Output Buffer is designed to pass the entire buffer back. Any data left in the sensor output buffer that was not requested will be deleted upon completion of the instruction.

<u>Read Back</u>

Following the Prepare Complete Output Buffer command described above, the requested data bytes are read back as follows:

Direction *(address byte R/$\overline{W}$ bit)*: 1; Read to master (user controller).

Output to sensor: Start condition, Address byte, ..., Stop condition
Note ...... are clock cycles.



Sensor Number
Returned data 2 to 4096
S 1 0 0 1 1 0 1 1 A X X X X X X X X A X X X X X X X X A X X X X X X X X A 0 0 0 0 0 0 0 0 A P
Byte X
Byte X-1
Byte 0
Terminator
Start Condition
Read
X = 2-4096 count must include returned terminator (0). The number of bytes was specified by a previous Status & Length acquisition sequence.
Stop Condition
X Master to Slave
X Slave to Master

Figure 16 - Read Back X Bytes

Upon completion, any remaining data in the sensors output buffer is deleted.

I2C Measurement Example

●The following example details the use of the READ? query to make a measurement. Read is often use programmatically because it establishes the beginning of the measurement when issued. If continuous triggering is active, FETCh? can be used to collect a "trailing" measurement. MEAS? may be inappropriate because the measurement time will vary due the use of automatic averaging. Measurements made with no power applied can result in a 30 second return time if using MEAS?

1. Test for ready see below
2. Write *SYST:PRES DEF* to the sensor using Header 06h, then wait 1ms
   Clears any settings and sets the sensor to a known state
3. Test for ready see below
4. Write *INIT:CONT 0* to the sensor using Header 06h, then wait 1ms
   Turns off continuous initiation - sensor will then initiate after READ? is issued
5. Test for ready see below
6. Write *AVER:COUN:AUTO 0* to the sensor using Header 06h, then wait 1ms
   Turns off automatic averaging
7. Test for ready see below
8. Write *FREQ 1000 MHZ (use your frequency)* to sensor using Header 06h, then wait 1ms
   Sets the measurement frequency
9. Test for ready see below
10. Write *AVER:COUN 10 (use your # of averages)* to sensor using Header 06h, then wait 1ms
    Sets the number of averages
11. Test for ready see below
12. Write *READ?* to sensor using Header 06h
    Initiates (Starts) the measurement
13. Wait the expected processing time (depends upon the number of averages)
14. Test for ready see below
15. Send Prepare Status & Length (no command) using Header 06h See Figure 13
16. Test for ready see below
17. Read 4 bytes from sensor - See Figure 14
18. Test the Status Byte if No data, wait at 50ms, or a time dependent upon the number of averages, then return to #16
19. Using value returned in #17, write the number of bytes to the sensor that are to be read back using Header 0Ch. See figure 15
20. Test for ready see below
21. Read back the measurement, specify the number of bytes as indicated by #17 – See also Figure 16
22. Repeat 11-24 as required for the process (each time a measurement is required)

Test for Ready:

1. Write address only to the sensor See Figure 10
2. Test ACK/NACK if NACK, then wait 1ms and return to 1

USING SPI

Electrical Data Connection and Addressing

Enabling the SPI/I2C interface is controlled by Pin 2 (SPI/I2C Enable). Applying power to Pin 2 (from pin 6) disables USB connectivity and enables the SPI/I2C interface. Choosing SPI or I2C for a LB5900 sensor is done using Pin 1 of the interface cable. Grounding the pin will enable SPI and disable I2C, applying power (from Pin 6), selects I2C and disables SPI.

Data direction. To simplify connectivity, LadyBug utilizes the MOSI/MISO naming scheme. This allows the pins on the master and the slave to have the same name. For example, the output of the master must always be connected to the input of the slave. This line is called MOSI, **M**aster **O**ut **S**lave **I**n. The master's input is MISO, **M**aster **I**n **S**lave **O**ut. Some microcontroller manufacturers may not use this terminology, however the SPI connections function the same.

SPI sensors require a separate Slave Selection (SS) connection for each sensor used. For example if two sensors are used, SCLK, MOSI and MISO are all connected to each sensor and the microcontroller. The microcontroller must also have two SS outputs, one to select each sensor. Only one SS line may be active at any given time, the non-active line disables the sensor's MISO (the sensor's data output line in this case) line. The SS line is active low as shown in Figure 17.

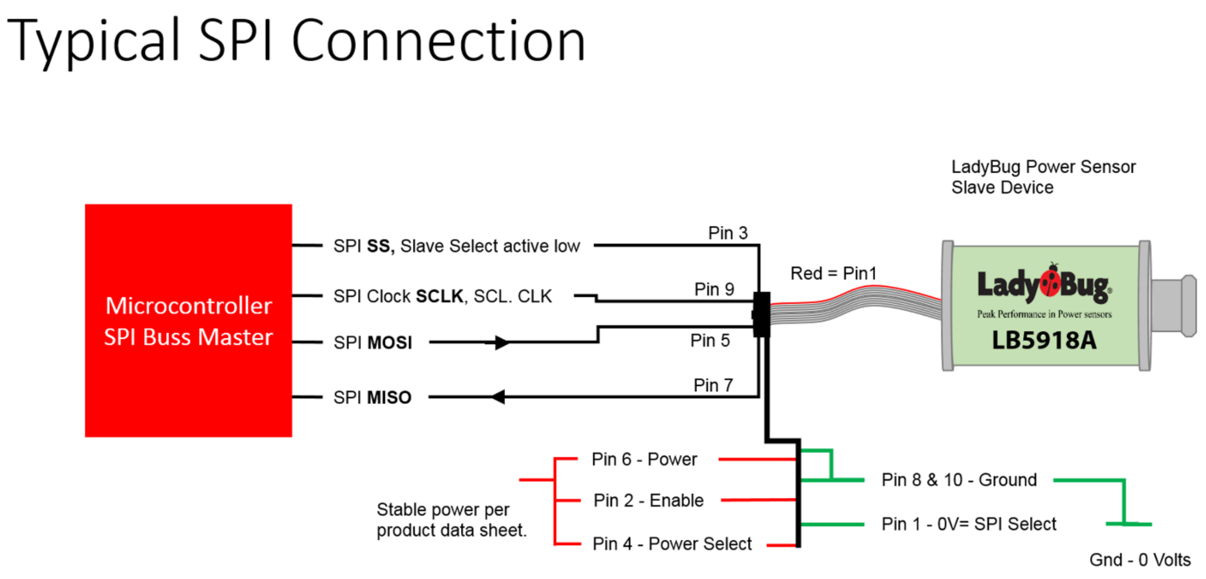


Figure 17 - Typical SPI connection

SPI Message Overview

LB5900 sensors utilize textual SCPI (Standard Commands for Programmable Instruments) commands. Commands are detailed in LadyBug's LB5900 Programming guide. Commands sent to; and data received from the sensor are encoded as ASCII text; however the command header, described below is binary along with the status byte.

SPI Clock Phase and Polarity

LadyBug sensors utilizes full-duplex SPI communication. Clocking mode / phase is an accepted standard mode based on the clock's polarity (CPOL) and phase (CPHA). The term phase indicates whether the bit-data is to be read on the leading or trailing edge of the clock; while the term CPOL indicates the base, or off state of the clock.

With LB5900 sensors, the clock standby condition is on or V+, and data is captured on the trailing edge of the clock. This is indicated as "Capture" in the timing diagram in Figure 18, which details the sensors CPOL=1 and CPHA=1 SPI timing.

The clock is always generated by the master (user controller) and both the master and sensor capture data on the trailing clock edge. The data state is expected to be stable for both ends of the SPI connection when the clock transition occurs.

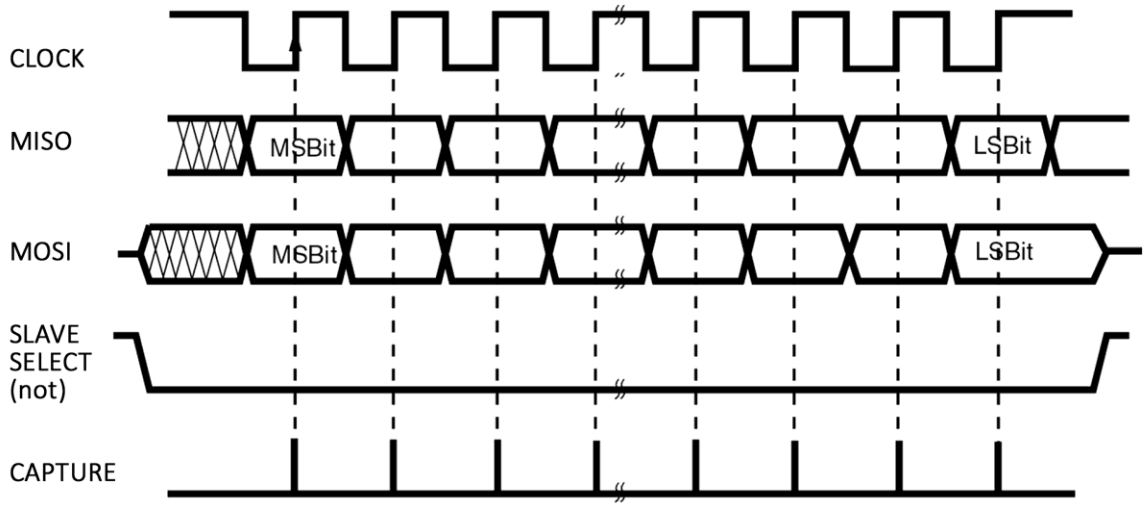


Figure 18 – Sensor Bit Timing Diagram

For the master (user controller) to receive data from the sensor, it must generate the clock, the sensor will synchronize the return data with this clock. Normally, the master accomplishes this by sending irrelevant data, often 0's, to its SPI module. The SPI module generates the clock signal. If zeros are sent, the MOSI line remains at 0 during the transmission.

Sensor SPI Module Reset

●In SPI Mode, while the clock line remains in the inactive state, pulsing the Slave Select line low twice for 1ms resets the Sensor's SPI module and issues a sensor DCL command. Refer to Figure 19, SPI Module Reset. This should be done at power up to eliminate erroneous data. The module can also be reset by holding the Slave Select line low for greater than one second while the clock line remains in the inactive state.

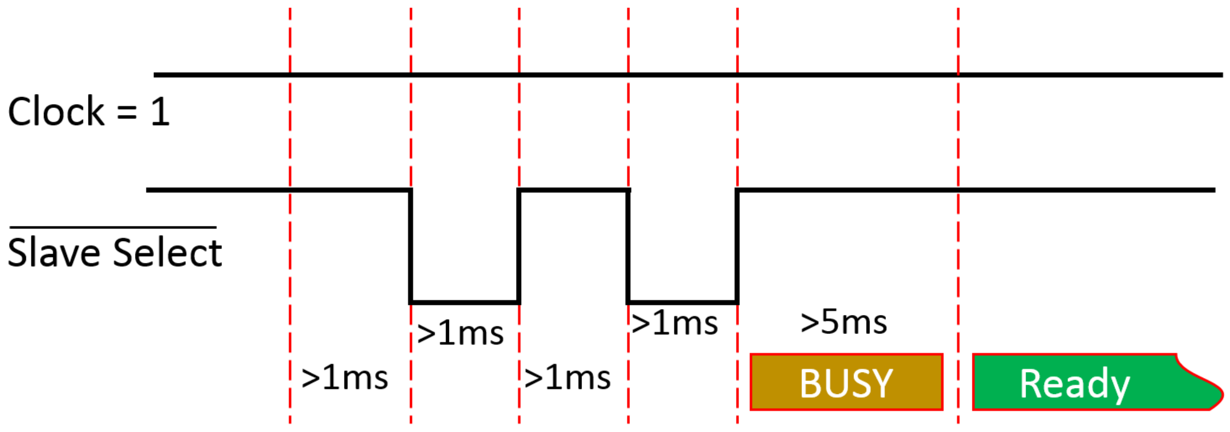


Figure 19 - SPI Module Reset

<u>Data format</u>
SPI utilizes hardware sensor addressing. The first data sent to the sensor is the command header, described next.

Command Header
All commands sent to the sensor include a binary-formatted header preceding the SCPI command or data. The first byte of the Command Header (Header) indicates how the message and any following sensor response will be handled. Figure 20 details the SPI Command Headers.

Command Header & Data Description	Length	Header
Write Command. Causes the sensor to execute the SCPI command that follows the header. Data returned with the command includes the Busy/Ready byte & previous communication status.	4 bytes plus command & terminator	F0h
Read Status & Length. Returns the Busy/Ready byte, previous SPI status, sensor status byte and the number of bytes that are in the buffer ready to read.	6 bytes total	06h
Read Complete Output Buffer. Returns the complete output buffer (up to 4096 bytes including terminator). The buffer will be purged upon completion. Shortest buffer is 2 bytes long.	3 bytes plus the buffer contents (buff minimum 2) Total Minimum 5.	0Ch

Figure 20 – SPI Command Header Table

Commands with no required response can be sent with a single communication if the sensor is not busy. Busyness detection is recommended. Queries require at least two communications, one to start a function, such as a measurement, then a second to read back the resultant data. After the request is made to the sensor, the master (user controller) is required to allow sufficient time for the sensor to process the function prior to requesting data. During the time the sensor is processing data, the master can address other sensors. For example, a routine could be created to start measurements on two sensors, then go back and read each one after a period of time has elapsed. Busyness detection is recommended in all cases.

Previous SPI communication status

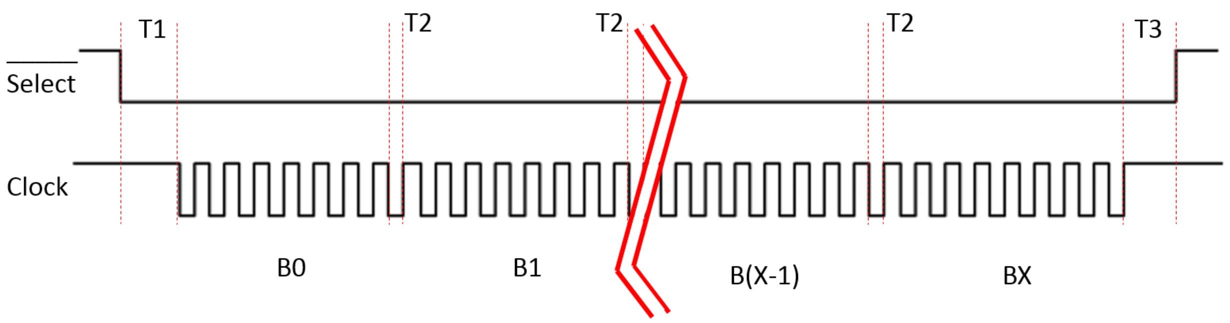
The second byte returned with all command headers is the results of the last SPI communication. These codes indicated any errors associated with the communication, and are detailed in Figure 21.

SPI Error Codes	**Header**
No Errors	E0h
Last communication was under clocked (not all data was received)	E1h
Last communication was over clocked (Excess data was sent and ignored)	E2h

Figure 21 – SPI Error Codes

Busyness

Each byte received by the sensor must be processed after reception. If the clock rate is relatively low, processing can be done during the clock cycle. However, it may be necessary to include a time allowance between bytes for processing as indicated below.



Item	Time
T1	>1us
T2	>0.5 µs*
T3	>1 µs*

*T3 Note: After T3 has elapsed, a processing time allowance must be included so that the sensor has time to process the request. Several milliseconds may be required for non-measurement commands. T2 Note: If clocking rate is below 1MHz, the T2 ½ cycle does not need to be stretched.

Figure 22 - Byte Timing Diagram and Table●

Processing begins when the SS ($\overline{Select}$) line is raised after a command is received from the master (user controller). During processing, the sensor has limited capability to respond. For example, after issuing a MEAS? or READ?, the process of collecting the averaged measurement begins. During this time the sensor will report as busy in one of the two ways detailed below.

Busy/Ready

The Busy/Ready byte is always the first byte returned with the Command Header. While waiting for a measurement to complete, checking using Command Header 06H is recommended. It will return the Busy/Ready byte, the communication error byte, the Sensor Status byte and the length of any available data. If the Busy/Ready byte is not zero, the sensor cannot process further instructions. When using Header 06h, the entire 6 bytes must be read in each time or an SPI error will be indicated on the next communication. When Busy/Ready is zero, the sensor can accept commands. This does not indicate that a measurement is complete and will occur while the sensor is processing measurements. Commands such as *DCL can be sent at this time to clear the sensor. Warning: The Sensor uses interrupts to process SPI communications; requests should be limited to less than one per millisecond. This will allow the sensor sufficient time to respond as well as process the commands.

Sensor Status Byte (STB)
Once Busy/Ready indicates the sensor is ready to process a command, the status byte can be utilized to determine if a measurement is complete. Status byte meaning is detailed in the sensor's programming guide. Bit 4 (Value 16) indicates that data is available. An alternate method is to use the three return length bytes, if these are zero, no data is ready. This indicates that the sensor has not completed the measurement (if one is in process).

Bit	Value	Meaning
0	1	Not in use
1	2	Device Status Condition-Future implementation
2	4	One or more errors are present in the error queue
3	8	Questionable Status Summary– Future implementation
4	16	The output buffer contains a message
5	32	Event Status Bit– Future implementation
6	64	Master Summary Status-Future implementation
7	128	Status Summary-Future implementation

Figure 23 - Sensor Status Byte

Measurement Processing Time
To assure the fastest measurement, the user should determine the time required to make the measurement in advance, and then limit requests to the sensor until such time has elapsed (during the measurement). The use of READ? or FETCh? are recommended. MEAS? will cause the sensor to adjust the number of averages and will result in a varying measurement time that could be very long if the measured power level is low.

Header Details
Command Headers are used to manage SPI communications. The headers include data length, Busyness information and status so that the user can properly communicate with the sensor.

The maximum allowable SPI baud rates allow communication at rates that are greater than power measurements can be made. As a result of this and other factors, checking the sensor for busyness is an important part of SPI communication.

Additional communication information is provided in example code and demonstration kits.

Command Header 06h - Read Status & Length

Figure 24 details the communications for Command Header 06h with an LB5900 sensor. Command Header 06h returns the Sensor Status and the length of any available data. The command is always be 6 bytes in total length, an error will result and be returned on the next commination if the length is not correct. As with all LB5900 SPI communications, data is clocked out of the sensor as the command and don't care bits are clocked in. As can be seen from Figure 24, don't care data is clocked in after the command to provide the proper number of clocks for the returned data.

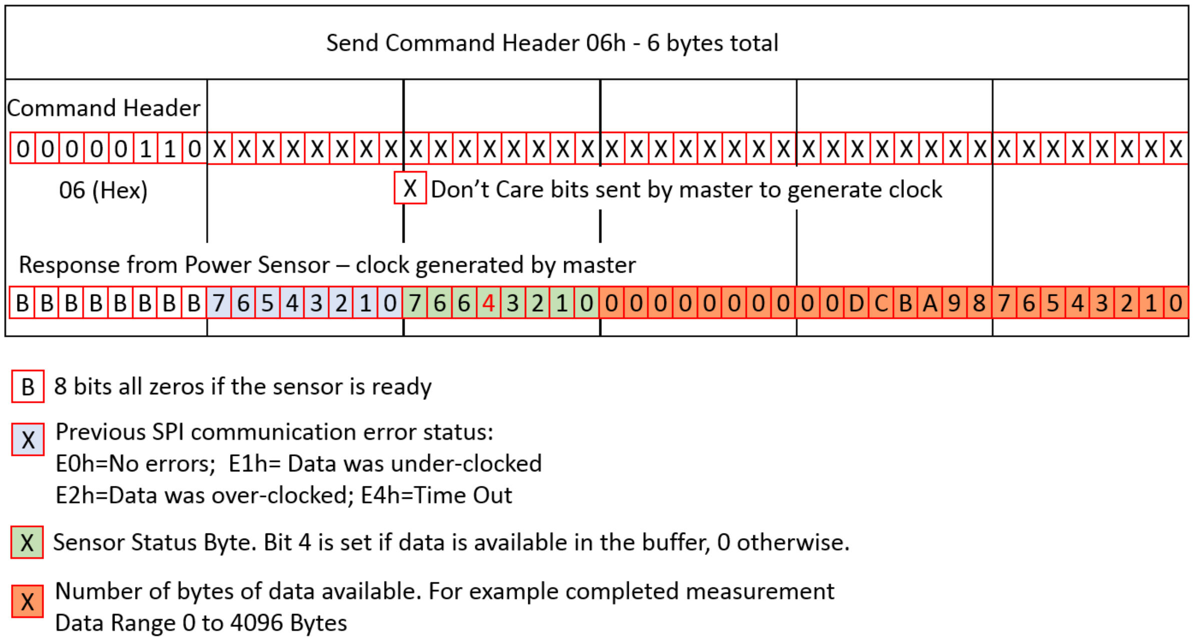

Send Command Header 06h - 6 bytes total
Command Header
0 0 0 0 0 1 1 0 X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
06 (Hex)
X Don't Care bits sent by master to generate clock
Response from Power Sensor – clock generated by master
B B B B B B B B 7 6 5 4 3 2 1 0 7 6 6 4 3 2 1 0 0 0 0 0 0 0 0 0 0 0 D C B A 9 8 7 6 5 4 3 2 1 0
B 8 bits all zeros if the sensor is ready
X Previous SPI communication error status:
E0h=No errors; E1h= Data was under-clocked
E2h=Data was over-clocked; E4h=Time Out
X Sensor Status Byte. Bit 4 is set if data is available in the buffer, 0 otherwise.
X Number of bytes of data available. For example completed measurement
Data Range 0 to 4096 Bytes

Figure 24 - Command Header 06h Details

Referring to Figure 24 and the first byte, as the command is clocked in, the sensor availability is returned. This first "busy" byte is the same with all Command Headers. If zero (binary) is returned, the sensor is available and the command will function as expected. This is separate from the Sensor's measurement completion flag. Measurement completion is indicated by bit 4 of the status byte, however this byte is not valid if the first byte indicates sensor is not ready. All six bytes should be read, even if zero is not returned.

The second byte indicates the status of the last communication as detailed in the Command Header section. This byte IS valid regardless of the sensor's busy byte.

The third byte will contain the Sensor's Status Byte if the busy byte is zero. Use bit 4 of this byte to determine if the sensor has data ready. This bit will be zero if the sensor has not completed a measurement, or other return data function is not ready; however the status byte is not valid unless the first "busy" byte is zero.

If bit 4 of the third byte is 1, it indicates that data is available and the binary value of the last three bytes of the 6-byte communication indicates the length of the available message (less the message terminator). The length of return can be between 0 and 4095 bytes including the required terminator.

Command Header F0h – Write Command

Figure 25 details the communications for Command Header F0h with an LB5900 sensor. This Command Header is used to issue any command or query, such as FETCh?' FREQ etc. The command does not implement reading back Query results, it simply issues the command. Results of any Query are read with Header 0Ch described later.

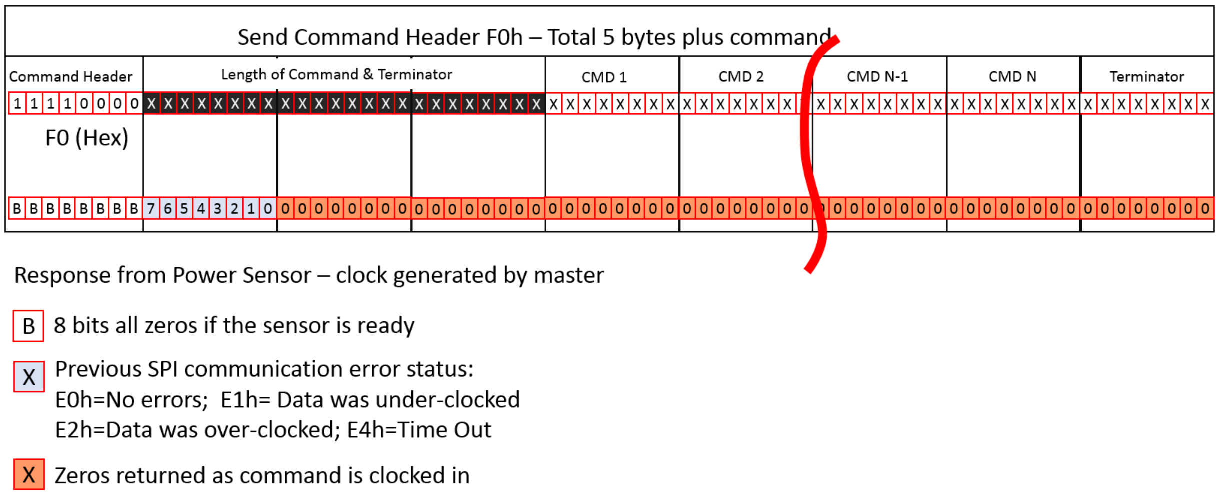

Send Command Header F0h – Total 5 bytes plus command
Command Header
Length of Command & Terminator
CMD 1
CMD 2
CMD N-1
CMD N
Terminator
1 1 1 1 0 0 0 0
F0 (Hex)
B B B B B B B B 7 6 5 4 3 2 1 0
Response from Power Sensor – clock generated by master
B 8 bits all zeros if the sensor is ready
X Previous SPI communication error status: E0h=No errors; E1h= Data was under-clocked E2h=Data was over-clocked; E4h=Time Out
X Zeros returned as command is clocked in

Figure 25 - Command Header F0h●

The length of Command Header F0h is dependent on the command being sent, however the minimum is 5 bytes, including the message terminator. The minimum 5 bytes including the terminator, is only the header itself, no command is included. This can be utilized to read the busy byte.

●Write Command Example:

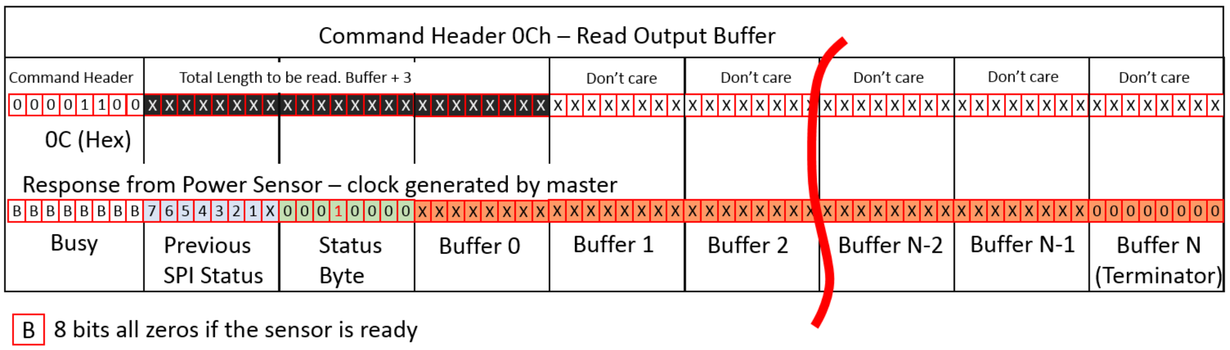
Note: In this example, a prior communication indicated that the sensor was ready.

To Sensor: *F000000E*SYST:PRES DEF00
Returned Data: 00E000000000000000000000000000000000

Note the only usable data in the return is the Busy byte (00); the Previous communication status (E0), the balance of the zeros are don't cares

Command Header 0Ch – Read Complete Output Buffer

●After using Command Header 06h to determine that there is data and the length of the data, Command Header 0Ch is used to read back the data. Referring to Figure 26, the command reads the entire buffer which cannot be longer than 4095 bytes plus the terminator. The data will be returned followed by the message terminator. As with all communications, the user controller must generate the required clocks. This is done by sending irrelevant data to the sensor.



B	8 bits all zeros if the sensor is ready
X	Previous SPI communication error status: E0h=No errors; E1h= Data was under-clocked E2h=Data was over-clocked; E4h=Time Out
X	Sensor Status Byte. Bit 4 is set if data is available in the buffer, 0 otherwise.
X	Returned message. Terminator is last byte

Figure 26 - Command Header 0Ch

Read Back Example:

Prior communication indicated that there are 16 bytes (010h) available.

To Sensor: *0C000010*000000000000000000000000000000
Returned message: *00E010*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

All shown in hex plus the headers shown in *italic*. Note that the full 16 byte message (15 xx and the 00 terminator), is returned starting with the last byte of the length specification in the 0C header. Since the return message header is 3 bytes and the 0C header is 4 bytes, only 15 bytes are required after the 0C header to read in the specified 16 bytes.

Communication Examples

The following examples detail communications between the User Controller and LB5900 sensor in SPI mode. The commands information is detailed in Hex data form in full bytes with bit data flow not shown. Please refer the SPI message information for bit flow, clocking and additional SPI enable information.

Each SPI transaction is started when the SS (active low) line is enabled (lowered). The complete transaction is made, and then the SS line returned to the inactive level (high), completing the transaction.

Measurement Example

The measurement example includes measurement setup, initiation of the measurement and reading back the data.

The following commands will be utilized in the order shown.

# 1 SYST:PRES DEF	Clears any settings
#2 TRIG:DEL:AUTO 0	Turns off Trigger delay
#3 INIT:CONT 0	Sets the sensor for single triggered measurements
#4 AVER:COUN:AUTO 0	Turns off automatic averaging
#5 FREQ 1020 MHZ	Sets the frequency
#6 AVER:COUN 10	Sets Averaging to 10
#7 READ?	Makes the measurement (Repeat as needed)

Sending the commands

It is recommended that Command Header 06h, Read Status and Length be utilized prior to sending each command. This will assure that the sensor is ready to accept the command. This is recommended because commands can be sent much faster than the sensor can process them.

To the Sensor: **060000000000** Send exactly 6 bytes
Return if busy:**FFE000000000** DO NOT SND COMAND-WAIT UNTIL REDY
Note the returned second byte is shown as E0 however it could be different and contain an error

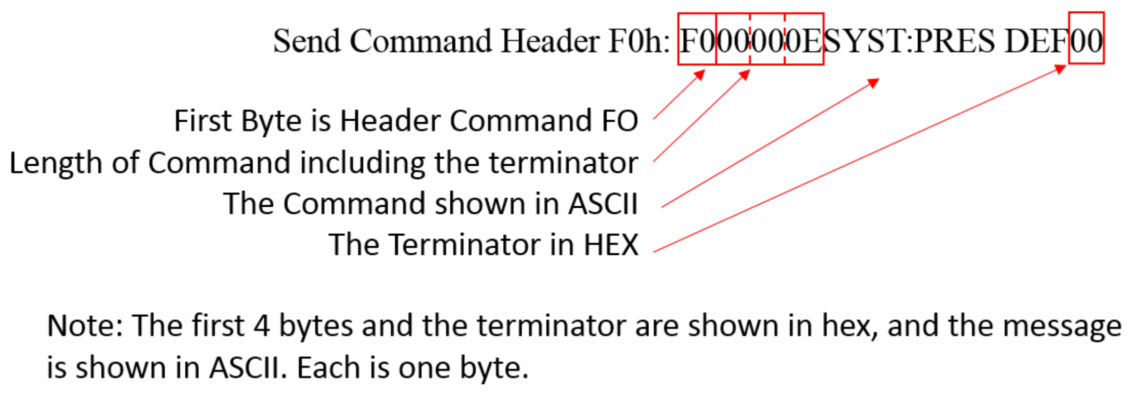
Return if ready:**00E000000000** SENSOR CAN ACCEPT A COMAND
Note the returned second byte is shown as E0 however it could be different and contain an error

The commands:

#1 SYST:PRES DEF **Sets the sensor to a known state**

To the Sensor: **F000000E**SYST:PRES DEF**00**
Expected return:00E000000000000000000000

It is recommended that the returned data from byte 2, Previous SPI Transaction error be tested. E0h is expected if no errors were encountered. Refer to SPI message overview for details.



Send Command Header F0h: F000000ESYST:PRES DEF00
First Byte is Header Command FO
Length of Command including the terminator
The Command shown in ASCII
The Terminator in HEX
Note: The first 4 bytes and the terminator are shown in hex, and the message is shown in ASCII. Each is one byte.

Figure 27 - Command Details●

As detailed in Figure 27, the command header is 4 bytes, the command header is shown in hexadecimal and is followed by the ASCII encoded message of 13 bytes including the space. Finally, the message terminator of 0, shown in hexadecimal. The total is 18 bytes, with the message and terminator length of 14, inserted in its hex form as 00000E. Note that below, HEX byte pairs are shown in bold.
At this point, a 1ms delay can be added or a “Read Status and Length” loop reading the status can utilized. For details on a “Read Status and Length” Loop refer to the READ? Command at the end of this section.

#2 TRIG:DEL:AUTO 0 **Turns off Trigger delay**

●To the sensor: **F0000010**TRIG:DEL:AUTO 0**00**

At this point, a 1ms delay can be added or a “Read Status and Length” loop reading the status can utilized. For details on a “Read Status and Length” Loop refer to the READ? Command at the end of this section.

#3 INIT:CONT 0 **Sets the Sensor for single measurements**

To the sensor: **F000000C**INIT:CONT 0**00**

At this point, a 1ms delay can be added or a “Read Status and Length” loop reading the status can utilized. For details on a “Read Status and Length” Loop refer to the READ? Command at the end of this section.

#4 AVER:COUN:AUTO 0 **Turns off automatic averaging**

● To the sensor: **F0000011**AVER:COUN:AUTO 0**00**

At this point, a 1ms delay can be added or a “Read Status and Length” loop reading the status can utilized. For details on a “Read Status and Length” Loop refer to the READ? Command at the end of this section.

#5 FREQ 1020 MHZ **Sets the frequency**

To the sensor: **F000000E**FREQ 1020 MHZ**00**

At this point, a 1ms delay can be added or a “Read Status and Length” loop reading the status can utilized. For details on a “Read Status and Length” Loop refer to the READ? Command at the end of this section.

#6 AVER:COUN 10 Sets Averaging to 10

To the sensor: **F000000D**AVER:COUN 10**00**

At this point, a 1ms delay can be added or a “Read Status and Length” loop reading the status can utilized. For details on a “Read Status and Length” Loop refer to the READ? Command at the end of this section.

#7 READ? Makes the measurement

Read initiates the measurement. Once complete the measurement will be read using Header 0Ch Read Complete Output Buffer. Depending on the number of averages and other settings, making the measurement will take a relatively long period of time. An estimated time will be utilized before testing for measurement completion.

To the sensor: **F0000006**READ?**00**

A delay of 250ms is added at this point in the controller program because it is known that the measurement will take at least that long.

Now test the sensor to see if the measurement is complete.

To the Sensor: **060000000000** Send all 6 bytes
Return if busy:**FFE000000000** Sensor not Ready
Note: The returned second byte is shown as E0 however it could contain an error
Note: The measurement completion state is not available because the sensor is not ready

Return if Sensor Ready but measurement is not complerte:**00E000000000**
Return if Sensor Ready and measurement is complerte:**00E010000010 (Hex)**

A loop can be setup to return back *before* the 250ms delay to prevent excess measurement interruption. Note that the ready return indicates that there are 16 bytes in the buffer.

●Get the measurement:
Send the header plus enough zeros to clock in the entire 16 byte message and terminator. In the example below, the first of the return bytes (“-“) was clocked in by the last of the 4 byte header and an additional 15 bytes must be send to read back the 16 byte message.

●To the Sensor (in HEX): **0C00001000000000000000000000000000000**
Return: **00E010**-6.84101868E+01**00 Note: Bold indicates hex pair**

Measurement value: -6.84101868E+01 Default units are dBm

Additional SPI Messaging Example

●The example below details byte by byte communications. All data is shown in Hex.

```
=================================================================

ENTER COMMAND OR QUERY-->read?
--> 06 00 00 00 00 00    READ STB AND LENGTH
<-- 00 E0 00 00 00 00    READY   SPI=E0   STB=00   BUFFER EMPTY

--> F0 00 00 06 .....    read?\0
<-- 00 E0 ...........    READY   SPI=E0

--> 06 00 00 00 00 00    READ STB AND LENGTH
<-- FF E0 00 00 00 00    BUSY    SPI=E0   STB=00   BUFFER EMPTY

--> 06 00 00 00 00 00    READ STB AND LENGTH
<-- FF E0 00 00 00 00    BUSY    SPI=E0   STB=00   BUFFER EMPTY

--> 06 00 00 00 00 00    READ STB AND LENGTH
<-- FF E0 00 00 00 00    BUSY    SPI=E0   STB=00   BUFFER EMPTY

--> 06 00 00 00 00 00    READ STB AND LENGTH
<-- FF E0 10 00 00
10    BUSY    SPI=E0   STB=10   LENGTH=0x000010

--> 0C 00 00 10 .....    READ OUTPUT BUFFER
<-- 00 E0 10 ........    -3.72808420E+00\0

--> 06 00 00 00 00 00    READ STB AND LENGTH
<-- 00 E0 00 00 00 00    READY   SPI=E0   STB=00   BUFFER EMPTY

=================================================================
```

REVISIONS & UPDATES

Document Revision V2.3
No Notes

Document Revision V2.4
Firmware: Added message length error test
Firmware: Improved DCL and SPI module reset functions for SPI/I2C
Document: Message length specified for SPI 0Ch messages. Page 27●
Document: Detailed that concatenated messages are illegal in SPI/I2C
Document: Added Information regarding SPI/I2C DCL and module resets
Document: Added details to timing diagrams and text
Important changes marked with ● on Pages 28, 27, 24, 21

Document Revision V2.5
Clarification regarding SPI/I2C Selection during Startup
Important changes marked with ● on Pages 4, 6

Document Revision V2.6
Document: Minor text changes

Document Revision V2.7 July 2017
Document: Minor text changes
Document: DCL / Abort change - Page 4●
Document: Corrections to the I2C Section● - All pages
Document: Added I2C Status Byte information Page 13●
Document: I2C Example updated Page 20●
Document: SPI Trigger example Page 27●
Document: SPI Auto Average example Page 29●
Document: SPI Auto Average example Page 30●
Document: Command Header 0C Page 27●
Document: Added 06 Header example Page 26●
Document: Updated image Page 26●
Document: Added SPI messaging diagram Page 32●

Document Revision V2.72 July 2021
Document: Update company information Page 1
Document: Revise sensor image and text Page 5