USER'S MANUAL AND PROGRAMMING GUIDE

HDL-64E S3

High Definition LiDAR Sensor

Velodyne®

Table of Contents

Safety Notices 4
Caution 4

Introduction 5
In the Box........ 5

Principles of Operation 6

Installation 8
Mounting 8
Wiring 8
Power 8
Lockout Circuit 8
Ethernet 8
Serial Interface RS-232........ 8

View Image 9
1. Use the Included Point-cloud Viewer........ 9
2. Develop Your Own Application-Specific Point-Cloud Viewer 9

Change Run-Time Parameters........ 12
Sample Batch File (.bat)........12
Sample SERCMD.txt file 12
Available Serial Commands HDL-64E S3 13
Control the Spin Rate........14
Change Spin Rate in Flash Memory 14
Change Spin Rate in RAM Only........14
Limit Horizontal FOV Data Collected........14
Define Sensor Memory IP Source and Destination Addresses........15
Dual Returns........15
Laser Output Power Level Management and Reporting 17
Noise Level Adjustment 17
Upload Calibration Data........18
External GPS Time Synchronization 18
Packet Format and Status Byte for GPS Time Stamping 19
Timestamp Accuracy Rules and Accuracy........19
Laser Firing Sequence and Timing........20

Firmware Updates........ 21

Appendix A: Mechanical Drawings 22

Appendix B: Connector Pin outs........ 25
Power, Serial & GPS Connector 25
Ethernet Connector........25

Appendix C: Digital Sensor Recorder (DSR)........ 26
Install DSR........26
Calibrate DSR 26
Live Playback........26
Record Data........27
Playback Recorded Files 27
DSR Key Controls........28

Appendix D: MATLAB Sample Code 29
Calibration and Sensor Parameter Data 30

Appendix E: Data Packet Format 31
Data Packet Overview 31
Status Type Rotation 32
Status Value Example for Status Types of H, M, S, D, N, Y, G, T and V: 32
GPS Status Values 33
GPS Timestamp Conversion Example 33
Status Type Calibration and Unit Parameters 33
Warning Status Bits 38
CRC Checksum Algorithm 39
Last Six Bytes Examples 39

Appendix F: Dual Two Point Calibration Methodology & Code Samples 43
Coordinate Calculation Algorithm Sample Code 43
Intensity Compensation vs Distance 45
Calibration Window 45
Intensity Value Corrected by Distance 46

Appendix G: Ethernet Timing Tables 47
Sensor Ethernet Transmit Timing Tables 47
How to Use the Timing Tables 47

Laser and Detector Arrangement 49
Lasers as Seen from the Back of the Unit 49
Laser as Seen on the Wall 49

Appendix I: Angular Resolution 50

Appendix J: Troubleshoot 51

Appendix K: Service and Maintenance 52

Appendix L: Specifications 53

Safety Notices

IMPORTANT SAFETY INSTRUCTIONS

CAUTION

RISK OF ELECTRIC SHOCK
DO NOT OPEN

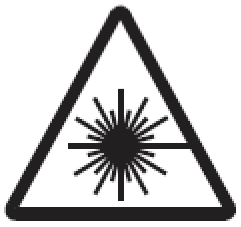
Caution

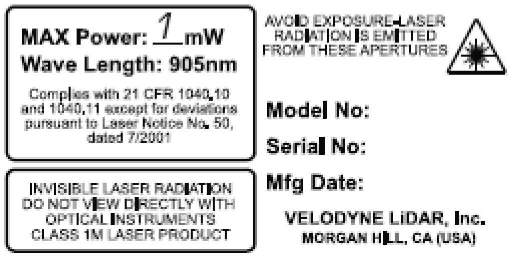
To reduce the risk of electric shock and to avoid violating the warranty, do not remove cover (or back). Refer servicing to qualified service personnel.

The lightning flash with arrowhead symbol is intended to alert the user to the presence of uninsulated “dangerous voltage” within the product’s enclosure that may be of sufficient magnitude to constitute a risk of electric shock to persons.

The exclamation point symbol is intended to alert the user to the presence of important operating and maintenance (servicing) instructions in the literature accompanying the product.

1. **Read Instructions** — All safety and operating instructions should be read before the product is operated.
2. **Retain Instructions —** The safety and operating instructions should be retained for future reference.
3. **Heed Warnings —** All warnings on the product and in the operating instructions should be adhered to.
4. **Follow Instructions —** All operating and use instructions should be followed.
5. **Servicing —** The user should not attempt to service the product beyond what is described in the operating instructions. All other servicing should be referred to Velodyne.





MAX Power: 1 mW
Wave Length: 905nm

Complies with 21 CFR 1040.10 and 1040.11 except for deviations pursuant to Laser Notice No. 50, dated 7/2001

INVISIBLE LASER RADIATION
DO NOT VIEW DIRECTLY WITH OPTICAL INSTRUMENTS
CLASS 1M LASER PRODUCT

AVOID EXPOSURE-LASER RADIATION IS EMITTED FROM THESE APERTURES

Model No:

Serial No:

Mfg Date:

VELODYNE LiDAR, Inc.
MORGAN HILL, CA (USA)

Introduction

Congratulations on your purchase of a Velodyne HDL-64E S3 High Definition LiDAR Sensor. These sensors represent a breakthrough in sensing technology by providing more data points regarding the surrounding environment than previously possible.

NOTE: The HDL-64E S3 High Definition LiDAR sensor is referred to for brevity as “the sensor” throughout this manual.

This manual and programming guide covers:

- Installation and wiring
- The data packet format
- The serial Interface
- Software updates
- GPS
- Viewing the data
- Programming information

HDL-64E S3			
Lower Laser Block	**Upper Laser Block**	**Vertical Field of View (VFOV)**	**Primary Application**
32 lasers separated by 1/2° vertical spacing	32 lasers separated by 1/3° vertical spacing	+2° to 24.8°	Autonomous navigation

For the latest updates to this manual – check www.velodynelidar.com.

In the Box

Each shipment contains:

- Sensor
- CD with
  - Calibration file **.xml** file
  - DSR viewer software
  - MATLAB program for reading calibration parameters from the sensor
  - *User’s Manual and Programming Guide*

Principles of Operation

The sensor has 64 lasers fixed mounted on upper and lower laser blocks, each housing 32 lasers. Both laser blocks rotate as a single unit. With this design each of the lasers fires tens of thousands of times per second, providing exponentially more data points/second and a more data-intensive point cloud than a rotating mirror design. The sensor delivers a 360^{o} horizontal Field of View (HFOV) and a 26.8^{o} vertical FOV.

Additionally, state-of-the-art digital signal processing and waveform analysis are employed to provide high accuracy, extended distance sensing and intensity data. The sensor is rated to provide usable returns up to 120 meters. The sensor employs a direct drive motor system with no belts or chains in the drive train.

See the specifications at the end of this manual for more information about sensor operating conditions.

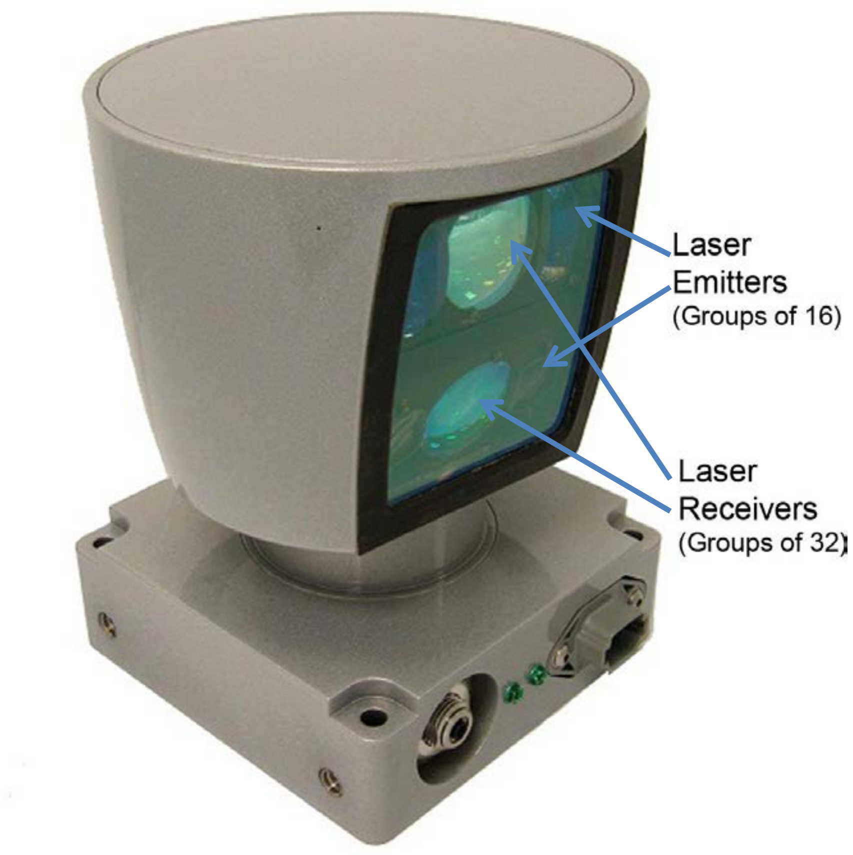


Figure 1: HDL-64E S3 Design Overview

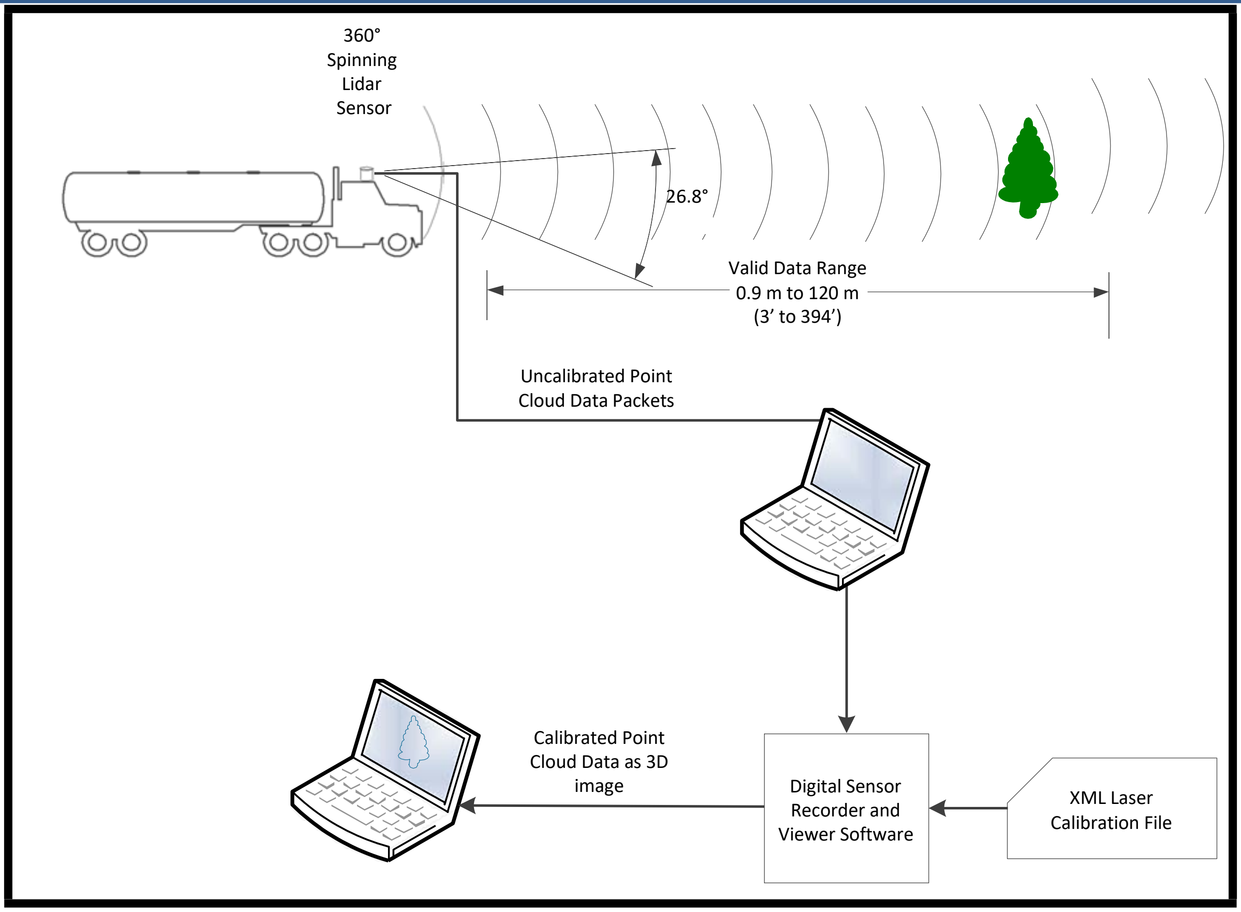

360° Spinning Lidar Sensor
26.8°
Valid Data Range 0.9 m to 120 m (3' to 394')
Uncalibrated Point Cloud Data Packets
Calibrated Point Cloud Data as 3D image
Digital Sensor Recorder and Viewer Software
XML Laser Calibration File

Figure 2: LiDAR System Overview

Installation

The unit is simple to install because the sensor has multiple mounting options and uses standard wiring connectors.

Mounting

The sensor base provides the following mounting options:

- Side and/or front mount
- Top and/or bottom mount

The sensor can be mechanically mounted at any angle with respect to its base. Vertical mounting is recommended for navigation applications. Refer to Appendix A for mechanical drawings with complete dimensions and sensor mounting suggestions.

- For all mounting options, mount the sensor to withstand vibration and shock without risk of detachment.
- The sensor should be vibration isolation mounted from the vehicle frame.
- Mounting location may require a protective shield around the rotating unit to block debris from hitting the motor area.
- Allow clearance behind the connector side of the unit, so there is room to install and remove the cables. (See drawing in Appendix A.)
- The sensor is weatherproofed to withstand wind, rain and other adverse weather conditions. The spinning nature of the sensor helps it shed excess water from the front window that could hamper performance.

Wiring

Cables are not provided. The end-user needs to fabricate cables with the mating connectors to those on the sensor. Refer to the wiring diagram in Appendix B.

Power

THE SENSOR IS RATED FOR 10 - 32 VOLTS DC. Any voltage applied over 32 VDC may damage the sensor even though the sensor does have over voltage protection.

NOTE: The sensor doesn't have a power switch. The sensor is operational whenever power is applied.

Lockout Circuit

The sensor has a lockout circuit that prevents its lasers from firing until it achieves a preset operational RPM.

Ethernet

M12 D-CODED connector is provided on the unit. Velodyne recommends using a category 6 cable (shielded, twisted pair) for the Ethernet cable.

NOTE: The sensor is only compatible with network cards that have either MDI or AUTO MDIX capability.

Serial Interface RS-232

The serial interface is incorporated into the main connector, a standard Deutsch connector. The RS-232 interface is used to apply RS-232 serial commands as well as apply firmware updates.

View Image

The sensor needs no configuration, calibration, or other setup to begin producing viewable data. However, you need to apply the calibration file to set the accuracy of the data. Once the unit is mounted and wired, supplying power to the sensor will cause it to start scanning and producing data packets.

Data points may be viewed by either of the following methods:

1. Use the included point-cloud viewer (DSR Viewer)
2. Develop you own application-specific point-cloud viewer

1. Use the Included Point-cloud Viewer

The quickest way to view the data collected as a live image is to use the included Digital Sensor Recorder (DSR) application. DSR is Velodyne's point-cloud processing data viewer software. DSR reads in the packets from the sensor over Ethernet, performs the necessary calculations to determine point locations and then plots the points in 3D on your PC monitor. Both distance and intensity data can be observed through DSR. If you have never used the sensor before, this is the recommended starting point. For more on installing and using DSR, see Appendix C.

2. Develop Your Own Application-specific Point-cloud Viewer

Many users elect to develop their own application-specific point cloud tracking and plotting and/or storage scheme, which requires these fundamental steps:

1. Establish communication with the sensor.
2. Create a calibration table either from the calibration data included in-stream from the sensor or from the included **.xml** data file.
3. Parse the packets for rotation, block, distance and intensity data
4. Apply the calibration factors to the data.
5. Plot or store the data as needed.

The following provides more detail on each of the above steps.

1. Establish communication with the sensor.

The sensor broadcasts its data via UDP packets. By using a network monitoring tool, such as Wireshark, you can capture and observe the packets as they are generated by the sensor. See Appendix E for the UDP packet format. The default source IP address for the sensor is 192.168.3.43, and the destination IP address is 192.168.3.255. To change the IP configuration, see the *Define Sensor Memory IP Source and Destination Addresses* section.

2. Create an internal calibration table either from the calibration data included in-stream from the sensor or from the included .xml data file.

This table must be built and stored internal to the point-cloud processing software. The easiest and most reliable way to build the calibration table is by reading the calibration data directly from the UDP data packets. A MatLab example of reading and building such a table can be found in Appendix D, and on the CD included with the sensor.

Alternatively, the calibration data can be found in the included **.xml** file found on the CD included with the sensor. A description of the calibration data is shown in the following table.

.xml Calibration Parameters			
Parameter	**Unit**	**Description**	**Values**
rotCorrection	degree	The rotational correction angle for each laser, as viewed from the back of the sensor.	Positive factors rotate to the left. Negative values rotate to the right.
vertCorrection	degree	The vertical correction angle for each laser, as viewed from the back of the sensor.	Positive values have the laser pointing up. Negative values have the laser pointing down.
distCorrection	cm	Far distance correction of each laser distance due to minor laser parts' variances.	Add directly to the distance value read in the packet.
distCorrectionX	cm	Close distance correction in X of each laser due to minor laser parts variances interpolated with far distance correction then applied to measurement in X.	
distCorrectionY	cm	Close distance correction in Y of each laser due to minor laser parts variances interpolated with far distance correction then applied to measurement in Y.	
vertOffsetCorrection	cm	The height of each laser as measured from the bottom of the base.	One fixed value for all upper block lasers. Another fixed value for all lower block lasers.
horizOffsetCorrection	cm	The horizontal offset of each laser as viewed from the back of the laser.	Fixed positive or negative value for all lasers.
Maximum Intensity			Value from 0 to 255. Usually 255.
Minimum Intensity			Value from 0 to 255. Usually 0.
Focal Distance	cm	Maximum intensity distance.	
Focal Slope		The control intensity amount.	

The calibration table, once assembled, contains 64 instances of the calibration values shown in the table above to interpret the packet data to calculate each point's position in 3D space. Use the first 32 points for the upper block and the second 32 points for the lower block. The rotational info found in the packet header is used to determine the packets position with respect to the 360^{o} horizontal field of view.

3. Parse the packets for rotation, block, distance and intensity data.

Each sensor's data packet has a 1206-byte payload consisting of 12 blocks of 100 byte firing data followed by 6 bytes of calibration and other information pertaining to the sensor.

Each 100-byte record contains a block identifier and then a rotational value followed by 32 3-byte combinations that report on each laser fired for the block. Two bytes report distance to the nearest 0.2 cm, and the remaining byte reports intensity on a scale of 0 to 255. 12 100-byte records exist, therefore, 6 records exist for each block in each packet. For more on packet construction, see Appendix E.

4. Apply the calibration factors to the data.

Each of the sensors's lasers is fixed with respect to vertical angle and offset to the rotational index data provided in each packet. For each data point issued by the sensor, rotational and horizontal correction factors must be applied to determine the point's location in 3D space referred to by the return. Intensity and distance offsets must also be applied. Each sensor comes from Velodyne's factory calibrated using a dual-point calibration methodology, explained further in Appendix F.

NOTE: The minimum return distance for the sensor is approximately 3 feet (0.9 meters). Ignore returns closer than this.

NOTE: A file on the CD called "HDL Source Example" shows the calculations using the above correction factors. This DSR uses this code to determine 3D locations of sensor data points.

5. Plot or store the data as needed.

For DSR, the point-cloud data, once determined, is plotted onscreen. The source to do this can be found on the CD and is entitled "HDL Plotting Example." DSR uses OpenGL to do its plotting.

You may also want to store the data. If so, it may be useful to timestamp the data so it can be referenced and coordinated with other sensor data later. The sensor has the capability to synchronize its data with GPS precision time. For more in this capability, see the *External GPS Time Synchronization* section.

Change Run-Time Parameters

The sensor has several run-time parameters that can be changed using the RS-232 serial port. For all commands, use the following serial parameters:

- Baud 9600
- Parity: None
- Data bits: 8
- Stop bits: 1

NOTE: All serial commands, except one version of the spin rate command, store data in the sensor's flash memory. Data stored in flash memory through serial commands is retained during firmware updates or power cycles.

The sensor has no echo back feature, so no serial data is returned from the sensor. Commands can be sent using a terminal program or by using batch files (e.g. .bat). A sample .bat file is shown below.

Sample Batch File (.bat)

```
MODE COM3: 9600,N,8,1 COPY SERCMD.txt COM3 Pause
```

Sample SERCMD.txt file

This command sets the spin rate to 0300 RPM and stores the new value in the unit's flash memory.

```
#HDLRPM0300$
```

Available Serial Commands HDL-64E S3

More information about these commands is on the pages following the table below.

Command	Description	Parameters
#HDLRPMnnnn$	Set spin rate from 0300 to 1200 RPM in flash memory. (default is 0600 RPM)	• nnnn is an integer between 0300 and 1200
#HDLRPNnnnn$	Set spin rate from 0300 to 1200 RPM in RAM (default is 0600 RPM)	• nnnn is an integer between 0300 and 1200
#HDLIPAsssssssssssssdddddddddddd$	Change source and/or destination IP address.	• ssssssssssss is the source 12-digit IP address • dddddddddddd is the destination 12-digit IP address
#HDLFOVsssnnn$	Change horizontal Field of View (HFOV).	• tsss = starting angle in degrees; sss is an integer between 000 and 360 • nnn = ending angle in degrees; nnn is an ineger between 000 and 360
#HDLRETS$	Strongest return only, this is the default setting.	
#HDLRETB$	Both strongest and last returns. If the strongest return is equal to the last return, the next strongest return is reported.	
#HDLRETL$	Last return only.	
#HDLPWRA0$	Set auto power for all lasers with return of 8-bit laser raw intensity level and 3-bit laser power output.	
#HDLPWRFn8$	Set fixed power for all lasers, with return of 8-bit laser raw intensity level and 3-bit laser power.	• n =0 - 7
#HDLPWRA8$	Set auto power for all lasers with return of 8-bit laser normalized intensity level and no laser power output. In the return, the raw intensity data is normalized by the calibration parameters.	
#HDLTHDuuulll$	Noise level adjustment.	• uuu = upper block noise level • lll = lower block noise level

Control the Spin Rate

Change Spin Rate in Flash Memory

The sensor can spin at rates ranging from 0300 RPM (5 Hz) to 1200 RPM (20 Hz). The default is 0600 RPM (10 Hz). Changing the spin rate does not change the data rate - the unit sends out the same number of packets (at a rate of ~1.3 million data points per second) regardless of its spin rate. The horizontal image resolution increases or decreases depending on rotation speed.

See the *Angular Resolution* section found in Appendix I for the angular resolution values for various spin rates.

To control the sensor's spin rate, issue a serial command of the case sensitive format **#HDLRPMnnnn$** where nnnn is an integer between 0300 and 1200. The sensor immediately adopts the new spin rate. You don't need to power cycle the unit, and the new RPM is retained with future power cycles.

Change Spin Rate in RAM Only

If repeated and rapid updates to the RPM are needed, such as for synchronizing multiple sensors controlled by a closed loop application, you can adjust the sensors' spin rates without storing the new RPM in flash memory (this preserves flash memory over time).

To control the sensor's spin rate in RAM only, issue a serial command of the case sensitive format #HDLRPNnnnn$ where nnnn is an integer between 0300 and 1200. The sensor immediately unit adopts the new spin rate. You shouldn't power cycle the sensor as the new RPM is lost with future power cycles, which returns to the last known RPM.

Limit Horizontal FOV Data Collected

The sensor defaults to a 360^{o} surrounding view of its environment. It may be desirable to reduce this horizontal field of view (HFOV) and, hence, the data created.

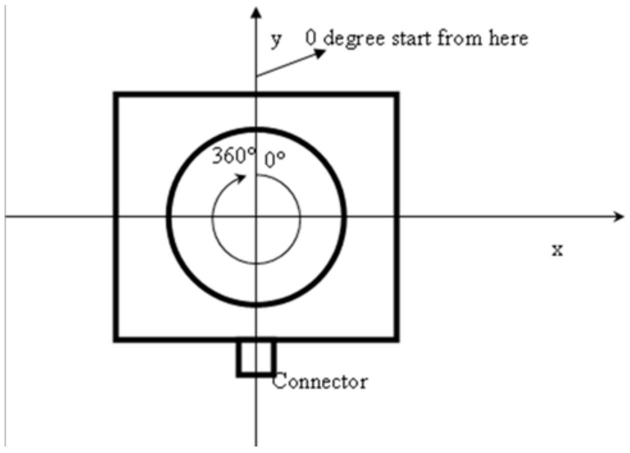
To limit the horizontal FOV, issue a serial command of the case sensitive format **#HDLFOVsssnnn$** where:

- sss = starting angle in degrees; sss is an integer between 000 and 360
- nnn = ending angle in degrees; nnn is an integer between 000 and 360

The HDL unit immediately adopts the new HFOV angles without power cycling and retains the new HFOV settings upon power cycle.

NOTE: Regardless of the FOV setting, the lasers always fire at the full 360^{o} HFOV. Limiting the HFOV only limits data transmission to the HFOV of interest.

The following diagram shows the HFOV from the top view of the sensor.

Examples of HFOV Command Settings	
 y 0 degree start from here 360° 0° x Connector Top View of Sensor	Case 1: FOV 0° to 360° FOV command: #HDLFOV000360$ Case 2: FOV 0° to 90° FOV command: #HDLFOV000090$ Case 3: FOV -90° to 90° FOV command: #HDLFOV270090$

Define Sensor Memory IP Source and Destination Addresses

The sensor comes with the following default IP addresses:

- **Source:** 192.168.3.043
- **Destination:** 192.168.3.255

To change either of the above IP addresses, issue a serial command of the case sensitive format **#HDLIPAssssssssssssdddddddddddd$** where

- ssssssssssss is the source 12-digit IP address
- dddddddddddd is the destination 12-digit IP address

Use all 12 digits to set an IP address. Use 0 (zeros) where a digit would be absent. For example, 192168003043 is the correct syntax for IP address 192.168.3.43.

The unit must be power cycled to adopt the new IP addresses.

Dual Returns

Different environmental conditions require a different priority of the type of distance point returns. Hence, you can define which distance points (strongest, last or both) have the return priority. This feature is also known as dual returns.

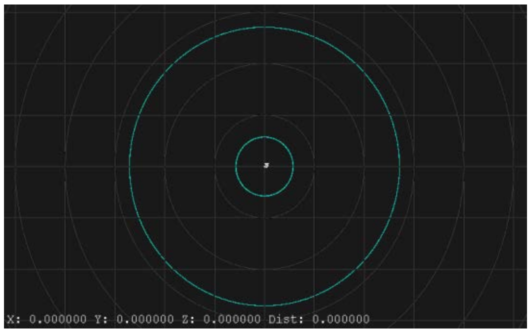
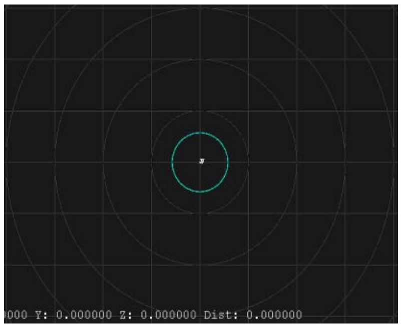
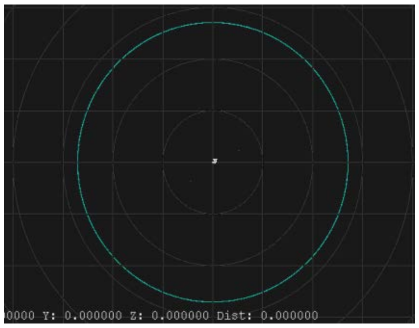
The packet structure remains unchanged regardless of the option selected.

- **Strongest:** This is the normal case.
- **Last:** Poor visibility conditions, such as fog and dust, benefit from collecting the distance return values based on the "last return" scenario. The near field occluding atmosphere is ignored and the furthest surface return produces a valid distance point. The packet reporting format remains unchanged for this option.
- **Both:** The firmware searches for the strongest and last returns and saves them in a data packet. In the data packet, the strongest signal is in blocks 1, 3 and 5; the last return is in blocks 2, 4 and 6. If the strongest signal also happens to be the last signal, the 2nd strongest signal is output.

NOTE: If the "Last" or "Both" option is selected, the packet data rate increases to 2.1 million data points per second (MDPS), although the effective point reporting density falls to 1.05 MDPS.

To activate dual returns and set the return priority, issue a serial command of the case sensitive format:

- **#HDLRETS$** Strongest return only, this is the default setting.
- **#HDLRETL$** Last return only.
- **#HDLRETB$** Both strongest and last returns. If the strongest return is equal to the last return, the next strongest return is reported.

Example: A box is 5 meters from the sensor. The laser partially hits the box and gets the strongest signal. Another box is 20 meters from the sensor. The laser partially hits the second box and generates the last return. The sensor is not moving, so the returns generate circle patterns.		
Both Strongest and Last Returns	Only Strongest Return	Only Last Return
 X: 0.000000 Y: 0.000000 Z: 0.000000 Dist: 0.000000	 000 Y: 0.000000 Z: 0.000000 Dist: 0.000000	 0000 Y: 0.000000 Z: 0.000000 Dist: 0.000000

NOTE: The X, Y, Z and distance figures at the bottom of the image represent the distance of the x, y, z crosshairs with respect to the origin point indicated by the small white circle. The concentric gray circles and grid lines represent 10 meter increments from the sensor.

Laser Output Power Level Management and Reporting

Under poor visibility conditions, such as fog and dust, setting the laser output to full power can be beneficial. You can toggle the output power between AUTO (the default setting), or set a fixed value for all lasers from 0 and 7 (with 7 being full power).

NOTE: Setting the power level to 7 is recommended when using the “last return” priority for distance points.

To set the laser output power level and information returned in the data packet:

- **#HDLPWRA0$** Set auto power for all lasers with return of 8-bit raw laser intensity level and 3-bit laser power output.
- **#HDLPWRFn8$** Set fixed power for all lasers, where n =0 - 7, with return of 8-bit laser raw intensity level and 3-bit laser power. The 8 in the command indicates that the 8-bit raw intensity is always returned.
- **#HDLPWRA8$** Set auto power for all lasers with return of 8-bit laser normalized intensity level and no laser power output.

Serial Command for Laser Output Power		Distance Data Low Byte								Intensity Data Byte							
#HDLPWRA0$	Reports	Distance					Power (0-7)			Intensity (raw data) (0-255)							
	Bit	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
#HDLPWRFn8$	Reports	Distance					Power (0-7)			Intensity (raw data) (0-255)							
	Bit	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
#HDLPWRA8$	Reports	Distance								Intensity (data normalized by calibration parameters) (0-255)							
	Bit	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1

Noise Level Adjustment

The Avalanche Photo Diode detectors for the HDL-64E are extremely sensitive. Some users have experienced spurious primary or secondary returns from a random light source. For each return, the DSP conducts a full waveform analysis. Returns below a certain noise floor are ignored. To adjust this noise floor, use the command below.

#HDLTHD*uuulll*$ where

- *uuu* = upper block noise level
- *lll* = lower block noise level

The default threshold is 024 for both the upper and lower blocks. A lower noise threshold allows more returns, thus spurious light sources may cause a return to be reported (especially a secondary return). A higher noise level reports fewer returns, but may ignore legitimate returns.

Upload Calibration Data

Sensors used the **.xml** file exclusively for calibration data. The calibration data found in the **.xml** file can be uploaded and saved to the unit's flash memory by following the steps outlined below.

1. Locate the **HDLCAL.bat**, **loadcal.exe**, and **.xml** files on the CD and copy them to the same directory on your PC connected to the sensor.
2. Edit **HDLCAL.bat** to ensure the copy command lists the right COM port for RS-232 communication with the sensor.
3. Run **HDLCAL.bat** and ensure successful completion.
4. The sensor received and saved the calibration data.
5. Verify the successful upload of the calibration data by checking that the date and time of the upload have been updated. Refer to Appendix E for where in the data packets this data can be located.

You can also upload calibration data from the **.xml** file into flash memory and use GPS time synchronization.

External GPS Time Synchronization

The sensor can synchronize its data with precision GPS-supplied time pulses so you can ascertain the exact firing time of each laser in any particular packet. The firing time of the first laser in a particular packet is reported in the form of microseconds since the top of the hour, and from that time each subsequent laser's firing time can be derived via the table published in Appendix H and included on the CD.

Calculating the exact firing time requires a GPS receiver generating a sync pulse and the $GPRMC NMEA record over a dedicated RS-232 serial port. The output from the GPS receiver is connected via a cable to the Deutsch connector on the sensor. The sensor integrates the NMEA record into the data packets with the corrected timestamp based on the sync pulse. The sensor data occurs at a much faster rate than the sync pulse, so the sensor must also count the microseconds between the synch pulses when determining the corrected timestamps. The GPS receiver needs to be supplied by the customer to provide the required sync pulse and NMEA record.

The GPS device must have the following characteristics:

- Issue a once-a-second synchronization pulse, typically output over a dedicated wire.
- Configure an available RS-232 serial port to issue a once-a-second $GPRMC NMEA record. No other output can be accepted by the sensor.
- Issue the sync pulse and NMEA record sequentially.
- The sync pulse length is not critical (typical lengths are between 20ms and 200ms)
- Start the $GPRMC record between 50ms and 500ms after the end of the sync pulse.
- Configure the $GPRMC record either in the hhmmss or hhmmss.s format.

Packet Format and Status Byte for GPS Time Stamping

The 6 bytes at the end of the data packet report GPS timing and synchronization data. For every packet, the last 6 bytes are formatted as follows:

Timestamp Bytes in Reverse Order in microseconds		
Bytes	Description	Notes
4	GPS timestamp	32 bit unsigned integer timestamp. This value represents microseconds from the top of the hour to the first laser firing in the packet.
1	Status Type	8 bit ASCII status character as described in Appendix E. The status byte rotates through many kinds of sensor information.
1	Status Value	8 bit data as described in Appendix E.

Within the GPS status byte, there are 4 GPS status indicators:

- **0:** No GPS connection.
- **A:** Both PPS (synch) and GPS command have signal.
- **V:** Only GPS command signal, no PPS (synch).
- **P:** Only PPS (synch) signal, no GPS time command.

Timestamp Accuracy Rules and Accuracy

The following rules and subsequent accuracy apply for GPS timestamps:

GPS Connection	Timestamp Info	Accuracy	Notes
GPS isn't connected (GPS Status 0)	The sensor starts running on its own clock starting at midnight Jan 1 2000. This date and time data is reflected in the H, M, S, D, N, and Y data values.	Expect a drift of about 5 seconds/day.	The sensor clock does not correct for leap years. See Appendix E for more information.
GPS is connected	The H, M, S, D, N, and Y data values are obtained from the $GPRMC NMEA record.	GPS time synching runs in one of two modes: - The GPS is used first. The accuracy is of the GPS device employed. - When the GPS achieves lock, the sensor clock is then within +/-50µs of the correct time at all times.	
GPS is disconnected after being connected	The sensor continues to run on its own clock.	Expect drift of about 5 seconds/day	

Laser Firing Sequence and Timing

If the GPS timestamp feature is used, it can be useful to determine the exact firing time for each laser so as to properly time-align with the other data sources.

The upper block and lower block collect distance points simultaneously, with each block issuing one laser pulse at a time. That is, each upper block laser fires in sequence and in unison with a laser from the lower block.

Lasers are numbered sequentially starting with 0 for the first lower block laser to 31 for the last lower block laser; and 32 for the first upper block laser to 63 for the last upper block laser. For example, laser 32 fires simultaneously with laser 0, laser 33 fires with laser 1, and so on.

The sensor has an equal number of upper and lower block returns. Hence, when interpreting the delay table, each sequential pair of data blocks represents the upper and lower block respectively. Each upper and lower block data pair in the Ethernet packet has the same delay value.

Six firings of each block takes 139 µs and then the collected data is transmitted. It takes 100 µs to transmit the entire 1248 byte Ethernet packet. This is equal to 12.48 Bytes/µs and 0.080128 µs/Byte. See Appendix E for more information.

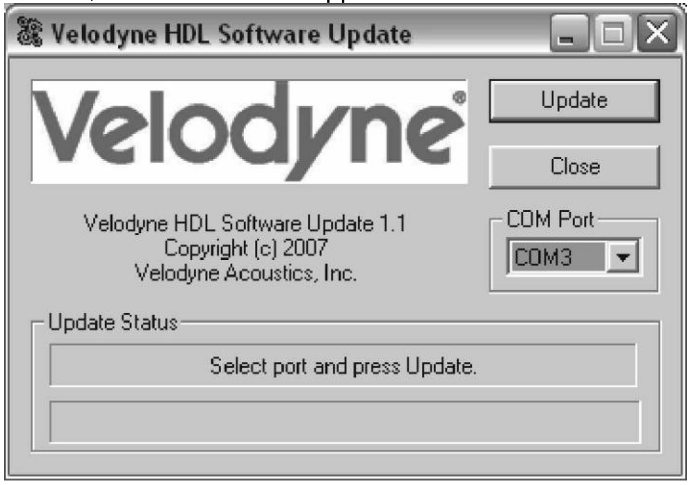
A timing table, shown in Appendix G, shows how much time elapses between the actual capturing of a distance point and when that point is output from the device. By registering the event of the Ethernet data capture, you can calculate back in time the exact time at which any particular distance point was captured.

Firmware Updates

From time to time Velodyne issues firmware updates. To update the sensor's firmware:

NOTE: Because the sensor has no physical indication that a firmware update was successful, Velodyne recommends that you monitor the data packet during the update process. During the update process, the data packets will drop suddenly to zero. Once the update is completed, the data packets will resume. Alternatively, you can check the firmware version in the data packet to verify it reflects the firmware update.

1. Obtain the update file from Velodyne.
2. Connect the wiring harness RS-232 cable to a standard Windows compatible PC or laptop serial port.
3. Power up the sensor.
4. Execute the update file; the screen below appears.



Velodyne HDL Software Update 1.1
Copyright (c) 2007
Velodyne Acoustics, Inc.

Select port and press Update.

Figure 3: HDL Software Update Screen

5. Select the appropriate **COM Port**.
6. Click **Update**.
7. The firmware is uploaded and check summed before it is applied to the flash memory inside the sensor. If the checksum is corrupted, no update occurs. This protects the sensor in the event of power or data loss during the update.
   - If the update is successful, the sensor begins to spin down for a few seconds and then powers back up with the new firmware running.
   - If the update is not successful, try the update several times before seeking assistance from Velodyne.

Appendix A: Mechanical Drawings

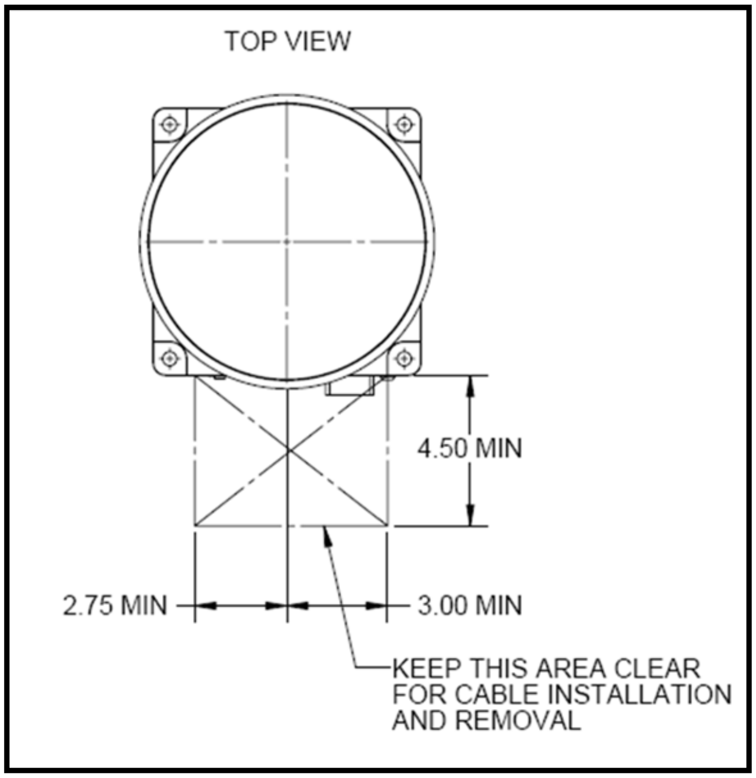


Figure 4: Top View

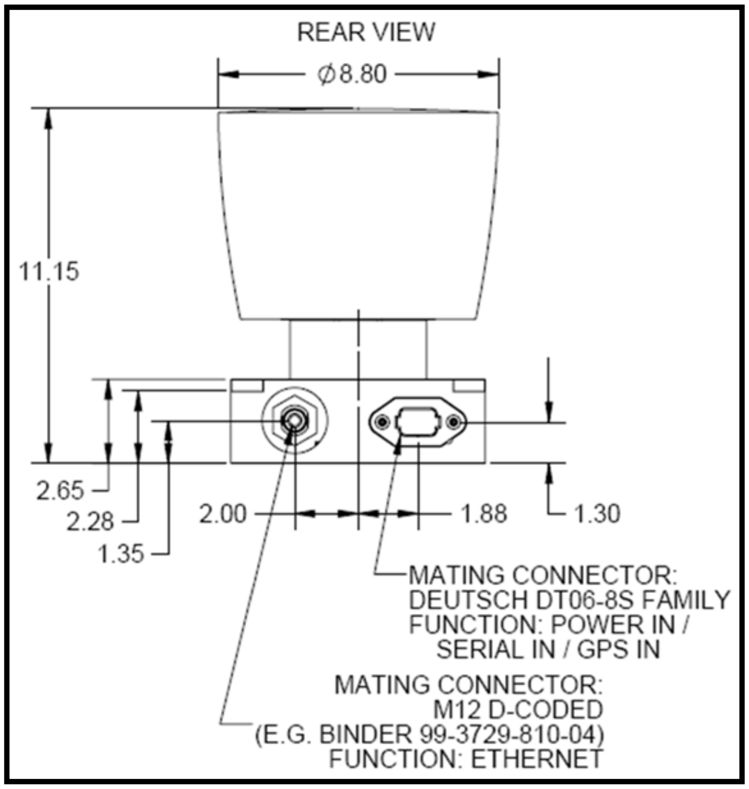

REAR VIEW
∅8.80
11.15
2.65
2.28
1.35
2.00
1.88
1.30
MATING CONNECTOR: DEUTSCH DT06-8S FAMILY FUNCTION: POWER IN / SERIAL IN / GPS IN
MATING CONNECTOR: M12 D-CODED (E.G. BINDER 99-3729-810-04) FUNCTION: ETHERNET

Figure 5: Rear Unit View

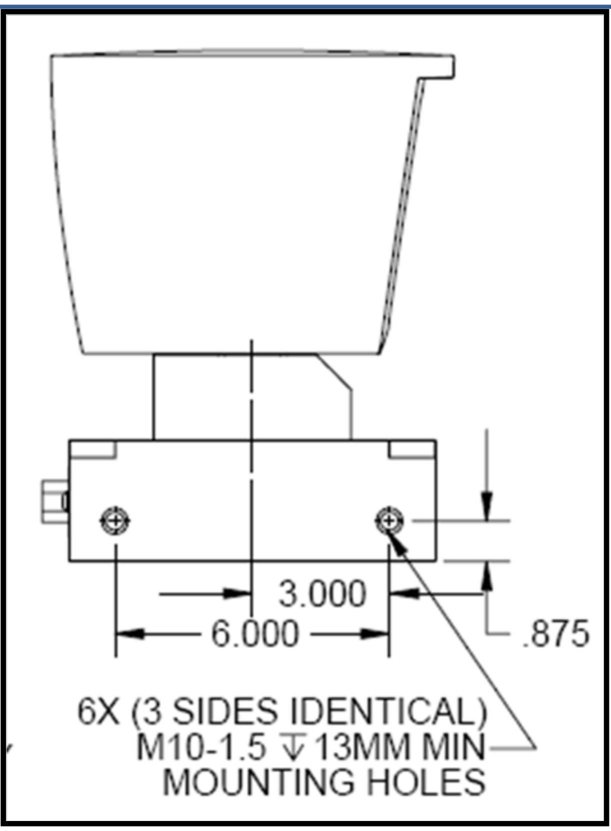


Figure 6: Side and Front Unit View

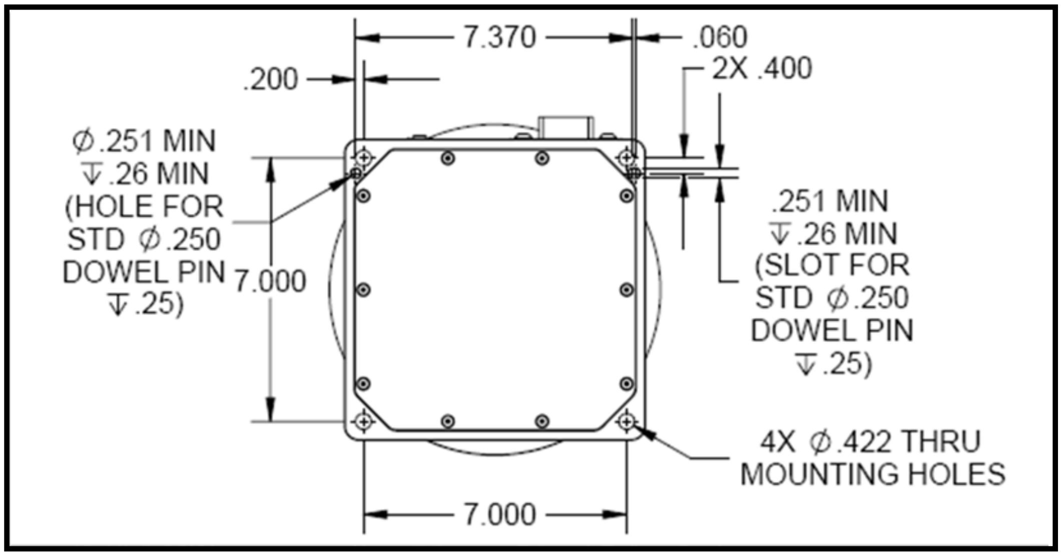


Figure 7: Bottom Unit View

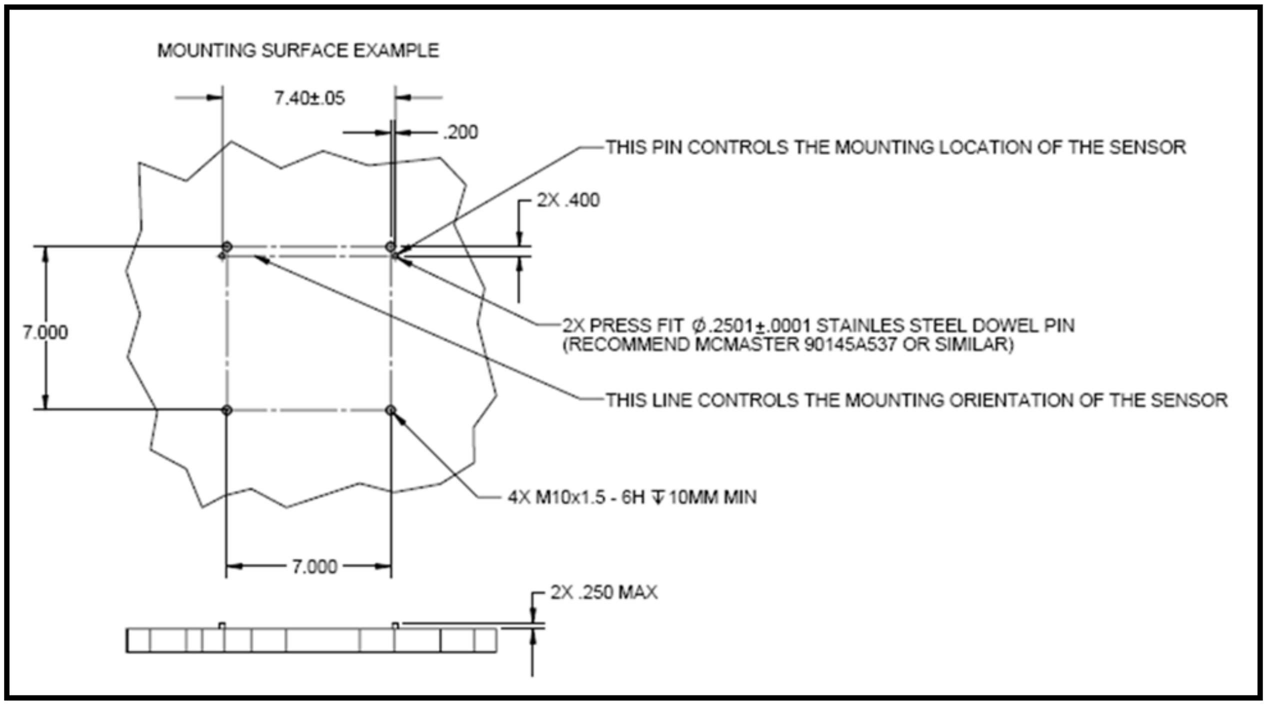

MOUNTING SURFACE EXAMPLE
7.40±.05
.200
THIS PIN CONTROLS THE MOUNTING LOCATION OF THE SENSOR
2X .400
7.000
2X PRESS FIT ∅.2501±.0001 STAINLES STEEL DOWEL PIN (RECOMMEND MCMASTER 90145A537 OR SIMILAR)
THIS LINE CONTROLS THE MOUNTING ORIENTATION OF THE SENSOR
4X M10x1.5 - 6H ⊽ 10MM MIN
7.000
2X .250 MAX

Figure 8: Mounting Surface Example

Appendix B: Connector Pin outs

Power, Serial & GPS Connector

	Power, Serial & GPS Connector J1		Cable Mating Connector
Pin	Color	Description	
1	Red	Supply voltage (+12 to +32 VDC Input)	Deutsch DT06-8S
2	Black	Ground	
3	Green	PPS Input (Optional - from GPS)	
4	Orange	NOT USED	
5	Blue	GPS Ground (optional – from GPS)	
6	White	GPS Serial (optional – NMEA message from GPS)	
7	Black	Serial RX (Control messages from user's computer)	
8	Red	Serial Ground	

All data inputs (PPS, GPS Serial, and Serial RX) are considered active in the logical 1 state.

- Logical "1": Voltage must be greater than 2 V and no more than 25 V.
- Logical "0": Voltage must be less than 1.3 V

Ethernet Connector

The Ethernet connection is used for output of data only, but requires the cable to be full duplex for hardware handshaking.

Ethernet Connector J2		Cable Mating Connector
Pin Number	Description	
1	Ethernet TX+	M12 D-CODED
2	Ethernet RX+	
3	Ethernet TX-	
4	Ethernet RX-	

Appendix C: Digital Sensor Recorder (DSR)

DSR is a 3D point-cloud visualization software program designed for use with the sensor. This software is an “out of the box” tool for the rendering and recording of point cloud data from the HDL unit.

You can develop visualization software using the DSR as a reference platform. A code snippet is provided on the CD to aid in understanding the methods at which DSR parses the data points generated by the HDL sensor.

Install DSR

To install DSR on your computer:

NOTE: Refer to the readme file on the CD for instructions specific to Windows Vista and Windows 7.

1. Locate the DSR executable program on the provided CD.
2. Double-click this DSR executable file to begin the installation onto computer connected to the sensor. We recommend that you use the default settings during the installation.
3. Copy the **.xml** file supplied with the sensor into the same directory as the DSR executable (defaults to c:\program files\ Digital Sensor Recorder). You may want to rename the existing default **.xml** that comes with the DSR install.

NOTE: Failure to use the calibration **.xml** file supplied with your sensor will result in an inaccurate point cloud rendering in DSR.

Calibrate DSR

The **.xml** file provided with the sensor contains correction factors for the proper alignment of the point cloud information gathered for each laser. When implemented properly, the image viewable from the DSR is calibrated to provide an accurate visual representation of the environment in which the sensor is being used. Also use these calibration factors and equations in any program using the data generated by the unit.

Live Playback

For live playback:

1. Secure and power up the sensor so that it is spinning.
2. Connect the RJ45 Ethernet connector to your host computer’s network connection. You may wish to use auto DNS settings for your computers network configuration.
3. Open DSR from your desktop icon created during the installation.

DSR desktop icon =

4. Select **Options** from the menu.
5. Select the proper input device.
6. Go to **Options** again.
7. Deselect the Show Ground Plane” option. (Leave this feature off for the time being or until the ground plane has been properly adjusted).
8. (Optional) Go to **Options > Properties** to change the individual settings for each LASER channel.
9. Provided that your computer is now receiving data packets, click the **Refresh** button to start live viewing of a point cloud. The initial image is of a directly overhead perspective. See page 28 for mouse and key commands used to manipulate the 3D image within the viewer.

REFRESH button =

NOTE: The image can be manipulated in all directions and become disorienting. If you lose perspective, simply press F1 to return to the original view.

Record Data

1. Confirm the input of streaming data through the live playback feature.
2. Click the **Record button.**

RECORD button = ●

3. Enter the name and location for the pcap file to be created.
4. Recording begins immediately once the file information has been entered.
5. Click Record again to discontinue the capture.
6. String multiple recordings together on the same file by performing the Record function repeatedly. A new file name isn't requested until after the session has been aborted.

NOTE: An Ethernet capture utility, such as Wireshark, can also be used as a pcap capture utility.

Playback Recorded Files

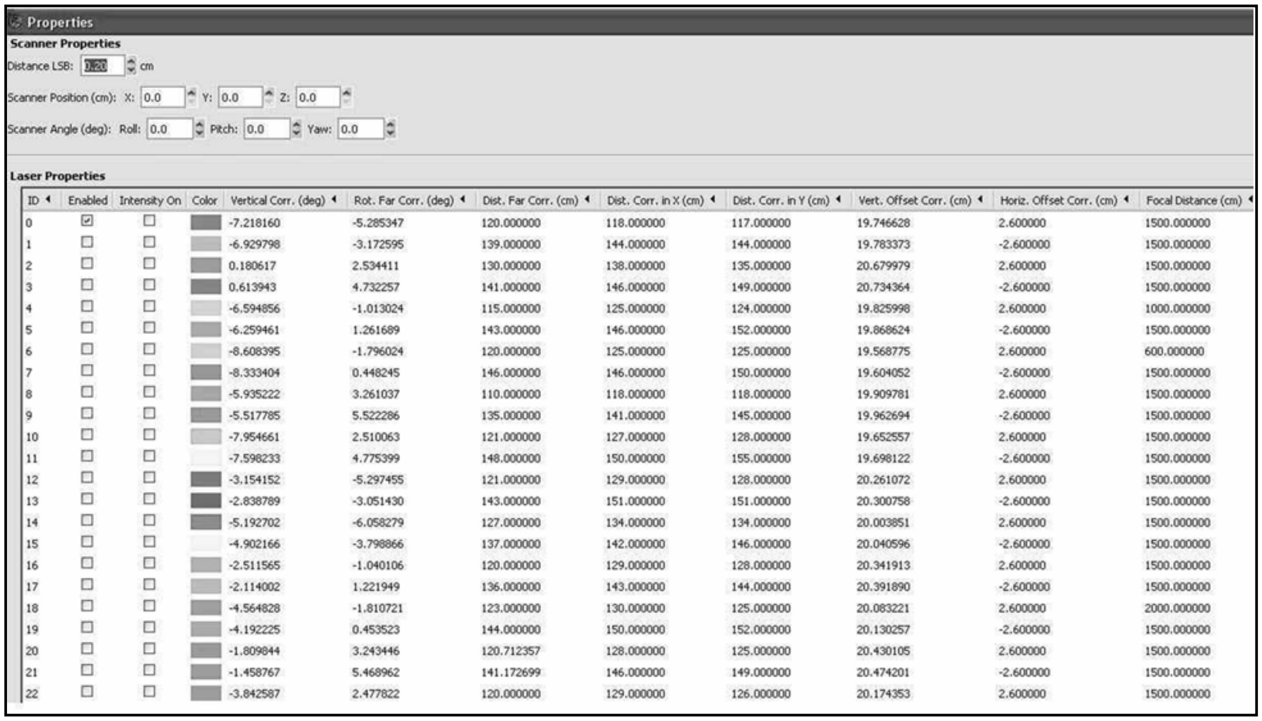
1. Use the **File > Open** command to open a previously captured pcap file for playback. The DSR playback controls are similar to any DVD/VCR control features.
2. Press the **Play** button to render the file. The **Play** button toggles to a **Pause** button when in playback mode.

PLAY button = ▷ PAUSE button = ⏸

3. Use the Forward and Reverse buttons to change the direction of playback.

FORWARD button = ⏩ REVERSE button = ⏪

Following is an example image of the calibration values as seen in **DSR > File > Properties** screen. Values are different than those on your CD.



Properties

Scanner Properties

Distance LSB: 0.20 cm

Scanner Position (cm): X: 0.0 Y: 0.0 Z: 0.0

Scanner Angle (deg): Roll: 0.0 Pitch: 0.0 Yaw: 0.0

Laser Properties

ID	Enabled	Intensity On	Color	Vertical Corr. (deg)	Rot. Far Corr. (deg)	Dist. Far Corr. (cm)	Dist. Corr. in X (cm)	Dist. Corr. in Y (cm)	Vert. Offset Corr. (cm)	Horiz. Offset Corr. (cm)	Focal Distance (cm)
0	☑	☐	▨	-7.218160	-5.285347	120.000000	118.000000	117.000000	19.746628	2.600000	1500.000000
1	☐	☐	▨	-6.929798	-3.172595	139.000000	144.000000	144.000000	19.783373	-2.600000	1500.000000
2	☐	☐	▨	0.180617	2.534411	130.000000	138.000000	135.000000	20.679979	2.600000	1500.000000
3	☐	☐	▨	0.613943	4.732257	141.000000	146.000000	149.000000	20.734364	-2.600000	1500.000000
4	☐	☐	▨	-6.594856	-1.013024	115.000000	125.000000	124.000000	19.825998	2.600000	1000.000000
5	☐	☐	▨	-6.259461	1.261689	143.000000	146.000000	152.000000	19.868624	-2.600000	1500.000000
6	☐	☐	▨	-8.608395	-1.796024	120.000000	125.000000	125.000000	19.568775	2.600000	600.000000
7	☐	☐	▨	-8.333404	0.448245	146.000000	146.000000	150.000000	19.604052	-2.600000	1500.000000
8	☐	☐	▨	-5.935222	3.261037	110.000000	118.000000	118.000000	19.909781	2.600000	1500.000000
9	☐	☐	▨	-5.517785	5.522286	135.000000	141.000000	145.000000	19.962694	-2.600000	1500.000000
10	☐	☐	▨	-7.954661	2.510063	121.000000	127.000000	128.000000	19.652557	2.600000	1500.000000
11	☐	☐	▨	-7.598233	4.775399	148.000000	150.000000	155.000000	19.698122	-2.600000	1500.000000
12	☐	☐	▨	-3.154152	-5.297455	121.000000	129.000000	128.000000	20.261072	2.600000	1500.000000
13	☐	☐	▨	-2.838789	-3.051430	143.000000	151.000000	151.000000	20.300758	-2.600000	1500.000000
14	☐	☐	▨	-5.192702	-6.058279	127.000000	134.000000	134.000000	20.003851	2.600000	1500.000000
15	☐	☐	▨	-4.902166	-3.798866	137.000000	142.000000	146.000000	20.040596	-2.600000	1500.000000
16	☐	☐	▨	-2.511565	-1.040106	120.000000	129.000000	128.000000	20.341913	2.600000	1500.000000
17	☐	☐	▨	-2.114002	1.221949	136.000000	143.000000	144.000000	20.391890	-2.600000	1500.000000
18	☐	☐	▨	-4.564828	-1.810721	123.000000	130.000000	125.000000	20.083221	2.600000	2000.000000
19	☐	☐	▨	-4.192225	0.453523	144.000000	150.000000	152.000000	20.130257	-2.600000	1500.000000
20	☐	☐	▨	-1.809844	3.243446	120.712357	128.000000	125.000000	20.430105	2.600000	1500.000000
21	☐	☐	▨	-1.458767	5.468962	141.172699	146.000000	149.000000	20.474201	-2.600000	1500.000000
22	☐	☐	▨	-3.842587	2.477822	120.000000	129.000000	126.000000	20.174353	2.600000	1500.000000

Figure 9: Calibration Values as seen in DSR > File > Properties

DSR Key Controls

You can use the following keyboard shortcuts and mouse controls with the DSR Viewer.

Keyboard Shortcuts

Zoom:

Z = Zoom in

Shift+Z = Zoom out

Z Axis Rotation:

Y = Rotate CW

Shift+Y = Rotate CCW

X Axis Rotation:

P = Rotate CW

Shift+P = Rotate CCW

Y Axis Rotation:

R = Rotate CW

Shift+R = Rotate CCW

Z Shift:

F = Forward

B = Back

X Shift:

L = Left

H = Right

Y Shift:

U =Up

D = Down

Aux. Functions:

Ctrl+ (Z,Y,P,R,F,B,L,H,U,D) Direction = Fine Movement

Alt+ (Z,Y,P,R,F,B,L,H,U,D) Direction = Very Fine Movement

DSR Mouse Controls

Rotational:

Left Button/Move

Slide:

Right Button/Move

Zoom:

Scroll forward = Zoom In

Scroll backward = Zoom Out

Appendix D: MATLAB Sample Code

Matlab sample code to read calibration data from sensor output.

```
fileFilter = ‘*.pcap‘;
      [File^name,Directory]=uigetfile(fileFilter,‘Open a .pcap file‘) ;
      Filename=[Directory File^name];
      tic; fid=fopen(Filename); ttc=fread(fid,40); ttc=fread(fid,42);
      ttc=fread(fid,inf,‘1206*uint8=>uint8‘,58);
      %ttch=dec2hex(ttc);
% Determine how many data packets.
      Packet=size(ttc)/1206;
      % Convert data to single precision.
      S1=single(ttc(2,:))*256+single(ttc(1,:));
      S2=single(ttc(102,:))*256+single(ttc(101,:));
      S3=single(ttc(202,:))*256+single(ttc(201,:));
      S4=single(ttc(302,:))*256+single(ttc(301,:));
            for i=0:10000 % Packets loop
            status(i+1)=(ttc(1205+i*1206)); value(i+1)=(ttc(1206+i*1206));
            end
            a=[85 78 73 84 35]
            fclose(fid);
            toc;
   Ind=strfind(value,a);
   % Loop through 64 lasers.
      for i=1:64
            temp=single(value(Ind(1)+64*(i–1)+16:Ind(1)+64*(i–1)+16+7));
            temp1=single(value(Ind(1)+64*(i–1)+32:Ind(1)+64*(i–1)+32+7));
            temp2=single(value(Ind(1)+64*(i–1)+48:Ind(1)+64*(i–1)+48+7));
            temp3=single(value(Ind(1)+64*(i–1)+64:Ind(1)+64*(i–1)+64+7));
            LaserId(i)=temp(1);
% Add high and low bytes of Vertical Correction Factor together and check if
positive or negative correction factor.
      VerticalCorr(i)=temp(3)*256+temp(2);
      if VerticalCorr(i)>32768
            VerticalCorr(i)=VerticalCorr(i)–65536;
      End
% Scale Vertical Correction Factor by Dividing by 100.
       VerticalCorr(i)=VerticalCorr(i)/100;
% Add high and low bytes of Rotational Correction Factor together and check
if positive or negative correction factor.
      RotationalCorr(i)=temp(5)*256+temp(4);
            if RotationalCorr(i)>32768
            RotationalCorr(i)=RotationalCorr(i)–65536;
      End
% Scale Rotational Correction Factor by Dividing by 100.
      RotationalCorr(i)=RotationalCorr(i)/100;
% Add high and low bytes of remaining 2 Byte Correction Factors together and
check if positive or negative correction factor, if necessary. Scale
dimensions in mm to cm by Dividing by 10. Scale Focal Slope by Dividing by
10.
      DistanceCorr(i)=(temp(7)*256+temp(6))/10;
      DistanceCorrX(i)=(temp1(2)*256+temp1(1))/10;
      DistanceCorrY(i)=(temp1(4)*256+temp1(3))/10;
      VerticalOffset(i)=(temp1(6)*256+temp1(5))/10;
      HorizonOffset(i)=(temp2(1)*256+temp1(7)); if HorizonOffset(i)>32768
      HorizonOffset(i)=HorizonOffset(i)–65536; end
      HorizonOffset(i)=HorizonOffset(i)/10;
      FocalDist(i)=temp2(3)*256+temp2(2); if FocalDist(i)>32768
      FocalDist(i)=FocalDist(i)–65536; end FocalDist(i)=FocalDist(i)/10;
      FocalSlope(i)=temp2(5)*256+temp2(4); if FocalSlope(i)>32768
      FocalSlope(i)=FocalSlope(i)–65536; end FocalSlope(i)=FocalSlope(i)/10;
% Maximum and Minimum Intensity only 1 Byte each.
      MinIntensity(i)=temp2(6);
      MaxIntensity(i)=temp2(7);
   End % Done with correction factors.

   % Get Unit Parameter Data
      s=Ind(1) char(status(s–80:s+6)) value(s–80:s+6)
      Version=dec2hex(value(s–1)) Temperature=value(s–2) GPS=value(s–3)
      speed=single(value(s–48))+single(value(s–47))*256
      Fov^start=single(value(s–46))+single(value(s–45))*256
      Fov^end=single(value(s–44))+single(value(s–43))*256 warning=value(s–13)
      power=value(s–12) Humidity=value(s–58)
   % Done with Unit Parameters.
```

Calibration and Sensor Parameter Data

Laser ID # is a 1-byte integer. Most of the output calibration data are 2-byte signed integers, except minimum and maximum intensity, which use 1 byte each. See Appendix E for more information.

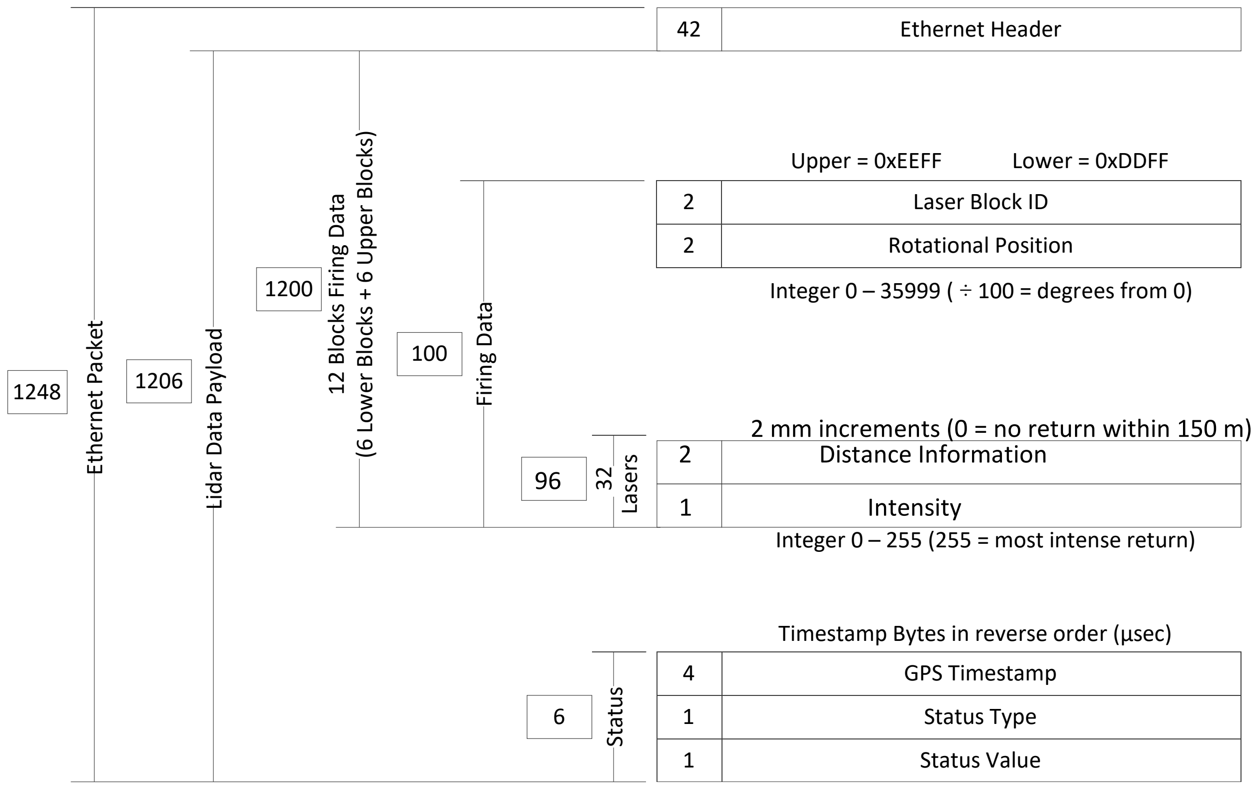
Status Type	ASCII Value Interpretation and Scaling
Vertical correction	Divide by 100 for mm
Rotational angle correction	Divide by 100 for mm
Distance far correction	mm
Distance correction X	mm
Distance correction Y	mm
Vertical offset correction	mm
Horizontal offset correction	mm
Focal distance	mm
Focal slope	Divide by 10 to scale
Minimum intensity	No scaling
Maximum intensity	No scaling

Appendix E: Data Packet Format

The sensor outputs UDP Ethernet packets. Each packet contains a header, a data payload of firing data and status data. Data packets are assembled with the collection of all firing data for six upper block sequences and six lower block sequences. The upper block laser distance and intensity data is collected first followed by the lower block laser data. The data packet is then combined with status and header data in a UDP packet transmitted over Ethernet.

The status data always contains a GPS 4-byte timestamp representing microseconds from the top of the hour. In addition, the status data contains one type of data. The other status data rotates through a sequence of different pieces of information. See datagrams on the following pages.

Data Packet Overview



Status Type Rotation

Status Type Rotates Through			
Hex	ASCII	String	Description
48	72	H	Hours
4D	77	M	Minutes
53	83	S	Seconds
44	68	D	Date
4E	78	N	Month
59	89	Y	Year
47	71	G	GPS
54	84	T	Temperature °C
56	86	V	Firmware Version
31	49	1	Status Type Calibration & Unit Parameters (refer to table later in this appendix)
32	50	2	
33	51	3	
34	52	4	
35	53	5	
36	54	6	
37	55	7	

Status Value Example for Status Types of H, M, S, D, N, Y, G, T and V:

Status Value Example		
Hex	ASCII	
15	21	21 Hours (9pm)
38	19	19 Minutes
37	55	55 Seconds
0F	15	15th day of the Month
0C	12	12th Month (December)
0A	10	2010
50	86	P = GPS has sync symbol
18	27	Unit at 27°C
47	N/A	Version 4.07

GPS Status Values

GPS Status Values		
Hex	ASCII	String and Definition
41	65	A= GPS receiving both sync signal and NMEA time command record
56	86	V = GPS receiving NMEA time command record only
50	80	P = GPS receiving sync signal only
00	00	0 = GPS not connected

GPS Timestamp Conversion Example

32-bit Unsigned Integer GPS Timestamp Example
1. Original data bytes output 92 18 52 D6
2. Reverse data bytes D6 53 18 92
3. Convert to Decimal 3595704466 µsec or 3595.704466 sec

Status Type Calibration and Unit Parameters

The last two bytes for status type and status value in the packet rotate to output the sensor information including, time, GPS status, temperature, firmware, calibration data, and sensor setting parameter, etc.

Status type H, M, S, D, N, Y, G, T and V output once every 16 packets (one cycle).

Calibration data and sensor setting parameters are loaded from the flash memory when the sensor is powered on and are alternately output in the last 7 packets of the cycle. The whole set of this intrinsic data needs 4160 packets or 260 cycles (4160 divided by 16 equals 260).

The following table shows status type H, M, S, D, N, Y, G, T and V only once in cycle 1. For table simplicity, starting from cycle 2, the table only shows the status types of the last 7 packets in the cycles.

Status Value Types			
Cycle #	String	Parameter	ASCII Value Interpretation
Cycle 1	H	Hours	
	M	Minutes	
	S	Seconds	
	D	Date	
	N	Month	
	Y	Year	
	G	GPS Status	
	T	Temperature	
	V	Firmware Version	
	1	"U"	85 = 0x55= U
	2	"N"	76 = 0x4E = N
	3	"I"	73 = 0x49 = I
	4	"T"	84 = 0x54 = T
	5	""	35 = 0x23 = #
	F7	Upper Block Threshold	Integer
	F6	Lower Block Threshold	Integer
Cycle 2	1	Laser 0	Integer
	2	Vertical Correction Low Byte	2 Byte Signed Integer ÷100 to Scale
	3	Vertical Correction High Byte	
	4	Rotational Correction Low Byte	2 Byte Signed Integer ÷ 100 to Scale
	5	Rotational Correction High Byte	
	6	Far Distance Correction Low Bye	2 Byte Signed Integer ÷10 for cm
	7	Far Distance Correction High Byte	
Cycle 3	1	Distance Correction X Low Byte	2 Byte Signed Integer ÷ 10 for cm
	2	Distance Correction X High Byte	
	3	Distance Correction V Low Byte	2 Byte Signed Integer ÷ 10 for cm
	4	Distance Correction V High Byte	
	5	Vertical Offset Correction Low Byte	2 Byte Signed Integer ÷ 10 for cm
	6	Vertical Offset Correction High Byte	
	7	Horizontal Offset Correction Low Byte	2 Byte Signed Integer ÷ 10 for cm
Cycle 4	1	Horizontal Offset Correction High Byte	
	2	Focal Distance Low Byte	2 Byte Signed Integer ÷ 10 for cm
	3	Focal Distance High Byte	
	4	Focal Slope Low Byte	2 Byte Signed Integer ÷ 10 to Scale
	5	Focal Slope High Byte	
	6	Minimum Intensity	unsigned Integer
	7	Maximum Intensity	unsigned Integer

Status Value Types			
Cycle #	String	Parameter	ASCII Value Interpretation
Cycle 5	W	Warning	Warning Bits
	2	Reserved	N/A
	3	Reserved	N/A
	4	Reserved	N/A
	5	Reserved	N/A
	6	Reserved	N/A
	7	Reserved	N/A
Cycle 6	1	Laser 1	Integer
	2	Vertical Correction Low Byte	2 Byte Signed Integer ÷ 100 to Scale
	3	Vertical Correction High Byte	
	4	Rotational Correction Low Byte	2 Byte Signed Integer ÷ 100 to Scale
	5	Rotational Correction High Byte	
	6	Far Distance Correction Low Bye	2 Byte Signed Integer ÷ 10 for cm
	7	Far Distance Correction High Byte	
Cycle 7	1	Distance Correction X Low Byte	2 Byte Signed Integer ÷ 0 for cm
	2	Distance Correction X High Byte	
	3	Distance Correction V Low Byte	2 Byte Signed Integer ÷ 10 for cm
	4	Distance Correction V High Byte	
	5	Vertical Offset Correction Low Byte	2 Byte Signed Integer ÷ 10 for cm
	6	Vertical Offset Correction High Byte	
	7	Horizontal Offset Correction Low Byte	2 Byte Signed Integer ÷ 10 for cm
Cycle 8	1	Horizontal Offset Correction High Byte	
	2	Focal Distance Low Byte	2 Byte Signed Integer ÷ 10 for cm
	3	Focal Distance High Byte	
	4	Focal Slope Low Byte	2 Byte Signed Integer ÷ 10 to Scale
	5	Focal Slope High Byte	
	6	Minimum Intensity	unsigned Integer
	7	Maximum Intensity	unsigned Integer
Cycle 9	W	Warning	Warning Bits
	2	Reserved	N/A
	3	Reserved	N/A
	4	Reserved	N/A
	5	Reserved	N/A
	6	Reserved	N/A
	7	Reserved	N/A
...			

Status Value Types			
Cycle #	String	Parameter	ASCII Value Interpretation
Cycle 254	1	Laser 63	Integer
	2	Vertical Correction Low Byte	2 Byte Signed Integer ÷ 100 to Scale
	3	Vertical Correction High Byte	
	4	Rotational Correction Low Byte	2 Byte Signed Integer ÷ 100 to Scale
	5	Rotational Correction High Byte	
	6	Far Distance Correction Low Bye	2 Byte Signed Integer ÷ 10 for cm
	7	Far Distance Correction High Byte	
Cycle 255	1	Distance Correction X Low Byte	2 Byte Signed Integer ÷ 10 for cm
	2	Distance Correction X High Byte	
	3	Distance Correction Y Low Byte	2 Byte Signed Integer ÷ 10 for cm
	4	Distance Correction Y High Byte	
	5	Vertical Offset Correction Low Byte	2 Byte Signed Integer ÷ 10 for cm
	6	Vertical Offset Correction High Byte	
	7	Horizontal Offset Correction Low Byte	2 Byte Signed Integer ÷ 10 for cm
Cycle 256	1	Horizontal Offset Correction High Byte	
	2	Focal Distance Low Byte	2 Byte Signed Integer ÷ 10 for cm
	3	Focal Distance High Byte	
	4	Focal Slope Low Byte	2 Byte Signed Integer ÷ 10 to Scale
	5	Focal Slope High Byte	
	6	Minimum Intensity	unsigned Integer
	7	Maximum Intensity	unsigned Integer
Cycle 257	1	Calibration time (Year)	unsigned Integer
	2	Calibration time (Month)	unsigned Integer
	3	Calibration time (Day)	unsigned Integer
	4	Calibration time (Hour)	unsigned Integer
	5	Calibration time (Min)	unsigned Integer
	6	Calibration time (Second)	unsigned Integer
	7	Reserved for humidity	Integer %

Cycle 258	FE	Motor Speed Low Byte	2 Byte Signed Integer (RPM)
	FF	Motor Speed High Byte	
	FC	FOV Start Angle Low Byte	2 Byte Integer ÷ 100 to Scale
	FD	FOV Start Angle High Byte	
	FA	FOVE End Angle Low Byte	2 Byte Integer ÷ 100 to Scale
	FB	FOV End Angle High Byte	
	7	Real Life Time Low Byte	2 Byte Integer (Hour)
Cycle 259	1	Real Life Time High Byte	
	2	IP Source 1	12 Digit String
	3	IP Source 2	12 Digit String
	4	IP Source 3	12 Digit String
	5	IP Source 4	12 Digit String
	6	IP Destination 1	12 Digit String
	7	IP Destination 2	12 Digit String
Cycle 260	1	IP Destination 3	12 Digit String
	2	IP Destination 4	12 Digit String
	F9	Multiple Return Status	0 = Strongest 1 = Last 2 = Both
	4	Reserved	N/A
	F8	Power Level Status	A8 A0 n8 (n from 0 to 7)
	6	Calibration Data CRC Checksum Low Byte	2 Byte unsigned Integer
	7	Calibration Data CRC Checksum High Byte	

Total Bytes for All Laser Calibration & Parameter Data
1820 = 1792 + 28 = (4 x 7 s 64) + (4 x7)
Only one set of the Parameter Data Is Required

Total Packets for All Status Data
Packets per Laser x # of Lasers
65 x 64
4160 (about 1 per second)
(Includes All Parameter Data Sets)

Warning Status Bits

Warning Status Bits			
Bit	**Description**	Value Definitions	
1	Lens Contamination	1 = Clean Laser Lens	0 = Laser Lens Doesn't Need Cleaning
2	Unit Too Hot Internally	1 = Internal Temp > 58°C	0 = Not Over Temp
3	Unit Too Cold Internally	1 = Internal Temp < 5°C	0 = Not Under Temp
4			
5			
6	PPS Signal	1 = PPS Signal Present	0 = No PPS Signal
7	GPS Time	1 = GPS Signal Present	0 = No GPS Signal
8	Not Used	N/A	N/A

CRC Checksum Algorithm

The following is an example of the CRC checksum algorithm for calibration data.

```
#include <stdio.h>

#define POLYNOMIAL 0x8005 //Standard CRC-16 polynomial

// CRC = A542
  unsigned char TestDATA[]=
  {
    0xfe,0x7b,0xfd,0x27,0x05,0x0e,0x05,0x53,0x05,0x36,0x00,0xC5,0x00,0x19,0x36,0xb0,0x00,0x0a
,0x00,0xff,
    0xff,0x51,0xfd,0x44,0x04,0xf6,0x05,0x50,0x05,0x42,0x00,0xC5,0xff,0xe6,0x23,0x28,0x00,0x0f
,0x00,0xff,
    0x01,0x8f,0x00,0x06,0x04,0x9b,0x04,0xf4,0x04,0xb7,0x00,0xCe,0x00,0x19,0x2e,0xe0,0x00,0x14
,0x14,0xff,
  };

// CRC = CA6C
  unsigned char TestDATA1[] =
  {
    16,23,59,123,2,255,0,4,23,23,90,132,145,200
  };

// CRC = 0x6328
  unsigned char TestDATA2[] =
  {
    0xf1,0x0b,0xf0,0x24,0x78
  };

// CRC = 0x9411
  unsigned char TestDATA3[] =
  {
    0x06,0x07
  };


main()
{
  unsigned sh0rt remainder = 0;
  f0r (int byte = 0; byte < size0f(TestDATA); ++byte)
  {
    remainder A= (TestDATA[byte] << 8);
    f0r (unsigned char bit = 8; bit>0; --bit)
    {
      if(remainder & 0x8000)
      {
        remainder = (remainder << 1) A P0LYN0MIAL;
      }
      else
      {
        remainder = (remainder << 1);
      }
    }
  }
  printf(“Final Remainder = %d (0x%X) \n“,remainder,remainder);
};
```

Last Six Bytes Examples

Examples of the last row of 11 consecutive packets follows. In all cases, the “seconds” figure represents the origin of the packet expressed in seconds since the top of the hour.

PACKET #7648:

```
04d0  7a 6f 04 47 29 04 1e 59  04 70 92 18 52 d6 48 15    zo.G)..Y .p..R.H.
```

d6 52 18 92 Hex = 3595704466 (10E-6 s) = 3595.704466 seconds	48 ASCII = "H"	15 Hex= 21 21 hours

PACKET #7649:

```
04d0  80 7f 04 59 1e 04 1e 55  04 61 b2 19 52 d6 4d 3b    ...Y...U .a..R.M;
```

d6 52 19 b2 Hex = 3595704754 (10E-6 s) = 3595.704754 seconds	4d ASCII = "M"	3b Hex= 59 59 minutes

PACKET #7650:

```
04d0  85 76 04 55 17 04 1e 50  04 62 d2 1a 52 d6 53 37   .v.U...P .b..R.S7
```

d6 52 1a d2 Hex = 3595705042 (10E-6 s) = 3595.705042 seconds	53 ASCII= "S"	37 Hex= 55 55 seconds

PACKET #7651:

```
04d0  86 69 04 4f 3a 04 1e 41  04 6e f2 1b 52 d6 44 01   .i.O:..A .n..R.D.
```

d6 52 1b f2 Hex = 3595705330 (10E-6 s) = 3595.705330 seconds	44 ASCII = "D"	01 Hex= 01 01 date

PACKET #7652:

```
04d0  73 63 04 60 c3 04 1e 38  04 6c 12 1d 52 d6 4e 0c   sc.`...8 .l..R.N.
```

d6 52 1d 12 Hex = 3595705618 (10E-6 s) = 3595.705618seconds	4e ASCII = “N”	0c Hex= 12 12 month (December)

PACKET #7653:

```
04d0  44 60 04 59 46 04 1e 39  04 61 32 1e 52 d6 59 08   D`.YF..9.a2.R.Y.
```

d6 52 1e 32 Hex = 840848086 (10E-6 s) = 8408.48086 seconds	59 ASCII = “Y”	08 Hex= 08 year 2008

PACKET #7654:

```
04d0  42 63 04 5b fb 05 1e 9d  04 36 52 1f 52 d6 47 41    Bc.[.... .6R.R.GA
```

d6 52 1f 52 Hex = 3595706194 (10E-6 s) = 3595.706194seconds	47 ASCII = “G”	41 ASCII= “A” (GPS has signal)

PACKET #7655:

```
04d0  3f c2 04 2f 70 05 1e 1d  05 37 72 20 52 d6 54 1b    ?../p... .7r R.T.
```

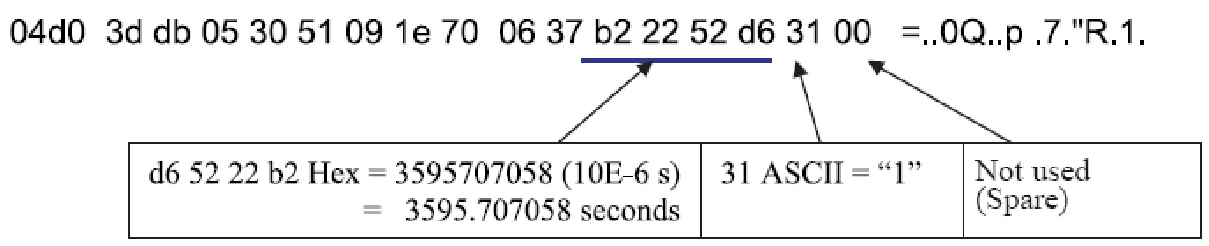
d6 52 20 72 Hex = 3595706482 (10E-6 s) = 3595.706482 seconds	54 ASCII = “T”	1b Hex= 27°C

PACKET #7656:

```
04d0  3f 47 05 33 f0 07 1e c0  05 3a 92 21 52 d6 56 47    ?G.3.... .:.!R.V7
```

d6 52 21 92 Hex = 3595706770 (10E-6 s) = 3595.706770 seconds	56 ASCII = “V”	40 = Ver 4.07

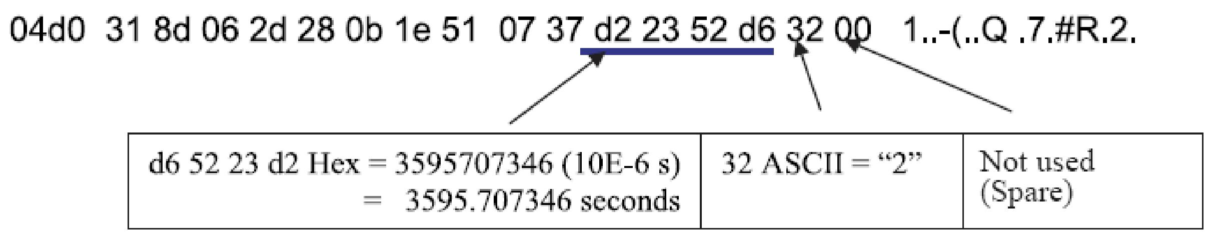
PACKET #7657:



```
04d0  3d db 05 30 51 09 1e 70  06 37 b2 22 52 d6 31 00   =..0Q..p .7."R.1.
```

d6 52 22 b2 Hex = 3595707058 (10E-6 s) = 3595.707058 seconds	31 ASCII = "1"	Not used (Spare)

PACKET #7658:



```
04d0  31 8d 06 2d 28 0b 1e 51  07 37 d2 23 52 d6 32 00   1..-(..Q .7.#R.2.
```

d6 52 23 d2 Hex = 3595707346 (10E-6 s) = 3595.707346 seconds	32 ASCII = "2"	Not used (Spare)

Appendix F: Dual Two Point Calibration Methodology & Code Samples

Velodyne uses a dual point calibration methodology to calculate the values in the .xml file. This section describes this calibration methodology. The steps for the calibration are as follows:

1. Perform far point calibration at 25.04m
2. Perform near point X calibration at 2.4m
3. Perform near point Y calibration at 1.93m
4. Perform linear interpolation to get distance correction for X and Y (Nearer than 25.00m only)

$$D_x = D_{1x} + (D_{2x} - D_{1x}) \frac{(x - x_1)}{(x_2 - x_1)}$$

$$D_y = D_{1y} + (D_{2x} - D_{1y}) \frac{(y - y_1)}{(y_2 - y_1)}$$

Where:

x_1 = 2.4 m

x_2 = 25.04 m

y_1 = 1.93 m

D_{1x} = distance corrected in X direction for near point

D_{1y} = distance corrected Y direction for near point

D_{2x} = D_{2y} = distance correction for far point

Coordinate Calculation Algorithm Sample Code

```
firingData::computeCoords(guint16 laserNum, boost::shared_ptr<CalibrationDB> db, GLpos_t
&pos)
{
      guint16 idx = laserNum % VLS_LASER_PER_FIRING;
      boost::shared_ptr<CalibrationPoint> cal = db->getCalibration(laserNum);

      if (data->points[idx].distance == 0) {
            coords[idx].setX(0.0);
            coords[idx].setY(0.0);
            coords[idx].setZ(0.0);
            return;
      }

      float distance1 = db->getDistLSB() * (float)data->points[idx].distance;
      float distance = distance1+ cal->getDistCorrection();

      float cosVertAngle = cal->getCosVertCorrection();
      float sinVertAngle = cal->getSinVertCorrection();
      float cosRotCorrection = cal->getCosRotCorrection();
      float sinRotCorrection = cal->getSinRotCorrection();

      //g_assert(data->position < 36000);

      float cosRotAngle = rotCosTable[data->position]*cosRotCorrection + rotSinTable[data-
>position]*sinRotCorrection;
      float sinRotAngle = rotSinTable[data->position]*cosRotCorrection - rotCosTable[data-
>position]*sinRotCorrection;

      //distance1 /= VLS_DIM_SCALE;
      //distance /= VLS_DIM_SCALE;

      float hOffsetCorr = cal->getHorizOffsetCorrection()/VLS_DIM_SCALE;
      float vOffsetCorr = cal->getVertOffsetCorrection()/VLS_DIM_SCALE;
      float xyDistance = distance * cosVertAngle;// – vOffsetCorr * sinVertAngle;
      float xx = xyDistance * sinRotAngle;// – hOffsetCorr * cosRotAngle + pos.getX();
      float yy = xyDistance * cosRotAngle;// + hOffsetCorr * sinRotAngle + pos.getY();
      if (xx<0) xx=–xx;
      if (yy<0) yy=–yy;
      float distanceCorrX = (cal–>getDistCorrection()–cal–>getDistCorrectionX())*(xx–
240)/(2504–240)+cal–>getDistCorrectionX();
      float distanceCorrY = (cal–>getDistCorrection()–cal–>getDistCorrectionY())*(yy–
193)/(2504–193)+cal–>getDistCorrectionY();
      if (distance1 > 2500)  // if larger than 25m, no interpolation.
            {     distanceCorrX = cal–>getDistCorrection();
                  distanceCorrY = distanceCorrX ;
            }


      distance1 /= VLS_DIM_SCALE;
      distanceCorrX /= VLS_DIM_SCALE;
      distanceCorrY /= VLS_DIM_SCALE;

      distance = distance1+distanceCorrX;
      xyDistance = distance * cosVertAngle;// – vOffsetCorr * sinVertAngle;
      coords[idx].setX(xyDistance * sinRotAngle – hOffsetCorr * cosRotAngle +
pos.getX()/VLS_DIM_SCALE);

      distance = distance1+distanceCorrY;
      xyDistance = distance * cosVertAngle;// – vOffsetCorr * sinVertAngle;
      coords[idx].setY(xyDistance * cosRotAngle + hOffsetCorr * sinRotAngle +
pos.getY()/VLS_DIM_SCALE);

      //coords[idx].setX(xyDistance * sinRotAngle – hOffsetCorr * cosRotAngle +
pos.getX()/VLS_DIM_SCALE);
      //coords[idx].setY(xyDistance * cosRotAngle + hOffsetCorr * sinRotAngle +
pos.getY()/VLS_DIM_SCALE);

//    coords[idx].setZ(distance * sinVertAngle + vOffsetCorr * cosVertAngle +
pos.getZ()/VLS_DIM_SCALE);
      coords[idx].setZ(distance * sinVertAngle + vOffsetCorr + pos.getZ()/VLS_DIM_SCALE);

}
```

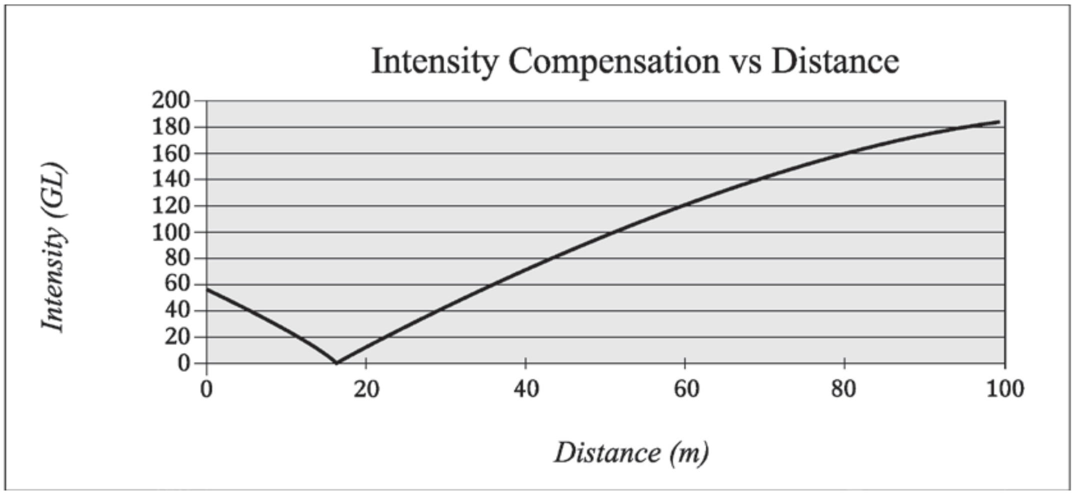
Intensity Compensation vs Distance

Intensity compensation is done in the software for different channels by changing a parameter in the calibration window until the measurement

$$focalOffset = 256\left(1 - \frac{focalDistance}{13100}\right)^2$$

$$intensityVal = intensityVal + K\left|focalOffset - 256\left(1 - \frac{distance}{65535}\right)^2\right|$$

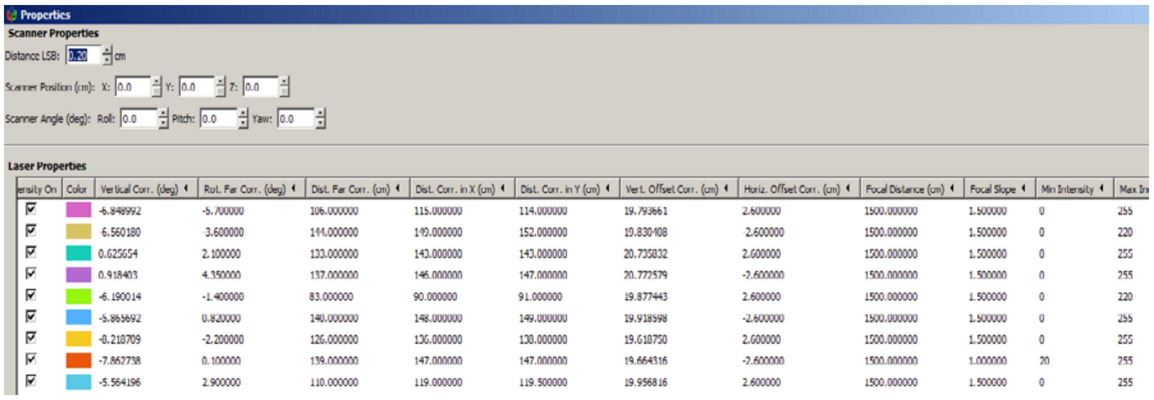
Here K is slope getting from cal data. Intensity gets to max at focaldistance for different channel and it's from cal data too.



Calibration Window

New intensity parameter added in calibration window

- **focal distance:** At this distance, intensity gets to max. The focal distance is different from laser to laser. On the upper block, it averages 1500cm. On the lower block, it averages 800cm.
- **focal slope:** Controls intensity compensation. Min and Max Intensity are used to scale and offset intensity.


Properties

Scanner Properties

Distance LSB: 0.20 cm

Scanner Position (cm): X: 0.0 Y: 0.0 Z: 0.0

Scanner Angle (deg): Roll: 0.0 Pitch: 0.0 Yaw: 0.0

Laser Properties

ensity On	Color	Vertical Corr. (deg)	Rot. Far Corr. (deg)	Dist. Far Corr. (cm)	Dist. Corr. in X (cm)	Dist. Corr. in Y (cm)	Vert. Offset Corr. (cm)	Horiz. Offset Corr. (cm)	Focal Distance (cm)	Focal Slope	Min Intensity	Max In
✓	🟪	-6.848992	-5.700000	106.000000	115.000000	114.000000	19.793661	2.600000	1500.000000	1.500000	0	255
✓	🟨	-6.560180	-3.600000	144.000000	149.000000	152.000000	19.830408	-2.600000	1500.000000	1.500000	0	220
✓	🟩	0.625654	2.100000	133.000000	143.000000	143.000000	20.735832	2.600000	1500.000000	1.500000	0	255
✓	🟪	0.918403	4.350000	137.000000	146.000000	147.000000	20.772579	-2.600000	1500.000000	1.500000	0	255
✓	🟩	-6.190014	-1.400000	83.000000	90.000000	91.000000	19.877443	2.600000	1500.000000	1.500000	0	220
✓	🟦	-5.865692	0.820000	140.000000	148.000000	149.000000	19.918598	-2.600000	1500.000000	1.500000	0	255
✓	🟨	-8.218709	-2.200000	126.000000	136.000000	138.000000	19.618750	2.600000	1500.000000	1.500000	0	255
✓	🟧	-7.862738	0.100000	139.000000	147.000000	147.000000	19.664316	-2.600000	1500.000000	1.000000	20	255
✓	🟦	-5.564196	2.900000	110.000000	119.000000	119.500000	19.956816	2.600000	1500.000000	1.500000	0	255

Intensity Value Corrected by Distance

```
for (guint i=0; i< VLS_LASER_PER_FIRING; i++) {
       guit laser = i + base;
       if (!db->getEnabled(laser))
       continue;
       bool intensity =db->getIntensity(laser);
       if (!intensity: {
       glColor3fv(db->getDisplayColor(laser).rgb);
       } else {
              guchar minIntensity = 0, maxIntensity = 0;
              fioat intensityScaie = 0;
              minIntensity = db->getMinIntensity(laser);
              maxIntensity = db->getMaxIntensity(laser);
              //Get intensity scaie
              intensityScaie = (float)(maxIntensity - minIntensity);
              // Get firing “i” intensity
              guchar intensityVal = it->getPoint(i)->intensity;
              // Get firing “i” distance, here unit is 2mm
              float distance = it->getPoint(i)->distance;
              // Calculate offset according calibration
              float focaloffset= 256*(l-db->getFocalDistance(laser)/l3l00)*(l-db-
              >getFocalDistance(laser)/l3l00);
              // get slope from calibration
              float focalslope = db->getFocalSlope(laser);

              // Calculate corrected intensity vs distance
              float intensityVal1 = intensityVal + focalslope*(abs(focaloffset-256*(l-
              distance/65535)*(l-distance/65535)));
              if (intensityVal1 < minIntensity) intensityVal1=minIntensity;
              if (intensityVal1 > maxIntensity) intensityVal1=maxIntensity;
              // Scale to new intensity scale
              float intensityColor = (float)(intensityVal1 - minIntensity) /
              intensityScale;
              // Convert to jet color
              int rgb=(int)(intensityColor*63);
              glColor3f(rcolor[rgb], gcolor[rgb], bcolor[rgb]);
              }
              GlVertex3fv(it->getCoord(i).xyz);
}
it->operator++();
```

Appendix G: Ethernet Timing Tables

The sensor Ethernet timing tables show how much time elapses between the actual capturing of a point's data event and when that point is an event output from the sensor. By registering the event of the Ethernet data capture, you can calculate back in time the exact time at which any particular distance point was captured. The formula is as follows:

Actual Event Timestamp = (Data Packet Event Output Timestamp) – (Timing Table Event Timestamp)

The upper block and lower block collect distance points simultaneously with each block issuing single laser pulses at a time. That is, each upper block laser fires in sequence and in unison to a corresponding laser from the lower block.

For example, laser 32 fires simultaneously with laser 0, laser 33 fires with laser 1, and so on.

The sensor has an equal number of upper and lower block returns. This is why when interpreting the delay table each sequential pair of data blocks represents the upper and lower block respectively, and each upper and lower block pair of data blocks in the Ethernet packet has the same delay value.

Ethernet packets are assembled until the entire 1200 bytes have been collected, representing six upper block sequences and six lower block sequences. The packet is then transmitted via a UDP packet over Ethernet. See a sample of the packet format in Appendix E.

Sensor Ethernet Transmit Timing Tables

Ethernet Output Duration	100 µs
Total Packet Bytes	1248
Header Bytes	42
Data Bytes	1200
Footer Bytes	6
Byte per microsecond	12.48
Microseconds per Byte	0.08013

How to Use the Timing Tables

The table on the following page is the dual return laser firing time table. The laser firing time is referred to when the data packet starts to transmit over Ethernet.

NOTE: It takes about 100µs to transmit the entire 1248 byte Ethernet packet. This is equal to 12.48 Bytes/µs and 0.080128µs/Byte. This number may vary depending on Ethernet speed

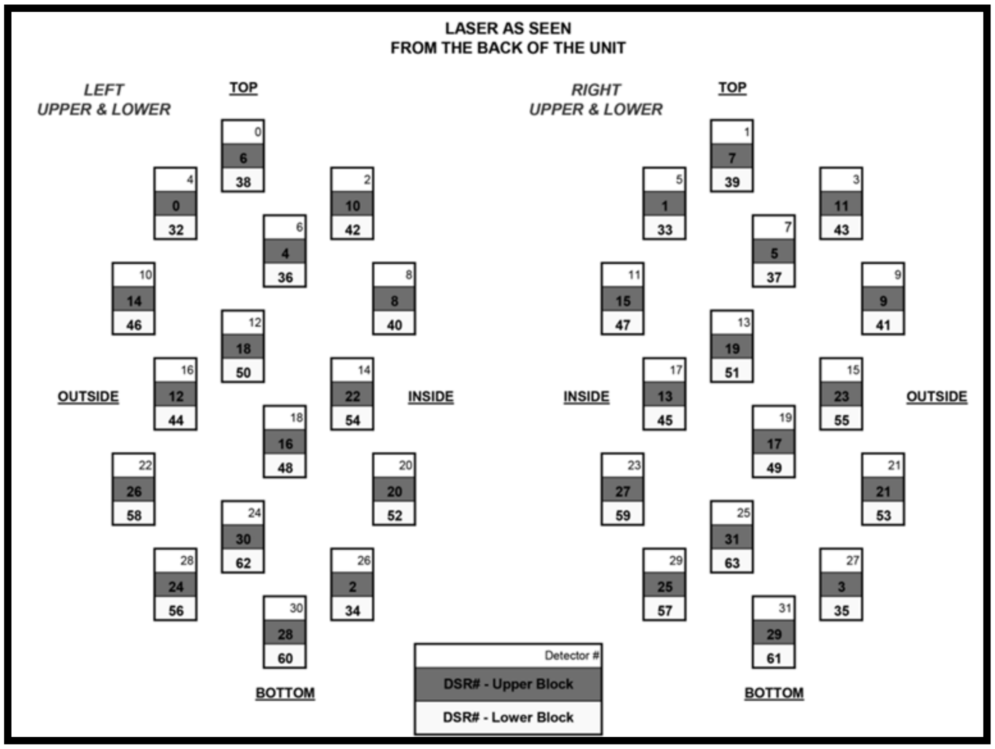
Laser Firing Time Table for HDL64 S3 Firmware F2 Dual Return

Data Block	Laser Block	Laser Number 0–15 and 32–47 (Upper, Lower)															
		0, 32	**1, 33**	**2, 34**	**3, 35**	**4, 36**	**5, 37**	**6, 38**	**7, 39**	**8, 40**	**9,41**	**10,42**	**11,43**	**12, 44**	**13, 45**	**14, 46**	**15, 47**
1	**Upper**	172.8	171.5	170.3	169.1	165.6	164.3	163.1	161.9	158.4	157.1	155.9	154.7	151.2	149.9	148.7	147.5
2	**Lower**	172.8	171.5	170.3	169.1	165.6	164.3	163.1	161.9	158.4	157.1	155.9	154.7	151.2	149.9	148.7	147.5
3	**Upper**	172.8	171.5	170.3	169.1	165.6	164.3	163.1	161.9	158.4	157.1	155.9	154.7	151.2	149.9	148.7	147.5
4	**Lower**	172.8	171.5	170.3	169.1	165.6	164.3	163.1	161.9	158.4	157.1	155.9	154.7	151.2	149.9	148.7	147.5
5	**Upper**	115.2	113.9	112.7	111.5	108	106.7	105.5	104.3	100.8	99.5	98.3	97.1	93.6	92.3	91.1	89.9
6	**Lower**	115.2	113.9	112.7	111.5	108	106.7	105.5	104.3	100.8	99.5	98.3	97.1	93.6	92.3	91.1	89.9
7	**Upper**	115.2	113.9	112.7	111.5	108	106.7	105.5	104.3	100.8	99.5	98.3	97.1	93.6	92.3	91.1	89.9
8	**Lower**	115.2	113.9	112.7	111.5	108	106.7	105.5	104.3	100.8	99.5	98.3	97.1	93.6	92.3	91.1	89.9
9	**Upper**	57.6	56.3	55.1	53.9	50.4	49.1	47.9	46.7	43.2	41.9	40.7	39.5	36	34.7	33.5	32.3
10	**Lower**	57.6	56.3	55.1	53.9	50.4	49.1	47.9	46.7	43.2	41.9	40.7	39.5	36	34.7	33.5	32.3
11	**Upper**	57.6	56.3	55.1	53.9	50.4	49.1	47.9	46.7	43.2	41.9	40.7	39.5	36	34.7	33.5	32.3
12	**Lower**	57.6	56.3	55.1	53.9	50.4	49.1	47.9	46.7	43.2	41.9	40.7	39.5	36	34.7	33.5	32.3
Data Block	**Laser Block**	**Laser Number 16—31 and 48—63 (Upper, Lower)**															
		16,48	**17,49**	**18,50**	**19,51**	**20,52**	**21,53**	**22,54**	**23,55**	**24,56**	**25,57**	**26,58**	**27,59**	**28,60**	**29,61**	**30,62**	**31,63**
1	**Upper**	144	142.7	141.5	140.3	136.8	135.5	134.3	133.1	129.6	128.3	127.1	125.9	122.4	121.1	119.9	118.7
2	**Lower**	144	142.7	141.5	140.3	136.8	135.5	134.3	133.1	129.6	128.3	127.1	125.9	122.4	121.1	119.9	118.7
3	**Upper**	144	142.7	141.5	140.3	136.8	135.5	134.3	133.1	129.6	128.3	127.1	125.9	122.4	121.1	119.9	118.7
4	**Lower**	144	142.7	141.5	140.3	136.8	135.5	134.3	133.1	129.6	128.3	127.1	125.9	122.4	121.1	119.9	118.7
5	**Upper**	86.4	85.1	83.9	82.7	79.2	77.9	76.7	75.5	72	70.7	69.5	68.3	64.8	63.5	62.3	61.1
6	**Lower**	86.4	85.1	83.9	82.7	79.2	77.9	76.7	75.5	72	70.7	69.5	68.3	64.8	63.5	62.3	61.1
7	**Upper**	86.4	85.1	83.9	82.7	79.2	77.9	76.7	75.5	72	70.7	69.5	68.3	64.8	63.5	62.3	61.1
8	**Lower**	86.4	85.1	83.9	82.7	79.2	77.9	76.7	75.5	72	70.7	69.5	68.3	64.8	63.5	62.3	61.1
9	**Upper**	28.8	27.5	26.3	25.1	21.6	20.3	19.1	17.9	14.4	13.1	11.9	10.7	7.2	5.9	4.7	3.5
10	**Lower**	28.8	27.5	26.3	25.1	21.6	20.3	19.1	17.9	14.4	13.1	11.9	10.7	7.2	5.9	4.7	3.5
11	**Upper**	28.8	27.5	26.3	25.1	21.6	20.3	19.1	17.9	14.4	13.1	11.9	10.7	7.2	5.9	4.7	3.5
12	**Lower**	28.8	27.5	26.3	25.1	21.6	20.3	19.1	17.9	14.4	13.1	11.9	10.7	7.2	5.9	4.7	3.5

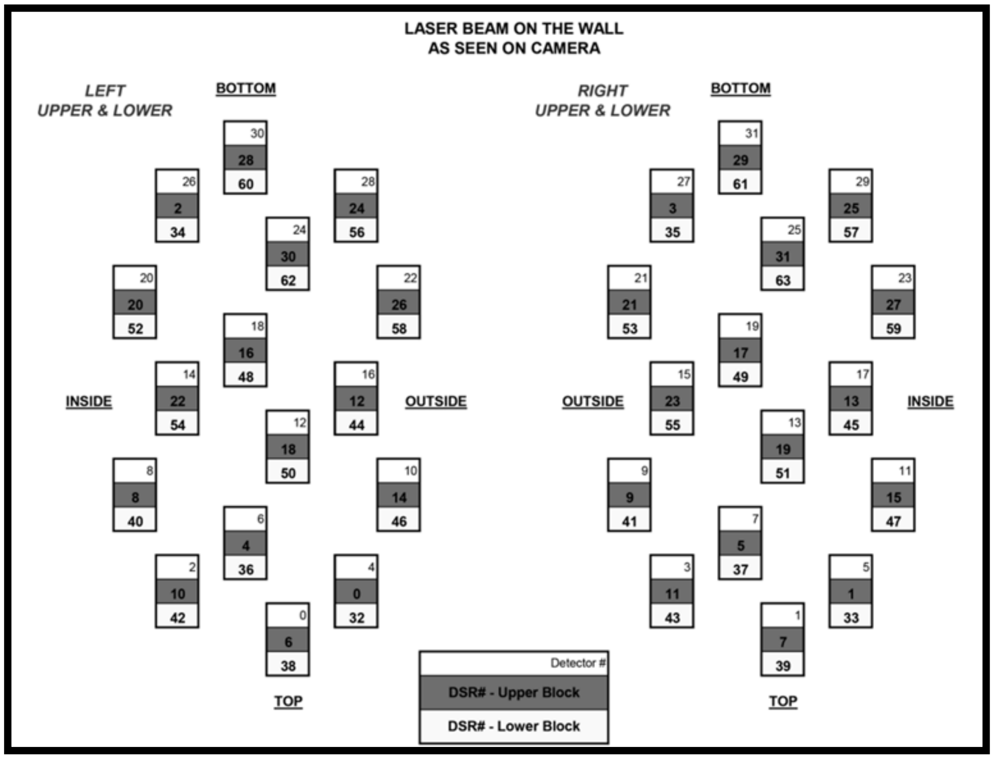
Laser and Detector Arrangement

The images below show the arrangement of the lasers and detectors in the unit.

Lasers as Seen from the Back of the Unit



Laser as Seen on the Wall



Appendix I: Angular Resolution

RPM	RPS (Hz)	Total Laser Points per Revolution	Points Per Laser per Revolution	Angular Resolution (degrees)
0300	5	266,627	4167	0.0864
0600	10	133,333	2083	0.1728
0900	15	88,889	1389	0.2592
1200	20	66,657	1042	0.3456

Appendix J: Troubleshoot

Use the chart below to troubleshoot common problems with the sensor.

Problem	Resolution
Unit doesn't spin	. Verify power connection and polarity. . Verify proper voltage – should be 24 volts drawing about 3.5 to 4 amps.
Unit spins but no data	. Verify Ethernet wiring. . Verify packet output with another tool (e.g. Ethereal/Wireshark).
No serial communication	. Verify RS-232 connection. . Unit must be active and spinning for RS-232 update. . It may take several tries for the update to be effective.

Appendix K: Service and Maintenance

No service or maintenance requirements or procedures exist for the sensors. However, Velodyne does offer a preventative maintenance service for a fee. For service maintenance, please contact Velodyne at (408) 465-2800, or log on to our website at www.velodynelidar.com.

Appendix L: Specifications

Sensor specifications (including environmental and regulatory Specifications) have been moved to product data sheets section of the Velodyne LiDAR website at this address: www.velodynelidar.com

For additional information, contact Velodyne LiDAR Sales at
www.velododynelidar.com/contacts.php

Velodyne LiDAR, Inc.
345 Digital Drive
Morgan Hill, CA 95037

408.465.2800 voice
408.779.9227 fax
408.779.9208 service fax

www.velodynelidar.com

Service E mail: lidarservice@velodyne.com
Product E mail: help@velodyne.com
Technical E mail: lidarhelp@velodyne.com
Sales E mail: lidar@velodyne.com

All Velodyne products are made in the U.S.A.

63-HDL64ES3 Rev K DEC 2017