

YDLIDAR SDK

V1.4.6

Generated by Doxygen 1.8.11

Contents

1	CYdLidar(YDLIDAR SDK API)	1
2	YDLidarDriver	3
3	README	5
4	Namespace Index	11
4.1	Namespace List	11
5	Class Index	13
5.1	Class List	13
6	File Index	15
6.1	File List	15
7	Namespace Documentation	17
7.1	angles Namespace Reference	17
7.1.1	Function Documentation	18
7.1.1.1	find_min_max_delta(double from, double left_limit, double right_limit, double &result_min_delta, double &result_max_delta)	18
7.1.1.2	from_degrees(double degrees)	18
7.1.1.3	normalize_angle(double angle)	18
7.1.1.4	normalize_angle_positive(double angle)	18
7.1.1.5	shortest_angular_distance(double from, double to)	18
7.1.1.6	shortest_angular_distance_with_limits(double from, double to, double left_limit, double right_limit, double &shortest_angle)	19
7.1.1.7	to_degrees(double radians)	19

7.1.1.8	<code>two_pi_complement(double angle)</code>	19
7.2	impl Namespace Reference	19
7.2.1	Function Documentation	20
7.2.1.1	<code>getCurrentTime()</code>	20
7.2.1.2	<code>getHDTimer()</code>	20
7.3	serial Namespace Reference	20
7.3.1	Detailed Description	20
7.3.2	Enumeration Type Documentation	22
7.3.2.1	<code>bytesize_t</code>	22
7.3.2.2	<code>flowcontrol_t</code>	22
7.3.2.3	<code>parity_t</code>	22
7.3.2.4	<code>stopbits_t</code>	23
7.3.3	Function Documentation	23
7.3.3.1	<code>is_standardbaudrate(unsigned long baudrate, speed_t &baud)</code>	23
7.3.3.2	<code>list_ports()</code>	23
7.3.3.3	<code>set_common_props(termios *tio)</code>	23
7.3.3.4	<code>set_databits(termios *tio, serial::bytesize_t databits)</code>	23
7.3.3.5	<code>set_flowcontrol(termios *tio, serial::flowcontrol_t flowcontrol)</code>	23
7.3.3.6	<code>set_parity(termios *tio, serial::parity_t parity)</code>	23
7.3.3.7	<code>set_stopbits(termios *tio, serial::stopbits_t stopbits)</code>	23
7.3.3.8	<code>timespec_from_ms(const uint32_t millis)</code>	23
7.4	ydlidar Namespace Reference	23
7.4.1	Enumeration Type Documentation	24
7.4.1.1	<code>YDLIDAR_MODLES</code>	24
7.4.1.2	<code>YDLIDAR_RATE</code>	25
7.4.2	Function Documentation	25
7.4.2.1	<code>ConvertLidarToUserSmample(int model, int rate)</code>	25
7.4.2.2	<code>ConvertUserToLidarSmample(int model, int m_SampleRate, int defaultRate)</code>	25
7.4.2.3	<code>fileExists(const std::string filename)</code>	25
7.4.2.4	<code>hasIntensity(int model)</code>	25

7.4.2.5	hasSampleRate(int model)	26
7.4.2.6	hasScanFrequencyCtrl(int model)	26
7.4.2.7	hasZeroAngle(int model)	26
7.4.2.8	init(int argc, char *argv[])	26
7.4.2.9	isOctaveLidar(int model)	26
7.4.2.10	isOldVersionTOFLidar(int model, int Major, int Minor)	27
7.4.2.11	isSerialNumbValid(const LaserDebug &info)	27
7.4.2.12	isSupportLidar(int model)	27
7.4.2.13	isSupportMotorCtrl(int model)	27
7.4.2.14	isSupportScanFrequency(int model, double frequency)	27
7.4.2.15	isTOFLidar(int type)	28
7.4.2.16	isValidSampleRate(std::map< int, int > smap)	28
7.4.2.17	isValidValue(uint8_t value)	28
7.4.2.18	isVersionValid(const LaserDebug &info)	28
7.4.2.19	lidarModelDefaultSampleRate(int model)	28
7.4.2.20	lidarModelToString(int model)	28
7.4.2.21	ok()	28
7.4.2.22	ParseLaserDebugInfo(const LaserDebug &info, device_info &value)	28
7.4.2.23	shutdownNow()	28
8	Class Documentation	29
8.1	cmd_packet Struct Reference	29
8.1.1	Member Data Documentation	29
8.1.1.1	cmd_flag	29
8.1.1.2	data	29
8.1.1.3	size	29
8.1.1.4	syncByte	29
8.2	CYdLidar Class Reference	30
8.2.1	Detailed Description	32
8.2.2	Constructor & Destructor Documentation	35
8.2.2.1	CYdLidar()	35

8.2.2.2	~CYdLidar()	35
8.2.3	Member Function Documentation	35
8.2.3.1	CalculateSampleRate(int count, double scan_time)	35
8.2.3.2	checkCalibrationAngle(const std::string &serialNumber)	35
8.2.3.3	checkCOMMs()	36
8.2.3.4	checkHardware()	36
8.2.3.5	checkLidarAbnormal()	36
8.2.3.6	checkSampleRate()	36
8.2.3.7	checkScanFrequency()	36
8.2.3.8	checkStatus()	36
8.2.3.9	disconnecting()	36
8.2.3.10	doProcessSimple(LaserScan &outscan, bool &hardwareError)	37
8.2.3.11	getAngleOffset() const	37
8.2.3.12	getDeviceHealth()	37
8.2.3.13	getDeviceInfo()	37
8.2.3.14	getHardwareVersion() const	37
8.2.3.15	getSerialNumber() const	37
8.2.3.16	getSoftVersion() const	37
8.2.3.17	handleDeviceInfoPackage(int count)	37
8.2.3.18	handleSingleChannelDevice()	37
8.2.3.19	initialize()	37
8.2.3.20	isAngleOffsetCorrected() const	38
8.2.3.21	isRangeIgnore(double angle) const	38
8.2.3.22	isRangeValid(double reading) const	38
8.2.3.23	parsePackageNode(const node_info &node, LaserDebug &info)	38
8.2.3.24	printfVersionInfo(const device_info &info)	38
8.2.3.25	PropertyBuilderByName(float, MaxRange, private) ; PropertyBuilderBy↵ Name(float	39
8.2.3.26	PropertyBuilderByName(float, MaxAngle, private)	39
8.2.3.27	PropertyBuilderByName(float, MinAngle, private)	40
8.2.3.28	PropertyBuilderByName(int, SampleRate, private)	40

8.2.3.29	PropertyBuilderByName(float, ScanFrequency, private)	41
8.2.3.30	PropertyBuilderByName(bool, FixedResolution, private)	41
8.2.3.31	PropertyBuilderByName(bool, Reversion, private)	41
8.2.3.32	PropertyBuilderByName(bool, Inverted, private)	42
8.2.3.33	PropertyBuilderByName(bool, AutoReconnect, private)	42
8.2.3.34	PropertyBuilderByName(int, SerialBaudrate, private)	42
8.2.3.35	PropertyBuilderByName(int, AbnormalCheckCount, private)	43
8.2.3.36	PropertyBuilderByName(std::string, SerialPort, private)	43
8.2.3.37	PropertyBuilderByName(std::vector< float >, IgnoreArray, private)	44
8.2.3.38	PropertyBuilderByName(float, OffsetTime, private)	44
8.2.3.39	PropertyBuilderByName(bool, SingleChannel, private)	44
8.2.3.40	PropertyBuilderByName(int, LidarType, private)	45
8.2.3.41	turnOff()	45
8.2.3.42	turnOn()	45
8.2.4	Member Data Documentation	46
8.2.4.1	defalutSampleRate	46
8.2.4.2	frequencyOffset	46
8.2.4.3	global_nodes	46
8.2.4.4	isScanning	46
8.2.4.5	last_node_time	46
8.2.4.6	lidar_model	46
8.2.4.7	lidarPtr	46
8.2.4.8	m_AngleOffset	46
8.2.4.9	m_FixedSize	46
8.2.4.10	m_isAngleOffsetCorrected	46
8.2.4.11	m_lidarHardVer	46
8.2.4.12	m_lidarSerialNum	46
8.2.4.13	m_lidarSoftVer	46
8.2.4.14	m_ParseSuccess	46
8.2.4.15	m_PointTime	46

8.2.4.16	<code>m_UserSampleRate</code>	46
8.2.4.17	<code>Major</code>	46
8.2.4.18	<code>Minjor</code>	46
8.2.4.19	<code>MinRange</code>	46
8.2.4.20	<code>private</code>	46
8.2.4.21	<code>SampleRateMap</code>	46
8.3	<code>device_health</code> Struct Reference	47
8.3.1	Member Data Documentation	47
8.3.1.1	<code>error_code</code>	47
8.3.1.2	<code>status</code>	47
8.4	<code>device_info</code> Struct Reference	47
8.4.1	Member Data Documentation	47
8.4.1.1	<code>firmware_version</code>	47
8.4.1.2	<code>hardware_version</code>	48
8.4.1.3	<code>model</code>	48
8.4.1.4	<code>serialnum</code>	48
8.5	Event Class Reference	48
8.5.1	Member Enumeration Documentation	49
8.5.1.1	anonymous enum	49
8.5.2	Constructor & Destructor Documentation	49
8.5.2.1	<code>Event(bool isAutoReset=true, bool isSignal=false)</code>	49
8.5.2.2	<code>~Event()</code>	49
8.5.3	Member Function Documentation	49
8.5.3.1	<code>release()</code>	49
8.5.3.2	<code>set(bool isSignal=true)</code>	49
8.5.3.3	<code>wait(unsigned long timeout=0xFFFFFFFF)</code>	49
8.5.4	Member Data Documentation	49
8.5.4.1	<code>_cond_catr</code>	49
8.5.4.2	<code>_cond_locker</code>	49
8.5.4.3	<code>_cond_var</code>	49

8.5.4.4	_is_signalled	49
8.5.4.5	_isAutoReset	49
8.6	function_state Struct Reference	49
8.6.1	Member Data Documentation	50
8.6.1.1	state	50
8.7	LaserConfig Struct Reference	50
8.7.1	Detailed Description	50
8.7.2	Member Function Documentation	51
8.7.2.1	operator=(const LaserConfig &data)	51
8.7.3	Member Data Documentation	51
8.7.3.1	angle_increment	51
8.7.3.2	max_angle	51
8.7.3.3	max_range	51
8.7.3.4	min_angle	51
8.7.3.5	min_range	51
8.7.3.6	scan_time	51
8.7.3.7	time_increment	51
8.8	LaserDebug Struct Reference	51
8.8.1	Member Data Documentation	52
8.8.1.1	MaxDebugIndex	52
8.8.1.2	W1F6GNoise_W1F5SNoise_W1F4MotorCtl_W4F0SnYear	52
8.8.1.3	W2F5Output2K4K5K_W5F0Date	52
8.8.1.4	W3F4BoradHardVer_W4F0Moth	52
8.8.1.5	W3F4CusMajor_W4F0CusMinor	52
8.8.1.6	W3F4HardwareVer_W4F0FirewareMajor	52
8.8.1.7	W4F3Model_W3F0DebugInfTranVer	52
8.8.1.8	W7F0SnNumH	52
8.8.1.9	W7F0SnNumL	52
8.9	LaserPoint Struct Reference	52
8.9.1	Member Function Documentation	53

8.9.1.1	operator=(const LaserPoint &data)	53
8.9.2	Member Data Documentation	53
8.9.2.1	angle	53
8.9.2.2	intensity	53
8.9.2.3	range	53
8.10	LaserScan Struct Reference	53
8.10.1	Member Function Documentation	54
8.10.1.1	operator=(const LaserScan &data)	54
8.10.2	Member Data Documentation	54
8.10.2.1	config	54
8.10.2.2	points	54
8.10.2.3	stamp	54
8.11	lidar_ans_header Struct Reference	54
8.11.1	Member Data Documentation	55
8.11.1.1	size	55
8.11.1.2	subType	55
8.11.1.3	syncByte1	55
8.11.1.4	syncByte2	55
8.11.1.5	type	55
8.12	Locker Class Reference	55
8.12.1	Member Enumeration Documentation	56
8.12.1.1	LOCK_STATUS	56
8.12.2	Constructor & Destructor Documentation	56
8.12.2.1	Locker()	56
8.12.2.2	~Locker()	56
8.12.3	Member Function Documentation	56
8.12.3.1	getLockHandle()	56
8.12.3.2	init()	56
8.12.3.3	lock(unsigned long timeout=0xFFFFFFFF)	56
8.12.3.4	release()	56

8.12.3.5	unlock()	56
8.12.4	Member Data Documentation	56
8.12.4.1	_lock	56
8.13	serial::MillisecondTimer Class Reference	56
8.13.1	Constructor & Destructor Documentation	57
8.13.1.1	MillisecondTimer(const uint32_t millis)	57
8.13.2	Member Function Documentation	57
8.13.2.1	remaining()	57
8.13.2.2	timespec_now()	57
8.13.3	Member Data Documentation	57
8.13.3.1	expiry	57
8.14	node_info Struct Reference	57
8.14.1	Member Data Documentation	58
8.14.1.1	angle_q6_checkbit	58
8.14.1.2	debug_info	58
8.14.1.3	distance_q2	58
8.14.1.4	index	58
8.14.1.5	scan_frequency	58
8.14.1.6	stamp	58
8.14.1.7	sync_flag	58
8.14.1.8	sync_quality	58
8.15	node_package Struct Reference	58
8.15.1	Member Data Documentation	59
8.15.1.1	checksum	59
8.15.1.2	nowPackageNum	59
8.15.1.3	package_CT	59
8.15.1.4	package_Head	59
8.15.1.5	packageFirstSampleAngle	59
8.15.1.6	packageLastSampleAngle	59
8.15.1.7	packageSample	59

8.16	node_packages Struct Reference	59
8.16.1	Member Data Documentation	60
8.16.1.1	checksum	60
8.16.1.2	nowPackageNum	60
8.16.1.3	package_CT	60
8.16.1.4	package_Head	60
8.16.1.5	packageFirstSampleAngle	60
8.16.1.6	packageLastSampleAngle	60
8.16.1.7	packageSampleDistance	60
8.17	offset_angle Struct Reference	60
8.17.1	Member Data Documentation	60
8.17.1.1	angle	60
8.18	PackageNode Struct Reference	60
8.18.1	Member Data Documentation	61
8.18.1.1	PakageSampleDistance	61
8.18.1.2	PakageSampleQuality	61
8.19	serial::PortInfo Struct Reference	61
8.19.1	Detailed Description	61
8.19.2	Member Data Documentation	62
8.19.2.1	description	62
8.19.2.2	device_id	62
8.19.2.3	hardware_id	62
8.19.2.4	port	62
8.20	sampling_rate Struct Reference	62
8.20.1	Member Data Documentation	62
8.20.1.1	rate	62
8.21	scan_exposure Struct Reference	63
8.21.1	Member Data Documentation	63
8.21.1.1	exposure	63
8.22	scan_frequency Struct Reference	63

8.22.1	Member Data Documentation	63
8.22.1.1	frequency	63
8.23	scan_heart_beat Struct Reference	63
8.23.1	Member Data Documentation	64
8.23.1.1	enable	64
8.24	scan_points Struct Reference	64
8.24.1	Member Data Documentation	64
8.24.1.1	flag	64
8.25	scan_rotation Struct Reference	64
8.25.1	Member Data Documentation	65
8.25.1.1	rotation	65
8.26	ScopedLocker Class Reference	65
8.26.1	Constructor & Destructor Documentation	65
8.26.1.1	ScopedLocker(Locker &l)	65
8.26.1.2	~ScopedLocker()	65
8.26.2	Member Function Documentation	65
8.26.2.1	forceUnlock()	65
8.26.3	Member Data Documentation	65
8.26.3.1	_binded	65
8.27	serial::Serial::ScopedReadLock Class Reference	66
8.27.1	Constructor & Destructor Documentation	66
8.27.1.1	ScopedReadLock(Serial::SerialImpl *pimpl)	66
8.27.1.2	~ScopedReadLock()	66
8.27.1.3	ScopedReadLock(const ScopedReadLock &)	66
8.27.2	Member Function Documentation	66
8.27.2.1	operator=(ScopedReadLock)	67
8.27.3	Member Data Documentation	67
8.27.3.1	pimpl_	67
8.28	serial::Serial::ScopedWriteLock Class Reference	67
8.28.1	Constructor & Destructor Documentation	68

8.28.1.1	ScopedWriteLock(Serial::SerialImpl *pimpl)	68
8.28.1.2	~ScopedWriteLock()	68
8.28.1.3	ScopedWriteLock(const ScopedWriteLock &)	68
8.28.2	Member Function Documentation	68
8.28.2.1	operator=(ScopedWriteLock)	68
8.28.3	Member Data Documentation	68
8.28.3.1	pimpl_	68
8.29	serial::Serial Class Reference	68
8.29.1	Detailed Description	70
8.29.2	Constructor & Destructor Documentation	70
8.29.2.1	Serial(const std::string &port=""/>, uint32_t baudrate=9600, Timeout timeout=↵ Timeout(), bytesize_t bytesize=eightbits, parity_t parity=parity_none, stopbits_↵ t stopbits=stopbits_one, flowcontrol_t flowcontrol=flowcontrol_none)	70
8.29.2.2	~Serial()	71
8.29.2.3	Serial(const Serial &)	71
8.29.3	Member Function Documentation	71
8.29.3.1	available()	71
8.29.3.2	closePort()	71
8.29.3.3	flush()	71
8.29.3.4	flushInput()	71
8.29.3.5	flushOutput()	71
8.29.3.6	getBaudrate() const	72
8.29.3.7	getBytesize() const	72
8.29.3.8	getByteTime()	72
8.29.3.9	getCD()	72
8.29.3.10	getCTS()	72
8.29.3.11	getDSR()	72
8.29.3.12	getFlowcontrol() const	73
8.29.3.13	getParity() const	73
8.29.3.14	getPort() const	73
8.29.3.15	getRI()	73

8.29.3.16 getStopbits() const	73
8.29.3.17 getTimeout() const	74
8.29.3.18 isOpen()	74
8.29.3.19 open()	74
8.29.3.20 operator=(const Serial &)	74
8.29.3.21 read(uint8_t *buffer, size_t size)	74
8.29.3.22 read(std::vector< uint8_t > &buffer, size_t size=1)	75
8.29.3.23 read(std::string &buffer, size_t size=1)	75
8.29.3.24 read(size_t size=1)	75
8.29.3.25 read_(uint8_t *buffer, size_t size)	76
8.29.3.26 readData(uint8_t *data, size_t size)	76
8.29.3.27 readline(std::string &buffer, size_t size=65536, std::string eol="" "n")	76
8.29.3.28 readline(size_t size=65536, std::string eol="" "n")	76
8.29.3.29 readlines(size_t size=65536, std::string eol="" "n")	77
8.29.3.30 sendBreak(int duration)	77
8.29.3.31 setBaudrate(uint32_t baudrate)	77
8.29.3.32 setBreak(bool level=true)	77
8.29.3.33 setBytesize(bytesize_t bytesize)	77
8.29.3.34 setDTR(bool level=true)	78
8.29.3.35 setFlowcontrol(flowcontrol_t flowcontrol)	78
8.29.3.36 setParity(parity_t parity)	78
8.29.3.37 setPort(const std::string &port)	78
8.29.3.38 setRTS(bool level=true)	79
8.29.3.39 setStopbits(stopbits_t stopbits)	79
8.29.3.40 setTimeout(Timeout &timeout)	79
8.29.3.41 setTimeout(uint32_t inter_byte_timeout, uint32_t read_timeout_constant, uint32_t read_timeout_multiplier, uint32_t write_timeout_constant, uint32_ t write_timeout_multiplier)	80
8.29.3.42 waitByteTimes(size_t count)	80
8.29.3.43 waitForChange()	80
8.29.3.44 waitfordata(size_t data_count, uint32_t timeout, size_t *returned_size)	80

8.29.3.45 waitReadable()	80
8.29.3.46 write(const uint8_t *data, size_t size)	81
8.29.3.47 write(const std::vector< uint8_t > &data)	81
8.29.3.48 write(const std::string &data)	81
8.29.3.49 write_(const uint8_t *data, size_t length)	81
8.29.3.50 writeData(const uint8_t *data, size_t size)	82
8.29.4 Member Data Documentation	82
8.29.4.1 pimpl_	82
8.30 serial::Serial::SerialImpl Class Reference	82
8.30.1 Constructor & Destructor Documentation	84
8.30.1.1 SerialImpl(const string &port, unsigned long baudrate, bytesize_t bytesize, parity_t parity, stopbits_t stopbits, flowcontrol_t flowcontrol)	84
8.30.1.2 ~SerialImpl()	84
8.30.2 Member Function Documentation	84
8.30.2.1 available()	84
8.30.2.2 close()	84
8.30.2.3 flush()	84
8.30.2.4 flushInput()	84
8.30.2.5 flushOutput()	84
8.30.2.6 getBaudrate() const	84
8.30.2.7 getBytesize() const	84
8.30.2.8 getByteTime()	84
8.30.2.9 getCD()	84
8.30.2.10 getCTS()	84
8.30.2.11 getDSR()	85
8.30.2.12 getFlowcontrol() const	85
8.30.2.13 getParity() const	85
8.30.2.14 getPort() const	85
8.30.2.15 getRI()	85
8.30.2.16 getStopbits() const	85
8.30.2.17 getTermios(termios *tio)	85

8.30.2.18	<code>getTimeout() const</code>	85
8.30.2.19	<code>isOpen() const</code>	85
8.30.2.20	<code>open()</code>	85
8.30.2.21	<code>read(uint8_t *buf, size_t size=1)</code>	85
8.30.2.22	<code>readLock()</code>	85
8.30.2.23	<code>readUnlock()</code>	85
8.30.2.24	<code>sendBreak(int duration)</code>	85
8.30.2.25	<code>setBaudrate(unsigned long baudrate)</code>	85
8.30.2.26	<code>setBreak(bool level)</code>	85
8.30.2.27	<code>setBytesize(bytesize_t bytesize)</code>	85
8.30.2.28	<code>setCustomBaudRate(unsigned long baudrate)</code>	85
8.30.2.29	<code>setDTR(bool level)</code>	85
8.30.2.30	<code>setFlowcontrol(flowcontrol_t flowcontrol)</code>	85
8.30.2.31	<code>setParity(parity_t parity)</code>	85
8.30.2.32	<code>setPort(const string &port)</code>	85
8.30.2.33	<code>setRTS(bool level)</code>	85
8.30.2.34	<code>setStandardBaudRate(speed_t baudrate)</code>	86
8.30.2.35	<code>setStopbits(stopbits_t stopbits)</code>	86
8.30.2.36	<code>setTermios(const termios *tio)</code>	86
8.30.2.37	<code>setTimeout(Timeout &timeout)</code>	86
8.30.2.38	<code>waitByteTimes(size_t count)</code>	86
8.30.2.39	<code>waitForChange()</code>	86
8.30.2.40	<code>waitfordata(size_t data_count, uint32_t timeout, size_t *returned_size)</code>	86
8.30.2.41	<code>waitReadable(uint32_t timeout)</code>	86
8.30.2.42	<code>write(const uint8_t *data, size_t length)</code>	86
8.30.2.43	<code>writeLock()</code>	86
8.30.2.44	<code>writeUnlock()</code>	86
8.30.3	Member Data Documentation	86
8.30.3.1	<code>baudrate_</code>	86
8.30.3.2	<code>byte_time_ns_</code>	86

8.30.3.3	bytesize_	86
8.30.3.4	fd_	86
8.30.3.5	flowcontrol_	86
8.30.3.6	is_open_	86
8.30.3.7	parity_	86
8.30.3.8	pid	86
8.30.3.9	port_	87
8.30.3.10	read_mutex	87
8.30.3.11	rtscts_	87
8.30.3.12	stopbits_	87
8.30.3.13	timeout_	87
8.30.3.14	write_mutex	87
8.30.3.15	xonxoff_	87
8.31	serial::termios2 Struct Reference	87
8.31.1	Member Data Documentation	87
8.31.1.1	c_cc	87
8.31.1.2	c_cflag	87
8.31.1.3	c_iflag	87
8.31.1.4	c_ispeed	87
8.31.1.5	c_lflag	87
8.31.1.6	c_line	87
8.31.1.7	c_oflag	87
8.31.1.8	c_ospeed	87
8.32	Thread Class Reference	88
8.32.1	Constructor & Destructor Documentation	88
8.32.1.1	Thread()	88
8.32.1.2	~Thread()	88
8.32.1.3	Thread(thread_proc_t proc, void *param)	88
8.32.2	Member Function Documentation	88
8.32.2.1	createThread(thread_proc_t proc, void *param=NULL)	88

8.32.2.2	createThreadAux(void *param)	88
8.32.2.3	getHandle()	89
8.32.2.4	getParam()	89
8.32.2.5	join(unsigned long timeout=-1)	89
8.32.2.6	operator==(const Thread &right)	89
8.32.2.7	terminate()	89
8.32.2.8	ThreadCreateObjectFunctor(CLASS *pthis)	89
8.32.3	Member Data Documentation	89
8.32.3.1	_func	89
8.32.3.2	_handle	89
8.32.3.3	_param	89
8.33	serial::Timeout Struct Reference	89
8.33.1	Detailed Description	90
8.33.2	Constructor & Destructor Documentation	90
8.33.2.1	Timeout(uint32_t inter_byte_timeout=0, uint32_t read_timeout_constant_↵=0, uint32_t read_timeout_multiplier=0, uint32_t write_timeout_constant=0, uint32_t write_timeout_multiplier=0)	90
8.33.3	Member Function Documentation	90
8.33.3.1	max()	90
8.33.3.2	simpleTimeout(uint32_t timeout)	90
8.33.4	Member Data Documentation	90
8.33.4.1	inter_byte_timeout	90
8.33.4.2	read_timeout_constant	90
8.33.4.3	read_timeout_multiplier	90
8.33.4.4	write_timeout_constant	90
8.33.4.5	write_timeout_multiplier	91
8.34	ydlidar::YDlidarDriver Class Reference	91
8.34.1	Detailed Description	96
8.34.2	Member Enumeration Documentation	96
8.34.2.1	anonymous enum	96
8.34.3	Constructor & Destructor Documentation	96

8.34.3.1	YDlidarDriver()	96
8.34.3.2	~YDlidarDriver()	96
8.34.4	Member Function Documentation	96
8.34.4.1	ascendScanData(node_info *nodebuffer, size_t count)	96
8.34.4.2	cacheScanData()	97
8.34.4.3	checkAutoConnecting()	97
8.34.4.4	checkDeviceInfo(uint8_t *recvBuffer, uint8_t byte, int recvPos, int recvSize, int pos)	97
8.34.4.5	checkTransDelay()	97
8.34.4.6	clearDTR()	97
8.34.4.7	connect(const char *port_path, uint32_t baudrate)	98
8.34.4.8	createThread()	99
8.34.4.9	disableDataGrabbing()	99
8.34.4.10	disconnect()	99
8.34.4.11	flushSerial()	99
8.34.4.12	getData(uint8_t *data, size_t size)	99
8.34.4.13	getDeviceInfo(device_info &info, uint32_t timeout=DEFAULT_TIMEOUT)	100
8.34.4.14	getHealth(device_health &health, uint32_t timeout=DEFAULT_TIMEOUT)	100
8.34.4.15	getSamplingRate(sampling_rate &rate, uint32_t timeout=DEFAULT_TIMEOUT)	101
8.34.4.16	getScanFrequency(scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)	101
8.34.4.17	getSDKVersion()	102
8.34.4.18	getZeroOffsetAngle(offset_angle &angle, uint32_t timeout=DEFAULT_TIMEOUT)	102
8.34.4.19	grabScanData(node_info *nodebuffer, size_t &count, uint32_t timeout=DEFAULT_TIMEOUT)	102
8.34.4.20	isconnected() const	103
8.34.4.21	isscanning() const	103
8.34.4.22	lidarPortList()	103
8.34.4.23	PropertyBuilderByName(bool, SingleChannel, private)	104
8.34.4.24	PropertyBuilderByName(int, LidarType, private)	104
8.34.4.25	PropertyBuilderByName(uint32_t, PointTime, private)	105
8.34.4.26	reset(uint32_t timeout=DEFAULT_TIMEOUT)	105

8.34.4.27 sendCommand(uint8_t cmd, const void *payload=NULL, size_t payloadsize=0)	106
8.34.4.28 sendData(const uint8_t *data, size_t size)	106
8.34.4.29 setAutoReconnect(const bool &enable)	106
8.34.4.30 setDTR()	107
8.34.4.31 setIntensities(const bool &intensities)	107
8.34.4.32 setSamplingRate(sampling_rate &rate, uint32_t timeout=DEFAULT_TIMEOUT)	107
8.34.4.33 setScanFrequencyAdd(scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)	108
8.34.4.34 setScanFrequencyAddMic(scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)	108
8.34.4.35 setScanFrequencyDis(scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)	109
8.34.4.36 setScanFrequencyDisMic(scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)	109
8.34.4.37 startAutoScan(bool force=false, uint32_t timeout=DEFAULT_TIMEOUT)	110
8.34.4.38 startMotor()	110
8.34.4.39 startScan(bool force=false, uint32_t timeout=DEFAULT_TIMEOUT)	110
8.34.4.40 stop()	111
8.34.4.41 stopMotor()	111
8.34.4.42 stopScan(uint32_t timeout=DEFAULT_TIMEOUT)	111
8.34.4.43 waitDevicePackage(uint32_t timeout=DEFAULT_TIMEOUT)	112
8.34.4.44 waitForData(size_t data_count, uint32_t timeout=DEFAULT_TIMEOUT, size_t *returned_size=NULL)	112
8.34.4.45 waitPackage(node_info *node, uint32_t timeout=DEFAULT_TIMEOUT)	113
8.34.4.46 waitResponseHeader(lidar_ans_header *header, uint32_t timeout=DEFAULT_TIMEOUT)	113
8.34.4.47 waitScanData(node_info *nodebuffer, size_t &count, uint32_t timeout=DEFAULT_TIMEOUT)	113
8.34.5 Member Data Documentation	114
8.34.5.1 _dataEvent	114
8.34.5.2 _lock	114
8.34.5.3 _serial	114
8.34.5.4 _serial_lock	114

8.34.5.5	_thread	114
8.34.5.6	async_size	114
8.34.5.7	asyncRecvPos	114
8.34.5.8	Checksum	114
8.34.5.9	ChecksumCal	115
8.34.5.10	ChecksumResult	115
8.34.5.11	FirstSampleAngle	115
8.34.5.12	get_device_health_success	115
8.34.5.13	get_device_info_success	115
8.34.5.14	globalRecvBuffer	115
8.34.5.15	has_device_header	115
8.34.5.16	has_package_error	115
8.34.5.17	header_	115
8.34.5.18	headerBuffer	115
8.34.5.19	health_	115
8.34.5.20	healthBuffer	115
8.34.5.21	info_	115
8.34.5.22	infoBuffer	115
8.34.5.23	IntervalSampleAngle	115
8.34.5.24	IntervalSampleAngle_LastPackage	115
8.34.5.25	isAutoconnting	115
8.34.5.26	isAutoReconnect	115
8.34.5.27	isConnected	115
8.34.5.28	isScanning	116
8.34.5.29	isSupportMotorDtrCtrl	116
8.34.5.30	last_device_byte	116
8.34.5.31	LastSampleAngle	116
8.34.5.32	LastSampleAngleCal	116
8.34.5.33	m_baudrate	116
8.34.5.34	m_intensities	116

8.34.5.35 m_sampling_rate	116
8.34.5.36 model	116
8.34.5.37 package	116
8.34.5.38 package_index	116
8.34.5.39 package_Sample_Index	116
8.34.5.40 packages	117
8.34.5.41 PackageSampleBytes	117
8.34.5.42 retryCount	117
8.34.5.43 sample_rate	117
8.34.5.44 SampleNumlAndCTCal	117
8.34.5.45 scan_frequency	117
8.34.5.46 scan_node_buf	117
8.34.5.47 scan_node_count	117
8.34.5.48 serial_port	117
8.34.5.49 trans_delay	117
8.34.5.50 Valu8Tou16	117
9 File Documentation	119
9.1 include/angles.h File Reference	119
9.1.1 Macro Definition Documentation	120
9.1.1.1 M_PI	120
9.2 include/CYdLidar.h File Reference	120
9.3 include/help_info.h File Reference	121
9.4 include/lock.h File Reference	123
9.4.1 Macro Definition Documentation	124
9.4.1.1 LOCK	124
9.4.1.2 UNLOCK	124
9.4.2 Function Documentation	124
9.4.2.1 check_group_uucp()	124
9.4.2.2 check_lock_pid(const char *file, int openpid)	124
9.4.2.3 check_lock_status(const char *)	125

9.4.2.4	fhs_lock(const char *, int)	125
9.4.2.5	fhs_unlock(const char *, int)	125
9.4.2.6	is_device_locked(const char *)	125
9.4.2.7	lfs_lock(const char *, int)	125
9.4.2.8	lfs_unlock(const char *, int)	125
9.4.2.9	lib_lock_dev_lock(const char *, int)	125
9.4.2.10	lib_lock_dev_unlock(const char *, int)	125
9.4.2.11	lock_device(const char *)	125
9.4.2.12	unlock_device(const char *)	125
9.4.2.13	uucp_lock(const char *, int)	125
9.4.2.14	uucp_unlock(const char *, int)	125
9.5	include/locker.h File Reference	125
9.6	include/serial.h File Reference	126
9.7	include/thread.h File Reference	128
9.7.1	Macro Definition Documentation	129
9.7.1.1	CLASS_THREAD	129
9.7.1.2	UNUSED	129
9.8	include/timer.h File Reference	129
9.8.1	Macro Definition Documentation	130
9.8.1.1	BEGIN_STATIC_CODE	130
9.8.1.2	END_STATIC_CODE	130
9.8.1.3	getms	130
9.8.1.4	getTime	130
9.8.2	Function Documentation	130
9.8.2.1	delay(uint32_t ms)	130
9.9	include/utils.h File Reference	130
9.9.1	Macro Definition Documentation	131
9.9.1.1	YDLIDAR_API	131
9.10	include/v8stdint.h File Reference	131
9.10.1	Macro Definition Documentation	133

9.10.1.1	<code>__attribute__</code>	133
9.10.1.2	<code>__small_endian</code>	133
9.10.1.3	<code>_access</code>	133
9.10.1.4	<code>INVALID_TIMESTAMP</code>	133
9.10.1.5	<code>IS_FAIL</code>	133
9.10.1.6	<code>IS_OK</code>	133
9.10.1.7	<code>IS_TIMEOUT</code>	133
9.10.1.8	<code>RESULT_FAIL</code>	133
9.10.1.9	<code>RESULT_OK</code>	133
9.10.1.10	<code>RESULT_TIMEOUT</code>	133
9.10.1.11	<code>UNUSED</code>	133
9.10.2	Typedef Documentation	133
9.10.2.1	<code>result_t</code>	133
9.10.2.2	<code>signal_handler_t</code>	133
9.10.2.3	<code>thread_proc_t</code>	133
9.10.3	Enumeration Type Documentation	133
9.10.3.1	anonymous enum	133
9.10.4	Function Documentation	134
9.10.4.1	<code>set_signal_handler(int signal_value, signal_handler_t signal_handler)</code>	134
9.10.4.2	<code>signal_handler(int signal_value)</code>	134
9.10.4.3	<code>trigger_interrupt_guard_condition(int signal_value)</code>	134
9.10.5	Variable Documentation	134
9.10.5.1	<code>g_signal_status</code>	134
9.10.5.2	<code>old_signal_handler</code>	134
9.11	<code>include/ydlidar_driver.h</code> File Reference	134
9.12	<code>include/ydlidar_protocol.h</code> File Reference	135
9.12.1	Macro Definition Documentation	139
9.12.1.1	<code>_countof</code>	139
9.12.1.2	<code>GLASSNOISEINTENSITY</code>	139
9.12.1.3	<code>INTENSITY_NORMAL_PACKAGE_SIZE</code>	139

9.12.1.4	LIDAR_ANS_SYNC_BYTE1	139
9.12.1.5	LIDAR_ANS_SYNC_BYTE2	139
9.12.1.6	LIDAR_ANS_TYPE_DEVHEALTH	139
9.12.1.7	LIDAR_ANS_TYPE_DEVINFO	139
9.12.1.8	LIDAR_ANS_TYPE_MEASUREMENT	139
9.12.1.9	LIDAR_CMD_ADD_EXPOSURE	139
9.12.1.10	LIDAR_CMD_DIS_EXPOSURE	139
9.12.1.11	LIDAR_CMD_DISABLE_CONST_FREQ	139
9.12.1.12	LIDAR_CMD_DISABLE_LOW_POWER	139
9.12.1.13	LIDAR_CMD_ENABLE_CONST_FREQ	139
9.12.1.14	LIDAR_CMD_ENABLE_LOW_POWER	139
9.12.1.15	LIDAR_CMD_FORCE_SCAN	139
9.12.1.16	LIDAR_CMD_FORCE_STOP	139
9.12.1.17	LIDAR_CMD_GET_AIMSPEED	139
9.12.1.18	LIDAR_CMD_GET_DEVICE_HEALTH	139
9.12.1.19	LIDAR_CMD_GET_DEVICE_INFO	139
9.12.1.20	LIDAR_CMD_GET_EAI	139
9.12.1.21	LIDAR_CMD_GET_OFFSET_ANGLE	139
9.12.1.22	LIDAR_CMD_GET_SAMPLING_RATE	139
9.12.1.23	LIDAR_CMD_RESET	140
9.12.1.24	LIDAR_CMD_RUN_INVERSION	140
9.12.1.25	LIDAR_CMD_RUN_POSITIVE	140
9.12.1.26	LIDAR_CMD_SAVE_SET_EXPOSURE	140
9.12.1.27	LIDAR_CMD_SCAN	140
9.12.1.28	LIDAR_CMD_SET_AIMSPEED_ADD	140
9.12.1.29	LIDAR_CMD_SET_AIMSPEED_ADDMIC	140
9.12.1.30	LIDAR_CMD_SET_AIMSPEED_DIS	140
9.12.1.31	LIDAR_CMD_SET_AIMSPEED_DISMIC	140
9.12.1.32	LIDAR_CMD_SET_LOW_EXPOSURE	140
9.12.1.33	LIDAR_CMD_SET_SAMPLING_RATE	140

9.12.1.34 LIDAR_CMD_STATE_MODEL_MOTOR	140
9.12.1.35 LIDAR_CMD_STOP	140
9.12.1.36 LIDAR_CMD_SYNC_BYTE	140
9.12.1.37 LIDAR_CMDFLAG_HAS_PAYLOAD	140
9.12.1.38 LIDAR_RESP_MEASUREMENT_ANGLE_SAMPLE_SHIFT	140
9.12.1.39 LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT	140
9.12.1.40 LIDAR_RESP_MEASUREMENT_CHECKBIT	140
9.12.1.41 LIDAR_RESP_MEASUREMENT_DISTANCE_SHIFT	140
9.12.1.42 LIDAR_RESP_MEASUREMENT_QUALITY_SHIFT	140
9.12.1.43 LIDAR_RESP_MEASUREMENT_SYNCBIT	140
9.12.1.44 LIDAR_STATUS_ERROR	140
9.12.1.45 LIDAR_STATUS_OK	140
9.12.1.46 LIDAR_STATUS_WARNING	141
9.12.1.47 M_PI	141
9.12.1.48 Node_Default_Quality	141
9.12.1.49 Node_NotSync	141
9.12.1.50 Node_Sync	141
9.12.1.51 NORMAL_PACKAGE_SIZE	141
9.12.1.52 PackagePaidBytes	141
9.12.1.53 PackageSampleMaxLngth	141
9.12.1.54 PH	141
9.12.1.55 PropertyBuilderByName	141
9.12.1.56 SUNNOISEINTENSITY	141
9.12.2 Enumeration Type Documentation	141
9.12.2.1 CT	141
9.12.2.2 LidarTypeID	141
9.12.3 Function Documentation	142
9.12.3.1 __attribute__((packed))	142
9.12.4 Variable Documentation	142
9.12.4.1 __attribute__	142

9.12.4.2	angle	142
9.12.4.3	angle_q6_checkbit	142
9.12.4.4	checkSum	142
9.12.4.5	cmd_flag	142
9.12.4.6	data	142
9.12.4.7	debug_info	142
9.12.4.8	distance_q2	142
9.12.4.9	enable	142
9.12.4.10	error_code	142
9.12.4.11	exposure	142
9.12.4.12	firmware_version	142
9.12.4.13	flag	143
9.12.4.14	frequency	143
9.12.4.15	hardware_version	143
9.12.4.16	index	143
9.12.4.17	model	143
9.12.4.18	nowPackageNum	143
9.12.4.19	package_CT	143
9.12.4.20	package_Head	143
9.12.4.21	packageFirstSampleAngle	143
9.12.4.22	packageLastSampleAngle	143
9.12.4.23	packageSample	143
9.12.4.24	packageSampleDistance	143
9.12.4.25	PakageSampleDistance	143
9.12.4.26	PakageSampleQuality	143
9.12.4.27	rate	143
9.12.4.28	rotation	143
9.12.4.29	scan_frequence	143
9.12.4.30	serialnum	144
9.12.4.31	size	144

9.12.4.32 stamp	144
9.12.4.33 state	144
9.12.4.34 status	144
9.12.4.35 subType	144
9.12.4.36 sync_flag	144
9.12.4.37 sync_quality	144
9.12.4.38 syncByte	144
9.12.4.39 syncByte1	144
9.12.4.40 syncByte2	144
9.12.4.41 type	144
9.13 README.md File Reference	144
9.14 samples/main.cpp File Reference	144
9.14.1 Function Documentation	145
9.14.1.1 main(int argc, char *argv[])	145
9.15 src/common.h File Reference	145
9.15.1 Macro Definition Documentation	146
9.15.1.1 SDKVersion	146
9.16 src/CYdLidar.cpp File Reference	146
9.17 src/impl/list_ports/list_ports_linux.cpp File Reference	146
9.18 src/impl/unix/list_ports_linux.cpp File Reference	146
9.19 src/impl/list_ports/list_ports_osx.cpp File Reference	146
9.20 src/impl/list_ports/list_ports_win.cpp File Reference	146
9.21 src/impl/windows/list_ports_win.cpp File Reference	146
9.22 src/impl/unix/unix.h File Reference	146
9.23 src/impl/unix/unix_serial.cpp File Reference	147
9.23.1 Macro Definition Documentation	148
9.23.1.1 BOTHER	148
9.23.1.2 SNCCS	148
9.23.1.3 TCGETS2	148
9.23.1.4 TCSETS2	148

9.23.1.5	TIOCINQ	148
9.24	src/impl/unix/unix_serial.h File Reference	148
9.25	src/impl/unix/unix_timer.cpp File Reference	149
9.26	src/impl/windows/win.h File Reference	150
9.27	src/impl/windows/win_serial.cpp File Reference	150
9.28	src/impl/windows/win_serial.h File Reference	150
9.29	src/impl/windows/win_timer.cpp File Reference	150
9.30	src/lock.c File Reference	150
9.30.1	Function Documentation	151
9.30.1.1	check_group_uucp()	151
9.30.1.2	check_lock_pid(const char *file, int openpid)	151
9.30.1.3	check_lock_status(const char *filename)	151
9.30.1.4	fhs_lock(const char *filename, int pid)	151
9.30.1.5	fhs_unlock(const char *filename, int openpid)	151
9.30.1.6	is_device_locked(const char *port_filename)	151
9.30.1.7	uucp_lock(const char *filename, int pid)	151
9.30.1.8	uucp_unlock(const char *filename, int openpid)	151
9.31	src/serial.cpp File Reference	151
9.32	src/ydlidar_driver.cpp File Reference	152
Index		153

Chapter 1

CYdLidar(YDLIDAR SDK API)

Library	CYdLidar
File	CYdLidar.h
Author	Tony [code at ydlidar com]
Source	https://github.com/ydlidar/YDLidar-SDK
Version	1.0.0
Sample	ydlidar test [G1 G2 G4 G6 S2 X2 X4)

This API calls Two LiDAR interface classes in the following sections:

- [YDlidarDriver](#)

Copyright

Copyright (c) 2018-2020 EAIBOT

Jump to the [::CYdLidar](#) interface documentation.

Chapter 2

YDlidarDriver

YDlidarDriver API

Library	YDlidarDriver
File	ydlidar_driver.h
Author	Tony [code at ydlidar com]
Source	https://github.com/ydlidar/YDLidar-SDK
Version	1.0.0

This YDlidarDriver support [TYPE_TRIANGLE](#) and [TYPE_TOF](#) LiDAR

Copyright

Copyright (c) 2018-2020 EAIBOT Jump to the [::ydlidar::YDlidarDriver](#) interface documentation.

Chapter 3

README

YDLIDAR SDK

1 Introduction

YDLIDAR(<https://www.ydlidar.com/>) series is a set of high-performance and low-cost LIDAR sensors, which is the perfect sensor of 2D SLAM, 3D reconstruction, multi-touch, and safety applications.

If you are using ROS (Robot Operating System), please use our open-source [ROS Driver](#).

1.1 Prerequisites

- Linux
- Windows 7/10, Visual Studio 2015/2017
- C++11 compiler

1.2 Release Notes

Title	Version	Data
SDK	1.4.6	2020-02-15

- [feat] the output points are fixed, when FixedResolution is set to true.

2 YDLidar SDK Communication Protocol

YDLidar SDK communication protocol opens to all users. It is the communication protocol between user programs and YDLIDAR products. The protocol consists of control commands and data format. Please refer to the [YDLidar SDK Communication Protocol](#) for detailed information.

3 YDLidar SDK

YDLidar SDK provides the implementation of control commands and Laser scan data transmission, as well as the C/C++ API. The basic structure of YDLidar SDK is shown as below:

Serial or network is used for communication between YDLidar SDK and LiDAR sensors. Please refer to the [YDLidar SDK Communication Protocol](#) for further information. [LaserScan](#) supports Laser Scan Data transmission, while Command handler receives and sends control commands. And the C++ API is based on Command and [LaserScan](#) Handler.

The YDLidar LiDAR sensors can be connected to host directly by serial or through the YDLidar Adapter board. YDLidar SDK supports both connection methods. When LiDAR units are connected to host directly by Serial, the host will establish communication with each LiDAR unit individually. And if the LiDAR units connect to host through Adapter board, then the host only communicates with the YDLidar Adapter board while the Adapter Board communicates with each LiDAR unit.

4 YDLidar SDK API

YDLidar SDK API provides a set of C++ style functions which can be conveniently integrated in C/C++ programs. Please refer to the [YDLidar SDK API Reference](#) for further information.

4.1 Installation

The installation procedures in Ubuntu 18.04/16.04/14.04 LTS and Windows 7/10 are shown here as examples. For Ubuntu 18.04/16.04/14.04 32-bit LTS and Mac, you can get it in [YDLidar-SDK wiki](#).

4.1.1 Ubuntu 18.04/16.04/14.04 LTS

Dependencies

YDLidar SDK requires [CMake 2.8.2+](#) as dependencies. You can install these packages using apt:

```
1 sudo apt install cmake pkg-config
```

Compile YDLidar SDK

In the YDLidar SDK directory, run the following commands to compile the project:

```
1 git clone https://github.com/YDLIDAR/YDLidar-SDK.git
2 cd YDLidar-SDK/build
3 cmake ..
4 make
5 sudo make install
```

4.1.2 Windows 7/10

Dependencies

YDLidar SDK supports Visual Studio 2015/2017 and requires **CMake 2.8.2+** as dependencies. **vcpkg** is recommended for building the dependency libraries as follows: For the 32-bit project:

```
1 .\vcpkg install cmake
2 .\vcpkg integrate install
```

For the 64-bit project:

```
1 .\vcpkg install cmake:x64-windows
2 .\vcpkg integrate install
```

Then, in the YDLidar SDK directory, run the following commands to create the Visual Studio solution file. Please replace [vcpkgroot] with your vcpkg installation path. Generate the 32-bit project:

```
1 cd build && \
2 cmake .. -DCMAKE_TOOLCHAIN_FILE=[vcpkgroot]\scripts\buildsystems\vcpkg.cmake"
```

Generate the 64-bit project:

```
1 cd build && \
2 cmake .. -G "Visual Studio 15 2017 Win64"
   -DCMAKE_TOOLCHAIN_FILE=[vcpkgroot]\scripts\buildsystems\vcpkg.cmake"
```

Compile YDLidar SDK

You can now compile the YDLidar SDK in Visual Studio.

4.2 Run YDLidar SDK Sample

Three samples are provided in samples, which demonstrate how to configure YDLidar LiDAR units and receive the laser scan data when directly connecting YDLidar SDK to LiDAR units or by using a YDLidar Adapter board, respectively. The sequence diagram is shown as below:

4.2.1 Ubuntu 18.04/16.04 /14.04 LTS

For Ubuntu 18.04/16.04/14.04 LTS, run the *ydliadar_test* if connect with the Triangle LiDAR unit(s) or TOF LiDAR unit(s):

```
1 ./ydliadar_test
```

4.2.2 Windows 7/10

After compiling the YDLidar SDK as shown in section 4.1.2, you can find *ydliadar_test.exe* in the {sdk} or {sdk} folder, respectively, which can be run directly.

Then you can see SDK initializing the information as below:

Then you can see SDK Scanning the information as below:

LIDAR	Model	Baudrate	Sample↔ Rate(K)	Range(m)	Frequency HZ)	Intenstiy(b	Single↔ Channel	voltage(↔ V)
G4	5	230400	9/8/4	0.↔ 28/0.26/0.↔ 1~16	5~12	false	false	4.8~5.2
X4	6	128000	5	0.12~10	5~12(P↔ WM)	false	false	4.8~5.2
X2/X2L	6	115200	3	0.10~8.0	4~8(P↔ WM)	false	true	4.8~5.2
G4PRO	7	230400	9/8/4	0.↔ 28/0.26/0.↔ 1~16	5~12	false	false	4.8~5.2
F4PRO	8	230400	4/6	0.12~12	5~12	false	false	4.8~5.2
R2	9	230400	5	0.12~16	5~12	false	false	4.8~5.2
G6	13	512000	18/16/8	0.↔ 28/0.26/0.↔ 1~25	5~12	false	false	4.8~5.2
G2A	14	230400	5	0.12~12	5~12	false	false	4.8~5.2
G2	15	230400	5	0.28~16	5~12	true(8)	false	4.8~5.2
G2C	16	115200	4	0.1~12	5~12	false	false	4.8~5.2
G4B	17	512000	10	0.12~16	5~12	true(10)	false	4.8~5.2
G4C	18	115200	4	0.1~12	5~12	false	false	4.8~5.2
G1	19	230400	9	0.28~16	5~12	false	false	4.8~5.2
TX8	100	115200	4	0.1~8	4~8(P↔ WM)	false	true	4.8~5.2
TX20	100	115200	4	0.1~20	4~8(P↔ WM)	false	true	4.8~5.2
TG15	100	512000	20/18/10	0.05~15	3~16	false	false	4.8~5.2
TG30	101	512000	20/18/10	0.05~30	3~16	false	false	4.8~5.2
TG50	102	512000	20/18/10	0.05~50	3~16	false	false	4.8~5.2

Note: PWM option speed control requires external PWM wave.

8 Licence

The SDK itself is licensed under BSD license

9 Support

You can get support from YDLidar with the following methods:

- Send email to support@ydlidar.com with a clear description of your problem and your setup
- Github Issues

10 Contact EAI

If you have any extra questions, please feel free to [contact us](#)

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

angles	17
impl	19
serial	20
ydlidar	23

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

cmd_packet	29
CYdLidar	30
device_health	47
device_info	47
Event	48
function_state	49
LaserConfig	
A struct for returning configuration from the YDLIDAR	50
LaserDebug	51
LaserPoint	52
LaserScan	53
lidar_ans_header	54
Locker	55
serial::MillisecondTimer	56
node_info	57
node_package	58
node_packages	59
offset_angle	60
PackageNode	60
serial::PortInfo	61
sampling_rate	62
scan_exposure	63
scan_frequency	63
scan_heart_beat	63
scan_points	64
scan_rotation	64
ScopedLocker	65
serial::Serial::ScopedReadLock	66
serial::Serial::ScopedWriteLock	67
serial::Serial	68
serial::Serial::SerialImpl	82
serial::termios2	87
Thread	88
serial::Timeout	89
ydlidar::YDLidarDriver	91

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

include/angles.h	119
include/CYdLidar.h	120
include/help_info.h	121
include/lock.h	123
include/locker.h	125
include/serial.h	126
include/thread.h	128
include/timer.h	129
include/utils.h	130
include/v8stdint.h	131
include/ydlidar_driver.h	134
include/ydlidar_protocol.h	135
samples/main.cpp	144
src/common.h	145
src/CYdLidar.cpp	146
src/lock.c	150
src/serial.cpp	151
src/ydlidar_driver.cpp	152
src/impl/list_ports/list_ports_linux.cpp	146
src/impl/list_ports/list_ports_osx.cpp	146
src/impl/list_ports/list_ports_win.cpp	146
src/impl/unix/list_ports_linux.cpp	146
src/impl/unix/unix.h	146
src/impl/unix/unix_serial.cpp	147
src/impl/unix/unix_serial.h	148
src/impl/unix/unix_timer.cpp	149
src/impl/windows/list_ports_win.cpp	146
src/impl/windows/win.h	150
src/impl/windows/win_serial.cpp	150
src/impl/windows/win_serial.h	150
src/impl/windows/win_timer.cpp	150

Chapter 7

Namespace Documentation

7.1 angles Namespace Reference

Functions

- static double [from_degrees](#) (double degrees)
Convert degrees to radians.
- static double [to_degrees](#) (double radians)
Convert radians to degrees.
- static double [normalize_angle_positive](#) (double angle)
normalize_angle_positive
- static double [normalize_angle](#) (double angle)
normalize
- static double [shortest_angular_distance](#) (double from, double to)
shortest_angular_distance
- static double [two_pi_complement](#) (double angle)
returns the angle in $[-2\pi, 2\pi]$ going the other way along the unit circle.
- static bool [find_min_max_delta](#) (double from, double left_limit, double right_limit, double &result_min_delta, double &result_max_delta)
This function is only intended for internal use and not intended for external use. If you do use it, read the documentation very carefully. Returns the min and max amount (in radians) that can be moved from "from" angle to "left_limit" and "right_limit".
- static bool [shortest_angular_distance_with_limits](#) (double from, double to, double left_limit, double right_limit, double &shortest_angle)
Returns the delta from "from_angle" to "to_angle" making sure it does not violate limits specified by left_limit and right_limit. The valid interval of angular positions is [left_limit, right_limit]. E.g., [-0.25, 0.25] is a 0.5 radians wide interval that contains 0. But [0.25, -0.25] is a $2\pi - 0.5$ wide interval that contains π (but not 0). The value of shortest_angle is the angular difference between "from" and "to" that lies within the defined valid interval. E.g. shortest_angular_distance_with_limits(-0.5, 0.5, 0.25, -0.25, ss) evaluates ss to $2\pi - 1.0$ and returns true while shortest_angular_distance_with_limits(-0.5, 0.5, -0.25, 0.25, ss) returns false since -0.5 and 0.5 do not lie in the interval [-0.25, 0.25].

7.1.1 Function Documentation

7.1.1.1 `static bool angles::find_min_max_delta (double from, double left_limit, double right_limit, double & result_min_delta, double & result_max_delta) [static]`

This function is only intended for internal use and not intended for external use. If you do use it, read the documentation very carefully. Returns the min and max amount (in radians) that can be moved from "from" angle to "left_limit" and "right_limit".

Returns

returns false if "from" angle does not lie in the interval [left_limit,right_limit]

Parameters

<i>from</i>	- "from" angle - must lie in [-M_PI, M_PI)
<i>left_limit</i>	- left limit of valid interval for angular position - must lie in [-M_PI, M_PI], left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
<i>right_limit</i>	- right limit of valid interval for angular position - must lie in [-M_PI, M_PI], left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
<i>result_min_delta</i>	- minimum (delta) angle (in radians) that can be moved from "from" position before hitting the joint stop
<i>result_max_delta</i>	- maximum (delta) angle (in radians) that can be moved from "from" position before hitting the joint stop

7.1.1.2 `static double angles::from_degrees (double degrees) [inline],[static]`

Convert degrees to radians.

7.1.1.3 `static double angles::normalize_angle (double angle) [inline],[static]`

normalize

Normalizes the angle to be -M_PI circle to +M_PI circle It takes and returns radians.

7.1.1.4 `static double angles::normalize_angle_positive (double angle) [inline],[static]`

normalize_angle_positive

Normalizes the angle to be 0 to 2*M_PI It takes and returns radians.

7.1.1.5 `static double angles::shortest_angular_distance (double from, double to) [inline],[static]`

shortest_angular_distance

Given 2 angles, this returns the shortest angular difference. The inputs and outputs are of course radians.

The result would always be $-\pi \leq \text{result} \leq \pi$. Adding the result to "from" will always get you an equivalent angle to "to".

7.1.1.6 `static bool angles::shortest_angular_distance_with_limits (double from, double to, double left_limit, double right_limit, double & shortest_angle) [inline],[static]`

Returns the delta from "from_angle" to "to_angle" making sure it does not violate limits specified by *left_limit* and *right_limit*. The valid interval of angular positions is [*left_limit*,*right_limit*]. E.g., [-0.25,0.25] is a 0.5 radians wide interval that contains 0. But [0.25,-0.25] is a $2 \times M_PI - 0.5$ wide interval that contains M_PI (but not 0). The value of *shortest_angle* is the angular difference between "from" and "to" that lies within the defined valid interval. E.g. `shortest_angular_distance_with_limits(-0.5,0.5,0.25,-0.25,ss)` evaluates *ss* to $2 \times M_PI - 1.0$ and returns true while `shortest_angular_distance_with_limits(-0.5,0.5,-0.25,0.25,ss)` returns false since -0.5 and 0.5 do not lie in the interval [-0.25,0.25].

Returns

true if "from" and "to" positions are within the limit interval, false otherwise

Parameters

<i>from</i>	- "from" angle
<i>to</i>	- "to" angle
<i>left_limit</i>	- left limit of valid interval for angular position, left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
<i>right_limit</i>	- right limit of valid interval for angular position, left and right limits are specified on the unit circle w.r.t to a reference pointing inwards
<i>shortest_angle</i>	- result of the shortest angle calculation

7.1.1.7 `static double angles::to_degrees (double radians) [inline],[static]`

Convert radians to degrees.

7.1.1.8 `static double angles::two_pi_complement (double angle) [inline],[static]`

returns the angle in $[-2 \times M_PI, 2 \times M_PI]$ going the other way along the unit circle.

Parameters

<i>angle</i>	The angle to which you want to turn in the range $[-2 \times M_PI, 2 \times M_PI]$ E.g. <code>two_pi_complement(-M_PI/4)</code> returns $7 \times M_PI/4$ <code>two_pi_complement(M_PI/4)</code> returns $-7 \times M_PI/4$
--------------	---

7.2 impl Namespace Reference

Functions

- `uint32_t getHDTimer ()`
- `uint64_t getCurrentTime ()`

7.2.1 Function Documentation

7.2.1.1 `uint64_t impl::getCurrentTime ()`

7.2.1.2 `uint32_t impl::getHDTimer ()`

7.3 serial Namespace Reference

Classes

- class [MillisecondTimer](#)
- struct [PortInfo](#)
- class [Serial](#)
- struct [termios2](#)
- struct [Timeout](#)

Enumerations

- enum [bytesize_t](#) { [fivebits](#) = 5, [sixbits](#) = 6, [sevenbits](#) = 7, [eightbits](#) = 8 }
- enum [parity_t](#) { [parity_none](#) = 0, [parity_odd](#) = 1, [parity_even](#) = 2, [parity_mark](#) = 3, [parity_space](#) = 4 }
- enum [stopbits_t](#) { [stopbits_one](#) = 1, [stopbits_two](#) = 2, [stopbits_one_point_five](#) }
- enum [flowcontrol_t](#) { [flowcontrol_none](#) = 0, [flowcontrol_software](#), [flowcontrol_hardware](#) }

Functions

- `std::vector< PortInfo > list_ports ()`
- `timespec timespec_from_ms (const uint32_t millis)`
- `static void set_common_props (termios *tio)`
- `static void set_databits (termios *tio, serial::bytesize_t databits)`
- `static void set_parity (termios *tio, serial::parity_t parity)`
- `static void set_stopbits (termios *tio, serial::stopbits_t stopbits)`
- `static void set_flowcontrol (termios *tio, serial::flowcontrol_t flowcontrol)`
- `static bool is_standardbaudrate (unsigned long baudrate, speed_t &baud)`

7.3.1 Detailed Description

`setup_port` - Configure the port, eg. baud rate, data bits, etc.

Parameters

<i>fd</i>	: The serial port
<i>speed</i>	: The baud rate
<i>data_bits</i>	: The data bits
<i>parity</i>	: The parity bits
<i>stop_bits</i>	: The stop bits

Returns

Return 0 if everything is OK, otherwise -1 with some error msg.

Note

Here are termios structure members:

Member	Description
c_cflag	Control options
c_lflag	Line options
c_iflag	Input options
c_oflag	Output options
c_cc	Control characters
c_ispeed	Input baud (new interface)
c_ospeed	Output baud (new interface)

The `c_cflag` member controls the baud rate, number of data bits, parity, stop bits, and hardware flow control. There are constants for all of the supported configurations. Constant Description

CBAUD	Bit mask for baud rate
B0	0 baud (drop DTR)
B50	50 baud
B75	75 baud
B110	110 baud
B134	134.5 baud
B150	150 baud
B200	200 baud
B300	300 baud
B600	600 baud
B1200	1200 baud
B1800	1800 baud
B2400	2400 baud
B4800	4800 baud
B9600	9600 baud
B19200	19200 baud
B38400	38400 baud
B57600	57,600 baud
B76800	76,800 baud
B115200	115,200 baud
EXTA	External rate clock
EXTB	External rate clock
CSIZE	Bit mask for data bits
CS5	5 data bits
CS6	6 data bits
CS7	7 data bits
CS8	8 data bits
CSTOPB	2 stop bits (1 otherwise)
CREAD	Enable receiver
PARENB	Enable parity bit
PARODD	Use odd parity instead of even
HUPCL	Hangup (drop DTR) on last close
CLOCAL	Local line - do not change "owner" of port
IOBLK	Block job control output
CNEW_RTSCCTS	Enable hardware flow control (not supported on all platforms)

The input modes member `c_iflag` controls any input processing that is done to characters received on the port. Like the `c_cflag` field, the final value stored in `c_iflag` is the bitwise OR of the desired options.

Constant	Description
INPCK	Enable parity check
IGNPAR	Ignore parity errors
PARMRK	Mark parity errors
ISTRIP	Strip parity bits
IXON	Enable software flow control (outgoing)
IXOFF	Enable software flow control (incoming)
IXANY	Allow any character to start flow again
IGNBRK	Ignore break condition
BRKINT	Send a SIGINT when a break condition is detected
INLCR	Map NL to CR
IGNCR	Ignore CR
ICRNL	Map CR to NL
IUCLC	Map uppercase to lowercase
IMAXBEL	Echo BEL on input line too long

Here are some examples of setting parity checking:
No parity (8N1):

```
options.c_cflag &= ~PARENB  
options.c_cflag &= ~CSTOPB  
options.c_cflag &= ~CSIZE;  
options.c_cflag |= CS8;
```

Even parity (7E1):

```
options.c_cflag |= PARENB  
options.c_cflag &= ~PARODD  
options.c_cflag &= ~CSTOPB  
options.c_cflag &= ~CSIZE;  
options.c_cflag |= CS7;
```

Odd parity (7O1):

```
options.c_cflag |= PARENB  
options.c_cflag |= PARODD  
options.c_cflag &= ~CSTOPB  
options.c_cflag &= ~CSIZE;  
options.c_cflag |= CS7;
```

7.3.2 Enumeration Type Documentation

7.3.2.1 enum serial::bytesize_t

Enumeration defines the possible bytesizes for the serial port.

Enumerator

fivebits
sixbits
sevenbits
eightbits

7.3.2.2 enum serial::flowcontrol_t

Enumeration defines the possible flowcontrol types for the serial port.

Enumerator

flowcontrol_none
flowcontrol_software
flowcontrol_hardware

7.3.2.3 enum serial::parity_t

Enumeration defines the possible parity types for the serial port.

Enumerator

parity_none
parity_odd
parity_even
parity_mark
parity_space

7.3.2.4 enum serial::stopbits_t

Enumeration defines the possible stopbit types for the serial port.

Enumerator

stopbits_one
stopbits_two
stopbits_one_point_five

7.3.3 Function Documentation

7.3.3.1 static bool serial::is_standardbaudrate (unsigned long *baudrate*, speed_t & *baud*) [inline], [static]

7.3.3.2 std::vector<PortInfo> serial::list_ports ()

7.3.3.3 static void serial::set_common_props (termios * *tio*) [inline], [static]

7.3.3.4 static void serial::set_databits (termios * *tio*, serial::bytesize_t *databits*) [inline], [static]

7.3.3.5 static void serial::set_flowcontrol (termios * *tio*, serial::flowcontrol_t *flowcontrol*) [inline], [static]

7.3.3.6 static void serial::set_parity (termios * *tio*, serial::parity_t *parity*) [inline], [static]

7.3.3.7 static void serial::set_stopbits (termios * *tio*, serial::stopbits_t *stopbits*) [inline], [static]

7.3.3.8 timespec serial::timespec_from_ms (const uint32_t *millis*)

7.4 ydlidar Namespace Reference

Classes

- class [YDlidarDriver](#)

Enumerations

- enum [YDLIDAR_MODLES](#) {
[YDLIDAR_F4](#) = 1, [YDLIDAR_T1](#) = 2, [YDLIDAR_F2](#) = 3, [YDLIDAR_S4](#) = 4,
[YDLIDAR_G4](#) = 5, [YDLIDAR_X4](#) = 6, [YDLIDAR_G4PRO](#) = 7, [YDLIDAR_F4PRO](#) = 8,
[YDLIDAR_R2](#) = 9, [YDLIDAR_G10](#) = 10, [YDLIDAR_S4B](#) = 11, [YDLIDAR_S2](#) = 12,
[YDLIDAR_G6](#) = 13, [YDLIDAR_G2A](#) = 14, [YDLIDAR_G2B](#) = 15, [YDLIDAR_G2C](#) = 16,
[YDLIDAR_G4B](#) = 17, [YDLIDAR_G4C](#) = 18, [YDLIDAR_G1](#) = 19, [YDLIDAR_TG15](#) = 100,
[YDLIDAR_TG30](#) = 101, [YDLIDAR_TG50](#) = 102, [YDLIDAR_Tail](#) }
- enum [YDLIDAR_RATE](#) { [YDLIDAR_RATE_4K](#) = 0, [YDLIDAR_RATE_8K](#) = 1, [YDLIDAR_RATE_9K](#) = 2, [YDLIDAR_RATE_10K](#) = 3 }

Functions

- `std::string lidarModelToString` (int `model`)
lidarModelToString
- `int lidarModelDefaultSampleRate` (int `model`)
lidarModelDefaultSampleRate
- `bool isOctaveLidar` (int `model`)
isOctaveLidar
- `bool hasSampleRate` (int `model`)
hasSampleRate
- `bool hasZeroAngle` (int `model`)
hasZeroAngle
- `bool hasScanFrequencyCtrl` (int `model`)
hasScanFrequencyCtrl
- `bool isSupportLidar` (int `model`)
isSupportLidar
- `bool hasIntensity` (int `model`)
hasIntensity
- `bool isSupportMotorCtrl` (int `model`)
isSupportMotorCtrl
- `bool isSupportScanFrequency` (int `model`, double `frequency`)
isSupportScanFrequency
- `bool isTOFLidar` (int `type`)
- `bool isOldVersionTOFLidar` (int `model`, int `Major`, int `Minor`)
- `bool isValidSampleRate` (std::map< int, int > `smap`)
- `int ConvertUserToLidarSmaple` (int `model`, int `m_SampleRate`, int `defaultRate`)
- `int ConvertLidarToUserSmaple` (int `model`, int `rate`)
- `bool isValidValue` (uint8_t `value`)
- `bool isVersionValid` (const `LaserDebug` &`info`)
- `bool isSerialNumbValid` (const `LaserDebug` &`info`)
- `bool ParseLaserDebugInfo` (const `LaserDebug` &`info`, `device_info` &`value`)
- `void init` (int `argc`, char *`argv`[])
- `bool ok` ()
- `void shutdownNow` ()
- `bool fileExists` (const std::string `filename`)

7.4.1 Enumeration Type Documentation

7.4.1.1 enum ydlidar::YDLIDAR_MODLES

Enumerator

- YDLIDAR_F4** F4雷达型号代号.
- YDLIDAR_T1** T1雷达型号代号.
- YDLIDAR_F2** F2雷达型号代号.
- YDLIDAR_S4** S4雷达型号代号.
- YDLIDAR_G4** G4雷达型号代号.
- YDLIDAR_X4** X4雷达型号代号.
- YDLIDAR_G4PRO** G4PRO雷达型号代号.
- YDLIDAR_F4PRO** F4PRO雷达型号代号.

YDLIDAR_R2 R2雷达型号代号.

YDLIDAR_G10 G10雷达型号代号.

YDLIDAR_S4B S4B雷达型号代号.

YDLIDAR_S2 S2雷达型号代号.

YDLIDAR_G6 G6雷达型号代号.

YDLIDAR_G2A G2A雷达型号代号.

YDLIDAR_G2B G2雷达型号代号.

YDLIDAR_G2C G2C雷达型号代号.

YDLIDAR_G4B G4B雷达型号代号.

YDLIDAR_G4C G4C雷达型号代号.

YDLIDAR_G1 G1雷达型号代号.

YDLIDAR_TG15 TG15雷达型号代号.

YDLIDAR_TG30 T30雷达型号代号.

YDLIDAR_TG50 TG50雷达型号代号.

YDLIDAR_Tail

7.4.1.2 enum ydlidar::YDLIDAR_RATE

Enumerator

YDLIDAR_RATE_4K

YDLIDAR_RATE_8K

YDLIDAR_RATE_9K

YDLIDAR_RATE_10K

7.4.2 Function Documentation

7.4.2.1 int ydlidar::ConvertLidarToUserSmample (int *model*, int *rate*) [inline]

7.4.2.2 int ydlidar::ConvertUserToLidarSmample (int *model*, int *m_SampleRate*, int *defaultRate*) [inline]

7.4.2.3 bool ydlidar::fileExists (const std::string *filename*) [inline]

7.4.2.4 bool ydlidar::hasIntensity (int *model*) [inline]

hasIntensity

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.5 `bool ydlidar::hasSampleRate (int model)` `[inline]`

hasSampleRate

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.6 `bool ydlidar::hasScanFrequencyCtrl (int model)` `[inline]`

hasScanFrequencyCtrl

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.7 `bool ydlidar::hasZeroAngle (int model)` `[inline]`

hasZeroAngle

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.8 `void ydlidar::init (int argc, char * argv[])` `[inline]`

7.4.2.9 `bool ydlidar::isOctaveLidar (int model)` `[inline]`

isOctaveLidar

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.10 `bool ydlidar::isOldVersionTOFLidar (int model, int Major, int Minor)` `[inline]`

7.4.2.11 `bool ydlidar::isSerialNumbValid (const LaserDebug & info)` `[inline]`

7.4.2.12 `bool ydlidar::isSupportLidar (int model)` `[inline]`

isSupportLidar

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.13 `bool ydlidar::isSupportMotorCtrl (int model)` `[inline]`

isSupportMotorCtrl

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.14 `bool ydlidar::isSupportScanFrequency (int model, double frequency)` `[inline]`

isSupportScanFrequency

Parameters

<i>model</i>	
<i>frequency</i>	

Returns

7.4.2.15 `bool ydlidar::isTOFLidar (int type)` `[inline]`

7.4.2.16 `bool ydlidar::isValidSampleRate (std::map< int, int > smap)` `[inline]`

7.4.2.17 `bool ydlidar::isValidValue (uint8_t value)` `[inline]`

7.4.2.18 `bool ydlidar::isVersionValid (const LaserDebug & info)` `[inline]`

7.4.2.19 `int ydlidar::lidarModelDefaultSampleRate (int model)` `[inline]`

`lidarModelDefaultSampleRate`

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.20 `std::string ydlidar::lidarModelToString (int model)` `[inline]`

`lidarModelToString`

Parameters

<i>model</i>	
--------------	--

Returns

7.4.2.21 `bool ydlidar::ok ()` `[inline]`

7.4.2.22 `bool ydlidar::ParseLaserDebugInfo (const LaserDebug & info, device_info & value)` `[inline]`

7.4.2.23 `void ydlidar::shutdownNow ()` `[inline]`

Chapter 8

Class Documentation

8.1 cmd_packet Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- uint8_t [syncByte](#)
- uint8_t [cmd_flag](#)
- uint8_t [size](#)
- uint8_t [data](#)

8.1.1 Member Data Documentation

8.1.1.1 uint8_t cmd_packet::cmd_flag

8.1.1.2 uint8_t cmd_packet::data

8.1.1.3 uint8_t cmd_packet::size

8.1.1.4 uint8_t cmd_packet::syncByte

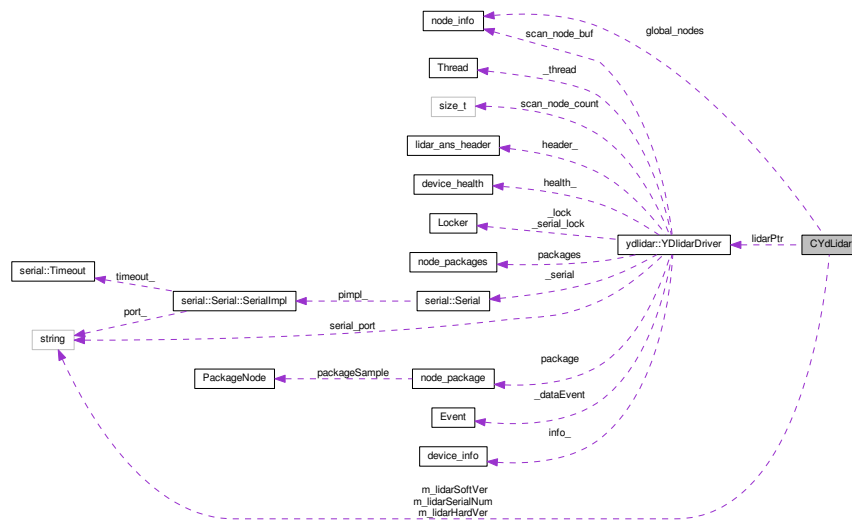
The documentation for this struct was generated from the following file:

- include/[ydlidar_protocol.h](#)

8.2 CYdLidar Class Reference

```
#include <CYdLidar.h>
```

Collaboration diagram for CYdLidar:



Public Member Functions

- [CYdLidar](#) ()
Constructor.
- virtual [~CYdLidar](#) ()
Destructor: turns the laser off.
- bool [initialize](#) ()
initialize
- bool [doProcessSimple](#) ([LaserScan](#) &outscan, bool &hardwareError)
- bool [turnOn](#) ()
See base class docs.
- bool [turnOff](#) ()
See base class docs.
- void [disconnecting](#) ()
Closes the comms with the laser. Shouldn't have to be directly needed by the user.
- float [getAngleOffset](#) () const
- bool [isAngleOffsetCorrected](#) () const
- std::string [getSoftVersion](#) () const
get lidar software version
- std::string [getHardwareVersion](#) () const
get lidar hardware version
- std::string [getSerialNumber](#) () const
get lidar serial number

Protected Member Functions

- bool [checkCOMMs](#) ()
- bool [checkStatus](#) ()
- bool [checkHardware](#) ()
- bool [getDeviceHealth](#) ()
- bool [getDeviceInfo](#) ()
- void [checkSampleRate](#) ()
checkSampleRate
- bool [CalculateSampleRate](#) (int count, double scan_time)
CalculateSampleRate.
- bool [checkScanFrequency](#) ()
- bool [checkLidarAbnormal](#) ()
- void [checkCalibrationAngle](#) (const std::string &serialNumber)
checkCalibrationAngle
- bool [isRangeValid](#) (double reading) const
isRangeValid
- bool [isRangeIgnore](#) (double angle) const
isRangeIgnore
- void [handleSingleChannelDevice](#) ()
handleSingleChannelDevice
- void [parsePackageNode](#) (const [node_info](#) &node, [LaserDebug](#) &info)
parsePackageNode
- void [handleDeviceInfoPackage](#) (int count)
handleDeviceInfoPackage
- void [printfVersionInfo](#) (const [device_info](#) &info)
printfVersionInfo

Private Member Functions

- [PropertyBuilderByName](#) (float, MaxRange, [private](#)) ; [PropertyBuilderByName](#)(float
Set and Get LiDAR Maximum effective range.
- [PropertyBuilderByName](#) (float, MaxAngle, [private](#))
Set and Get LiDAR Maximum effective angle.
- [PropertyBuilderByName](#) (float, MinAngle, [private](#))
Set and Get LiDAR Minimum effective angle.
- [PropertyBuilderByName](#) (int, SampleRate, [private](#))
Set and Get LiDAR Sampling rate.
- [PropertyBuilderByName](#) (float, ScanFrequency, [private](#))
Set and Get LiDAR Scan frequency.
- [PropertyBuilderByName](#) (bool, FixedResolution, [private](#))
Set and Get LiDAR Fixed angular resolution.
.
- [PropertyBuilderByName](#) (bool, Reversion, [private](#))
Set and Get LiDAR Reversion.
true: LiDAR data rotated 180 degrees.
false: Keep raw Data.
default: false
.
- [PropertyBuilderByName](#) (bool, Inverted, [private](#))

Set and Get LiDAR inverted.
true: Data is counterclockwise
false: Data is clockwise
Default: clockwise.

- [PropertyBuilderByName](#) (bool, AutoReconnect, [private](#))
Set and Get LiDAR Automatically reconnect flag.
Whether to support hot plug.
- [PropertyBuilderByName](#) (int, SerialBaudrate, [private](#))
Set and Get LiDAR baudrate or network port.
- [PropertyBuilderByName](#) (int, AbnormalCheckCount, [private](#))
Set and Get LiDAR Maximum number of abnormal checks.
- [PropertyBuilderByName](#) (std::string, SerialPort, [private](#))
Set and Get LiDAR Serial port or network IP address.
- [PropertyBuilderByName](#) (std::vector< float >, IgnoreArray, [private](#))
Set and Get LiDAR filtering angle area.
- [PropertyBuilderByName](#) (float, OffsetTime, [private](#))
- [PropertyBuilderByName](#) (bool, SingleChannel, [private](#))
Set and Get LiDAR single channel. Whether LiDAR communication channel is a single-channel.
- [PropertyBuilderByName](#) (int, LidarType, [private](#))
Set and Get LiDAR Type.

Private Attributes

- [MinRange](#)
- [private](#)
- bool [isScanning](#)
- int [m_FixedSize](#)
- float [m_AngleOffset](#)
- bool [m_isAngleOffsetCorrected](#)
- float [frequencyOffset](#)
- int [lidar_model](#)
- uint8_t [Major](#)
- uint8_t [Minjor](#)
- [YDLidarDriver](#) * [lidarPtr](#)
- uint64_t [m_PointTime](#)
- uint64_t [last_node_time](#)
- [node_info](#) * [global_nodes](#)
- std::map< int, int > [SampleRateMap](#)
- bool [m_ParseSuccess](#)
- std::string [m_lidarSoftVer](#)
- std::string [m_lidarHardVer](#)
- std::string [m_lidarSerialNum](#)
- int [defalutSampleRate](#)
- int [m_UserSampleRate](#)

8.2.1 Detailed Description

"Dataset"

LIDAR	Model	Baudrate	Sample↔ Rate(K)	Range(m)	Frequency (HZ)	Intenstiy(t	Single↔ Channel	voltage(↔ V)
F4	1	115200	4	0.12~12	5~12	false	false	4.8~5.2
S4	4	115200	4	0.↔ 10~8.0	5~12 (PWM)	false	false	4.8~5.2
S4B	4/11	153600	4	0.↔ 10~8.0	5~12(P↔ WM)	true(8)	false	4.8~5.2
S2	4/12	115200	3	0.↔ 10~8.0	4~8(P↔ WM)	false	true	4.8~5.2
G4	5	230400	9/8/4	0.↔ 28/0.26/0.↔ 1~16	5~12	false	false	4.8~5.2
X4	6	128000	5	0.12~10	5~12(P↔ WM)	false	false	4.8~5.2
X2/X2L	6	115200	3	0.↔ 10~8.0	4~8(P↔ WM)	false	true	4.8~5.2
G4PRO	7	230400	9/8/4	0.↔ 28/0.26/0.↔ 1~16	5~12	false	false	4.8~5.2
F4PRO	8	230400	4/6	0.12~12	5~12	false	false	4.8~5.2
R2	9	230400	5	0.12~16	5~12	false	false	4.8~5.2
G6	13	512000	18/16/8	0.↔ 28/0.26/0.↔ 1~25	5~12	false	false	4.8~5.2
G2A	14	230400	5	0.12~12	5~12	false	false	4.8~5.2
G2	15	230400	5	0.28~16	5~12	true(8)	false	4.8~5.2
G2C	16	115200	4	0.1~12	5~12	false	false	4.8~5.2
G4B	17	512000	10	0.12~16	5~12	true(10)	false	4.8~5.2
G4C	18	115200	4	0.1~12	5~12	false	false	4.8~5.2
G1	19	230400	9	0.28~16	5~12	false	false	4.8~5.2
TX8	100	115200	4	0.01~8	4~8(P↔ WM)	false	true	4.8~5.2
TX20	100	115200	4	0.01~8	4~8(P↔ WM)	false	true	4.8~5.2
TG15	100	512000	20/18/10	0.01~30	3~16	false	false	4.8~5.2
TG30	101	512000	20/18/10	0.01~30	3~16	false	false	4.8~5.2
TG50	102	512000	20/18/10	0.01~50	3~16	false	false	4.8~5.2

Dataset:

example: G4 LiDAR

```

CYdLidar laser;
laser.setMaxAngle(180);
laser.setMinAngle(-180);
laser.setMinRange(0.1);
laser.setMaxRange(16.0);
laser.setSerialPort("/dev/ydlidar");
laser.setSerialBaudrate(230400);
laser.setFixedResolution(false);
laser.setReversion(true);
laser.setInverted(true);
laser.setScanFrequency(10.0);
laser.setSampleRate(9);
laser.setAutoReconnect(true);
std::vector<float> ignore_array;
laser.setIgnoreArray(ignore_array);
laser.setSingleChannel(false);
laser.setLidarType(TYPE_TRIANGLE);
laser.setDeviceType(YDLIDAR_TYPE_SERIAL);

```

```
laser.setIntensity(false);  
laser.setAbnormalCheckCount(4);  
laser.setSupportMotorDtrCtrl(false);
```

example: S2 LiDAR

```
CYdLidar laser;  
laser.setMaxAngle(180);  
laser.setMinAngle(-180);  
laser.setMinRange(0.1);  
laser.setMaxRange(8.0);  
laser.setSerialPort("/dev/ydlidar");  
laser.setSerialBaudrate(115200);  
laser.setFixedResolution(false);  
laser.setReversion(false);  
laser.setInverted(true);  
laser.setScanFrequency(6.0);  
laser.setSampleRate(3);  
laser.setAutoReconnect(true);  
std::vector<float> ignore_array;  
laser.setIgnoreArray(ignore_array);  
laser.setSingleChannel(true);  
laser.setLidarType(TYPE_TRIANGLE);  
laser.setDeviceType(YDLIDAR_TYPE_SERIAL);  
laser.setIntensity(false);  
laser.setAbnormalCheckCount(4);  
laser.setSupportMotorDtrCtrl(true);
```

example: TG30 LiDAR

```
CYdLidar laser;  
laser.setMaxAngle(180);  
laser.setMinAngle(-180);  
laser.setMinRange(0.01);  
laser.setMaxRange(32.0);  
laser.setSerialPort("/dev/ydlidar");  
laser.setSerialBaudrate(512000);  
laser.setFixedResolution(false);  
laser.setReversion(true);  
laser.setInverted(true);  
laser.setScanFrequency(10.0);  
laser.setSampleRate(20);  
laser.setAutoReconnect(true);  
std::vector<float> ignore_array;  
laser.setIgnoreArray(ignore_array);  
laser.setSingleChannel(false);  
laser.setLidarType(TYPE_TOF);  
laser.setDeviceType(YDLIDAR_TYPE_SERIAL);  
laser.setIntensity(false);  
laser.setAbnormalCheckCount(4);  
laser.setSupportMotorDtrCtrl(false);
```

example: TX8 LiDAR

```
CYdLidar laser;  
laser.setMaxAngle(180);  
laser.setMinAngle(-180);  
laser.setMinRange(0.1);  
laser.setMaxRange(8.0);  
laser.setSerialPort("/dev/ydlidar");  
laser.setSerialBaudrate(115200);  
laser.setFixedResolution(false);  
laser.setReversion(false);  
laser.setInverted(true);  
laser.setScanFrequency(6.0);  
laser.setSampleRate(4);  
laser.setAutoReconnect(true);  
std::vector<float> ignore_array;  
laser.setIgnoreArray(ignore_array);  
laser.setSingleChannel(true);  
laser.setLidarType(TYPE_TOF);  
laser.setDeviceType(YDLIDAR_TYPE_SERIAL);  
laser.setIntensity(false);  
laser.setAbnormalCheckCount(4);  
laser.setSupportMotorDtrCtrl(true);
```

example: T15 LiDAR

```
CYdLidar laser;
```



```

laser.setMaxAngle(180);
laser.setMinAngle(-180);
laser.setMinRange(0.01);
laser.setMaxRange(64.0);
laser.setSerialPort("192.168.1.11");
laser.setSerialBaudrate(8000);
laser.setFixedResolution(false);
laser.setReversion(true);
laser.setInverted(true);
laser.setScanFrequency(20.0);
laser.setSampleRate(20);
laser.setAutoReconnect(true);
std::vector<float> ignore_array;
laser.setIgnoreArray(ignore_array);
laser.setSingleChannel(false);
laser.setLidarType(TYPE_TOF_NET);
laser.setDeviceType(YDLIDAR_TYPE_TCP);
laser.setIntensity(true);
laser.setAbnormalCheckCount(4);
laser.setSupportMotorDtrCtrl(false);

```

Provides a platform independent class to for LiDAR development. This class is designed to serial or socket communication development in a platform independent manner.

- LiDAR types
 1. [ydlidar::YDLidarDriver](#) Class
 2. [ydlidar::ETLidarDriver](#) Class

8.2.2 Constructor & Destructor Documentation

8.2.2.1 CYdLidar::CYdLidar ()

Constructor.

8.2.2.2 CYdLidar::~~CYdLidar () [virtual]

Destructor: turns the laser off.

8.2.3 Member Function Documentation

8.2.3.1 bool CYdLidar::CalculateSampleRate (int *count*, double *scan_time*) [protected]

CalculateSampleRate.

Parameters

<i>count</i>	
--------------	--

Returns

8.2.3.2 void CYdLidar::checkCalibrationAngle (const std::string & *serialNumber*) [protected]

checkCalibrationAngle

Parameters

<i>serialNumber</i>	
---------------------	--

8.2.3.3 `bool CYdLidar::checkCOMMs ()` [protected]

Returns true if communication has been established with the device. If it's not, try to create a comms channel.

Returns

false on error.

8.2.3.4 `bool CYdLidar::checkHardware ()` [protected]

Returns true if the normal scan runs with the device. If it's not,

Returns

false on error.

8.2.3.5 `bool CYdLidar::checkLidarAbnormal ()` [protected]

returns true if the lidar data is normal, If it's not

8.2.3.6 `void CYdLidar::checkSampleRate ()` [protected]

checkSampleRate

8.2.3.7 `bool CYdLidar::checkScanFrequency ()` [protected]

Retruns true if the scan frequency is set to user's frequency is successful, If it's not

8.2.3.8 `bool CYdLidar::checkStatus ()` [protected]

Returns true if health status and device information has been obtained with the device. If it's not,

Returns

false on error.

8.2.3.9 `void CYdLidar::disconnecting ()`

Closes the comms with the laser. Shouldn't have to be directly needed by the user.

8.2.3.10 `bool CYdLidar::doProcessSimple (LaserScan & outscan, bool & hardwareError)`

8.2.3.11 `float CYdLidar::getAngleOffset () const`

8.2.3.12 `bool CYdLidar::getDeviceHealth () [protected]`

Returns true if the device is in good health, If it's not

Returns true if the device is connected & operative

8.2.3.13 `bool CYdLidar::getDeviceInfo () [protected]`

Returns true if the device information is correct, If it's not

8.2.3.14 `std::string CYdLidar::getHardwareVersion () const`

get lidar hardware version

8.2.3.15 `std::string CYdLidar::getSerialNumber () const`

get lidar serial number

8.2.3.16 `std::string CYdLidar::getSoftVersion () const`

get lidar software version

8.2.3.17 `void CYdLidar::handleDeviceInfoPackage (int count) [protected]`

handleDeviceInfoPackage

Parameters

<i>count</i>	
--------------	--

8.2.3.18 `void CYdLidar::handleSingleChannelDevice () [protected]`

handleSingleChannelDevice

8.2.3.19 `bool CYdLidar::initialize ()`

initialize

to connect and turns the laser on. Raises an exception on error.

8.2.3.20 `bool CYdLidar::isAngleOffsetCorrected () const`

8.2.3.21 `bool CYdLidar::isRangeIgnore (double angle) const` [protected]

isRangeIgnore

Parameters

<i>angle</i>	
--------------	--

Returns

8.2.3.22 `bool CYdLidar::isRangeValid (double reading) const` [protected]

isRangeValid

Parameters

<i>reading</i>	
----------------	--

Returns

8.2.3.23 `void CYdLidar::parsePackageNode (const node_info & node, LaserDebug & info)` [protected]

parsePackageNode

Parameters

<i>node</i>	
<i>info</i>	

8.2.3.24 `void CYdLidar::printfVersionInfo (const device_info & info)` [protected]

printfVersionInfo

Parameters

<i>info</i>	
-------------	--

8.2.3.25 CYdLidar::PropertyBuilderByName (float , MaxRange , private) [private]

Set and Get LiDAR Maximum effective range.

Note

The effective range beyond the maximum is set to zero.
the MaxRange should be greater than the MinRange.

Remarks

unit: m

See also

[PropertyBuilderByName](#) and DataSet
CYdLidar::setMaxRange and CYdLidar::getMaxRange Set and Get LiDAR Minimum effective range.

Note

The effective range less than the minimum is set to zero.
the MinRange should be less than the MaxRange.

Remarks

unit: m

See also

[PropertyBuilderByName](#) and Dataset
CYdLidar::setMinRange and CYdLidar::getMinRange

8.2.3.26 CYdLidar::PropertyBuilderByName (float , MaxAngle , private) [private]

Set and Get LiDAR Maximum effective angle.

Note

The effective angle beyond the maximum will be ignored.
the MaxAngle should be greater than the MinAngle

Remarks

unit: degree, Range:-180~180

See also

[PropertyBuilderByName](#) and Dataset
CYdLidar::setMaxAngle and CYdLidar::getMaxAngle

8.2.3.27 CYdLidar::PropertyBuilderByName (float , MinAngle , private) [private]

Set and Get LiDAR Minimum effective angle.

Note

The effective angle less than the minmum will be ignored.
the MinAngle should be less than the MaxAngle

Remarks

unit: degree, Range:-180~180

See also

[PropertyBuilderByName](#) and Dataset
CYdLidar::setMinAngle and CYdLidar::getMinAngle

8.2.3.28 CYdLidar::PropertyBuilderByName (int , SampleRate , private) [private]

Set and Get LiDAR Sampling rate.

Note

If the set sampling rate does no exist. the actual sampling rate is the LiDAR's default sampling rate.
Set the sampling rate to match the LiDAR.

Remarks

unit: kHz/s, Ranges: 2,3,4,5,6,8,9,10,16,18,20

G4/F4	4,8,9
F4PRO	4,6
G6	8,16,18
G4B	10
G1	9
G2A/G2/R2/X4	5
S4/S4B/G4C/TX8/TX20	4
G2C	4
S2	3
TG15/TG30/TG50	10,18,20
T5/T15	20

See also

CYdLidar::setSampleRate and CYdLidar::getSampleRate

8.2.3.29 CYdLidar::PropertyBuilderByName (float , ScanFrequency , private) [private]

Set and Get LiDAR Scan frequency.

Note

If the LiDAR is a single channel, the scanning frequency needs to be adjusted by external PWM.
Set the scan frequency to match the LiDAR.

Remarks

unit: Hz

S2/X2/X2L/TX8/TX20	4~8(PWM)
F4/F4PRO/G4/G4PRO/R2	5~12
G6/G2A/G2/G2C/G4B/G4C/G1	5~12
S4/S4B/X4	5~12(PWM)
TG15/TG30/TG50	3~16
T5/T15	5~40

See also

CYdLidar::setScanFrequency and CYdLidar::getScanFrequency

8.2.3.30 CYdLidar::PropertyBuilderByName (bool , FixedResolution , private) [private]

Set and Get LiDAR Fixed angular resolution.

Note

The Lidar scanning frequency will change slightly due to various reasons. so the number of points per circle will also change slightly.
if a fixed angular resolution is required. a fixed number of points is required.

If set to true, the angle_increment of the fixed angle resolution in [LaserConfig](#) will be a fixed value.

See also

CYdLidar::setFixedResolution and CYdLidar::getFixedResolution

8.2.3.31 CYdLidar::PropertyBuilderByName (bool , Reversion , private) [private]

Set and Get LiDAR Reversion.

true: LiDAR data rotated 180 degrees.

false: Keep raw Data.

default: false

Note

Refer to the table below for the LiDAR Reversion.

This is currently related to your coordinate system and install direction. Whether to reverse it depends on your actual scene.

LiDAR	reversion
G1/G2/G2A/G2C/F4/F4PRO/R2	true
G4/G4PRO/G4B/G4C/G6	true
TG15/TG30/TG50	true
T5/T15	true
S2/X2/X2L/X4/S4/S4B	false
TX8/TX20	false

Reversion Table

See also

CYdLidar::setReversion and CYdLidar::getReversion

8.2.3.32 CYdLidar::PropertyBuilderByName (bool, Inverted, private) [private]

Set and Get LiDAR inverted.
 true: Data is counterclockwise
 false: Data is clockwise
 Default: clockwise.

Note

If set to true, LiDAR data direction is positive counterclockwise. otherwise it is positive clockwise.

See also

CYdLidar::setInverted and CYdLidar::getInverted

8.2.3.33 CYdLidar::PropertyBuilderByName (bool, AutoReconnect, private) [private]

Set and Get LiDAR Automatically reconnect flag.
 Whether to support hot plug.

See also

CYdLidar::setAutoReconnect and CYdLidar::getAutoReconnect

8.2.3.34 CYdLidar::PropertyBuilderByName (int, SerialBaudrate, private) [private]

Set and Get LiDAR baudrate or network port.

Note

Refer to the table below for the LiDAR Baud Rate.
 Set the baudrate or network port to match the LiDAR.

F4/S2/X2/X2L/S4/TX8/TX20/G4C	115200
X4	128000
S4B	153600
G1/G2/R2/G4/G4PRO/F4PRO	230400
G2A/G2C	230400
G6/G4B/TG15/TG30/TG50	512000
T5/T15(network)	8000

Remarks

See also

CYdLidar::setSerialBaudrate and CYdLidar::getSerialBaudrate

8.2.3.35 CYdLidar::PropertyBuilderByName (int , AbnormalCheckCount , private) [private]

Set and Get LiDAR Maximum number of abnormal checks.

Note

When the LiDAR Turn On, if the number of times of abnormal data acquisition is greater than the current AbnormalCheckCount, the LiDAR Fails to Turn On.

The Minimum abnormal value is Two, if it is less than the Minimum Value, it will be set to the Minimum Value.

See also

CYdLidar::setAbnormalCheckCount and CYdLidar::getAbnormalCheckCount

8.2.3.36 CYdLidar::PropertyBuilderByName (std::string , SerialPort , private) [private]

Set and Get LiDAR Serial port or network IP address.

Note

If it is serial port, your need to ensure that the serial port had read and write permissions.
If it is a network, make sure the network can ping.

See also

CYdLidar::setSerialPort and CYdLidar::getSerialPort

8.2.3.37 CYdLidar::PropertyBuilderByName (std::vector< float > , IgnoreArray , private) [private]

Set and Get LiDAR filtering angle area.

Note

If the LiDAR angle is in the IgnoreArray, the current range will be set to zero.
Filtering angles need to appear in pairs.

The purpose of the current paramter is to filter out the angular area set by user

example: Filters 10 degrees to 30 degrees and 80 degrees to 90 degrees.

```
CYdLidar laser;//Defining an CYdLidar instance.
std::vector<float> ignore_array;
ignore_array.push_back(10.0);
ignore_array.push_back(30.0);
ignore_array.push_back(80.0);
ignore_array.push_back(90.0);
laser.setIgnoreArray(ignore_array);
```

See also

CYdLidar::setIgnoreArray and CYdLidar::getIgnoreArray

8.2.3.38 CYdLidar::PropertyBuilderByName (float , OffsetTime , private) [private]

8.2.3.39 CYdLidar::PropertyBuilderByName (bool , SingleChannel , private) [private]

Set and Get LiDAR single channel. Whether LiDAR communication channel is a single-channel.

Note

For a single-channel LiDAR, if the settings are reversed.
an error will occur in obtaining device information and the LiDAR will Faied to Start.
For dual-channel LiDAR, if th setttings are reversed.
the device information cannot be obtained.
Set the single channel to match the LiDAR.

G1/G2/G2A/G2C	false
G4/G4B/G4PRO/G6/F4/F4PRO	false
S4/S4B/X4/R2/G4C	false
S2/X2/X2L	true
TG15/TG30/TG50	false
TX8/TX20	true
T5/T15	false
	true

Remarks

See also

CYdLidar::setSingleChannel and CYdLidar::getSingleChannel

8.2.3.40 CYdLidar::PropertyBuilderByName (int, LidarType, private) [private]

Set and Get LiDAR Type.

Note

Refer to the table below for the LiDAR Type.
Set the LiDAR Type to match the LiDAR.

G1/G2A/G2/G2C	TYPE_TRIANGLE
G4/G4B/G4C/G4PRO	TYPE_TRIANGLE
G6/F4/F4PRO	TYPE_TRIANGLE
S4/S4B/X4/R2/S2/X2/X2L	TYPE_TRIANGLE
TG15/TG30/TG50/TX8/TX20	TYPE_TOF
T5/T15	TYPE_TOF_NET

Remarks

See also

[LidarTypeID](#)

CYdLidar::setLidarType and CYdLidar::getLidarType

8.2.3.41 bool CYdLidar::turnOff ()

See base class docs.

8.2.3.42 bool CYdLidar::turnOn ()

See base class docs.

8.2.4 Member Data Documentation

- 8.2.4.1 `int CYdLidar::defalutSampleRate` [private]
- 8.2.4.2 `float CYdLidar::frequencyOffset` [private]
- 8.2.4.3 `node_info* CYdLidar::global_nodes` [private]
- 8.2.4.4 `bool CYdLidar::isScanning` [private]
- 8.2.4.5 `uint64_t CYdLidar::last_node_time` [private]
- 8.2.4.6 `int CYdLidar::lidar_model` [private]
- 8.2.4.7 `YDLidarDriver* CYdLidar::lidarPtr` [private]
- 8.2.4.8 `float CYdLidar::m_AngleOffset` [private]
- 8.2.4.9 `int CYdLidar::m_FixedSize` [private]
- 8.2.4.10 `bool CYdLidar::m_isAngleOffsetCorrected` [private]
- 8.2.4.11 `std::string CYdLidar::m_lidarHardVer` [private]
- 8.2.4.12 `std::string CYdLidar::m_lidarSerialNum` [private]
- 8.2.4.13 `std::string CYdLidar::m_lidarSoftVer` [private]
- 8.2.4.14 `bool CYdLidar::m_ParseSuccess` [private]
- 8.2.4.15 `uint64_t CYdLidar::m_PointTime` [private]
- 8.2.4.16 `int CYdLidar::m_UserSampleRate` [private]
- 8.2.4.17 `uint8_t CYdLidar::Major` [private]
- 8.2.4.18 `uint8_t CYdLidar::Minjor` [private]
- 8.2.4.19 `CYdLidar::MinRange` [private]
- 8.2.4.20 `CYdLidar::private` [private]
- 8.2.4.21 `std::map<int, int> CYdLidar::SampleRateMap` [private]

The documentation for this class was generated from the following files:

- [include/CYdLidar.h](#)
- [src/CYdLidar.cpp](#)

8.3 device_health Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t status](#)
健康状态
- [uint16_t error_code](#)
错误代码

8.3.1 Member Data Documentation

8.3.1.1 uint16_t device_health::error_code

错误代码

8.3.1.2 uint8_t device_health::status

健康状态

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.4 device_info Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t model](#)
雷达型号
- [uint16_t firmware_version](#)
固件版本号
- [uint8_t hardware_version](#)
硬件版本号
- [uint8_t serialnum](#) [16]
序列号

8.4.1 Member Data Documentation

8.4.1.1 uint16_t device_info::firmware_version

固件版本号

8.4.1.2 uint8_t device_info::hardware_version

硬件版本号

8.4.1.3 uint8_t device_info::model

雷达型号

8.4.1.4 uint8_t device_info::serialnum[16]

系列号

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.5 Event Class Reference

```
#include <locker.h>
```

Public Types

- enum { [EVENT_OK](#) = 1, [EVENT_TIMEOUT](#) = 2, [EVENT_FAILED](#) = 0 }

Public Member Functions

- [Event](#) (bool isAutoReset=true, bool isSignal=false)
- [~Event](#) ()
- void [set](#) (bool isSignal=true)
- unsigned long [wait](#) (unsigned long timeout=0xFFFFFFFF)

Protected Member Functions

- void [release](#) ()

Protected Attributes

- pthread_condattr_t [_cond_cattr](#)
- pthread_cond_t [_cond_var](#)
- pthread_mutex_t [_cond_locker](#)
- bool [_is_signalled](#)
- bool [_isAutoReset](#)

8.5.1 Member Enumeration Documentation

8.5.1.1 anonymous enum

Enumerator

EVENT_OK
EVENT_TIMEOUT
EVENT_FAILED

8.5.2 Constructor & Destructor Documentation

8.5.2.1 `Event::Event (bool isAutoReset = true, bool isSignal = false)` [inline],[explicit]

8.5.2.2 `Event::~~Event ()` [inline]

8.5.3 Member Function Documentation

8.5.3.1 `void Event::release ()` [inline],[protected]

8.5.3.2 `void Event::set (bool isSignal = true)` [inline]

8.5.3.3 `unsigned long Event::wait (unsigned long timeout = 0xFFFFFFFF)` [inline]

8.5.4 Member Data Documentation

8.5.4.1 `pthread_condattr_t Event::_cond_catr` [protected]

8.5.4.2 `pthread_mutex_t Event::_cond_locker` [protected]

8.5.4.3 `pthread_cond_t Event::_cond_var` [protected]

8.5.4.4 `bool Event::_is_signalled` [protected]

8.5.4.5 `bool Event::_isAutoReset` [protected]

The documentation for this class was generated from the following file:

- [include/locker.h](#)

8.6 function_state Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t state](#)

8.6.1 Member Data Documentation

8.6.1.1 [uint8_t function_state::state](#)

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.7 LaserConfig Struct Reference

A struct for returning configuration from the YDLIDAR.

```
#include <ydlidar_protocol.h>
```

Public Member Functions

- [LaserConfig](#) & [operator=](#) (const [LaserConfig](#) &data)

Public Attributes

- float [min_angle](#)
Start angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.
- float [max_angle](#)
Stop angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.
- float [angle_increment](#)
angle resoltuion [rad]
- float [time_increment](#)
Scan resoltuion [s].
- float [scan_time](#)
Time between scans.
- float [min_range](#)
Minimum range [m].
- float [max_range](#)
Maximum range [m].

8.7.1 Detailed Description

A struct for returning configuration from the YDLIDAR.

8.7.2 Member Function Documentation

8.7.2.1 `LaserConfig& LaserConfig::operator= (const LaserConfig & data)` `[inline]`

8.7.3 Member Data Documentation

8.7.3.1 `float LaserConfig::angle_increment`

angle resoltuion [rad]

8.7.3.2 `float LaserConfig::max_angle`

Stop angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.

8.7.3.3 `float LaserConfig::max_range`

Maximum range [m].

8.7.3.4 `float LaserConfig::min_angle`

Start angle for the laser scan [rad]. 0 is forward and angles are measured clockwise when viewing YDLIDAR from the top.

8.7.3.5 `float LaserConfig::min_range`

Minimum range [m].

8.7.3.6 `float LaserConfig::scan_time`

Time between scans.

8.7.3.7 `float LaserConfig::time_increment`

Scan resoltuion [s].

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.8 LaserDebug Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- `uint8_t` [W3F4CusMajor_W4F0CusMinor](#)
- `uint8_t` [W4F3Model_W3F0DebugInfTranVer](#)
- `uint8_t` [W3F4HardwareVer_W4F0FirewareMajor](#)
- `uint8_t` [W3F4BoradHardVer_W4F0Moth](#)
- `uint8_t` [W2F5Output2K4K5K_W5F0Date](#)
- `uint8_t` [W1F6GNoise_W1F5SNoise_W1F4MotorCtl_W4F0SnYear](#)
- `uint8_t` [W7F0SnNumH](#)
- `uint8_t` [W7F0SnNumL](#)
- `uint8_t` [MaxDebugIndex](#)

8.8.1 Member Data Documentation

8.8.1.1 `uint8_t` `LaserDebug::MaxDebugIndex`

8.8.1.2 `uint8_t` `LaserDebug::W1F6GNoise_W1F5SNoise_W1F4MotorCtl_W4F0SnYear`

8.8.1.3 `uint8_t` `LaserDebug::W2F5Output2K4K5K_W5F0Date`

8.8.1.4 `uint8_t` `LaserDebug::W3F4BoradHardVer_W4F0Moth`

8.8.1.5 `uint8_t` `LaserDebug::W3F4CusMajor_W4F0CusMinor`

8.8.1.6 `uint8_t` `LaserDebug::W3F4HardwareVer_W4F0FirewareMajor`

8.8.1.7 `uint8_t` `LaserDebug::W4F3Model_W3F0DebugInfTranVer`

8.8.1.8 `uint8_t` `LaserDebug::W7F0SnNumH`

8.8.1.9 `uint8_t` `LaserDebug::W7F0SnNumL`

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.9 LaserPoint Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Member Functions

- [LaserPoint](#) & `operator=` (const [LaserPoint](#) &data)

Public Attributes

- float [angle](#)
lidar angle [rad]
- float [range](#)
lidar range [m]
- float [intensity](#)
lidar intensity

8.9.1 Member Function Documentation

8.9.1.1 `LaserPoint& LaserPoint::operator= (const LaserPoint & data)` `[inline]`

8.9.2 Member Data Documentation

8.9.2.1 `float LaserPoint::angle`

lidar angle [rad]

8.9.2.2 `float LaserPoint::intensity`

lidar intensity

8.9.2.3 `float LaserPoint::range`

lidar range [m]

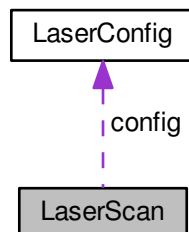
The documentation for this struct was generated from the following file:

- `include/ydlidar_protocol.h`

8.10 LaserScan Struct Reference

```
#include <ydlidar_protocol.h>
```

Collaboration diagram for LaserScan:



Public Member Functions

- [LaserScan](#) & `operator=` (const [LaserScan](#) &`data`)

Public Attributes

- `uint64_t` [stamp](#)
System time when first range was measured in nanoseconds.
- `std::vector< LaserPoint >` [points](#)
Array of lidar points.
- [LaserConfig](#) [config](#)
Configuration of scan.

8.10.1 Member Function Documentation

8.10.1.1 `LaserScan& LaserScan::operator= (const LaserScan & data)` `[inline]`

8.10.2 Member Data Documentation

8.10.2.1 `LaserConfig LaserScan::config`

Configuration of scan.

8.10.2.2 `std::vector<LaserPoint> LaserScan::points`

Array of lidar points.

8.10.2.3 `uint64_t LaserScan::stamp`

System time when first range was measured in nanoseconds.

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.11 lidar_ans_header Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- `uint8_t` [syncByte1](#)
- `uint8_t` [syncByte2](#)
- `uint32_t` [size](#): 30
- `uint32_t` [subType](#): 2
- `uint8_t` [type](#)

8.11.1 Member Data Documentation

8.11.1.1 `uint32_t lidar_ans_header::size`

8.11.1.2 `uint32_t lidar_ans_header::subType`

8.11.1.3 `uint8_t lidar_ans_header::syncByte1`

8.11.1.4 `uint8_t lidar_ans_header::syncByte2`

8.11.1.5 `uint8_t lidar_ans_header::type`

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.12 Locker Class Reference

```
#include <locker.h>
```

Public Types

- enum `LOCK_STATUS` { `LOCK_OK` = 0, `LOCK_TIMEOUT` = -1, `LOCK_FAILED` = -2 }

Public Member Functions

- `Locker` ()
- `~Locker` ()
- `Locker::LOCK_STATUS lock` (unsigned long timeout=0xFFFFFFFF)
- void `unlock` ()
- `pthread_mutex_t * getLockHandle` ()

Protected Member Functions

- void `init` ()
- void `release` ()

Protected Attributes

- `pthread_mutex_t _lock`

8.12.1 Member Enumeration Documentation

8.12.1.1 enum Locker::LOCK_STATUS

Enumerator

LOCK_OK
LOCK_TIMEOUT
LOCK_FAILED

8.12.2 Constructor & Destructor Documentation

8.12.2.1 Locker::Locker () [inline]

8.12.2.2 Locker::~~Locker () [inline]

8.12.3 Member Function Documentation

8.12.3.1 pthread_mutex_t* Locker::getLockHandle () [inline]

8.12.3.2 void Locker::init () [inline], [protected]

8.12.3.3 Locker::LOCK_STATUS Locker::lock (unsigned long *timeout* = 0xFFFFFFFF) [inline]

8.12.3.4 void Locker::release () [inline], [protected]

8.12.3.5 void Locker::unlock () [inline]

8.12.4 Member Data Documentation

8.12.4.1 pthread_mutex_t Locker::_lock [protected]

The documentation for this class was generated from the following file:

- [include/locker.h](#)

8.13 serial::MillisecondTimer Class Reference

```
#include <unix_serial.h>
```

Public Member Functions

- [MillisecondTimer](#) (const uint32_t millis)
- int64_t [remaining](#) ()

Static Private Member Functions

- static timespec [timespec_now](#) ()

Private Attributes

- timespec [expiry](#)

8.13.1 Constructor & Destructor Documentation

8.13.1.1 `serial::MillisecondTimer::MillisecondTimer (const uint32_t millis)` `[explicit]`

8.13.2 Member Function Documentation

8.13.2.1 `int64_t serial::MillisecondTimer::remaining ()`

8.13.2.2 `timespec serial::MillisecondTimer::timespec_now ()` `[static]`, `[private]`

8.13.3 Member Data Documentation

8.13.3.1 `timespec serial::MillisecondTimer::expiry` `[private]`

The documentation for this class was generated from the following files:

- `src/impl/unix/unix_serial.h`
- `src/impl/unix/unix_serial.cpp`

8.14 node_info Struct Reference

```
#include <yddlidar_protocol.h>
```

Public Attributes

- uint8_t [sync_flag](#)
- uint16_t [sync_quality](#)
- uint16_t [angle_q6_checkbit](#)
信号质量
- uint16_t [distance_q2](#)
测距点角度
- uint64_t [stamp](#)
当前测距点距离
- uint8_t [scan_frequence](#)
时间戳
- uint8_t [debug_info](#) [12]
特定版本此值才有效,无效值是0
- uint8_t [index](#)

8.14.1 Member Data Documentation

8.14.1.1 `uint16_t node_info::angle_q6_checkbit`

信号质量

8.14.1.2 `uint8_t node_info::debug_info[12]`

特定版本此值才有效,无效值是0

8.14.1.3 `uint16_t node_info::distance_q2`

测距点角度

8.14.1.4 `uint8_t node_info::index`

8.14.1.5 `uint8_t node_info::scan_frequence`

时间戳

8.14.1.6 `uint64_t node_info::stamp`

当前测距点距离

8.14.1.7 `uint8_t node_info::sync_flag`

8.14.1.8 `uint16_t node_info::sync_quality`

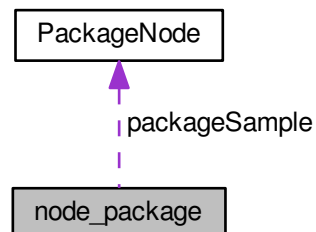
The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.15 `node_package` Struct Reference

```
#include <ydlidar_protocol.h>
```

Collaboration diagram for `node_package`:



Public Attributes

- uint16_t [package_Head](#)
- uint8_t [package_CT](#)
- uint8_t [nowPackageNum](#)
- uint16_t [packageFirstSampleAngle](#)
- uint16_t [packageLastSampleAngle](#)
- uint16_t [checksum](#)
- [PackageNode](#) [packageSample](#) [[PackageSampleMaxLngth](#)]

8.15.1 Member Data Documentation

8.15.1.1 uint16_t node_package::checksum

8.15.1.2 uint8_t node_package::nowPackageNum

8.15.1.3 uint8_t node_package::package_CT

8.15.1.4 uint16_t node_package::package_Head

8.15.1.5 uint16_t node_package::packageFirstSampleAngle

8.15.1.6 uint16_t node_package::packageLastSampleAngle

8.15.1.7 [PackageNode](#) node_package::packageSample[[PackageSampleMaxLngth](#)]

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.16 node_packages Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- uint16_t [package_Head](#)
- uint8_t [package_CT](#)
- uint8_t [nowPackageNum](#)
- uint16_t [packageFirstSampleAngle](#)
- uint16_t [packageLastSampleAngle](#)
- uint16_t [checksum](#)
- uint16_t [packageSampleDistance](#) [[PackageSampleMaxLngth](#)]

8.16.1 Member Data Documentation

8.16.1.1 `uint16_t node_packages::checkSum`

8.16.1.2 `uint8_t node_packages::nowPackageNum`

8.16.1.3 `uint8_t node_packages::package_CT`

8.16.1.4 `uint16_t node_packages::package_Head`

8.16.1.5 `uint16_t node_packages::packageFirstSampleAngle`

8.16.1.6 `uint16_t node_packages::packageLastSampleAngle`

8.16.1.7 `uint16_t node_packages::packageSampleDistance[PackageSampleMaxLngh]`

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.17 offset_angle Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- `int32_t` [angle](#)

8.17.1 Member Data Documentation

8.17.1.1 `int32_t offset_angle::angle`

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.18 PackageNode Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- uint8_t [PackageSampleQuality](#)
- uint16_t [PackageSampleDistance](#)

8.18.1 Member Data Documentation

8.18.1.1 uint16_t PackageNode::PackageSampleDistance

8.18.1.2 uint8_t PackageNode::PackageSampleQuality

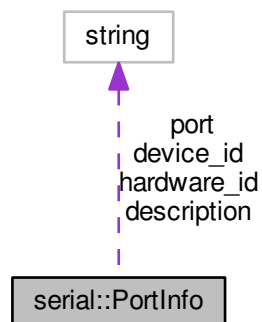
The documentation for this struct was generated from the following file:

- include/[ydlidar_protocol.h](#)

8.19 serial::PortInfo Struct Reference

```
#include <serial.h>
```

Collaboration diagram for serial::PortInfo:



Public Attributes

- std::string [port](#)
- std::string [description](#)
- std::string [hardware_id](#)
- std::string [device_id](#)

8.19.1 Detailed Description

Structure that describes a serial device.

8.19.2 Member Data Documentation

8.19.2.1 `std::string serial::PortInfo::description`

Human readable description of serial device if available.

8.19.2.2 `std::string serial::PortInfo::device_id`

Hardware Device ID or "" if not available.

8.19.2.3 `std::string serial::PortInfo::hardware_id`

Hardware ID (e.g. VID:PID of USB serial devices) or "n/a" if not available.

8.19.2.4 `std::string serial::PortInfo::port`

Address of the serial port (this can be passed to the constructor of [Serial](#)).

The documentation for this struct was generated from the following file:

- [include/serial.h](#)

8.20 `sampling_rate` Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- `uint8_t rate`
采样频率

8.20.1 Member Data Documentation

8.20.1.1 `uint8_t sampling_rate::rate`

采样频率

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.21 scan_exposure Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t exposure](#)
低光功率模式

8.21.1 Member Data Documentation

8.21.1.1 [uint8_t scan_exposure::exposure](#)

低光功率模式

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.22 scan_frequency Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint32_t frequency](#)
扫描频率

8.22.1 Member Data Documentation

8.22.1.1 [uint32_t scan_frequency::frequency](#)

扫描频率

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.23 scan_heart_beat Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t enable](#)
掉电保护状态

8.23.1 Member Data Documentation

8.23.1.1 [uint8_t scan_heart_beat::enable](#)

掉电保护状态

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.24 [scan_points](#) Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t flag](#)

8.24.1 Member Data Documentation

8.24.1.1 [uint8_t scan_points::flag](#)

The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.25 [scan_rotation](#) Struct Reference

```
#include <ydlidar_protocol.h>
```

Public Attributes

- [uint8_t rotation](#)

8.25.1 Member Data Documentation

8.25.1.1 uint8_t scan_rotation::rotation

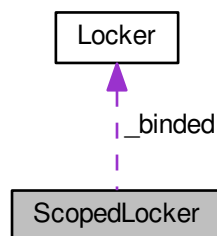
The documentation for this struct was generated from the following file:

- [include/ydlidar_protocol.h](#)

8.26 ScopedLocker Class Reference

```
#include <locker.h>
```

Collaboration diagram for ScopedLocker:



Public Member Functions

- [ScopedLocker](#) ([Locker](#) &l)
- void [forceUnlock](#) ()
- [~ScopedLocker](#) ()

Public Attributes

- [Locker](#) & [_binded](#)

8.26.1 Constructor & Destructor Documentation

8.26.1.1 `ScopedLocker::ScopedLocker (Locker & l)` [`inline`], [`explicit`]

8.26.1.2 `ScopedLocker::~~ScopedLocker ()` [`inline`]

8.26.2 Member Function Documentation

8.26.2.1 `void ScopedLocker::forceUnlock ()` [`inline`]

8.26.3 Member Data Documentation

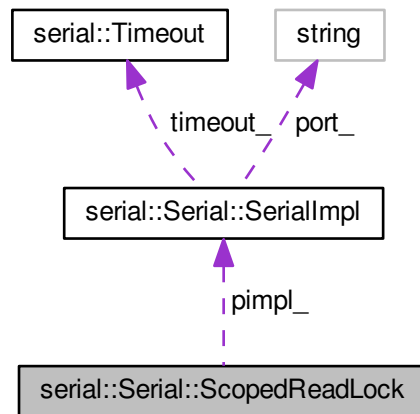
8.26.3.1 `Locker& ScopedLocker::_binded`

The documentation for this class was generated from the following file:

- [include/locker.h](#)

8.27 serial::Serial::ScopedReadLock Class Reference

Collaboration diagram for serial::Serial::ScopedReadLock:



Public Member Functions

- [ScopedReadLock](#) ([Serial::SerialImpl](#) *pimpl)
- [~ScopedReadLock](#) ()

Private Member Functions

- [ScopedReadLock](#) (const [ScopedReadLock](#) &)
- const [ScopedReadLock](#) & operator= ([ScopedReadLock](#))

Private Attributes

- [Serial::SerialImpl](#) * pimpl_

8.27.1 Constructor & Destructor Documentation

8.27.1.1 `serial::Serial::ScopedReadLock::ScopedReadLock (Serial::SerialImpl * pimpl)` `[inline]`, `[explicit]`

8.27.1.2 `serial::Serial::ScopedReadLock::~~ScopedReadLock ()` `[inline]`

8.27.1.3 `serial::Serial::ScopedReadLock::ScopedReadLock (const ScopedReadLock &)` `[private]`

8.27.2 Member Function Documentation

8.27.2.1 `const ScopedReadLock& serial::Serial::ScopedReadLock::operator= (ScopedReadLock)` [private]

8.27.3 Member Data Documentation

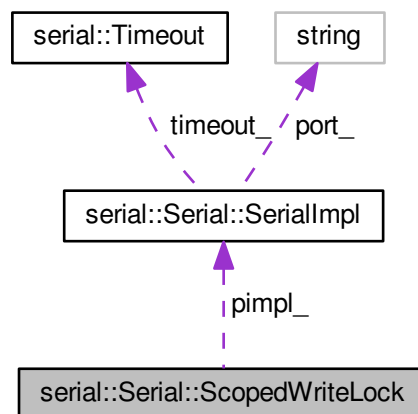
8.27.3.1 `Serial::SerialImpl* serial::Serial::ScopedReadLock::pimpl_` [private]

The documentation for this class was generated from the following file:

- [src/serial.cpp](#)

8.28 serial::Serial::ScopedWriteLock Class Reference

Collaboration diagram for serial::Serial::ScopedWriteLock:



Public Member Functions

- [ScopedWriteLock](#) ([Serial::SerialImpl](#) *pimpl)
- [~ScopedWriteLock](#) ()

Private Member Functions

- [ScopedWriteLock](#) (const [ScopedWriteLock](#) &)
- const [ScopedWriteLock](#) & [operator=](#) ([ScopedWriteLock](#))

Private Attributes

- [Serial::SerialImpl](#) * [pimpl_](#)

8.28.1 Constructor & Destructor Documentation

8.28.1.1 `serial::Serial::ScopedWriteLock::ScopedWriteLock (Serial::SerialImpl * pimpl)` `[inline]`, `[explicit]`

8.28.1.2 `serial::Serial::ScopedWriteLock::~~ScopedWriteLock ()` `[inline]`

8.28.1.3 `serial::Serial::ScopedWriteLock::ScopedWriteLock (const ScopedWriteLock &)` `[private]`

8.28.2 Member Function Documentation

8.28.2.1 `const ScopedWriteLock& serial::Serial::ScopedWriteLock::operator= (ScopedWriteLock)` `[private]`

8.28.3 Member Data Documentation

8.28.3.1 `Serial::SerialImpl* serial::Serial::ScopedWriteLock::pimpl_` `[private]`

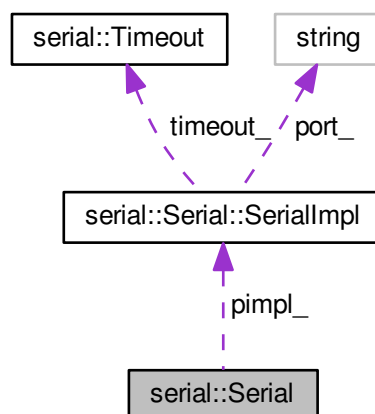
The documentation for this class was generated from the following file:

- [src/serial.cpp](#)

8.29 serial::Serial Class Reference

```
#include <serial.h>
```

Collaboration diagram for serial::Serial:



Classes

- class [ScopedReadLock](#)
- class [ScopedWriteLock](#)
- class [SerialImpl](#)

Public Member Functions

- [Serial](#) (const std::string &port="", uint32_t baudrate=9600, [Timeout](#) timeout=[Timeout\(\)](#), [bytesize_t](#) bytesize=[eightbits](#), [parity_t](#) parity=[parity_none](#), [stopbits_t](#) stopbits=[stopbits_one](#), [flowcontrol_t](#) flowcontrol=[flowcontrol_none](#))
- virtual [~Serial](#) ()
- bool [open](#) ()
- bool [isOpen](#) ()
- void [closePort](#) ()
- [size_t](#) [available](#) ()
- bool [waitReadable](#) ()
- void [waitByteTimes](#) ([size_t](#) count)
- int [waitfordata](#) ([size_t](#) data_count, uint32_t timeout, [size_t](#) *returned_size)
waitfordata
- virtual [size_t](#) [writeData](#) (const uint8_t *data, [size_t](#) size)
writeData
- virtual [size_t](#) [readData](#) (uint8_t *data, [size_t](#) size)
readData
- [size_t](#) [read](#) (uint8_t *buffer, [size_t](#) size)
- [size_t](#) [read](#) (std::vector< uint8_t > &buffer, [size_t](#) size=1)
- [size_t](#) [read](#) (std::string &buffer, [size_t](#) size=1)
- std::string [read](#) ([size_t](#) size=1)
- [size_t](#) [readline](#) (std::string &buffer, [size_t](#) size=65536, std::string eol="\n")
- std::string [readline](#) ([size_t](#) size=65536, std::string eol="\n")
- std::vector< std::string > [readlines](#) ([size_t](#) size=65536, std::string eol="\n")
- [size_t](#) [write](#) (const uint8_t *data, [size_t](#) size)
- [size_t](#) [write](#) (const std::vector< uint8_t > &data)
- [size_t](#) [write](#) (const std::string &data)
- void [setPort](#) (const std::string &port)
- std::string [getPort](#) () const
- void [setTimeout](#) ([Timeout](#) &timeout)
- void [setTimeout](#) (uint32_t inter_byte_timeout, uint32_t read_timeout_constant, uint32_t read_timeout_↵ multiplier, uint32_t write_timeout_constant, uint32_t write_timeout_multiplier)
- [Timeout](#) [getTimeout](#) () const
- bool [setBaudrate](#) (uint32_t baudrate)
- uint32_t [getBaudrate](#) () const
- bool [setBytesize](#) ([bytesize_t](#) bytesize)
- [bytesize_t](#) [getBytesize](#) () const
- bool [setParity](#) ([parity_t](#) parity)
- [parity_t](#) [getParity](#) () const
- bool [setStopbits](#) ([stopbits_t](#) stopbits)
- [stopbits_t](#) [getStopbits](#) () const
- bool [setFlowcontrol](#) ([flowcontrol_t](#) flowcontrol)
- [flowcontrol_t](#) [getFlowcontrol](#) () const
- void [flush](#) ()
- void [flushInput](#) ()
- void [flushOutput](#) ()

- void [sendBreak](#) (int duration)
- bool [setBreak](#) (bool level=true)
- bool [setRTS](#) (bool level=true)
- bool [setDTR](#) (bool level=true)
- bool [waitForChange](#) ()
- bool [getCTS](#) ()
- bool [getDSR](#) ()
- bool [getRI](#) ()
- bool [getCD](#) ()
- int [getByteTime](#) ()

Private Member Functions

- [Serial](#) (const [Serial](#) &)
- [Serial](#) & [operator=](#) (const [Serial](#) &)
- size_t [read_](#) (uint8_t *buffer, size_t [size](#))
- size_t [write_](#) (const uint8_t *data, size_t length)

Private Attributes

- [SerialImpl](#) * [pimpl_](#)

8.29.1 Detailed Description

Class that provides a portable serial port interface.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 `serial::Serial::Serial (const std::string & port = " ", uint32_t baudrate = 9600, Timeout timeout = Timeout () , bytesize_t bytesize = eightbits, parity_t parity = parity_none, stopbits_t stopbits = stopbits_one, flowcontrol_t flowcontrol = flowcontrol_none) [explicit]`

Creates a [Serial](#) object and opens the port if a port is specified, otherwise it remains closed until [serial::Serial::open](#) is called.

Parameters

<i>port</i>	A std::string containing the address of the serial port, which would be something like 'COM1' on Windows and '/dev/ttyS0' on Linux.
<i>baudrate</i>	An unsigned 32-bit integer that represents the baudrate
<i>timeout</i>	A serial::Timeout struct that defines the timeout conditions for the serial port.

See also

[serial::Timeout](#)

Parameters

<i>bytesize</i>	Size of each byte in the serial transmission of data, default is eightbits, possible values are: fivebits, sixbits, sevenbits, eightbits
<i>parity</i>	Method of parity, default is parity_none, possible values are: parity_none, parity_odd, parity_even
<i>stopbits</i>	Number of stop bits used, default is stopbits_one, possible values are: stopbits_one, stopbits_one_point_five, stopbits_two
<i>flowcontrol</i>	Type of flowcontrol used, default is flowcontrol_none, possible values are: flowcontrol_none, flowcontrol_software, flowcontrol_hardware

Exceptions

<i>serial::PortNotOpenedException</i>	
<i>serial::IOException</i>	
<i>std::invalid_argument</i>	

8.29.2.2 serial::Serial::~~Serial () [virtual]

Destructor

8.29.2.3 serial::Serial::Serial (const Serial &) [private]

8.29.3 Member Function Documentation

8.29.3.1 size_t serial::Serial::available ()

Return the number of characters in the buffer.

8.29.3.2 void serial::Serial::closePort ()

Closes the serial port.

8.29.3.3 void serial::Serial::flush ()

Flush the input and output buffers

8.29.3.4 void serial::Serial::flushInput ()

Flush only the input buffer

8.29.3.5 void serial::Serial::flushOutput ()

Flush only the output buffer

8.29.3.6 `uint32_t serial::Serial::getBaudrate () const`

Gets the baudrate for the serial port.

Returns

An integer that sets the baud rate for the serial port.

See also

[Serial::setBaudrate](#)

\

8.29.3.7 `bytesize_t serial::Serial::getBytesize () const`

Gets the bytesize for the serial port.

See also

[Serial::setBytesize](#)

\

8.29.3.8 `int serial::Serial::getByteTime ()`

Returns the singal byte time.

8.29.3.9 `bool serial::Serial::getCD ()`

Returns the current status of the CD line.

8.29.3.10 `bool serial::Serial::getCTS ()`

Returns the current status of the CTS line.

8.29.3.11 `bool serial::Serial::getDSR ()`

Returns the current status of the DSR line.

8.29.3.12 flowcontrol_t serial::Serial::getFlowcontrol () const

Gets the flow control for the serial port.

See also

[Serial::setFlowcontrol](#)

\

8.29.3.13 parity_t serial::Serial::getParity () const

Gets the parity for the serial port.

See also

[Serial::setParity](#)

\

8.29.3.14 string serial::Serial::getPort () const

Gets the serial port identifier.

See also

[Serial::setPort](#)

Exceptions

<i>std::invalid_argument</i>	
------------------------------	--

8.29.3.15 bool serial::Serial::getRI ()

Returns the current status of the RI line.

8.29.3.16 stopbits_t serial::Serial::getStopbits () const

Gets the stopbits for the serial port.

See also

[Serial::setStopbits](#)

\

8.29.3.17 `serial::Timeout serial::Serial::getTimeout () const`

Gets the timeout for reads in seconds.

Returns

A [Timeout](#) struct containing the `inter_byte_timeout`, and read and write timeout constants and multipliers.

See also

[Serial::setTimeout](#)

8.29.3.18 `bool serial::Serial::isOpen ()`

Gets the open status of the serial port.

Returns

Returns true if the port is open, false otherwise.

8.29.3.19 `bool serial::Serial::open ()`

Opens the serial port as long as the port is set and the port isn't already open.

If the port is provided to the constructor then an explicit call to open is not needed.

See also

[Serial::Serial](#)

Returns

Returns true if the port is open, false otherwise.

8.29.3.20 `Serial& serial::Serial::operator= (const Serial &) [private]`

8.29.3.21 `size_t serial::Serial::read (uint8_t * buffer, size_t size)`

Read a given amount of bytes from the serial port into a given buffer.

The read function will return in one of three cases:

- The number of requested bytes was read.
 - In this case the number of bytes requested will match the `size_t` returned by read.
- A timeout occurred, in this case the number of bytes read will not match the amount requested, but no exception will be thrown. One of two possible timeouts occurred:
 - The inter byte timeout expired, this means that number of milliseconds elapsed between receiving bytes from the serial port exceeded the inter byte timeout.
 - The total timeout expired, which is calculated by multiplying the read timeout multiplier by the number of requested bytes and then added to the read timeout constant. If that total number of milliseconds elapses after the initial call to read a timeout will occur.
- An exception occurred, in this case an actual exception will be thrown.

Parameters

<i>buffer</i>	An uint8_t array of at least the requested size.
<i>size</i>	A size_t defining how many bytes to be read.

Returns

A size_t representing the number of bytes read as a result of the call to read.

8.29.3.22 `size_t serial::Serial::read (std::vector< uint8_t > & buffer, size_t size = 1)`

Read a given amount of bytes from the serial port into a give buffer.

Parameters

<i>buffer</i>	A reference to a std::vector of uint8_t.
<i>size</i>	A size_t defining how many bytes to be read.

Returns

A size_t representing the number of bytes read as a result of the call to read.

8.29.3.23 `size_t serial::Serial::read (std::string & buffer, size_t size = 1)`

Read a given amount of bytes from the serial port into a give buffer.

Parameters

<i>buffer</i>	A reference to a std::string.
<i>size</i>	A size_t defining how many bytes to be read.

Returns

A size_t representing the number of bytes read as a result of the call to read.

8.29.3.24 `string serial::Serial::read (size_t size = 1)`

Read a given amount of bytes from the serial port and return a string containing the data.

Parameters

<i>size</i>	A size_t defining how many bytes to be read.
-------------	--

Returns

A `std::string` containing the data read from the port.

8.29.3.25 `size_t serial::Serial::read_ (uint8_t * buffer, size_t size)` [private]

8.29.3.26 `size_t serial::Serial::readData (uint8_t * data, size_t size)` [virtual]

readData**Parameters**

<i>data</i>	
<i>size</i>	

Returns

8.29.3.27 `size_t serial::Serial::readline (std::string & buffer, size_t size = 65536, std::string eol = "\n")`

Reads in a line or until a given delimiter has been processed.

Reads from the serial port until a single line has been read.

Parameters

<i>buffer</i>	A <code>std::string</code> reference used to store the data.
<i>size</i>	A maximum length of a line, defaults to 65536 (2^{16})
<i>eol</i>	A string to match against for the EOL.

Returns

A `size_t` representing the number of bytes read.

8.29.3.28 `std::string serial::Serial::readline (size_t size = 65536, std::string eol = "\n")`

Reads in a line or until a given delimiter has been processed.

Reads from the serial port until a single line has been read.

Parameters

<i>size</i>	A maximum length of a line, defaults to 65536 (2^{16})
<i>eol</i>	A string to match against for the EOL.

Returns

A std::string containing the line.

8.29.3.29 `vector< string > serial::Serial::readlines (size_t size = 65536, std::string eol = "\n")`

Reads in multiple lines until the serial port times out.

This requires a timeout > 0 before it can be run. It will read until a timeout occurs and return a list of strings.

Parameters

<i>size</i>	A maximum length of combined lines, defaults to 65536 (2^{16})
<i>eol</i>	A string to match against for the EOL.

Returns

A vector<string> containing the lines.

8.29.3.30 `void serial::Serial::sendBreak (int duration)`

Sends the RS-232 break signal. See tcsendbreak(3).

8.29.3.31 `bool serial::Serial::setBaudrate (uint32_t baudrate)`

Sets the baudrate for the serial port.

Possible baudrates depends on the system but some safe baudrates include: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 56000, 57600, 115200. Some other baudrates that are supported by some comports: 128000, 153600, 230400, 256000, 460800, 921600.

Parameters

<i>baudrate</i>	An integer that sets the baud rate for the serial port.
-----------------	---

8.29.3.32 `bool serial::Serial::setBreak (bool level = true)`

Set the break condition to a given level. Defaults to true.

8.29.3.33 `bool serial::Serial::setBytesize (bytesize_t bytesize)`

Sets the bytesize for the serial port.

Parameters

<i>bytesize</i>	Size of each byte in the serial transmission of data, default is eightbits, possible values are: fivebits, sixbits, sevenbits, eightbits
-----------------	--

\

8.29.3.34 `bool serial::Serial::setDTR (bool level = true)`

Set the DTR handshaking line to the given level. Defaults to true.

8.29.3.35 `bool serial::Serial::setFlowcontrol (flowcontrol_t flowcontrol)`

Sets the flow control for the serial port.

Parameters

<i>flowcontrol</i>	Type of flowcontrol used, default is flowcontrol_none, possible values are: flowcontrol_none, flowcontrol_software, flowcontrol_hardware
--------------------	--

\

8.29.3.36 `bool serial::Serial::setParity (parity_t parity)`

Sets the parity for the serial port.

Parameters

<i>parity</i>	Method of parity, default is parity_none, possible values are: parity_none, parity_odd, parity_even
---------------	---

\

8.29.3.37 `void serial::Serial::setPort (const std::string & port)`

Sets the serial port identifier.

Parameters

<i>port</i>	A const std::string reference containing the address of the serial port, which would be something like 'COM1' on Windows and '/dev/ttyS0' on Linux.
-------------	---

Exceptions

<i>std::invalid_argument</i>	
------------------------------	--

8.29.3.38 `bool serial::Serial::setRTS (bool level = true)`

Set the RTS handshaking line to the given level. Defaults to true.

8.29.3.39 `bool serial::Serial::setStopbits (stopbits_t stopbits)`

Sets the stopbits for the serial port.

Parameters

<i>stopbits</i>	Number of stop bits used, default is stopbits_one, possible values are: stopbits_one, stopbits_one_point_five, stopbits_two
-----------------	---

\

8.29.3.40 `void serial::Serial::setTimeout (serial::Timeout & timeout)`

Sets the timeout for reads and writes using the [Timeout](#) struct.

There are two timeout conditions described here:

- The inter byte timeout:
 - The inter_byte_timeout component of [serial::Timeout](#) defines the maximum amount of time, in milliseconds, between receiving bytes on the serial port that can pass before a timeout occurs. Setting this to zero will prevent inter byte timeouts from occurring.
- Total time timeout:
 - The constant and multiplier component of this timeout condition, for both read and write, are defined in [serial::Timeout](#). This timeout occurs if the total time since the read or write call was made exceeds the specified time in milliseconds.
 - The limit is defined by multiplying the multiplier component by the number of requested bytes and adding that product to the constant component. In this way if you want a read call, for example, to timeout after exactly one second regardless of the number of bytes you asked for then set the read_timeout_constant component of [serial::Timeout](#) to 1000 and the read_timeout_multiplier to zero. This timeout condition can be used in conjunction with the inter byte timeout condition with out any problems, timeout will simply occur when one of the two timeout conditions is met. This allows users to have maximum control over the trade-off between responsiveness and efficiency.

Read and write functions will return in one of three cases. When the reading or writing is complete, when a timeout occurs, or when an exception occurs.

A timeout of 0 enables non-blocking mode.

Parameters

<i>timeout</i>	A serial::Timeout struct containing the inter byte timeout, and the read and write timeout constants and multipliers.
----------------	---

See also

[serial::Timeout](#)

8.29.3.41 `void serial::Serial::setTimeout (uint32_t inter_byte_timeout, uint32_t read_timeout_constant, uint32_t read_timeout_multiplier, uint32_t write_timeout_constant, uint32_t write_timeout_multiplier)` `[inline]`

Sets the timeout for reads and writes.

8.29.3.42 `void serial::Serial::waitByteTimes (size_t count)`

Block for a period of time corresponding to the transmission time of count characters at present serial settings. This may be used in conjunction with waitReadable to read larger blocks of data from the port.

8.29.3.43 `bool serial::Serial::waitForChange ()`

Blocks until CTS, DSR, RI, CD changes or something interrupts it.

Can throw an exception if an error occurs while waiting. You can check the status of CTS, DSR, RI, and CD once this returns. Uses TIOCMWAIT via ioctl if available (mostly only on Linux) with a resolution of less than +-1ms and as good as +-0.2ms. Otherwise a polling method is used which can give +-2ms.

Returns

Returns true if one of the lines changed, false if something else occurred.

8.29.3.44 `int serial::Serial::waitfordata (size_t data_count, uint32_t timeout, size_t * returned_size)`

waitfordata

Parameters

<i>data_count</i>	
<i>timeout</i>	
<i>returned_size</i>	

Returns

8.29.3.45 `bool serial::Serial::waitReadable ()`

Block until there is serial data to read or read_timeout_constant number of milliseconds have elapsed. The return value is true when the function exits with the port in a readable state, false otherwise (due to timeout or select interruption).

8.29.3.46 `size_t serial::Serial::write (const uint8_t * data, size_t size)`

Write a string to the serial port.

Parameters

<i>data</i>	A const reference containing the data to be written to the serial port.
<i>size</i>	A size_t that indicates how many bytes should be written from the given data buffer.

Returns

A size_t representing the number of bytes actually written to the serial port.

Exceptions

<i>serial::PortNotOpenedException</i>	
<i>serial::SerialException</i>	
<i>serial::IOException</i>	

8.29.3.47 `size_t serial::Serial::write (const std::vector< uint8_t > & data)`

Write a string to the serial port.

Parameters

<i>data</i>	A const reference containing the data to be written to the serial port.
-------------	---

Returns

A size_t representing the number of bytes actually written to the serial port.

8.29.3.48 `size_t serial::Serial::write (const std::string & data)`

Write a string to the serial port.

Parameters

<i>data</i>	A const reference containing the data to be written to the serial port.
-------------	---

Returns

A size_t representing the number of bytes actually written to the serial port.

8.29.3.49 `size_t serial::Serial::write_ (const uint8_t * data, size_t length)` [private]

8.29.3.50 `size_t serial::Serial::writeData (const uint8_t * data, size_t size)` [virtual]

writeData

Parameters

<i>data</i>	
<i>size</i>	

Returns

8.29.4 Member Data Documentation

8.29.4.1 `SerialImpl* serial::Serial::pimpl_` [private]

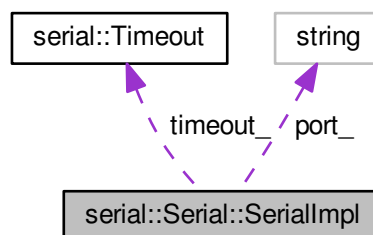
The documentation for this class was generated from the following files:

- [include/serial.h](#)
- [src/serial.cpp](#)

8.30 serial::Serial::SerialImpl Class Reference

```
#include <unix_serial.h>
```

Collaboration diagram for serial::Serial::SerialImpl:



Public Member Functions

- [SerialImpl](#) (const string &port, unsigned long baudrate, [bytesize_t](#) bytesize, [parity_t](#) parity, [stopbits_t](#) stopbits, [flowcontrol_t](#) flowcontrol)
- virtual [~SerialImpl](#) ()
- bool [open](#) ()
- void [close](#) ()
- bool [isOpen](#) () const
- [size_t](#) [available](#) ()
- bool [waitReadable](#) (uint32_t timeout)
- void [waitByteTimes](#) ([size_t](#) count)
- int [waitfordata](#) ([size_t](#) data_count, uint32_t timeout, [size_t](#) *returned_size)
- [size_t](#) [read](#) (uint8_t *buf, [size_t](#) size=1)
- [size_t](#) [write](#) (const uint8_t *data, [size_t](#) length)
- void [flush](#) ()
- void [flushInput](#) ()
- void [flushOutput](#) ()
- void [sendBreak](#) (int duration)
- bool [setBreak](#) (bool level)
- bool [setRTS](#) (bool level)
- bool [setDTR](#) (bool level)
- bool [waitForChange](#) ()
- bool [getCTS](#) ()
- bool [getDSR](#) ()
- bool [getRI](#) ()
- bool [getCD](#) ()
- uint32_t [getByteTime](#) ()
- void [setPort](#) (const string &port)
- string [getPort](#) () const
- void [setTimeout](#) ([Timeout](#) &timeout)
- [Timeout](#) [getTimeout](#) () const
- bool [setBaudrate](#) (unsigned long baudrate)
- bool [setStandardBaudRate](#) ([speed_t](#) baudrate)
- bool [setCustomBaudRate](#) (unsigned long baudrate)
- unsigned long [getBaudrate](#) () const
- bool [setBytesize](#) ([bytesize_t](#) bytesize)
- [bytesize_t](#) [getBytesize](#) () const
- bool [setParity](#) ([parity_t](#) parity)
- [parity_t](#) [getParity](#) () const
- bool [setStopbits](#) ([stopbits_t](#) stopbits)
- [stopbits_t](#) [getStopbits](#) () const
- bool [setFlowcontrol](#) ([flowcontrol_t](#) flowcontrol)
- [flowcontrol_t](#) [getFlowcontrol](#) () const
- bool [setTermios](#) (const termios *tio)
- bool [getTermios](#) (termios *tio)
- int [readLock](#) ()
- int [readUnlock](#) ()
- int [writeLock](#) ()
- int [writeUnlock](#) ()

Private Attributes

- string `port_`
- int `fd_`
- pid_t `pid`
- bool `is_open_`
- bool `xonxoff_`
- bool `rtscts_`
- Timeout `timeout_`
- unsigned long `baudrate_`
- uint32_t `byte_time_ns_`
- parity_t `parity_`
- bytesize_t `bytesize_`
- stopbits_t `stopbits_`
- flowcontrol_t `flowcontrol_`
- pthread_mutex_t `read_mutex`
- pthread_mutex_t `write_mutex`

8.30.1 Constructor & Destructor Documentation

8.30.1.1 `serial::Serial::SerialImpl::SerialImpl (const string & port, unsigned long baudrate, bytesize_t bytesize, parity_t parity, stopbits_t stopbits, flowcontrol_t flowcontrol)` `[explicit]`

8.30.1.2 `serial::Serial::SerialImpl::~SerialImpl ()` `[virtual]`

8.30.2 Member Function Documentation

8.30.2.1 `size_t serial::Serial::SerialImpl::available ()`

8.30.2.2 `void serial::Serial::SerialImpl::close ()`

8.30.2.3 `void serial::Serial::SerialImpl::flush ()`

8.30.2.4 `void serial::Serial::SerialImpl::flushInput ()`

8.30.2.5 `void serial::Serial::SerialImpl::flushOutput ()`

8.30.2.6 `unsigned long serial::Serial::SerialImpl::getBaudrate () const`

8.30.2.7 `serial::bytesize_t serial::Serial::SerialImpl::getBytesize () const`

8.30.2.8 `uint32_t serial::Serial::SerialImpl::getByteTime ()`

8.30.2.9 `bool serial::Serial::SerialImpl::getCD ()`

8.30.2.10 `bool serial::Serial::SerialImpl::getCTS ()`

- 8.30.2.11 `bool serial::Serial::SerialImpl::getDSR ()`
- 8.30.2.12 `serial::flowcontrol_t serial::Serial::SerialImpl::getFlowcontrol () const`
- 8.30.2.13 `serial::parity_t serial::Serial::SerialImpl::getParity () const`
- 8.30.2.14 `string serial::Serial::SerialImpl::getPort () const`
- 8.30.2.15 `bool serial::Serial::SerialImpl::getRI ()`
- 8.30.2.16 `serial::stopbits_t serial::Serial::SerialImpl::getStopbits () const`
- 8.30.2.17 `bool serial::Serial::SerialImpl::getTermios (termios * tio)`
- 8.30.2.18 `serial::Timeout serial::Serial::SerialImpl::getTimeout () const`
- 8.30.2.19 `bool serial::Serial::SerialImpl::isOpen () const`
- 8.30.2.20 `bool serial::Serial::SerialImpl::open ()`
- 8.30.2.21 `size_t serial::Serial::SerialImpl::read (uint8_t * buf, size_t size = 1)`
- 8.30.2.22 `int serial::Serial::SerialImpl::readLock ()`
- 8.30.2.23 `int serial::Serial::SerialImpl::readUnlock ()`
- 8.30.2.24 `void serial::Serial::SerialImpl::sendBreak (int duration)`
- 8.30.2.25 `bool serial::Serial::SerialImpl::setBaudrate (unsigned long baudrate)`
- 8.30.2.26 `bool serial::Serial::SerialImpl::setBreak (bool level)`
- 8.30.2.27 `bool serial::Serial::SerialImpl::setBytesize (serial::bytesize_t bytesize)`
- 8.30.2.28 `bool serial::Serial::SerialImpl::setCustomBaudRate (unsigned long baudrate)`
- 8.30.2.29 `bool serial::Serial::SerialImpl::setDTR (bool level)`
- 8.30.2.30 `bool serial::Serial::SerialImpl::setFlowcontrol (serial::flowcontrol_t flowcontrol)`
- 8.30.2.31 `bool serial::Serial::SerialImpl::setParity (serial::parity_t parity)`
- 8.30.2.32 `void serial::Serial::SerialImpl::setPort (const string & port)`
- 8.30.2.33 `bool serial::Serial::SerialImpl::setRTS (bool level)`

8.30.2.34 `bool serial::Serial::SerialImpl::setStandardBaudRate (speed_t baudrate)`

8.30.2.35 `bool serial::Serial::SerialImpl::setStopbits (serial::stopbits_t stopbits)`

8.30.2.36 `bool serial::Serial::SerialImpl::setTermios (const termios * tio)`

8.30.2.37 `void serial::Serial::SerialImpl::setTimeout (serial::Timeout & timeout)`

8.30.2.38 `void serial::Serial::SerialImpl::waitByteTimes (size_t count)`

8.30.2.39 `bool serial::Serial::SerialImpl::waitForChange ()`

8.30.2.40 `int serial::Serial::SerialImpl::waitfordata (size_t data_count, uint32_t timeout, size_t * returned_size)`

8.30.2.41 `bool serial::Serial::SerialImpl::waitReadable (uint32_t timeout)`

8.30.2.42 `size_t serial::Serial::SerialImpl::write (const uint8_t * data, size_t length)`

Error

[Timeout](#)

Port ready to write

8.30.2.43 `int serial::Serial::SerialImpl::writeLock ()`

8.30.2.44 `int serial::Serial::SerialImpl::writeUnlock ()`

8.30.3 Member Data Documentation

8.30.3.1 `unsigned long serial::Serial::SerialImpl::baudrate_ [private]`

8.30.3.2 `uint32_t serial::Serial::SerialImpl::byte_time_ns_ [private]`

8.30.3.3 `bytesize_t serial::Serial::SerialImpl::bytesize_ [private]`

8.30.3.4 `int serial::Serial::SerialImpl::fd_ [private]`

8.30.3.5 `flowcontrol_t serial::Serial::SerialImpl::flowcontrol_ [private]`

8.30.3.6 `bool serial::Serial::SerialImpl::is_open_ [private]`

8.30.3.7 `parity_t serial::Serial::SerialImpl::parity_ [private]`

8.30.3.8 `pid_t serial::Serial::SerialImpl::pid [private]`

- 8.30.3.9 string serial::Serial::SerialImpl::port_ [private]
- 8.30.3.10 pthread_mutex_t serial::Serial::SerialImpl::read_mutex [private]
- 8.30.3.11 bool serial::Serial::SerialImpl::rtscts_ [private]
- 8.30.3.12 stopbits_t serial::Serial::SerialImpl::stopbits_ [private]
- 8.30.3.13 Timeout serial::Serial::SerialImpl::timeout_ [private]
- 8.30.3.14 pthread_mutex_t serial::Serial::SerialImpl::write_mutex [private]
- 8.30.3.15 bool serial::Serial::SerialImpl::xonxoff_ [private]

The documentation for this class was generated from the following files:

- [src/impl/unix/unix_serial.h](#)
- [src/impl/unix/unix_serial.cpp](#)

8.31 serial::termios2 Struct Reference

Public Attributes

- tcflag_t c_iflag
- tcflag_t c_oflag
- tcflag_t c_cflag
- tcflag_t c_lflag
- cc_t c_line
- cc_t c_cc [SNCCS]
- speed_t c_ispeed
- speed_t c_ospeed

8.31.1 Member Data Documentation

- 8.31.1.1 cc_t serial::termios2::c_cc[SNCCS]
- 8.31.1.2 tcflag_t serial::termios2::c_cflag
- 8.31.1.3 tcflag_t serial::termios2::c_iflag
- 8.31.1.4 speed_t serial::termios2::c_ispeed
- 8.31.1.5 tcflag_t serial::termios2::c_lflag
- 8.31.1.6 cc_t serial::termios2::c_line
- 8.31.1.7 tcflag_t serial::termios2::c_oflag
- 8.31.1.8 speed_t serial::termios2::c_ospeed

The documentation for this struct was generated from the following file:

- [src/impl/unix/unix_serial.cpp](#)

8.32 Thread Class Reference

```
#include <thread.h>
```

Public Member Functions

- [Thread](#) ()
- virtual [~Thread](#) ()
- [_size_t](#) [getHandle](#) ()
- int [terminate](#) ()
- void * [getParam](#) ()
- int [join](#) (unsigned long timeout=-1)
- bool [operator==](#) (const [Thread](#) &right)

Static Public Member Functions

- template<class CLASS , int(CLASS::*)(void) PROC>
static [Thread](#) [ThreadCreateObjectFunctor](#) (CLASS *pthis)
- template<class CLASS , int(CLASS::*)(void) PROC>
static [_size_t](#) THREAD_PROC [createThreadAux](#) (void *param)
- static [Thread](#) [createThread](#) ([thread_proc_t](#) proc, void *param=NULL)

Protected Member Functions

- [Thread](#) ([thread_proc_t](#) proc, void *param)

Protected Attributes

- void * [_param](#)
- [thread_proc_t](#) [_func](#)
- [_size_t](#) [_handle](#)

8.32.1 Constructor & Destructor Documentation

8.32.1.1 [Thread::Thread](#) () [inline],[explicit]

8.32.1.2 virtual [Thread::~~Thread](#) () [inline],[virtual]

8.32.1.3 [Thread::Thread](#) ([thread_proc_t](#) *proc*, void * *param*) [inline],[explicit],[protected]

8.32.2 Member Function Documentation

8.32.2.1 static [Thread](#) [Thread::createThread](#) ([thread_proc_t](#) *proc*, void * *param* =NULL) [inline],[static]

8.32.2.2 template<class CLASS , int(CLASS::*)(void) PROC> static [_size_t](#) THREAD_PROC [Thread::createThreadAux](#) (void * *param*) [inline],[static]

8.32.2.3 `_size_t Thread::getHandle ()` `[inline]`

8.32.2.4 `void* Thread::getParam ()` `[inline]`

8.32.2.5 `int Thread::join (unsigned long timeout = -1)` `[inline]`

8.32.2.6 `bool Thread::operator== (const Thread & right)` `[inline]`

8.32.2.7 `int Thread::terminate ()` `[inline]`

8.32.2.8 `template<class CLASS , int(CLASS::*)(void) PROC> static Thread Thread::ThreadCreateObjectFunctor (CLASS * this)` `[inline]`, `[static]`

8.32.3 Member Data Documentation

8.32.3.1 `thread_proc_t Thread::_func` `[protected]`

8.32.3.2 `_size_t Thread::_handle` `[protected]`

8.32.3.3 `void* Thread::_param` `[protected]`

The documentation for this class was generated from the following file:

- [include/thread.h](#)

8.33 serial::Timeout Struct Reference

```
#include <serial.h>
```

Public Member Functions

- [Timeout](#) (`uint32_t inter_byte_timeout` = 0, `uint32_t read_timeout_constant` = 0, `uint32_t read_timeout_↔ multiplier` = 0, `uint32_t write_timeout_constant` = 0, `uint32_t write_timeout_multiplier` = 0)

Static Public Member Functions

- static `uint32_t` [max](#) ()
- static [Timeout](#) [simpleTimeout](#) (`uint32_t` `timeout`)

Public Attributes

- `uint32_t` [inter_byte_timeout](#)
- `uint32_t` [read_timeout_constant](#)
- `uint32_t` [read_timeout_multiplier](#)
- `uint32_t` [write_timeout_constant](#)
- `uint32_t` [write_timeout_multiplier](#)

8.33.1 Detailed Description

Structure for setting the timeout of the serial port, times are in milliseconds.

In order to disable the interbyte timeout, set it to [Timeout::max\(\)](#).

8.33.2 Constructor & Destructor Documentation

8.33.2.1 `serial::Timeout::Timeout (uint32_t inter_byte_timeout_ = 0, uint32_t read_timeout_constant_ = 0, uint32_t read_timeout_multiplier_ = 0, uint32_t write_timeout_constant_ = 0, uint32_t write_timeout_multiplier_ = 0) [inline], [explicit]`

8.33.3 Member Function Documentation

8.33.3.1 `static uint32_t serial::Timeout::max () [inline], [static]`

8.33.3.2 `static Timeout serial::Timeout::simpleTimeout (uint32_t timeout) [inline], [static]`

Convenience function to generate [Timeout](#) structs using a single absolute timeout.

Parameters

<i>timeout</i>	A long that defines the time in milliseconds until a timeout occurs after a call to read or write is made.
----------------	--

Returns

[Timeout](#) struct that represents this simple timeout provided.

8.33.4 Member Data Documentation

8.33.4.1 `uint32_t serial::Timeout::inter_byte_timeout`

Number of milliseconds between bytes received to timeout on.

8.33.4.2 `uint32_t serial::Timeout::read_timeout_constant`

A constant number of milliseconds to wait after calling read.

8.33.4.3 `uint32_t serial::Timeout::read_timeout_multiplier`

A multiplier against the number of requested bytes to wait after calling read.

8.33.4.4 `uint32_t serial::Timeout::write_timeout_constant`

A constant number of milliseconds to wait after calling write.

8.33.4.5 uint32_t serial::Timeout::write_timeout_multiplier

A multiplier against the number of requested bytes to wait after calling write.

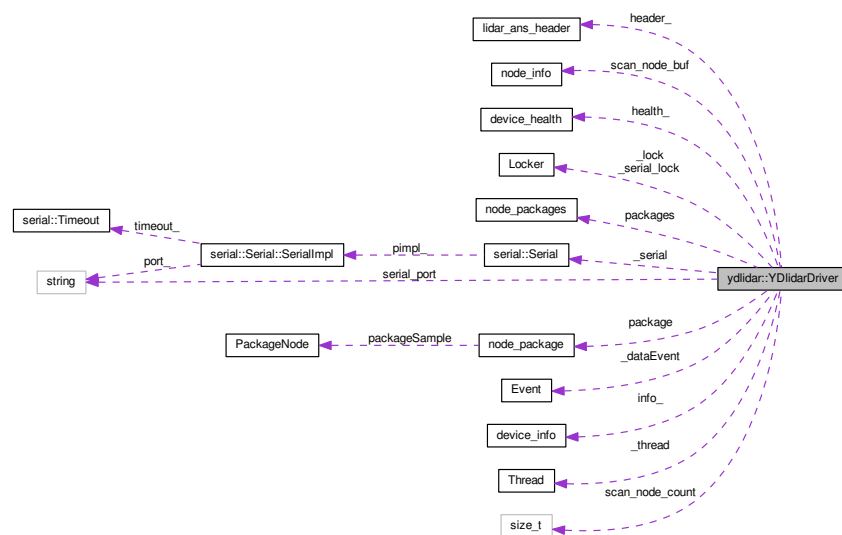
The documentation for this struct was generated from the following file:

- include/serial.h

8.34 ydlidar::YDlidarDriver Class Reference

```
#include <ydlidar_driver.h>
```

Collaboration diagram for ydlidar::YDlidarDriver:



Public Types

- enum { `DEFAULT_TIMEOUT` = 2000, `DEFAULT_HEART_BEAT` = 1000, `MAX_SCAN_NODES` = 3600, `DEFAULT_TIMEOUT_COUNT` = 1 }

Public Member Functions

- `PropertyBuilderByName` (bool, SingleChannel, private)
Set and Get LiDAR single channel. Whether LiDAR communication channel is a single-channel.
- `PropertyBuilderByName` (int, LidarType, private)
Set and Get LiDAR Type.
- `PropertyBuilderByName` (uint32_t, PointTime, private)
Set and Get Sampling interval.
- `YDlidarDriver` ()
- virtual `~YDlidarDriver` ()

- **result_t connect** (const char *port_path, uint32_t baudrate)
连接雷达
连接成功后, 必须使用::disconnect函数关闭
- **void disconnect** ()
断开雷达连接
- **bool isscanning** () const
扫图状态
- **bool isconnected** () const
连接雷达状态
- **void setIntensities** (const bool &intensities)
设置雷达是否带信号质量
连接成功后, 必须使用::disconnect函数关闭
- **void setAutoReconnect** (const bool &enable)
设置雷达异常自动重新连接
- **result_t getHealth** (device_health &health, uint32_t timeout=DEFAULT_TIMEOUT)
获取雷达设备健康状态
- **result_t getDeviceInfo** (device_info &info, uint32_t timeout=DEFAULT_TIMEOUT)
获取雷达设备信息
- **result_t startScan** (bool force=false, uint32_t timeout=DEFAULT_TIMEOUT)
开启扫描
- **result_t stop** ()
关闭扫描
- **result_t grabScanData** (node_info *nodebuffer, size_t &count, uint32_t timeout=DEFAULT_TIMEOUT)
获取激光数据
- **result_t ascendScanData** (node_info *nodebuffer, size_t count)
补偿激光角度
把角度限制在0到360度之间
- **result_t reset** (uint32_t timeout=DEFAULT_TIMEOUT)
重置激光雷达
- **result_t startMotor** ()
打开电机
- **result_t stopMotor** ()
关闭电机
- **result_t getScanFrequency** (scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)
获取激光雷达当前扫描频率
- **result_t setScanFrequencyAdd** (scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)
设置增加扫描频率1HZ
- **result_t setScanFrequencyDis** (scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)
设置减小扫描频率1HZ
- **result_t setScanFrequencyAddMic** (scan_frequency &frequency, uint32_t timeout=DEFAULT_TIMEOUT)

设置增加扫描频率 0.1HZ

- [result_t setScanFrequencyDisMic](#) ([scan_frequency](#) &[frequency](#), [uint32_t](#) timeout=[DEFAULT_TIMEOUT](#))

设置减小扫描频率 0.1HZ

- [result_t getSamplingRate](#) ([sampling_rate](#) &[rate](#), [uint32_t](#) timeout=[DEFAULT_TIMEOUT](#))

获取激光雷达当前采样频率

- [result_t setSamplingRate](#) ([sampling_rate](#) &[rate](#), [uint32_t](#) timeout=[DEFAULT_TIMEOUT](#))

设置激光雷达当前采样频率

- [result_t getZeroOffsetAngle](#) ([offset_angle](#) &[angle](#), [uint32_t](#) timeout=[DEFAULT_TIMEOUT](#))

获取激光雷达当前零位角

Static Public Member Functions

- static [std::string getSDKVersion](#) ()

获取当前SDK版本号
静态函数

- static [std::map< std::string, std::string > lidarPortList](#) ()

lidarPortList 获取雷达端口

Public Attributes

- [std::atomic< bool > isConnected](#)

串口连接状态

- [std::atomic< bool > isScanning](#)

扫图状态

- [std::atomic< bool > isAutoReconnect](#)

异常自动重新连接

- [std::atomic< bool > isAutoconnting](#)

是否正在自动连接中

- [node_info * scan_node_buf](#)

激光点信息

- [size_t scan_node_count](#)

激光点数

- [Event_dataEvent](#)

数据同步事件

- [Locker_lock](#)

线程锁

- [Locker_serial_lock](#)

串口锁

- [Thread_thread](#)

线程id

Protected Member Functions

- [result_t createThread \(\)](#)
创建解析雷达数据线程
- [result_t startAutoScan](#) (bool force=false, uint32_t timeout=DEFAULT_TIMEOUT)
重新连接开启扫描
- [result_t stopScan](#) (uint32_t timeout=DEFAULT_TIMEOUT)
stopScan
- [result_t checkDeviceInfo](#) (uint8_t *recvBuffer, uint8_t byte, int recvPos, int recvSize, int pos)
checkDeviceStatus
- [result_t waitDevicePackage](#) (uint32_t timeout=DEFAULT_TIMEOUT)
waitDevicePackage
- [result_t waitPackage](#) ([node_info](#) *node, uint32_t timeout=DEFAULT_TIMEOUT)
解包激光数据
- [result_t waitScanData](#) ([node_info](#) *nodebuffer, size_t &count, uint32_t timeout=DEFAULT_TIMEOUT)
发送数据到雷达
- int [cacheScanData](#) ()
激光数据解析线程
- [result_t sendCommand](#) (uint8_t cmd, const void *payload=NULL, size_t payloadsize=0)
发送数据到雷达
- [result_t waitResponseHeader](#) ([lidar_ans_header](#) *header, uint32_t timeout=DEFAULT_TIMEOUT)
等待激光数据包头
- [result_t waitForData](#) (size_t data_count, uint32_t timeout=DEFAULT_TIMEOUT, size_t *returned_size=NULL)
等待固定数量串口数据
- [result_t getData](#) (uint8_t *data, size_t size)
获取串口数据
- [result_t sendData](#) (const uint8_t *data, size_t size)
串口发送数据
- void [checkTransDelay](#) ()
checkTransDelay
- void [disableDataGrabbing](#) ()
关闭数据获取通道
- void [setDTR](#) ()
设置串口DTR
- void [clearDTR](#) ()
清除串口DTR
- void [flushSerial](#) ()
flushSerial
- [result_t checkAutoConnecting](#) ()
checkAutoConnecting

Private Attributes

- int [PackageSampleBytes](#)
一个包包含的激光点数
- [serial::Serial](#) * [_serial](#)
串口
- bool [m_intensities](#)
信号质量状态
- [uint32_t](#) [m_baudrate](#)
波特率
- bool [isSupportMotorDtrCtrl](#)
是否支持电机控制
- [uint32_t](#) [trans_delay](#)
串口传输一个 *byte* 时间
- int [m_sampling_rate](#)
采样频率
- int [model](#)
雷达型号
- int [sample_rate](#)
- [node_package](#) [package](#)
带信号质量协议包
- [node_packages](#) [packages](#)
不带信号质量协议包
- [uint16_t](#) [package_Sample_Index](#)
包采样点索引
- float [IntervalSampleAngle](#)
- float [IntervalSampleAngle_LastPackage](#)
- [uint16_t](#) [FirstSampleAngle](#)
起始采样角
- [uint16_t](#) [LastSampleAngle](#)
结束采样角
- [uint16_t](#) [Checksum](#)
校验和
- [uint8_t](#) [scan_frequence](#)
协议中雷达转速
- [uint16_t](#) [ChecksumCal](#)
- [uint16_t](#) [SampleNumAndCTCal](#)
- [uint16_t](#) [LastSampleAngleCal](#)
- bool [ChecksumResult](#)
- [uint16_t](#) [Valu8To16](#)
- [std::string](#) [serial_port](#)
雷达端口
- [uint8_t](#) * [globalRecvBuffer](#)
- int [retryCount](#)
- bool [has_device_header](#)
- [uint8_t](#) [last_device_byte](#)
- int [asyncRecvPos](#)
- [uint16_t](#) [async_size](#)
- [device_info](#) [info_](#)
- [device_health](#) [health_](#)
- [lidar_ans_header](#) [header_](#)
- [uint8_t](#) * [headerBuffer](#)

- uint8_t * [infoBuffer](#)
- uint8_t * [healthBuffer](#)
- bool [get_device_info_success](#)
- bool [get_device_health_success](#)
- int [package_index](#)
- bool [has_package_error](#)

8.34.1 Detailed Description

Class that provides a lidar interface.

8.34.2 Member Enumeration Documentation

8.34.2.1 anonymous enum

Enumerator

DEFAULT_TIMEOUT 默认超时时间.
DEFAULT_HEART_BEAT 默认检测掉电功能时间.
MAX_SCAN_NODES 最大扫描点数.
DEFAULT_TIMEOUT_COUNT

8.34.3 Constructor & Destructor Documentation

8.34.3.1 ydlidar::YDlidarDriver::YDlidarDriver ()

A constructor. A more elaborate description of the constructor.

8.34.3.2 ydlidar::YDlidarDriver::~~YDlidarDriver () [virtual]

A destructor. A more elaborate description of the destructor.

8.34.4 Member Function Documentation

8.34.4.1 result_t ydlidar::YDlidarDriver::ascendScanData (node_info * nodebuffer, size_t count)

补偿激光角度
把角度限制在0到360度之间

Parameters

in	<i>nodebuffer</i>	激光点信息
in	<i>count</i>	一圈激光点数

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

补偿之前，必须使用::grabScanData函数获取激光数据成功

8.34.4.2 int ydlidar::YDlidarDriver::cacheScanData () [protected]

激光数据解析线程

8.34.4.3 result_t ydlidar::YDlidarDriver::checkAutoConnecting () [protected]

checkAutoConnecting

8.34.4.4 result_t ydlidar::YDlidarDriver::checkDeviceInfo (uint8_t * *recvBuffer*, uint8_t *byte*, int *recvPos*, int *recvSize*, int *pos*) [protected]

checkDeviceStatus

Parameters

<i>byte</i>	
-------------	--

Returns**8.34.4.5 void ydlidar::YDlidarDriver::checkTransDelay () [protected]**

checkTransDelay

8.34.4.6 void ydlidar::YDlidarDriver::clearDTR () [protected]

清除串口DTR

8.34.4.7 `result_t ydlidar::YDlidarDriver::connect (const char * port_path, uint32_t baudrate)`

连接雷达

连接成功后，必须使用::disconnect函数关闭

Parameters

in	<i>port_path</i>	串口号
in	<i>baudrate</i>	波特率, YDLIDAR-SS雷达波特率: 230400 G2-SS-1

Returns

返回连接状态

Return values

0	成功
<	0 失败

Note

连接成功后, 必须使用::disconnect函数关闭

See also

函数::YDlidarDriver::disconnect ("::"是指定有连接功能,可以看文档里的disconnect变成绿,点击它可以跳转到disconnect.)

8.34.4.8 result_t ydlidar::YDlidarDriver::createThread () [protected]

创建解析雷达数据线程

Note

创建解析雷达数据线程之前, 必须使用::startScan函数开启扫图成功

8.34.4.9 void ydlidar::YDlidarDriver::disableDataGrabbing () [protected]

关闭数据获取通道

8.34.4.10 void ydlidar::YDlidarDriver::disconnect ()

断开雷达连接

8.34.4.11 void ydlidar::YDlidarDriver::flushSerial () [protected]

flushSerial

8.34.4.12 result_t ydlidar::YDlidarDriver::getData (uint8_t* data, size_t size) [protected]

获取串口数据

Parameters

in	<i>data</i>	数据指针
in	<i>size</i>	数据大小

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	获取成功
<i>RESULT_FAILE</i>	获取失败

8.34.4.13 result_t ydlidar::YDlidarDriver::getDeviceInfo (device_info & info, uint32_t timeout = DEFAULT_TIMEOUT)

获取雷达设备信息

Parameters

in	<i>info</i>	设备信息
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	获取成功
<i>RESULT_FAILE</i>	or RESULT_TIMEOUT 获取失败

8.34.4.14 result_t ydlidar::YDlidarDriver::getHealth (device_health & health, uint32_t timeout = DEFAULT_TIMEOUT)

获取雷达设备健康状态

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	获取成功
<i>RESULT_FAILE</i>	or RESULT_TIMEOUT 获取失败

8.34.4.15 `result_t ydlidar::YDlidarDriver::getSamplingRate (sampling_rate & rate, uint32_t timeout = DEFAULT_TIMEOUT)`

获取激光雷达当前采样频率

Parameters

in	<i>frequency</i>	采样频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.16 `result_t ydlidar::YDlidarDriver::getScanFrequency (scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT)`

获取激光雷达当前扫描频率

Parameters

in	<i>frequency</i>	扫描频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.17 `std::string ydlidar::YDlidarDriver::getSDKVersion () [static]`

获取当前SDK版本号
静态函数

Returns

返回当前SKD 版本号

8.34.4.18 `result_t ydlidar::YDlidarDriver::getZeroOffsetAngle (offset_angle & angle, uint32_t timeout = DEFAULT_TIMEOUT)`

获取激光雷达当前零位角

Parameters

in	<i>angle</i>	零位偏移角
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.19 `result_t ydlidar::YDlidarDriver::grabScanData (node_info * nodebuffer, size_t & count, uint32_t timeout = DEFAULT_TIMEOUT)`

获取激光数据

Parameters

in	<i>nodebuffer</i>	激光点信息
in	<i>count</i>	一圈激光点数
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	获取成功
<i>RESULT_FAILE</i>	获取失败

Note

获取之前，必须使用::startScan函数开启扫描

8.34.4.20 bool ydlidar::YDlidarDriver::isconnected () const

连接雷达状态

Returns

返回连接状态

Return values

<i>true</i>	成功
<i>false</i>	失败

8.34.4.21 bool ydlidar::YDlidarDriver::isscanning () const

扫图状态

Returns

返回当前雷达扫图状态

Return values

<i>true</i>	正在扫图
<i>false</i>	扫图关闭

8.34.4.22 std::map< std::string, std::string > ydlidar::YDlidarDriver::lidarPortList () [static]

lidarPortList 获取雷达端口

Returns

在线雷达列表

8.34.4.23 ydlidar::YDLidarDriver::PropertyBuilderByName (bool , SingleChannel , private)

Set and Get LiDAR single channel. Whether LiDAR communication channel is a single-channel.

Note

For a single-channel LiDAR, if the settings are reversed.
an error will occur in obtaining device information and the LiDAR will Failed to Start.
For dual-channel LiDAR, if th setttings are reversed.
the device information cannot be obtained.
Set the single channel to match the LiDAR.

G1/G2/G2A/G2C	false
G4/G4B/G4PRO/G6/F4/F4PRO	false
S4/S4B/X4/R2/G4C	false
S2/X2/X2L	true
TG15/TG30/TG50	false
TX8/TX20	true
T5/T15	false
	true

Remarks

See also

DriverInterface::setSingleChannel and DriverInterface::getSingleChannel

8.34.4.24 ydlidar::YDLidarDriver::PropertyBuilderByName (int , LidarType , private)

Set and Get LiDAR Type.

Note

Refer to the table below for the LiDAR Type.
Set the LiDAR Type to match the LiDAR.

G1/G2A/G2/G2C	TYPE_TRIANGLE
G4/G4B/G4C/G4PRO	TYPE_TRIANGLE
G6/F4/F4PRO	TYPE_TRIANGLE
S4/S4B/X4/R2/S2/X2/X2L	TYPE_TRIANGLE
TG15/TG30/TG50/TX8/TX20	TYPE_TOF
T5/T15	TYPE_TOF_NET

Remarks

See also

[LidarTypeID](#)

DriverInterface::setLidarType and DriverInterface::getLidarType

8.34.4.25 ydlidar::YDlidarDriver::PropertyBuilderByName (uint32_t , PointTime , private)

Set and Get Sampling interval.

Note

Negative correlation between sampling interval and lidar sampling rate.

sampling interval = 1e9 / sampling rate(/s)

Set the LiDAR sampling interval to match the LiDAR.

See also

DriverInterface::setPointTime and DriverInterface::getPointTime

8.34.4.26 result_t ydlidar::YDlidarDriver::reset (uint32_t timeout = DEFAULT_TIMEOUT)

重置激光雷达

Parameters

in	<i>timeout</i>	超时时间
----	----------------	------

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作, 如果在扫描中调用::stop函数停止扫描

8.34.4.27 `result_t ydlidar::YDlidarDriver::sendCommand (uint8_t cmd, const void * payload = NULL, size_t payloadsize = 0)` [protected]

发送数据到雷达

Parameters

in	<i>cmd</i>	命名码
in	<i>payload</i>	payload
in	<i>payloadsize</i>	payloadsize

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

8.34.4.28 `result_t ydlidar::YDlidarDriver::sendData (const uint8_t * data, size_t size)` [protected]

串口发送数据

Parameters

in	<i>data</i>	发送数据指针
in	<i>size</i>	数据大小

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	发送成功
<i>RESULT_FAILE</i>	发送失败

8.34.4.29 `void ydlidar::YDlidarDriver::setAutoReconnect (const bool & enable)`

设置雷达异常自动重新连接

Parameters

in	<i>enable</i>	是否开启自动重连: true 开启 false 关闭
----	---------------	----------------------------

8.34.4.30 void ydlidar::YDlidarDriver::setDTR () [protected]

设置串口DTR

8.34.4.31 void ydlidar::YDlidarDriver::setIntensities (const bool & *isintensities*)

设置雷达是否带信号质量
连接成功后, 必须使用::disconnect函数关闭

Parameters

in	<i>isintensities</i>	是否带信号质量: true 带信号质量 false 无信号质量
----	----------------------	---------------------------------

Note

只有S4B(波特率是153600)雷达支持带信号质量, 别的型号雷达暂不支持

8.34.4.32 result_t ydlidar::YDlidarDriver::setSamplingRate (sampling_rate & *rate*, uint32_t *timeout* = DEFAULT_TIMEOUT)

设置激光雷达当前采样频率

Parameters

in	<i>rate</i>	采样频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAIL</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.33 `result_t ydlidar::YDlidarDriver::setScanFrequencyAdd (scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT)`

设置增加扫描频率1HZ

Parameters

in	<i>frequency</i>	扫描频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.34 `result_t ydlidar::YDlidarDriver::setScanFrequencyAddMic (scan_frequency & frequency, uint32_t timeout = DEFAULT_TIMEOUT)`

设置增加扫描频率0.1HZ

Parameters

in	<i>frequency</i>	扫描频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.35 **result_t** ydlidar::YDlidarDriver::setScanFrequencyDis (**scan_frequency** & *frequency*, uint32_t *timeout* = **DEFAULT_TIMEOUT**)

设置减小扫描频率1HZ

Parameters

in	<i>frequency</i>	扫描频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.36 **result_t** ydlidar::YDlidarDriver::setScanFrequencyDisMic (**scan_frequency** & *frequency*, uint32_t *timeout* = **DEFAULT_TIMEOUT**)

设置减小扫描频率0.1HZ

Parameters

in	<i>frequency</i>	扫描频率
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

Note

停止扫描后再执行当前操作

8.34.4.37 `result_t ydlidar::YDlidarDriver::startAutoScan (bool force = false, uint32_t timeout = DEFAULT_TIMEOUT)`
[protected]

重新连接开启扫描

Parameters

in	<i>force</i>	扫描模式
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	开启成功
<i>RESULT_FAILE</i>	开启失败

Note

sdk 自动重新连接调用

8.34.4.38 `result_t ydlidar::YDlidarDriver::startMotor ()`

打开电机

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

8.34.4.39 `result_t ydlidar::YDlidarDriver::startScan (bool force = false, uint32_t timeout = DEFAULT_TIMEOUT)`

开启扫描

Parameters

in	<i>force</i>	扫描模式
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	开启成功
<i>RESULT_FAILE</i>	开启失败

Note

只用开启一次成功即可

8.34.4.40 result_t ydlidar::YDlidarDriver::stop ()

关闭扫描

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	关闭成功
<i>RESULT_FAILE</i>	关闭失败

8.34.4.41 result_t ydlidar::YDlidarDriver::stopMotor ()

关闭电机

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_FAILE</i>	失败

8.34.4.42 result_t ydlidar::YDlidarDriver::stopScan (uint32_t timeout = DEFAULT_TIMEOUT) [protected]

stopScan

Parameters

<i>timeout</i>	
----------------	--

Returns

8.34.4.43 **result_t** ydlidar::YDlidarDriver::waitDevicePackage (uint32_t *timeout* = DEFAULT_TIMEOUT)
[protected]

waitDevicePackage

Parameters

<i>timeout</i>	
----------------	--

Returns

8.34.4.44 **result_t** ydlidar::YDlidarDriver::waitForData (size_t *data_count*, uint32_t *timeout* = DEFAULT_TIMEOUT, size_t * *returned_size* = NULL) [protected]

等待固定数量串口数据

Parameters

in	<i>data_count</i>	等待数据大小
in	<i>timeout</i>	等待时间
in	<i>returned_size</i>	实际数据大小

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	获取成功
<i>RESULT_TIMEOUT</i>	等待超时
<i>RESULT_FAILE</i>	获取失败

Note

当timeout = -1 时, 将一直等待

8.34.4.45 **result_t** ydlidar::YDlidarDriver::waitPackage (**node_info** * *node*, uint32_t *timeout* = DEFAULT_TIMEOUT)
[protected]

解包激光数据

Parameters

in	<i>node</i>	解包后激光点信息
in	<i>timeout</i>	超时时间

8.34.4.46 **result_t** ydlidar::YDlidarDriver::waitResponseHeader (**lidar_ans_header** * *header*, uint32_t *timeout* = DEFAULT_TIMEOUT) [protected]

等待激光数据包头

Parameters

in	<i>header</i>	包头
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	获取成功
<i>RESULT_TIMEOUT</i>	等待超时
<i>RESULT_FAILE</i>	获取失败

Note

当timeout = -1 时, 将一直等待

8.34.4.47 **result_t** ydlidar::YDlidarDriver::waitScanData (**node_info** * *nodebuffer*, **size_t** & *count*, uint32_t *timeout* = DEFAULT_TIMEOUT) [protected]

发送数据到雷达

Parameters

in	<i>nodebuffer</i>	激光信息指针
in	<i>count</i>	激光点数大小
in	<i>timeout</i>	超时时间

Returns

返回执行结果

Return values

<i>RESULT_OK</i>	成功
<i>RESULT_TIMEOUT</i>	等待超时
<i>RESULT_FAILE</i>	失败

8.34.5 Member Data Documentation**8.34.5.1 Event ydlidar::YDlidarDriver::_dataEvent**

数据同步事件

8.34.5.2 Locker ydlidar::YDlidarDriver::_lock

线程锁

8.34.5.3 serial::Serial* ydlidar::YDlidarDriver::_serial [private]

串口

8.34.5.4 Locker ydlidar::YDlidarDriver::_serial_lock

串口锁

8.34.5.5 Thread ydlidar::YDlidarDriver::_thread

线程id

8.34.5.6 uint16_t ydlidar::YDlidarDriver::async_size [private]**8.34.5.7 int ydlidar::YDlidarDriver::asyncRecvPos** [private]**8.34.5.8 uint16_t ydlidar::YDlidarDriver::Checksum** [private]

校验和

8.34.5.9 `uint16_t ydlidar::YDlidarDriver::ChecksumCal` [private]

8.34.5.10 `bool ydlidar::YDlidarDriver::ChecksumResult` [private]

8.34.5.11 `uint16_t ydlidar::YDlidarDriver::FirstSampleAngle` [private]

起始采样角

8.34.5.12 `bool ydlidar::YDlidarDriver::get_device_health_success` [private]

8.34.5.13 `bool ydlidar::YDlidarDriver::get_device_info_success` [private]

8.34.5.14 `uint8_t* ydlidar::YDlidarDriver::globalRecvBuffer` [private]

8.34.5.15 `bool ydlidar::YDlidarDriver::has_device_header` [private]

8.34.5.16 `bool ydlidar::YDlidarDriver::has_package_error` [private]

8.34.5.17 `lidar_ans_header ydlidar::YDlidarDriver::header_` [private]

8.34.5.18 `uint8_t* ydlidar::YDlidarDriver::headerBuffer` [private]

8.34.5.19 `device_health ydlidar::YDlidarDriver::health_` [private]

8.34.5.20 `uint8_t* ydlidar::YDlidarDriver::healthBuffer` [private]

8.34.5.21 `device_info ydlidar::YDlidarDriver::info_` [private]

8.34.5.22 `uint8_t* ydlidar::YDlidarDriver::infoBuffer` [private]

8.34.5.23 `float ydlidar::YDlidarDriver::IntervalSampleAngle` [private]

8.34.5.24 `float ydlidar::YDlidarDriver::IntervalSampleAngle_LastPackage` [private]

8.34.5.25 `std::atomic<bool> ydlidar::YDlidarDriver::isAutoconnting`

是否正在自动连接中

8.34.5.26 `std::atomic<bool> ydlidar::YDlidarDriver::isAutoReconnect`

异常自动从新连接

8.34.5.27 `std::atomic<bool> ydlidar::YDlidarDriver::isConnected`

串口连接状态

8.34.5.28 `std::atomic<bool> ydlidar::YDlidarDriver::isScanning`

扫图状态

8.34.5.29 `bool ydlidar::YDlidarDriver::isSupportMotorDtrCtrl` [private]

是否支持电机控制

8.34.5.30 `uint8_t ydlidar::YDlidarDriver::last_device_byte` [private]

8.34.5.31 `uint16_t ydlidar::YDlidarDriver::LastSampleAngle` [private]

结束采样角

8.34.5.32 `uint16_t ydlidar::YDlidarDriver::LastSampleAngleCal` [private]

8.34.5.33 `uint32_t ydlidar::YDlidarDriver::m_baudrate` [private]

波特率

8.34.5.34 `bool ydlidar::YDlidarDriver::m_intensities` [private]

信号质量状体

8.34.5.35 `int ydlidar::YDlidarDriver::m_sampling_rate` [private]

采样频率

8.34.5.36 `int ydlidar::YDlidarDriver::model` [private]

雷达型号

8.34.5.37 `node_package ydlidar::YDlidarDriver::package` [private]

带信号质量协议包

8.34.5.38 `int ydlidar::YDlidarDriver::package_index` [private]

8.34.5.39 `uint16_t ydlidar::YDlidarDriver::package_Sample_Index` [private]

包采样点索引

8.34.5.40 `node_packages ydlidar::YDlidarDriver::packages` [private]

不带信好质量协议包

8.34.5.41 `int ydlidar::YDlidarDriver::PackageSampleBytes` [private]

一个包包含的激光点数

8.34.5.42 `int ydlidar::YDlidarDriver::retryCount` [private]

8.34.5.43 `int ydlidar::YDlidarDriver::sample_rate` [private]

8.34.5.44 `uint16_t ydlidar::YDlidarDriver::SampleNumIAndCTCal` [private]

8.34.5.45 `uint8_t ydlidar::YDlidarDriver::scan_frequence` [private]

协议中雷达转速

8.34.5.46 `node_info* ydlidar::YDlidarDriver::scan_node_buf`

激光点信息

8.34.5.47 `size_t ydlidar::YDlidarDriver::scan_node_count`

激光点数

8.34.5.48 `std::string ydlidar::YDlidarDriver::serial_port` [private]

雷达端口

8.34.5.49 `uint32_t ydlidar::YDlidarDriver::trans_delay` [private]

串口传输一个byte时间

8.34.5.50 `uint16_t ydlidar::YDlidarDriver::Valu8Tou16` [private]

The documentation for this class was generated from the following files:

- [include/ydlidar_driver.h](#)
- [src/ydlidar_driver.cpp](#)

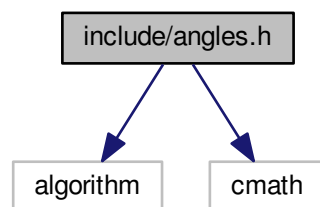
Chapter 9

File Documentation

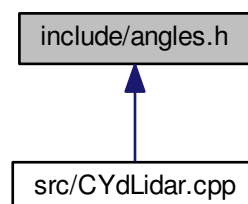
9.1 include/angles.h File Reference

```
#include <algorithm>
#include <cmath>
```

Include dependency graph for angles.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [angles](#)

Macros

- `#define M_PI 3.1415926`

Functions

- static double [angles::from_degrees](#) (double degrees)
Convert degrees to radians.
- static double [angles::to_degrees](#) (double radians)
Convert radians to degrees.
- static double [angles::normalize_angle_positive](#) (double [angle](#))
normalize_angle_positive
- static double [angles::normalize_angle](#) (double [angle](#))
normalize
- static double [angles::shortest_angular_distance](#) (double from, double to)
shortest_angular_distance
- static double [angles::two_pi_complement](#) (double [angle](#))
*returns the angle in $[-2*M_PI, 2*M_PI]$ going the other way along the unit circle.*
- static bool [angles::find_min_max_delta](#) (double from, double left_limit, double right_limit, double &result_min_delta, double &result_max_delta)
This function is only intended for internal use and not intended for external use. If you do use it, read the documentation very carefully. Returns the min and max amount (in radians) that can be moved from "from" angle to "left_limit" and "right_limit".
- static bool [angles::shortest_angular_distance_with_limits](#) (double from, double to, double left_limit, double right_limit, double &shortest_angle)
*Returns the delta from "from_angle" to "to_angle" making sure it does not violate limits specified by left_limit and right_limit. The valid interval of angular positions is $[left_limit, right_limit]$. E.g., $[-0.25, 0.25]$ is a 0.5 radians wide interval that contains 0. But $[0.25, -0.25]$ is a $2*M_PI-0.5$ wide interval that contains M_PI (but not 0). The value of shortest_angle is the angular difference between "from" and "to" that lies within the defined valid interval. E.g. `shortest_angular_distance_with_limits(-0.5, 0.5, 0.25, -0.25, ss)` evaluates ss to $2*M_PI-1.0$ and returns true while `shortest_angular_distance_with_limits(-0.5, 0.5, -0.25, 0.25, ss)` returns false since -0.5 and 0.5 do not lie in the interval $[-0.25, 0.25]$.*

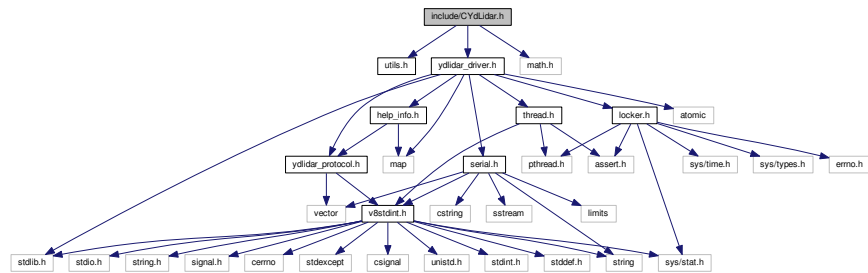
9.1.1 Macro Definition Documentation

9.1.1.1 `#define M_PI 3.1415926`

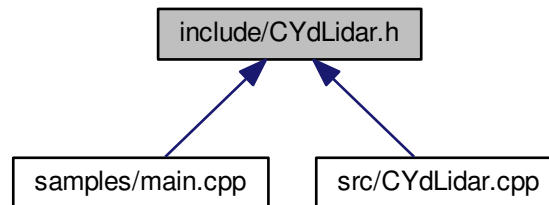
9.2 include/CYdLidar.h File Reference

```
#include "utils.h"
#include "ydlidar_driver.h"
#include <math.h>
```

Include dependency graph for CYdLidar.h:



This graph shows which files directly or indirectly include this file:



Classes

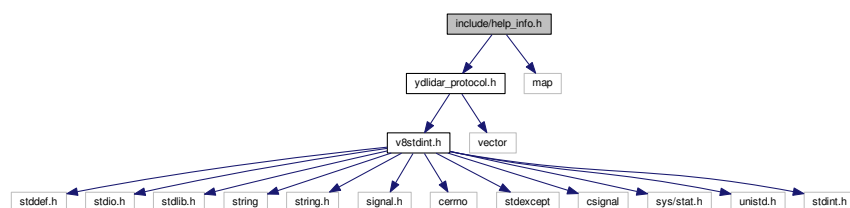
- class [CYdLidar](#)

9.3 include/help_info.h File Reference

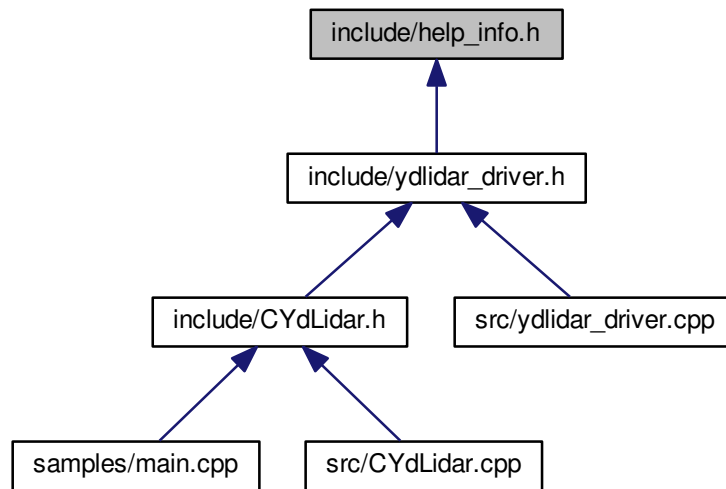
```
#include <ydlidar_protocol.h>
```

```
#include <map>
```

Include dependency graph for help_info.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [ydlidar](#)

Enumerations

- enum [ydlidar::YDLIDAR_MODULES](#) {
[ydlidar::YDLIDAR_F4](#) = 1, [ydlidar::YDLIDAR_T1](#) = 2, [ydlidar::YDLIDAR_F2](#) = 3, [ydlidar::YDLIDAR_S4](#) = 4,
[ydlidar::YDLIDAR_G4](#) = 5, [ydlidar::YDLIDAR_X4](#) = 6, [ydlidar::YDLIDAR_G4PRO](#) = 7, [ydlidar::YDLIDAR_F4PRO](#) = 8,
[ydlidar::YDLIDAR_R2](#) = 9, [ydlidar::YDLIDAR_G10](#) = 10, [ydlidar::YDLIDAR_S4B](#) = 11, [ydlidar::YDLIDAR_S2](#)
= 12,
[ydlidar::YDLIDAR_G6](#) = 13, [ydlidar::YDLIDAR_G2A](#) = 14, [ydlidar::YDLIDAR_G2B](#) = 15, [ydlidar::YDLIDAR_G2C](#) = 16,
[ydlidar::YDLIDAR_G4B](#) = 17, [ydlidar::YDLIDAR_G4C](#) = 18, [ydlidar::YDLIDAR_G1](#) = 19, [ydlidar::YDLIDAR_TG15](#) = 100,
[ydlidar::YDLIDAR_TG30](#) = 101, [ydlidar::YDLIDAR_TG50](#) = 102, [ydlidar::YDLIDAR_Tail](#) }
- enum [ydlidar::YDLIDAR_RATE](#) { [ydlidar::YDLIDAR_RATE_4K](#) = 0, [ydlidar::YDLIDAR_RATE_8K](#) = 1,
[ydlidar::YDLIDAR_RATE_9K](#) = 2, [ydlidar::YDLIDAR_RATE_10K](#) = 3 }

Functions

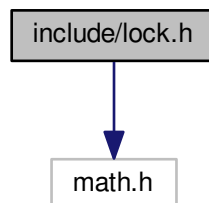
- std::string [ydlidar::lidarModelToString](#) (int [model](#))
[lidarModelToString](#)
- int [ydlidar::lidarModelDefaultSampleRate](#) (int [model](#))
[lidarModelDefaultSampleRate](#)
- bool [ydlidar::isOctaveLidar](#) (int [model](#))
[isOctaveLidar](#)

- bool [ydlidar::hasSampleRate](#) (int [model](#))
hasSampleRate
- bool [ydlidar::hasZeroAngle](#) (int [model](#))
hasZeroAngle
- bool [ydlidar::hasScanFrequencyCtrl](#) (int [model](#))
hasScanFrequencyCtrl
- bool [ydlidar::isSupportLidar](#) (int [model](#))
isSupportLidar
- bool [ydlidar::hasIntensity](#) (int [model](#))
hasIntensity
- bool [ydlidar::isSupportMotorCtrl](#) (int [model](#))
isSupportMotorCtrl
- bool [ydlidar::isSupportScanFrequency](#) (int [model](#), double [frequency](#))
isSupportScanFrequency
- bool [ydlidar::isTOFLidar](#) (int [type](#))
- bool [ydlidar::isOldVersionTOFLidar](#) (int [model](#), int Major, int Minor)
- bool [ydlidar::isValidSampleRate](#) (std::map< int, int > [smap](#))
- int [ydlidar::ConvertUserToLidarSmaple](#) (int [model](#), int m_SampleRate, int defaultRate)
- int [ydlidar::ConvertLidarToUserSmaple](#) (int [model](#), int [rate](#))
- bool [ydlidar::isValidValue](#) (uint8_t value)
- bool [ydlidar::isVersionValid](#) (const [LaserDebug](#) &info)
- bool [ydlidar::isSerialNumbValid](#) (const [LaserDebug](#) &info)
- bool [ydlidar::ParseLaserDebugInfo](#) (const [LaserDebug](#) &info, [device_info](#) &value)

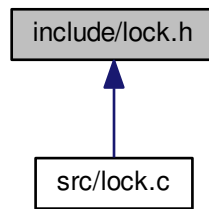
9.4 include/lock.h File Reference

```
#include <math.h>
```

Include dependency graph for lock.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define LOCK system_does_not_lock`
- `#define UNLOCK system_does_not_unlock`

Functions

- `int check_group_uucp ()`
- `int check_lock_pid (const char *file, int openpid)`
- `int lock_device (const char *)`
- `void unlock_device (const char *)`
- `int is_device_locked (const char *)`
- `int check_lock_status (const char *)`
- `int lfs_unlock (const char *, int)`
- `int lfs_lock (const char *, int)`
- `int lib_lock_dev_unlock (const char *, int)`
- `int lib_lock_dev_lock (const char *, int)`
- `void fhs_unlock (const char *, int)`
- `int fhs_lock (const char *, int)`
- `void uucp_unlock (const char *, int)`
- `int uucp_lock (const char *, int)`

9.4.1 Macro Definition Documentation

9.4.1.1 `#define LOCK system_does_not_lock`

9.4.1.2 `#define UNLOCK system_does_not_unlock`

9.4.2 Function Documentation

9.4.2.1 `int check_group_uucp ()`

9.4.2.2 `int check_lock_pid (const char * file, int openpid)`

9.4.2.3 int check_lock_status (const char *)

9.4.2.4 int fhs_lock (const char *, int)

9.4.2.5 void fhs_unlock (const char *, int)

9.4.2.6 int is_device_locked (const char *)

9.4.2.7 int lfs_lock (const char *, int)

9.4.2.8 int lfs_unlock (const char *, int)

9.4.2.9 int lib_lock_dev_lock (const char *, int)

9.4.2.10 int lib_lock_dev_unlock (const char *, int)

9.4.2.11 int lock_device (const char *)

9.4.2.12 void unlock_device (const char *)

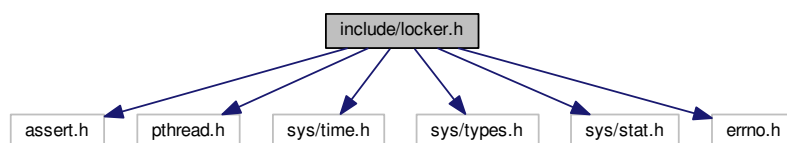
9.4.2.13 int uucp_lock (const char *, int)

9.4.2.14 void uucp_unlock (const char *, int)

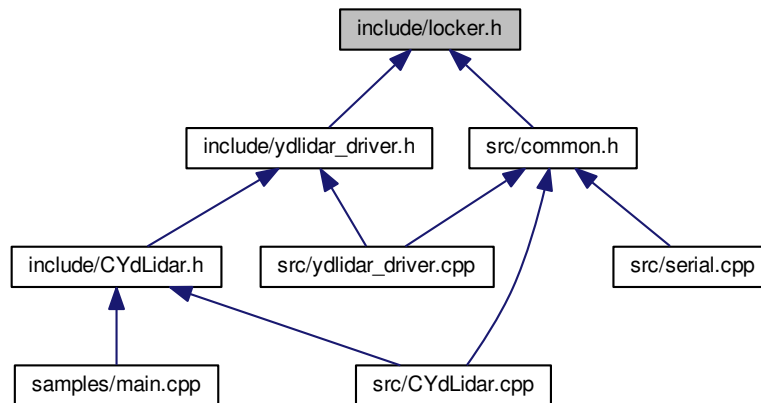
9.5 include/locker.h File Reference

```
#include <assert.h>
#include <pthread.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
```

Include dependency graph for locker.h:



This graph shows which files directly or indirectly include this file:



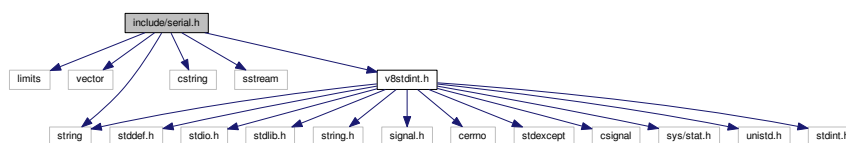
Classes

- class [Locker](#)
- class [Event](#)
- class [ScopedLocker](#)

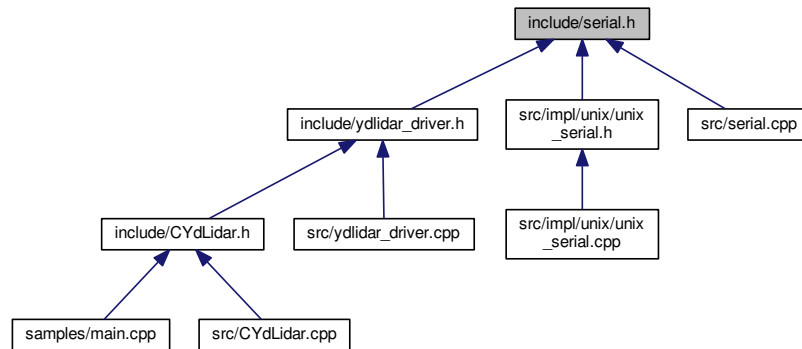
9.6 include/serial.h File Reference

```
#include <limits>
#include <vector>
#include <string>
#include <cstring>
#include <sstream>
#include "v8stdint.h"
```

Include dependency graph for serial.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [serial::Timeout](#)
- class [serial::Serial](#)
- struct [serial::PortInfo](#)

Namespaces

- [serial](#)

Enumerations

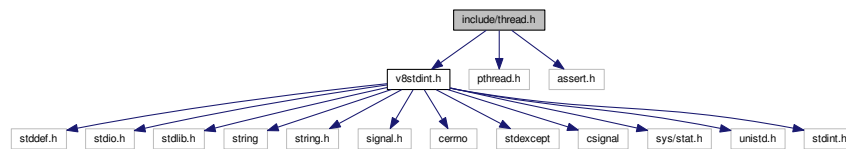
- enum [serial::bytesize_t](#) { [serial::fivebits](#) = 5, [serial::sixbits](#) = 6, [serial::sevenbits](#) = 7, [serial::eightbits](#) = 8 }
- enum [serial::parity_t](#) { [serial::parity_none](#) = 0, [serial::parity_odd](#) = 1, [serial::parity_even](#) = 2, [serial::parity_mark](#) = 3, [serial::parity_space](#) = 4 }
- enum [serial::stopbits_t](#) { [serial::stopbits_one](#) = 1, [serial::stopbits_two](#) = 2, [serial::stopbits_one_point_five](#) }
- enum [serial::flowcontrol_t](#) { [serial::flowcontrol_none](#) = 0, [serial::flowcontrol_software](#), [serial::flowcontrol_hardware](#) }

Functions

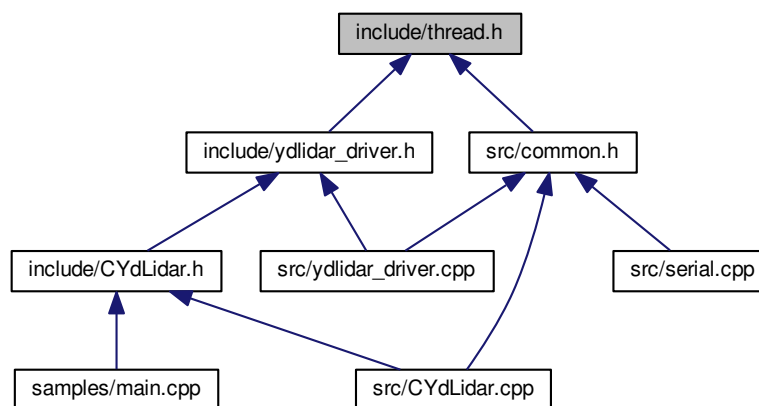
- `std::vector< PortInfo > serial::list_ports ()`

9.7 include/thread.h File Reference

```
#include "v8stdint.h"
#include <pthread.h>
#include <assert.h>
Include dependency graph for thread.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [Thread](#)

Macros

- #define [UNUSED](#)(x) (void)x
- #define [CLASS_THREAD](#)(c, x) [Thread::ThreadCreateObjectFunctor](#)<c, &c::x>(this)

9.7.1 Macro Definition Documentation

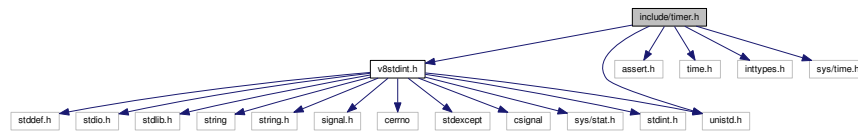
9.7.1.1 `#define CLASS_THREAD(c, x) Thread::ThreadCreateObjectFunctor<c, &c::x>(this)`

9.7.1.2 `#define UNUSED(x) (void)x`

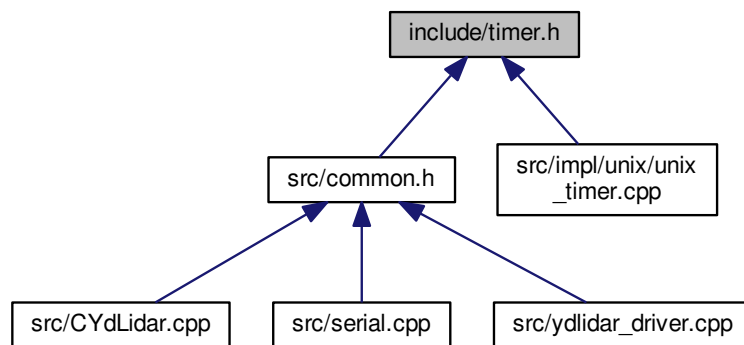
9.8 include/timer.h File Reference

```
#include "v8stdint.h"
#include <assert.h>
#include <time.h>
#include <inttypes.h>
#include <sys/time.h>
#include <unistd.h>
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [impl](#)

Macros

- `#define BEGIN_STATIC_CODE(_blockname_)`
- `#define END_STATIC_CODE(_blockname_) } _instance_##_blockname_;`
- `#define getms() impl::getHDTimer()`
- `#define getTime() impl::getCurrentTime()`

Functions

- static void `delay` (uint32_t ms)
- uint32_t `impl::getHDTimer` ()
- uint64_t `impl::getCurrentTime` ()

9.8.1 Macro Definition Documentation

9.8.1.1 `#define BEGIN_STATIC_CODE(_blockname_)`

Value:

```
static class _static_code_##_blockname_ { \
public: \
    _static_code_##_blockname_ ()
```

9.8.1.2 `#define END_STATIC_CODE(_blockname_) _instance_##_blockname_;`

9.8.1.3 `#define getms() impl::getHDTimer()`

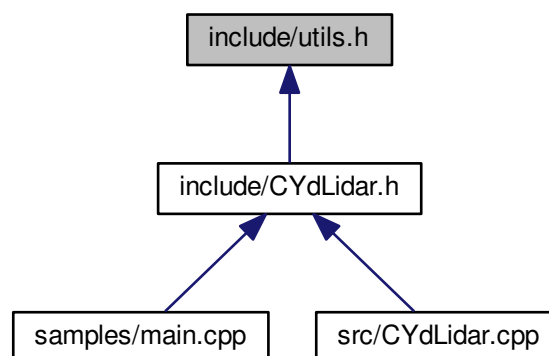
9.8.1.4 `#define getTime() impl::getCurrentTime()`

9.8.2 Function Documentation

9.8.2.1 static void `delay` (uint32_t *ms*) `[inline],[static]`

9.9 include/utils.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- `#define YDLIDAR_API`

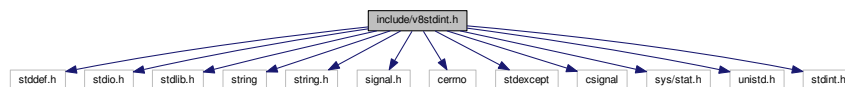
9.9.1 Macro Definition Documentation

9.9.1.1 `#define YDLIDAR_API`

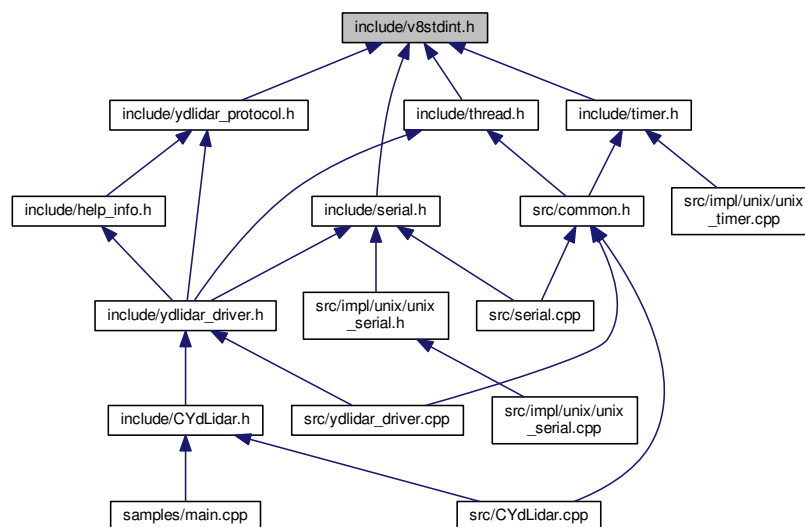
9.10 include/v8stdint.h File Reference

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <string.h>
#include <signal.h>
#include <cerrno>
#include <stdexcept>
#include <csignal>
#include <sys/stat.h>
#include <unistd.h>
#include <stdint.h>
```

Include dependency graph for v8stdint.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [ydlidar](#)

Macros

- `#define UNUSED(x) (void)x`
- `#define _access access`
- `#define __small_endian`
- `#define __attribute__(x)`
- `#define RESULT_OK 0`
- `#define RESULT_TIMEOUT -1`
- `#define RESULT_FAIL -2`
- `#define INVALID_TIMESTAMP (0)`
- `#define IS_OK(x) ((x) == RESULT_OK)`
- `#define IS_TIMEOUT(x) ((x) == RESULT_TIMEOUT)`
- `#define IS_FAIL(x) ((x) == RESULT_FAIL)`

Typedefs

- `typedef _size_t(THREAD_PROC * thread_proc_t) (void *)`
- `typedef int32_t result_t`
- `typedef void(* signal_handler_t) (int)`

Enumerations

- `enum { DEVICE_DRIVER_TYPE_SERIALPORT = 0x0, DEVICE_DRIVER_TYPE_TCP = 0x1 }`

Functions

- `signal_handler_t set_signal_handler (int signal_value, signal_handler_t signal_handler)`
- `void trigger_interrupt_guard_condition (int signal_value)`
- `void signal_handler (int signal_value)`
- `void ydlidar::init (int argc, char *argv[])`
- `bool ydlidar::ok ()`
- `void ydlidar::shutdownNow ()`
- `bool ydlidar::fileExists (const std::string filename)`

Variables

- `static volatile sig_atomic_t g_signal_status = 0`
- `static signal_handler_t old_signal_handler = 0`

9.10.1 Macro Definition Documentation

9.10.1.1 `#define __attribute__(x)`

9.10.1.2 `#define __small_endian`

9.10.1.3 `#define _access access`

9.10.1.4 `#define INVALID_TIMESTAMP (0)`

9.10.1.5 `#define IS_FAIL(x) ((x) == RESULT_FAIL)`

9.10.1.6 `#define IS_OK(x) ((x) == RESULT_OK)`

9.10.1.7 `#define IS_TIMEOUT(x) ((x) == RESULT_TIMEOUT)`

9.10.1.8 `#define RESULT_FAIL -2`

9.10.1.9 `#define RESULT_OK 0`

9.10.1.10 `#define RESULT_TIMEOUT -1`

9.10.1.11 `#define UNUSED(x) (void)x`

9.10.2 Typedef Documentation

9.10.2.1 `typedef int32_t result_t`

9.10.2.2 `typedef void(* signal_handler_t) (int)`

9.10.2.3 `typedef _size_t(THREAD_PROC * thread_proc_t) (void *)`

9.10.3 Enumeration Type Documentation

9.10.3.1 anonymous enum

Enumerator

DEVICE_DRIVER_TYPE_SERIALPORT

DEVICE_DRIVER_TYPE_TCP

9.10.4 Function Documentation

9.10.4.1 `signal_handler_t set_signal_handler(int signal_value, signal_handler_t signal_handler)` [inline]

9.10.4.2 `void signal_handler(int signal_value)` [inline]

9.10.4.3 `void trigger_interrupt_guard_condition(int signal_value)` [inline]

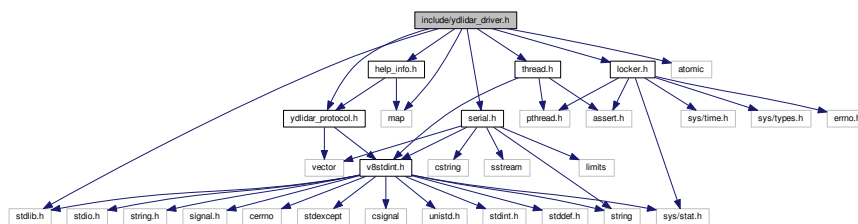
9.10.5 Variable Documentation

9.10.5.1 `volatile sig_atomic_t g_signal_status = 0` [static]

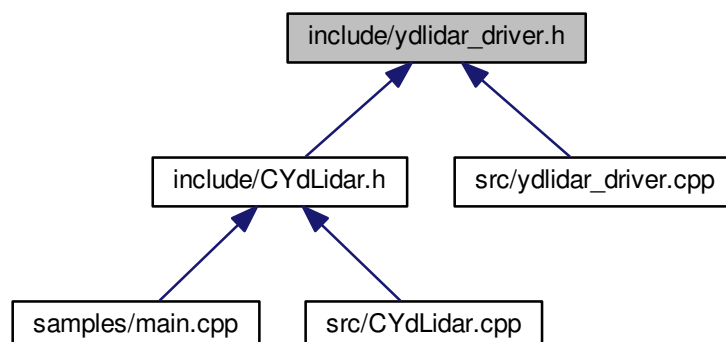
9.10.5.2 `signal_handler_t old_signal_handler = 0` [static]

9.11 include/ydlidar_driver.h File Reference

```
#include <stdlib.h>
#include <atomic>
#include <map>
#include "serial.h"
#include "locker.h"
#include "thread.h"
#include "ydlidar_protocol.h"
#include "help_info.h"
Include dependency graph for ydlidar_driver.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ydlidar::YDlidarDriver](#)

Namespaces

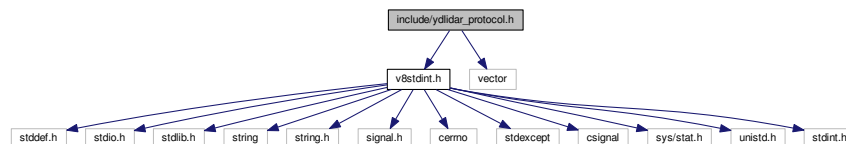
- [ydlidar](#)

9.12 include/ydlidar_protocol.h File Reference

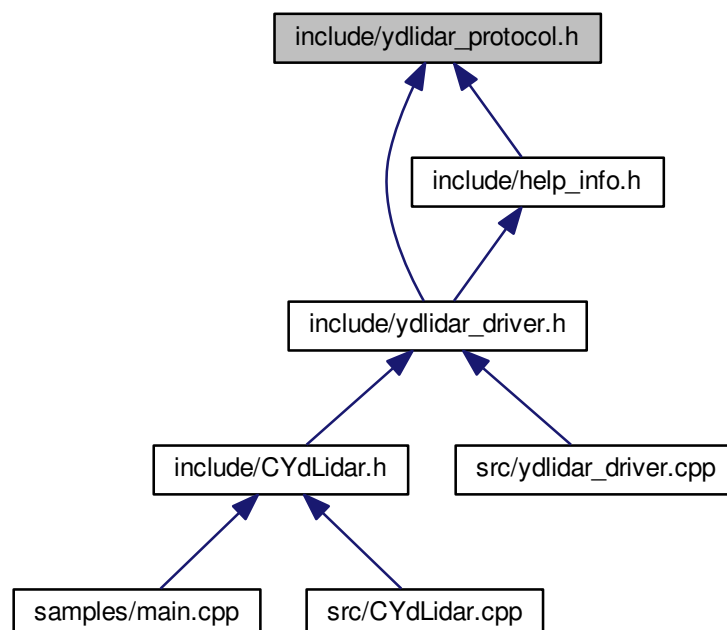
```
#include "v8stdint.h"
```

```
#include <vector>
```

Include dependency graph for ydlidar_protocol.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [node_info](#)
- struct [PackageNode](#)
- struct [node_package](#)
- struct [node_packages](#)
- struct [device_info](#)
- struct [device_health](#)
- struct [sampling_rate](#)
- struct [scan_frequency](#)
- struct [scan_rotation](#)
- struct [scan_exposure](#)
- struct [scan_heart_beat](#)
- struct [scan_points](#)
- struct [function_state](#)
- struct [offset_angle](#)
- struct [cmd_packet](#)
- struct [lidar_ans_header](#)
- struct [LaserPoint](#)
- struct [LaserDebug](#)
- struct [LaserConfig](#)

A struct for returning configuration from the YDLIDAR.

- struct [LaserScan](#)

Macros

- `#define PropertyBuilderByName(type, name, access_permission)`
- `#define _countof(_Array) (int)(sizeof(_Array) / sizeof(_Array[0]))`
- `#define M_PI 3.1415926`
- `#define SUNNOISEINTENSITY 0xff`
- `#define GLASSNOISEINTENSITY 0xfe`
- `#define LIDAR_CMD_STOP 0x65`
- `#define LIDAR_CMD_SCAN 0x60`
- `#define LIDAR_CMD_FORCE_SCAN 0x61`
- `#define LIDAR_CMD_RESET 0x80`
- `#define LIDAR_CMD_FORCE_STOP 0x00`
- `#define LIDAR_CMD_GET_EAI 0x55`
- `#define LIDAR_CMD_GET_DEVICE_INFO 0x90`
- `#define LIDAR_CMD_GET_DEVICE_HEALTH 0x92`
- `#define LIDAR_ANS_TYPE_DEVINFO 0x4`
- `#define LIDAR_ANS_TYPE_DEVHEALTH 0x6`
- `#define LIDAR_CMD_SYNC_BYTE 0xA5`
- `#define LIDAR_CMDFLAG_HAS_PAYLOAD 0x80`
- `#define LIDAR_ANS_SYNC_BYTE1 0xA5`
- `#define LIDAR_ANS_SYNC_BYTE2 0x5A`
- `#define LIDAR_ANS_TYPE_MEASUREMENT 0x81`
- `#define LIDAR_RESP_MEASUREMENT_SYNCBIT (0x1<<0)`
- `#define LIDAR_RESP_MEASUREMENT_QUALITY_SHIFT 2`
- `#define LIDAR_RESP_MEASUREMENT_CHECKBIT (0x1<<0)`
- `#define LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT 1`
- `#define LIDAR_RESP_MEASUREMENT_DISTANCE_SHIFT 2`
- `#define LIDAR_RESP_MEASUREMENT_ANGLE_SAMPLE_SHIFT 8`
- `#define LIDAR_CMD_RUN_POSITIVE 0x06`
- `#define LIDAR_CMD_RUN_INVERSION 0x07`

- `#define LIDAR_CMD_SET_AIMSPEED_ADDMIC 0x09`
- `#define LIDAR_CMD_SET_AIMSPEED_DISMIC 0x0A`
- `#define LIDAR_CMD_SET_AIMSPEED_ADD 0x0B`
- `#define LIDAR_CMD_SET_AIMSPEED_DIS 0x0C`
- `#define LIDAR_CMD_GET_AIMSPEED 0x0D`
- `#define LIDAR_CMD_SET_SAMPLING_RATE 0xD0`
- `#define LIDAR_CMD_GET_SAMPLING_RATE 0xD1`
- `#define LIDAR_STATUS_OK 0x0`
- `#define LIDAR_STATUS_WARNING 0x1`
- `#define LIDAR_STATUS_ERROR 0x2`
- `#define LIDAR_CMD_ENABLE_LOW_POWER 0x01`
- `#define LIDAR_CMD_DISABLE_LOW_POWER 0x02`
- `#define LIDAR_CMD_STATE_MODEL_MOTOR 0x05`
- `#define LIDAR_CMD_ENABLE_CONST_FREQ 0x0E`
- `#define LIDAR_CMD_DISABLE_CONST_FREQ 0x0F`
- `#define LIDAR_CMD_GET_OFFSET_ANGLE 0x93`
- `#define LIDAR_CMD_SAVE_SET_EXPOSURE 0x94`
- `#define LIDAR_CMD_SET_LOW_EXPOSURE 0x95`
- `#define LIDAR_CMD_ADD_EXPOSURE 0x96`
- `#define LIDAR_CMD_DIS_EXPOSURE 0x97`
- `#define PackageSampleMaxLength 0x100`
- `#define Node_Default_Quality (10)`
- `#define Node_Sync 1`
- `#define Node_NotSync 2`
- `#define PackagePaidBytes 10`
- `#define PH 0x55AA`
- `#define NORMAL_PACKAGE_SIZE 90`
- `#define INTENSITY_NORMAL_PACKAGE_SIZE 130`

Enumerations

- enum `CT` { `CT_Normal` = 0, `CT_RingStart` = 1, `CT_Tail` }
- enum `LidarTypeID` { `TYPE_TOF` = 0, `TYPE_TRIANGLE` = 1, `TYPE_Tail` }

Functions

- struct `node_info` `__attribute__((packed))`

Variables

- `uint8_t sync_flag`
- `uint16_t sync_quality`
- `uint16_t angle_q6_checkbit`
信号质量
- `uint16_t distance_q2`
测距点角度
- `uint64_t stamp`
当前测距点距离
- `uint8_t scan_frequence`
时间戳
- `uint8_t debug_info` [12]

特定版本此值才有效,无效值是0

- uint8_t [index](#)
- uint8_t [PackageSampleQuality](#)
- uint16_t [PackageSampleDistance](#)
- uint16_t [package_Head](#)
- uint8_t [package_CT](#)
- uint8_t [nowPackageNum](#)
- uint16_t [packageFirstSampleAngle](#)
- uint16_t [packageLastSampleAngle](#)
- uint16_t [checkSum](#)
- [PackageNode](#) [packageSample](#) [[PackageSampleMaxLngth](#)]
- uint16_t [packageSampleDistance](#) [[PackageSampleMaxLngth](#)]
- uint8_t [model](#)
雷达型号
- uint16_t [firmware_version](#)
固件版本号
- uint8_t [hardware_version](#)
硬件版本号
- uint8_t [serialnum](#) [16]
系列号
- uint8_t [status](#)
健康状况
- uint16_t [error_code](#)
错误代码
- uint8_t [rate](#)
采样频率
- uint32_t [frequency](#)
扫描频率
- uint8_t [rotation](#)
- uint8_t [exposure](#)
低光功率模式
- uint8_t [enable](#)
掉电保护状态
- uint8_t [flag](#)
- uint8_t [state](#)
- int32_t [angle](#)
- uint8_t [syncByte](#)
- uint8_t [cmd_flag](#)
- uint8_t [size](#)
- uint8_t [data](#)
- uint8_t [syncByte1](#)
- uint8_t [syncByte2](#)
- uint32_t [subType](#)
- uint8_t [type](#)
- struct [LaserPoint](#) [__attribute__](#)

9.12.1 Macro Definition Documentation

9.12.1.1 `#define _countof(_Array) (int)(sizeof(_Array) / sizeof(_Array[0]))`

9.12.1.2 `#define GLASSNOISEINTENSITY 0xfe`

9.12.1.3 `#define INTENSITY_NORMAL_PACKAGE_SIZE 130`

9.12.1.4 `#define LIDAR_ANS_SYNC_BYTE1 0xA5`

9.12.1.5 `#define LIDAR_ANS_SYNC_BYTE2 0x5A`

9.12.1.6 `#define LIDAR_ANS_TYPE_DEVHEALTH 0x6`

9.12.1.7 `#define LIDAR_ANS_TYPE_DEVINFO 0x4`

9.12.1.8 `#define LIDAR_ANS_TYPE_MEASUREMENT 0x81`

9.12.1.9 `#define LIDAR_CMD_ADD_EXPOSURE 0x96`

9.12.1.10 `#define LIDAR_CMD_DIS_EXPOSURE 0x97`

9.12.1.11 `#define LIDAR_CMD_DISABLE_CONST_FREQ 0x0F`

9.12.1.12 `#define LIDAR_CMD_DISABLE_LOW_POWER 0x02`

9.12.1.13 `#define LIDAR_CMD_ENABLE_CONST_FREQ 0x0E`

9.12.1.14 `#define LIDAR_CMD_ENABLE_LOW_POWER 0x01`

9.12.1.15 `#define LIDAR_CMD_FORCE_SCAN 0x61`

9.12.1.16 `#define LIDAR_CMD_FORCE_STOP 0x00`

9.12.1.17 `#define LIDAR_CMD_GET_AIMSPEED 0x0D`

9.12.1.18 `#define LIDAR_CMD_GET_DEVICE_HEALTH 0x92`

9.12.1.19 `#define LIDAR_CMD_GET_DEVICE_INFO 0x90`

9.12.1.20 `#define LIDAR_CMD_GET_EAI 0x55`

9.12.1.21 `#define LIDAR_CMD_GET_OFFSET_ANGLE 0x93`

9.12.1.22 `#define LIDAR_CMD_GET_SAMPLING_RATE 0xD1`

9.12.1.23 `#define LIDAR_CMD_RESET 0x80`

9.12.1.24 `#define LIDAR_CMD_RUN_INVERSION 0x07`

9.12.1.25 `#define LIDAR_CMD_RUN_POSITIVE 0x06`

9.12.1.26 `#define LIDAR_CMD_SAVE_SET_EXPOSURE 0x94`

9.12.1.27 `#define LIDAR_CMD_SCAN 0x60`

9.12.1.28 `#define LIDAR_CMD_SET_AIMSPEED_ADD 0x0B`

9.12.1.29 `#define LIDAR_CMD_SET_AIMSPEED_ADDMIC 0x09`

9.12.1.30 `#define LIDAR_CMD_SET_AIMSPEED_DIS 0x0C`

9.12.1.31 `#define LIDAR_CMD_SET_AIMSPEED_DISMIC 0x0A`

9.12.1.32 `#define LIDAR_CMD_SET_LOW_EXPOSURE 0x95`

9.12.1.33 `#define LIDAR_CMD_SET_SAMPLING_RATE 0xD0`

9.12.1.34 `#define LIDAR_CMD_STATE_MODEL_MOTOR 0x05`

9.12.1.35 `#define LIDAR_CMD_STOP 0x65`

9.12.1.36 `#define LIDAR_CMD_SYNC_BYTE 0xA5`

9.12.1.37 `#define LIDAR_CMDFLAG_HAS_PAYLOAD 0x80`

9.12.1.38 `#define LIDAR_RESP_MEASUREMENT_ANGLE_SAMPLE_SHIFT 8`

9.12.1.39 `#define LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT 1`

9.12.1.40 `#define LIDAR_RESP_MEASUREMENT_CHECKBIT (0x1 << 0)`

9.12.1.41 `#define LIDAR_RESP_MEASUREMENT_DISTANCE_SHIFT 2`

9.12.1.42 `#define LIDAR_RESP_MEASUREMENT_QUALITY_SHIFT 2`

9.12.1.43 `#define LIDAR_RESP_MEASUREMENT_SYNCBIT (0x1 << 0)`

9.12.1.44 `#define LIDAR_STATUS_ERROR 0x2`

9.12.1.45 `#define LIDAR_STATUS_OK 0x0`

9.12.1.46 `#define LIDAR_STATUS_WARNING 0x1`

9.12.1.47 `#define M_PI 3.1415926`

9.12.1.48 `#define Node_Default_Quality (10)`

9.12.1.49 `#define Node_NotSync 2`

9.12.1.50 `#define Node_Sync 1`

9.12.1.51 `#define NORMAL_PACKAGE_SIZE 90`

9.12.1.52 `#define PackagePaidBytes 10`

9.12.1.53 `#define PackageSampleMaxLngth 0x100`

9.12.1.54 `#define PH 0x55AA`

9.12.1.55 `#define PropertyBuilderByName(type, name, access_permission)`

Value:

```
access_permission:\
    type m_##name;\
    public:\
        inline void set##name(type v) {\
            m_##name = v;\
        }\
        inline type get##name() {\
            return m_##name;\
        }\
}
```

9.12.1.56 `#define SUNNOISEINTENSITY 0xff`

9.12.2 Enumeration Type Documentation

9.12.2.1 enum CT

Enumerator

CT_Normal
CT_RingStart
CT_Tail

9.12.2.2 enum LidarTypeID

Enumerator

TYPE_TOF
TYPE_TRIANGLE
TYPE_Tail

9.12.3 Function Documentation

9.12.3.1 struct node_info __attribute__((packed))

9.12.4 Variable Documentation

9.12.4.1 struct lidar_ans_header __attribute__((packed))

9.12.4.2 int32_t angle

9.12.4.3 uint16_t angle_q6_checkbit

信号质量

9.12.4.4 uint16_t checksum

9.12.4.5 uint8_t cmd_flag

9.12.4.6 uint8_t data

9.12.4.7 uint8_t debug_info[12]

特定版本此值才有效,无效值是0

9.12.4.8 uint16_t distance_q2

测距点角度

9.12.4.9 uint8_t enable

掉电保护状态

9.12.4.10 uint16_t error_code

错误代码

9.12.4.11 uint8_t exposure

低光功率模式

9.12.4.12 uint16_t firmware_version

固件版本号

9.12.4.13 uint8_t flag

9.12.4.14 uint32_t frequency

扫描频率

9.12.4.15 uint8_t hardware_version

硬件版本号

9.12.4.16 uint8_t index

9.12.4.17 uint8_t model

雷达型号

9.12.4.18 uint8_t nowPackageNum

9.12.4.19 uint8_t package_CT

9.12.4.20 uint16_t package_Head

9.12.4.21 uint16_t packageFirstSampleAngle

9.12.4.22 uint16_t packageLastSampleAngle

9.12.4.23 PackageNode packageSample[PackageSampleMaxLngth]

9.12.4.24 uint16_t packageSampleDistance[PackageSampleMaxLngth]

9.12.4.25 uint16_t PackageSampleDistance

9.12.4.26 uint8_t PackageSampleQuality

9.12.4.27 uint8_t rate

采样频率

9.12.4.28 uint8_t rotation

9.12.4.29 uint8_t scan_frequence

时间戳

9.12.4.30 uint8_t serialnum[16]

系列号

9.12.4.31 uint32_t size

9.12.4.32 uint64_t stamp

当前测距点距离

9.12.4.33 uint8_t state

9.12.4.34 uint8_t status

健康状态

9.12.4.35 uint32_t subType

9.12.4.36 uint8_t sync_flag

9.12.4.37 uint16_t sync_quality

9.12.4.38 uint8_t syncByte

9.12.4.39 uint8_t syncByte1

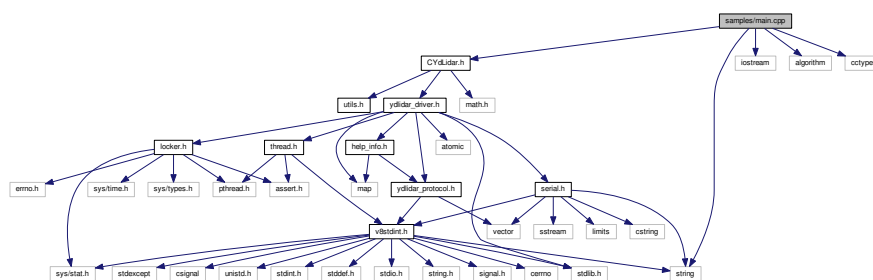
9.12.4.40 uint8_t syncByte2

9.12.4.41 uint8_t type

9.13 README.md File Reference

9.14 samples/main.cpp File Reference

```
#include "CYdLidar.h"
#include <iostream>
#include <string>
#include <algorithm>
#include <cctype>
Include dependency graph for main.cpp:
```



Functions

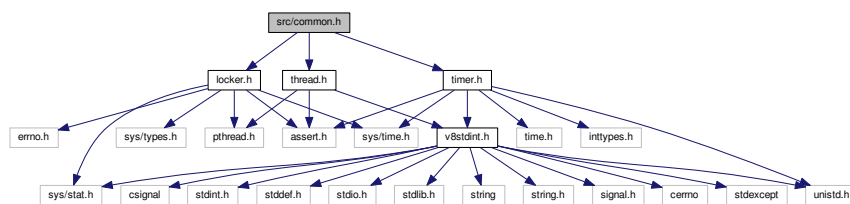
- int [main](#) (int argc, char *argv[])

9.14.1 Function Documentation

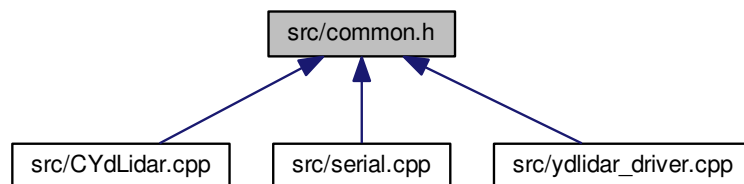
9.14.1.1 int main (int argc, char * argv[])

9.15 src/common.h File Reference

```
#include "locker.h"
#include "thread.h"
#include "timer.h"
Include dependency graph for common.h:
```



This graph shows which files directly or indirectly include this file:



Macros

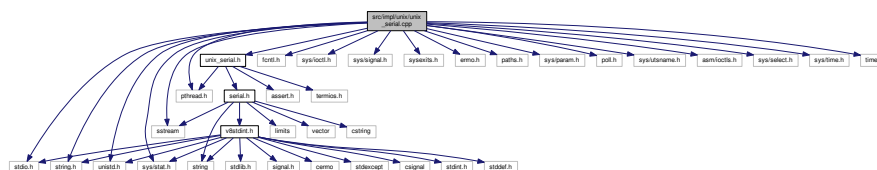
- #define [SDKVersion](#) "1.4.6"

Include dependency graph for unix.h:



9.23 src/impl/unix/unix_serial.cpp File Reference

Include dependency graph for `unix_serial.cpp`:



Classes

- struct [serial::termios2](#)

Namespaces

- [serial](#)

Macros

- #define [TIOCIHQ](#) 0x541B
- #define [SNCCS](#) 19
- #define [TCGETS2_IOR](#)('T', 0x2A, struct termios2)
- #define [TCSETS2_IOW](#)('T', 0x2B, struct termios2)
- #define [BOTHER](#) 0010000

Functions

- timespec [serial::timespec_from_ms](#) (const uint32_t millis)
- static void [serial::set_common_props](#) (termios *tio)
- static void [serial::set_databits](#) (termios *tio, [serial::bytesize_t](#) databits)
- static void [serial::set_parity](#) (termios *tio, [serial::parity_t](#) parity)
- static void [serial::set_stopbits](#) (termios *tio, [serial::stopbits_t](#) stopbits)
- static void [serial::set_flowcontrol](#) (termios *tio, [serial::flowcontrol_t](#) flowcontrol)
- static bool [serial::is_standardbaudrate](#) (unsigned long baudrate, speed_t &baud)

9.23.1 Macro Definition Documentation

9.23.1.1 #define [BOTHER](#) 0010000

9.23.1.2 #define [SNCCS](#) 19

9.23.1.3 #define [TCGETS2_IOR](#)('T', 0x2A, struct termios2)

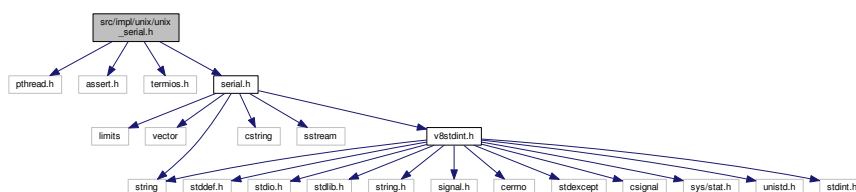
9.23.1.4 #define [TCSETS2_IOW](#)('T', 0x2B, struct termios2)

9.23.1.5 #define [TIOCIHQ](#) 0x541B

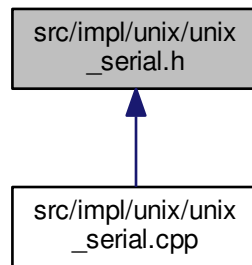
9.24 src/impl/unix/unix_serial.h File Reference

```
#include <pthread.h>
#include <assert.h>
#include <termios.h>
#include "serial.h"
```

Include dependency graph for unix_serial.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [serial::MillisecondTimer](#)
- class [serial::Serial::SerialImpl](#)

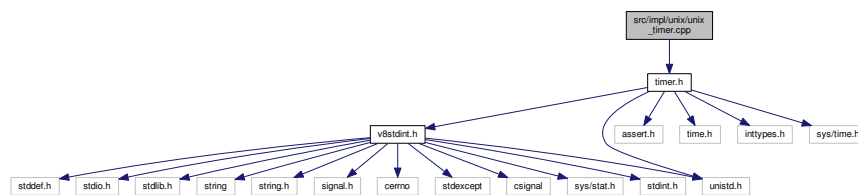
Namespaces

- [serial](#)

9.25 src/impl/unix/unix_timer.cpp File Reference

```
#include "timer.h"
```

Include dependency graph for unix_timer.cpp:



Namespaces

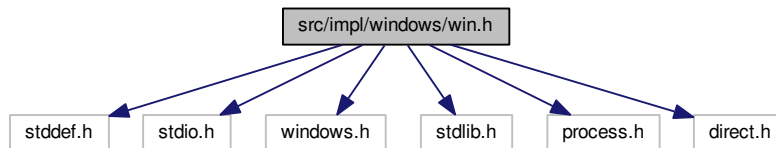
- [impl](#)

Functions

- `uint32_t impl::getHDTimer ()`
- `uint64_t impl::getCurrentTime ()`

9.26 src/impl/windows/win.h File Reference

```
#include <stddef.h>
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <process.h>
#include <direct.h>
Include dependency graph for win.h:
```



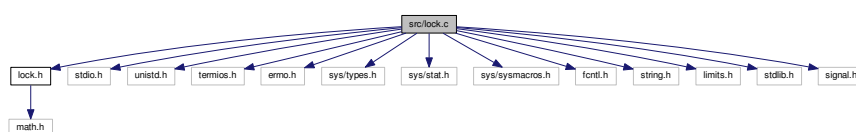
9.27 src/impl/windows/win_serial.cpp File Reference

9.28 src/impl/windows/win_serial.h File Reference

9.29 src/impl/windows/win_timer.cpp File Reference

9.30 src/lock.c File Reference

```
#include "lock.h"
#include <stdio.h>
#include <unistd.h>
#include <termios.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <fcntl.h>
#include <string.h>
#include <limits.h>
#include <stdlib.h>
#include <signal.h>
Include dependency graph for lock.c:
```



Functions

- int [fhs_lock](#) (const char *filename, int pid)
- int [uucp_lock](#) (const char *filename, int pid)
- int [check_lock_status](#) (const char *filename)
- void [fhs_unlock](#) (const char *filename, int openpid)
- void [uucp_unlock](#) (const char *filename, int openpid)
- int [check_lock_pid](#) (const char *file, int openpid)
- int [check_group_uucp](#) ()
- int [is_device_locked](#) (const char *port_filename)

9.30.1 Function Documentation

9.30.1.1 int [check_group_uucp](#) ()

9.30.1.2 int [check_lock_pid](#) (const char * *file*, int *openpid*)

9.30.1.3 int [check_lock_status](#) (const char * *filename*)

9.30.1.4 int [fhs_lock](#) (const char * *filename*, int *pid*)

9.30.1.5 void [fhs_unlock](#) (const char * *filename*, int *openpid*)

9.30.1.6 int [is_device_locked](#) (const char * *port_filename*)

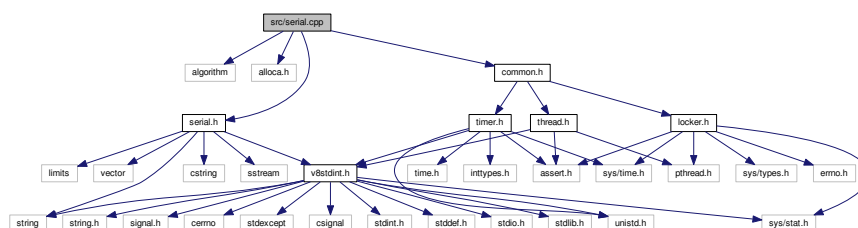
9.30.1.7 int [uucp_lock](#) (const char * *filename*, int *pid*)

9.30.1.8 void [uucp_unlock](#) (const char * *filename*, int *openpid*)

9.31 src/serial.cpp File Reference

```
#include <algorithm>
#include <alloca.h>
#include "serial.h"
#include "common.h"
```

Include dependency graph for serial.cpp:



Classes

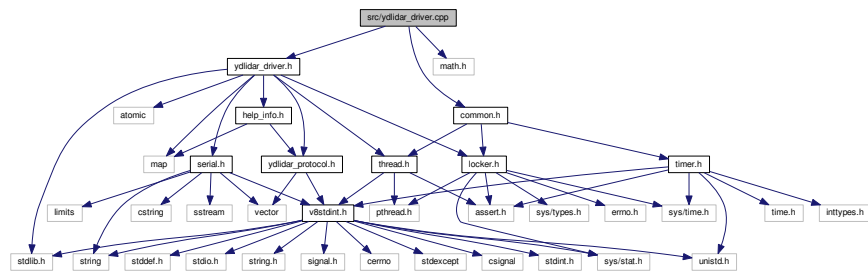
- class [serial::Serial::ScopedReadLock](#)
- class [serial::Serial::ScopedWriteLock](#)

Namespaces

- [serial](#)

9.32 src/ydlidar_driver.cpp File Reference

```
#include "ydlidar_driver.h"
#include "common.h"
#include <math.h>
Include dependency graph for ydlidar_driver.cpp:
```



Namespaces

- [ydlidar](#)

Index

- `__attribute__`
 - `v8stdint.h`, 133
 - `ydlidar_protocol.h`, 142
- `__small_endian`
 - `v8stdint.h`, 133
- `_access`
 - `v8stdint.h`, 133
- `_binded`
 - `ScopedLocker`, 65
- `_cond_cattr`
 - `Event`, 49
- `_cond_locker`
 - `Event`, 49
- `_cond_var`
 - `Event`, 49
- `_countof`
 - `ydlidar_protocol.h`, 139
- `_dataEvent`
 - `ydlidar::YDlidarDriver`, 114
- `_func`
 - `Thread`, 89
- `_handle`
 - `Thread`, 89
- `_isAutoReset`
 - `Event`, 49
- `_is_signalled`
 - `Event`, 49
- `_lock`
 - `Locker`, 56
 - `ydlidar::YDlidarDriver`, 114
- `_param`
 - `Thread`, 89
- `_serial`
 - `ydlidar::YDlidarDriver`, 114
- `_serial_lock`
 - `ydlidar::YDlidarDriver`, 114
- `_thread`
 - `ydlidar::YDlidarDriver`, 114
- `~CYdLidar`
 - `CYdLidar`, 35
- `~Event`
 - `Event`, 49
- `~Locker`
 - `Locker`, 56
- `~ScopedLocker`
 - `ScopedLocker`, 65
- `~ScopedReadLock`
 - `serial::Serial::ScopedReadLock`, 66
- `~ScopedWriteLock`
 - `serial::Serial::ScopedWriteLock`, 68
- `~Serial`
 - `serial::Serial`, 71
- `~SerialImpl`
 - `serial::Serial::SerialImpl`, 84
- `~Thread`
 - `Thread`, 88
- `~YDlidarDriver`
 - `ydlidar::YDlidarDriver`, 96
- `angle`
 - `LaserPoint`, 53
 - `offset_angle`, 60
 - `ydlidar_protocol.h`, 142
- `angle_increment`
 - `LaserConfig`, 51
- `angle_q6_checkbit`
 - `node_info`, 58
 - `ydlidar_protocol.h`, 142
- `angles`, 17
 - `find_min_max_delta`, 18
 - `from_degrees`, 18
 - `normalize_angle`, 18
 - `normalize_angle_positive`, 18
 - `shortest_angular_distance`, 18
 - `shortest_angular_distance_with_limits`, 18
 - `to_degrees`, 19
 - `two_pi_complement`, 19
- `angles.h`
 - `M_PI`, 120
- `ascendScanData`
 - `ydlidar::YDlidarDriver`, 96
- `async_size`
 - `ydlidar::YDlidarDriver`, 114
- `asyncRecvPos`
 - `ydlidar::YDlidarDriver`, 114
- `available`
 - `serial::Serial`, 71
 - `serial::Serial::SerialImpl`, 84
- `BEGIN_STATIC_CODE`
 - `timer.h`, 130
- `BOTHER`
 - `unix_serial.cpp`, 148
- `baudrate_`
 - `serial::Serial::SerialImpl`, 86
- `byte_time_ns_`
 - `serial::Serial::SerialImpl`, 86
- `bytesize_`
 - `serial::Serial::SerialImpl`, 86

- bytesize_t
 - serial, [22](#)
- c_cc
 - serial::termios2, [87](#)
- c_cflag
 - serial::termios2, [87](#)
- c_iflag
 - serial::termios2, [87](#)
- c_ispeed
 - serial::termios2, [87](#)
- c_lflag
 - serial::termios2, [87](#)
- c_line
 - serial::termios2, [87](#)
- c_oflag
 - serial::termios2, [87](#)
- c_ospeed
 - serial::termios2, [87](#)
- CLASS_THREAD
 - thread.h, [129](#)
- CT_Normal
 - ydlidar_protocol.h, [141](#)
- CT_RingStart
 - ydlidar_protocol.h, [141](#)
- CT_Tail
 - ydlidar_protocol.h, [141](#)
- CYdLidar, [30](#)
 - ~CYdLidar, [35](#)
 - CYdLidar, [35](#)
 - CalculateSampleRate, [35](#)
 - checkCOMMs, [36](#)
 - checkCalibrationAngle, [35](#)
 - checkHardware, [36](#)
 - checkLidarAbnormal, [36](#)
 - checkSampleRate, [36](#)
 - checkScanFrequency, [36](#)
 - checkStatus, [36](#)
 - defalutSampleRate, [46](#)
 - disconnecting, [36](#)
 - doProcessSimple, [36](#)
 - frequencyOffset, [46](#)
 - getAngleOffset, [37](#)
 - getDeviceHealth, [37](#)
 - getDeviceInfo, [37](#)
 - getHardwareVersion, [37](#)
 - getSerialNumber, [37](#)
 - getSoftVersion, [37](#)
 - global_nodes, [46](#)
 - handleDeviceInfoPackage, [37](#)
 - handleSingleChannelDevice, [37](#)
 - initialize, [37](#)
 - isAngleOffsetCorrected, [37](#)
 - isRangeIgnore, [38](#)
 - isRangeValid, [38](#)
 - isScanning, [46](#)
 - last_node_time, [46](#)
 - lidar_model, [46](#)
 - lidarPtr, [46](#)
 - m_AngleOffset, [46](#)
 - m_FixedSize, [46](#)
 - m_ParseSuccess, [46](#)
 - m_PointTime, [46](#)
 - m_UserSampleRate, [46](#)
 - m_isAngleOffsetCorrected, [46](#)
 - m_lidarHardVer, [46](#)
 - m_lidarSerialNum, [46](#)
 - m_lidarSoftVer, [46](#)
 - Major, [46](#)
 - MinRange, [46](#)
 - Minjor, [46](#)
 - parsePackageNode, [38](#)
 - printfVersionInfo, [38](#)
 - private, [46](#)
 - PropertyBuilderByName, [38–45](#)
 - SampleRateMap, [46](#)
 - turnOff, [45](#)
 - turnOn, [45](#)
- cacheScanData
 - ydlidar::YDlidarDriver, [97](#)
- CalculateSampleRate
 - CYdLidar, [35](#)
- check_group_uucp
 - lock.c, [151](#)
 - lock.h, [124](#)
- check_lock_pid
 - lock.c, [151](#)
 - lock.h, [124](#)
- check_lock_status
 - lock.c, [151](#)
 - lock.h, [124](#)
- checkAutoConnecting
 - ydlidar::YDlidarDriver, [97](#)
- checkCOMMs
 - CYdLidar, [36](#)
- checkCalibrationAngle
 - CYdLidar, [35](#)
- checkDeviceInfo
 - ydlidar::YDlidarDriver, [97](#)
- checkHardware
 - CYdLidar, [36](#)
- checkLidarAbnormal
 - CYdLidar, [36](#)
- checkSampleRate
 - CYdLidar, [36](#)
- checkScanFrequency
 - CYdLidar, [36](#)
- checkStatus
 - CYdLidar, [36](#)
- Checksum
 - ydlidar::YDlidarDriver, [114](#)
- checksum
 - node_package, [59](#)
 - node_packages, [60](#)
 - ydlidar_protocol.h, [142](#)
- ChecksumCal
 - ydlidar::YDlidarDriver, [114](#)

- ChecksumResult
 - ydlidar::YDLidarDriver, 115
- checkTransDelay
 - ydlidar::YDLidarDriver, 97
- clearDTR
 - ydlidar::YDLidarDriver, 97
- close
 - serial::Serial::SerialImpl, 84
- closePort
 - serial::Serial, 71
- cmd_flag
 - cmd_packet, 29
 - ydlidar_protocol.h, 142
- cmd_packet, 29
 - cmd_flag, 29
 - data, 29
 - size, 29
 - syncByte, 29
- common.h
 - SDKVersion, 146
- config
 - LaserScan, 54
- connect
 - ydlidar::YDLidarDriver, 97
- ConvertLidarToUserSmaple
 - ydlidar, 25
- ConvertUserToLidarSmaple
 - ydlidar, 25
- createThread
 - Thread, 88
 - ydlidar::YDLidarDriver, 99
- createThreadAux
 - Thread, 88
- CT
 - ydlidar_protocol.h, 141
- DEFAULT_HEART_BEAT
 - ydlidar::YDLidarDriver, 96
- DEFAULT_TIMEOUT_COUNT
 - ydlidar::YDLidarDriver, 96
- DEFAULT_TIMEOUT
 - ydlidar::YDLidarDriver, 96
- DEVICE_DRIVER_TYPE_SERIALPORT
 - v8stdint.h, 133
- DEVICE_DRIVER_TYPE_TCP
 - v8stdint.h, 133
- data
 - cmd_packet, 29
 - ydlidar_protocol.h, 142
- debug_info
 - node_info, 58
 - ydlidar_protocol.h, 142
- defalutSampleRate
 - CYdLidar, 46
- delay
 - timer.h, 130
- description
 - serial::PortInfo, 62
- device_health, 47
 - error_code, 47
 - status, 47
- device_id
 - serial::PortInfo, 62
- device_info, 47
 - firmware_version, 47
 - hardware_version, 47
 - model, 48
 - serialnum, 48
- disableDataGrabbing
 - ydlidar::YDLidarDriver, 99
- disconnect
 - ydlidar::YDLidarDriver, 99
- disconnecting
 - CYdLidar, 36
- distance_q2
 - node_info, 58
 - ydlidar_protocol.h, 142
- doProcessSimple
 - CYdLidar, 36
- END_STATIC_CODE
 - timer.h, 130
- EVENT_FAILED
 - Event, 49
- EVENT_OK
 - Event, 49
- EVENT_TIMEOUT
 - Event, 49
- eightbits
 - serial, 22
- enable
 - scan_heart_beat, 64
 - ydlidar_protocol.h, 142
- error_code
 - device_health, 47
 - ydlidar_protocol.h, 142
- Event, 48
 - _cond_cattr, 49
 - _cond_locker, 49
 - _cond_var, 49
 - _isAutoReset, 49
 - _is_signalled, 49
 - ~Event, 49
 - EVENT_FAILED, 49
 - EVENT_OK, 49
 - EVENT_TIMEOUT, 49
 - Event, 49
 - release, 49
 - set, 49
 - wait, 49
- expiry
 - serial::MillisecondTimer, 57
- exposure
 - scan_exposure, 63
 - ydlidar_protocol.h, 142
- fd_
 - serial::Serial::SerialImpl, 86

- fhs_lock
 - lock.c, [151](#)
 - lock.h, [125](#)
- fhs_unlock
 - lock.c, [151](#)
 - lock.h, [125](#)
- fileExists
 - ydlidar, [25](#)
- find_min_max_delta
 - angles, [18](#)
- firmware_version
 - device_info, [47](#)
 - ydlidar_protocol.h, [142](#)
- FirstSampleAngle
 - ydlidar::YDLidarDriver, [115](#)
- fivebits
 - serial, [22](#)
- flag
 - scan_points, [64](#)
 - ydlidar_protocol.h, [142](#)
- flowcontrol_
 - serial::Serial::SerialImpl, [86](#)
- flowcontrol_hardware
 - serial, [22](#)
- flowcontrol_none
 - serial, [22](#)
- flowcontrol_software
 - serial, [22](#)
- flowcontrol_t
 - serial, [22](#)
- flush
 - serial::Serial, [71](#)
 - serial::Serial::SerialImpl, [84](#)
- flushInput
 - serial::Serial, [71](#)
 - serial::Serial::SerialImpl, [84](#)
- flushOutput
 - serial::Serial, [71](#)
 - serial::Serial::SerialImpl, [84](#)
- flushSerial
 - ydlidar::YDLidarDriver, [99](#)
- forceUnlock
 - ScopedLocker, [65](#)
- frequency
 - scan_frequency, [63](#)
 - ydlidar_protocol.h, [143](#)
- frequencyOffset
 - CYdLidar, [46](#)
- from_degrees
 - angles, [18](#)
- function_state, [49](#)
 - state, [50](#)
- g_signal_status
 - v8stdint.h, [134](#)
- GLASSNOISEINTENSITY
 - ydlidar_protocol.h, [139](#)
- get_device_health_success
 - ydlidar::YDLidarDriver, [115](#)
- getAngleOffset
 - CYdLidar, [37](#)
- getBaudrate
 - serial::Serial, [71](#)
 - serial::Serial::SerialImpl, [84](#)
- getByteTime
 - serial::Serial, [72](#)
 - serial::Serial::SerialImpl, [84](#)
- getBytesize
 - serial::Serial, [72](#)
 - serial::Serial::SerialImpl, [84](#)
- getCTS
 - serial::Serial, [72](#)
 - serial::Serial::SerialImpl, [84](#)
- getCD
 - serial::Serial, [72](#)
 - serial::Serial::SerialImpl, [84](#)
- getCurrentTime
 - impl, [20](#)
- getDSR
 - serial::Serial, [72](#)
 - serial::Serial::SerialImpl, [84](#)
- getData
 - ydlidar::YDLidarDriver, [99](#)
- getDeviceHealth
 - CYdLidar, [37](#)
- getDeviceInfo
 - CYdLidar, [37](#)
 - ydlidar::YDLidarDriver, [100](#)
- getFlowcontrol
 - serial::Serial, [72](#)
 - serial::Serial::SerialImpl, [85](#)
- getHDTimer
 - impl, [20](#)
- getHandle
 - Thread, [88](#)
- getHardwareVersion
 - CYdLidar, [37](#)
- getHealth
 - ydlidar::YDLidarDriver, [100](#)
- getLockHandle
 - Locker, [56](#)
- getParam
 - Thread, [89](#)
- getParity
 - serial::Serial, [73](#)
 - serial::Serial::SerialImpl, [85](#)
- getPort
 - serial::Serial, [73](#)
 - serial::Serial::SerialImpl, [85](#)
- getRI
 - serial::Serial, [73](#)
 - serial::Serial::SerialImpl, [85](#)
- getSDKVersion
 - ydlidar::YDLidarDriver, [101](#)
- getSamplingRate

- ydlidar::YDLidarDriver, 101
- getScanFrequency
 - ydlidar::YDLidarDriver, 101
- getSerialNumber
 - CYdLidar, 37
- getSoftVersion
 - CYdLidar, 37
- getStopbits
 - serial::Serial, 73
 - serial::Serial::SerialImpl, 85
- getTermios
 - serial::Serial::SerialImpl, 85
- getTime
 - timer.h, 130
- getTimeout
 - serial::Serial, 73
 - serial::Serial::SerialImpl, 85
- getZeroOffsetAngle
 - ydlidar::YDLidarDriver, 102
- getms
 - timer.h, 130
- global_nodes
 - CYdLidar, 46
- globalRecvBuffer
 - ydlidar::YDLidarDriver, 115
- grabScanData
 - ydlidar::YDLidarDriver, 102
- handleDeviceInfoPackage
 - CYdLidar, 37
- handleSingleChannelDevice
 - CYdLidar, 37
- hardware_id
 - serial::PortInfo, 62
- hardware_version
 - device_info, 47
 - ydlidar_protocol.h, 143
- has_device_header
 - ydlidar::YDLidarDriver, 115
- has_package_error
 - ydlidar::YDLidarDriver, 115
- hasIntensity
 - ydlidar, 25
- hasSampleRate
 - ydlidar, 25
- hasScanFrequencyCtrl
 - ydlidar, 26
- hasZeroAngle
 - ydlidar, 26
- header_
 - ydlidar::YDLidarDriver, 115
- headerBuffer
 - ydlidar::YDLidarDriver, 115
- health_
 - ydlidar::YDLidarDriver, 115
- healthBuffer
 - ydlidar::YDLidarDriver, 115
- INTENSITY_NORMAL_PACKAGE_SIZE
 - ydlidar_protocol.h, 139
- INVALID_TIMESTAMP
 - v8stdint.h, 133
- IS_FAIL
 - v8stdint.h, 133
- IS_OK
 - v8stdint.h, 133
- IS_TIMEOUT
 - v8stdint.h, 133
- impl, 19
 - getCurrentTime, 20
 - getHDTimer, 20
- include/CYdLidar.h, 120
- include/angles.h, 119
- include/help_info.h, 121
- include/lock.h, 123
- include/locker.h, 125
- include/serial.h, 126
- include/thread.h, 128
- include/timer.h, 129
- include/utils.h, 130
- include/v8stdint.h, 131
- include/ydlidar_driver.h, 134
- include/ydlidar_protocol.h, 135
- index
 - node_info, 58
 - ydlidar_protocol.h, 143
- info_
 - ydlidar::YDLidarDriver, 115
- infoBuffer
 - ydlidar::YDLidarDriver, 115
- init
 - Locker, 56
 - ydlidar, 26
- initialize
 - CYdLidar, 37
- intensity
 - LaserPoint, 53
- inter_byte_timeout
 - serial::Timeout, 90
- IntervalSampleAngle
 - ydlidar::YDLidarDriver, 115
- IntervalSampleAngle_LastPackage
 - ydlidar::YDLidarDriver, 115
- is_device_locked
 - lock.c, 151
 - lock.h, 125
- is_open_
 - serial::Serial::SerialImpl, 86
- is_standardbaudrate
 - serial, 23
- isAngleOffsetCorrected
 - CYdLidar, 37
- isAutoReconnect
 - ydlidar::YDLidarDriver, 115
- isAutoconnting
 - ydlidar::YDLidarDriver, 115
- isConnected

- ydlidar::YDLidarDriver, [115](#)
- isOctaveLidar
 - ydlidar, [26](#)
- isOldVersionTOFLidar
 - ydlidar, [27](#)
- isOpen
 - serial::Serial, [74](#)
 - serial::Serial::SerialImpl, [85](#)
- isRangeIgnore
 - CYdLidar, [38](#)
- isRangeValid
 - CYdLidar, [38](#)
- isScanning
 - CYdLidar, [46](#)
 - ydlidar::YDLidarDriver, [115](#)
- isSerialNumbValid
 - ydlidar, [27](#)
- isSupportLidar
 - ydlidar, [27](#)
- isSupportMotorCtrl
 - ydlidar, [27](#)
- isSupportMotorDtrCtrl
 - ydlidar::YDLidarDriver, [116](#)
- isSupportScanFrequency
 - ydlidar, [27](#)
- isTOFLidar
 - ydlidar, [27](#)
- isValidSampleRate
 - ydlidar, [28](#)
- isValidValue
 - ydlidar, [28](#)
- isVersionValid
 - ydlidar, [28](#)
- isconnected
 - ydlidar::YDLidarDriver, [103](#)
- isscanning
 - ydlidar::YDLidarDriver, [103](#)
- join
 - Thread, [89](#)
- LIDAR_ANS_SYNC_BYTE1
 - ydlidar_protocol.h, [139](#)
- LIDAR_ANS_SYNC_BYTE2
 - ydlidar_protocol.h, [139](#)
- LIDAR_ANS_TYPE_DEVHEALTH
 - ydlidar_protocol.h, [139](#)
- LIDAR_ANS_TYPE_DEVINFO
 - ydlidar_protocol.h, [139](#)
- LIDAR_ANS_TYPE_MEASUREMENT
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_ADD_EXPOSURE
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_DIS_EXPOSURE
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_DISABLE_CONST_FREQ
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_DISABLE_LOW_POWER
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_ENABLE_CONST_FREQ
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_ENABLE_LOW_POWER
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_FORCE_SCAN
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_FORCE_STOP
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_GET_AIMSPEED
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_GET_DEVICE_HEALTH
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_GET_DEVICE_INFO
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_GET_EAI
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_GET_OFFSET_ANGLE
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_GET_SAMPLING_RATE
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_RESET
 - ydlidar_protocol.h, [139](#)
- LIDAR_CMD_RUN_INVERSION
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_RUN_POSITIVE
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SAVE_SET_EXPOSURE
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SCAN
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SET_AIMSPEED_ADDMIC
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SET_AIMSPEED_ADD
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SET_AIMSPEED_DISMIC
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SET_AIMSPEED_DIS
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SET_LOW_EXPOSURE
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SET_SAMPLING_RATE
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_STATE_MODEL_MOTOR
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_STOP
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMD_SYNC_BYTE
 - ydlidar_protocol.h, [140](#)
- LIDAR_CMDFLAG_HAS_PAYLOAD
 - ydlidar_protocol.h, [140](#)
- LIDAR_RESP_MEASUREMENT_ANGLE_SAMPLE_↔
 - SHIFT
 - ydlidar_protocol.h, [140](#)
- LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT
 - ydlidar_protocol.h, [140](#)
- LIDAR_RESP_MEASUREMENT_CHECKBIT
 - ydlidar_protocol.h, [140](#)
- LIDAR_RESP_MEASUREMENT_DISTANCE_SHIFT

- ydlidar_protocol.h, [140](#)
- LIDAR_RESP_MEASUREMENT_QUALITY_SHIFT
 - ydlidar_protocol.h, [140](#)
- LIDAR_RESP_MEASUREMENT_SYNCBIT
 - ydlidar_protocol.h, [140](#)
- LIDAR_STATUS_ERROR
 - ydlidar_protocol.h, [140](#)
- LIDAR_STATUS_OK
 - ydlidar_protocol.h, [140](#)
- LIDAR_STATUS_WARNING
 - ydlidar_protocol.h, [140](#)
- LOCK_FAILED
 - Locker, [56](#)
- LOCK_OK
 - Locker, [56](#)
- LOCK_STATUS
 - Locker, [56](#)
- LOCK_TIMEOUT
 - Locker, [56](#)
- LOCK
 - lock.h, [124](#)
- LaserConfig, [50](#)
 - angle_increment, [51](#)
 - max_angle, [51](#)
 - max_range, [51](#)
 - min_angle, [51](#)
 - min_range, [51](#)
 - operator=, [51](#)
 - scan_time, [51](#)
 - time_increment, [51](#)
- LaserDebug, [51](#)
 - MaxDebugIndex, [52](#)
 - W1F6GNoise_W1F5SNoise_W1F4MotorCtl_W4←
F0SnYear, [52](#)
 - W2F5Output2K4K5K_W5F0Date, [52](#)
 - W3F4BoradHardVer_W4F0Moth, [52](#)
 - W3F4CusMajor_W4F0CusMinor, [52](#)
 - W3F4HardwareVer_W4F0FirewareMajor, [52](#)
 - W4F3Model_W3F0DebugInfTranVer, [52](#)
 - W7F0SnNumH, [52](#)
 - W7F0SnNumL, [52](#)
- LaserPoint, [52](#)
 - angle, [53](#)
 - intensity, [53](#)
 - operator=, [53](#)
 - range, [53](#)
- LaserScan, [53](#)
 - config, [54](#)
 - operator=, [54](#)
 - points, [54](#)
 - stamp, [54](#)
- last_device_byte
 - ydlidar::YDlidarDriver, [116](#)
- last_node_time
 - CYdLidar, [46](#)
- LastSampleAngle
 - ydlidar::YDlidarDriver, [116](#)
- LastSampleAngleCal
 - ydlidar::YDlidarDriver, [116](#)
- lfs_lock
 - lock.h, [125](#)
- lfs_unlock
 - lock.h, [125](#)
- lib_lock_dev_lock
 - lock.h, [125](#)
- lib_lock_dev_unlock
 - lock.h, [125](#)
- lidar_ans_header, [54](#)
 - size, [55](#)
 - subType, [55](#)
 - syncByte1, [55](#)
 - syncByte2, [55](#)
 - type, [55](#)
- lidar_model
 - CYdLidar, [46](#)
- lidarModelDefaultSampleRate
 - ydlidar, [28](#)
- lidarModelToString
 - ydlidar, [28](#)
- lidarPortList
 - ydlidar::YDlidarDriver, [103](#)
- lidarPtr
 - CYdLidar, [46](#)
- LidarTypeID
 - ydlidar_protocol.h, [141](#)
- list_ports
 - serial, [23](#)
- lock
 - Locker, [56](#)
- lock.c
 - check_group_uucp, [151](#)
 - check_lock_pid, [151](#)
 - check_lock_status, [151](#)
 - fhs_lock, [151](#)
 - fhs_unlock, [151](#)
 - is_device_locked, [151](#)
 - uucp_lock, [151](#)
 - uucp_unlock, [151](#)
- lock.h
 - check_group_uucp, [124](#)
 - check_lock_pid, [124](#)
 - check_lock_status, [124](#)
 - fhs_lock, [125](#)
 - fhs_unlock, [125](#)
 - is_device_locked, [125](#)
 - LOCK, [124](#)
 - lfs_lock, [125](#)
 - lfs_unlock, [125](#)
 - lib_lock_dev_lock, [125](#)
 - lib_lock_dev_unlock, [125](#)
 - lock_device, [125](#)
 - UNLOCK, [124](#)
 - unlock_device, [125](#)
 - uucp_lock, [125](#)
 - uucp_unlock, [125](#)
- lock_device

- lock.h, 125
- Locker, 55
 - _lock, 56
 - ~Locker, 56
 - getLockHandle, 56
 - init, 56
 - LOCK_FAILED, 56
 - LOCK_OK, 56
 - LOCK_STATUS, 56
 - LOCK_TIMEOUT, 56
 - lock, 56
 - Locker, 56
 - release, 56
 - unlock, 56
- m_AngleOffset
 - CYdLidar, 46
- m_FixedSize
 - CYdLidar, 46
- m_ParseSuccess
 - CYdLidar, 46
- M_PI
 - angles.h, 120
 - ydlidar_protocol.h, 141
- m_PointTime
 - CYdLidar, 46
- m_UserSampleRate
 - CYdLidar, 46
- m_baudrate
 - ydlidar::YDLidarDriver, 116
- m_intensities
 - ydlidar::YDLidarDriver, 116
- m_isAngleOffsetCorrected
 - CYdLidar, 46
- m_lidarHardVer
 - CYdLidar, 46
- m_lidarSerialNum
 - CYdLidar, 46
- m_lidarSoftVer
 - CYdLidar, 46
- m_sampling_rate
 - ydlidar::YDLidarDriver, 116
- MAX_SCAN_NODES
 - ydlidar::YDLidarDriver, 96
- main
 - main.cpp, 145
- main.cpp
 - main, 145
- Major
 - CYdLidar, 46
- max
 - serial::Timeout, 90
- max_angle
 - LaserConfig, 51
- max_range
 - LaserConfig, 51
- MaxDebugIndex
 - LaserDebug, 52
- MillisecondTimer
 - serial::MillisecondTimer, 57
- min_angle
 - LaserConfig, 51
- min_range
 - LaserConfig, 51
- MinRange
 - CYdLidar, 46
- Minjor
 - CYdLidar, 46
- model
 - device_info, 48
 - ydlidar::YDLidarDriver, 116
 - ydlidar_protocol.h, 143
- NORMAL_PACKAGE_SIZE
 - ydlidar_protocol.h, 141
- Node_Default_Quality
 - ydlidar_protocol.h, 141
- Node_NotSync
 - ydlidar_protocol.h, 141
- Node_Sync
 - ydlidar_protocol.h, 141
- node_info, 57
 - angle_q6_checkbit, 58
 - debug_info, 58
 - distance_q2, 58
 - index, 58
 - scan_frequence, 58
 - stamp, 58
 - sync_flag, 58
 - sync_quality, 58
- node_package, 58
 - checksum, 59
 - nowPackageNum, 59
 - package_CT, 59
 - package_Head, 59
 - packageFirstSampleAngle, 59
 - packageLastSampleAngle, 59
 - packageSample, 59
- node_packages, 59
 - checksum, 60
 - nowPackageNum, 60
 - package_CT, 60
 - package_Head, 60
 - packageFirstSampleAngle, 60
 - packageLastSampleAngle, 60
 - packageSampleDistance, 60
- normalize_angle
 - angles, 18
- normalize_angle_positive
 - angles, 18
- nowPackageNum
 - node_package, 59
 - node_packages, 60
 - ydlidar_protocol.h, 143
- offset_angle, 60
 - angle, 60
- ok

- ydlidar, 28
- old_signal_handler
 - v8stdint.h, 134
- open
 - serial::Serial, 74
 - serial::Serial::SerialImpl, 85
- operator=
 - LaserConfig, 51
 - LaserPoint, 53
 - LaserScan, 54
 - serial::Serial, 74
 - serial::Serial::ScopedReadLock, 66
 - serial::Serial::ScopedWriteLock, 68
- operator==
 - Thread, 89
- package
 - ydlidar::YDlidarDriver, 116
- package_CT
 - node_package, 59
 - node_packages, 60
 - ydlidar_protocol.h, 143
- package_Head
 - node_package, 59
 - node_packages, 60
 - ydlidar_protocol.h, 143
- package_Sample_Index
 - ydlidar::YDlidarDriver, 116
- package_index
 - ydlidar::YDlidarDriver, 116
- packageFirstSampleAngle
 - node_package, 59
 - node_packages, 60
 - ydlidar_protocol.h, 143
- packageLastSampleAngle
 - node_package, 59
 - node_packages, 60
 - ydlidar_protocol.h, 143
- PackageNode, 60
 - PakageSampleDistance, 61
 - PakageSampleQuality, 61
- PackagePaidBytes
 - ydlidar_protocol.h, 141
- packageSample
 - node_package, 59
 - ydlidar_protocol.h, 143
- PackageSampleBytes
 - ydlidar::YDlidarDriver, 117
- packageSampleDistance
 - node_packages, 60
 - ydlidar_protocol.h, 143
- PackageSampleMaxLngth
 - ydlidar_protocol.h, 141
- packages
 - ydlidar::YDlidarDriver, 116
- PakageSampleDistance
 - PackageNode, 61
 - ydlidar_protocol.h, 143
- PakageSampleQuality
 - PackageNode, 61
 - ydlidar_protocol.h, 143
- parity_
 - serial::Serial::SerialImpl, 86
- parity_even
 - serial, 22
- parity_mark
 - serial, 22
- parity_none
 - serial, 22
- parity_odd
 - serial, 22
- parity_space
 - serial, 22
- parity_t
 - serial, 22
- ParseLaserDebugInfo
 - ydlidar, 28
- parsePackageNode
 - CYdLidar, 38
- PH
 - ydlidar_protocol.h, 141
- pid
 - serial::Serial::SerialImpl, 86
- pimpl_
 - serial::Serial, 82
 - serial::Serial::ScopedReadLock, 67
 - serial::Serial::ScopedWriteLock, 68
- points
 - LaserScan, 54
- port
 - serial::PortInfo, 62
- port_
 - serial::Serial::SerialImpl, 86
- printfVersionInfo
 - CYdLidar, 38
- private
 - CYdLidar, 46
- PropertyBuilderByName
 - CYdLidar, 38–45
 - ydlidar::YDlidarDriver, 104, 105
 - ydlidar_protocol.h, 141
- README.md, 144
- RESULT_FAIL
 - v8stdint.h, 133
- RESULT_OK
 - v8stdint.h, 133
- RESULT_TIMEOUT
 - v8stdint.h, 133
- range
 - LaserPoint, 53
- rate
 - sampling_rate, 62
 - ydlidar_protocol.h, 143
- read
 - serial::Serial, 74, 75
 - serial::Serial::SerialImpl, 85
- read_

- serial::Serial, 76
- read_mutex
 - serial::Serial::SerialImpl, 87
- read_timeout_constant
 - serial::Timeout, 90
- read_timeout_multiplier
 - serial::Timeout, 90
- readData
 - serial::Serial, 76
- readLock
 - serial::Serial::SerialImpl, 85
- readUnlock
 - serial::Serial::SerialImpl, 85
- readline
 - serial::Serial, 76
- readlines
 - serial::Serial, 77
- release
 - Event, 49
 - Locker, 56
- remaining
 - serial::MillisecondTimer, 57
- reset
 - ydlidar::YDlidarDriver, 105
- result_t
 - v8stdint.h, 133
- retryCount
 - ydlidar::YDlidarDriver, 117
- rotation
 - scan_rotation, 65
 - ydlidar_protocol.h, 143
- rtscts_
 - serial::Serial::SerialImpl, 87
- SDKVersion
 - common.h, 146
- SNCCS
 - unix_serial.cpp, 148
- SUNNOISEINTENSITY
 - ydlidar_protocol.h, 141
- sample_rate
 - ydlidar::YDlidarDriver, 117
- SampleNumIAndCTCal
 - ydlidar::YDlidarDriver, 117
- SampleRateMap
 - CYdLidar, 46
- samples/main.cpp, 144
- sampling_rate, 62
 - rate, 62
- scan_exposure, 63
 - exposure, 63
- scan_frequence
 - node_info, 58
 - ydlidar::YDlidarDriver, 117
 - ydlidar_protocol.h, 143
- scan_frequency, 63
 - frequency, 63
- scan_heart_beat, 63
 - enable, 64
- scan_node_buf
 - ydlidar::YDlidarDriver, 117
- scan_node_count
 - ydlidar::YDlidarDriver, 117
- scan_points, 64
 - flag, 64
- scan_rotation, 64
 - rotation, 65
- scan_time
 - LaserConfig, 51
- ScopedLocker, 65
 - _binded, 65
 - ~ScopedLocker, 65
 - forceUnlock, 65
 - ScopedLocker, 65
- ScopedReadLock
 - serial::Serial::ScopedReadLock, 66
- ScopedWriteLock
 - serial::Serial::ScopedWriteLock, 68
- sendBreak
 - serial::Serial, 77
 - serial::Serial::SerialImpl, 85
- sendCommand
 - ydlidar::YDlidarDriver, 105
- sendData
 - ydlidar::YDlidarDriver, 106
- Serial
 - serial::Serial, 70, 71
- serial, 20
 - bytesize_t, 22
 - eightbits, 22
 - fivebits, 22
 - flowcontrol_hardware, 22
 - flowcontrol_none, 22
 - flowcontrol_software, 22
 - flowcontrol_t, 22
 - is_standardbaudrate, 23
 - list_ports, 23
 - parity_even, 22
 - parity_mark, 22
 - parity_none, 22
 - parity_odd, 22
 - parity_space, 22
 - parity_t, 22
 - set_common_props, 23
 - set_databits, 23
 - set_flowcontrol, 23
 - set_parity, 23
 - set_stopbits, 23
 - sevenbits, 22
 - sixbits, 22
 - stopbits_one, 23
 - stopbits_one_point_five, 23
 - stopbits_t, 22
 - stopbits_two, 23
 - timespec_from_ms, 23
- serial::MillisecondTimer, 56
 - expiry, 57

- MillisecondTimer, 57
- remaining, 57
- timespec_now, 57
- serial::PortInfo, 61
 - description, 62
 - device_id, 62
 - hardware_id, 62
 - port, 62
- serial::Serial, 68
 - ~Serial, 71
 - available, 71
 - closePort, 71
 - flush, 71
 - flushInput, 71
 - flushOutput, 71
 - getBaudrate, 71
 - getByteTime, 72
 - getBytesize, 72
 - getCTS, 72
 - getCD, 72
 - getDSR, 72
 - getFlowcontrol, 72
 - getParity, 73
 - getPort, 73
 - getRI, 73
 - getStopbits, 73
 - getTimeout, 73
 - isOpen, 74
 - open, 74
 - operator=, 74
 - pimpl_, 82
 - read, 74, 75
 - read_, 76
 - readData, 76
 - readline, 76
 - readlines, 77
 - sendBreak, 77
 - Serial, 70, 71
 - setBaudrate, 77
 - setBreak, 77
 - setBytesize, 77
 - setDTR, 78
 - setFlowcontrol, 78
 - setParity, 78
 - setPort, 78
 - setRTS, 78
 - setStopbits, 79
 - setTimeout, 79, 80
 - waitByteTimes, 80
 - waitForChange, 80
 - waitReadable, 80
 - waitfordata, 80
 - write, 80, 81
 - write_, 81
 - writeData, 81
- serial::Serial::ScopedReadLock, 66
 - ~ScopedReadLock, 66
 - operator=, 66
- pimpl_, 67
- ScopedReadLock, 66
- serial::Serial::ScopedWriteLock, 67
 - ~ScopedWriteLock, 68
 - operator=, 68
 - pimpl_, 68
 - ScopedWriteLock, 68
- serial::Serial::SerialImpl, 82
 - ~SerialImpl, 84
 - available, 84
 - baudrate_, 86
 - byte_time_ns_, 86
 - bytesize_, 86
 - close, 84
 - fd_, 86
 - flowcontrol_, 86
 - flush, 84
 - flushInput, 84
 - flushOutput, 84
 - getBaudrate, 84
 - getByteTime, 84
 - getBytesize, 84
 - getCTS, 84
 - getCD, 84
 - getDSR, 84
 - getFlowcontrol, 85
 - getParity, 85
 - getPort, 85
 - getRI, 85
 - getStopbits, 85
 - getTermios, 85
 - getTimeout, 85
 - is_open_, 86
 - isOpen, 85
 - open, 85
 - parity_, 86
 - pid, 86
 - port_, 86
 - read, 85
 - read_mutex, 87
 - readLock, 85
 - readUnlock, 85
 - rtscts_, 87
 - sendBreak, 85
 - SerialImpl, 84
 - setBaudrate, 85
 - setBreak, 85
 - setBytesize, 85
 - setCustomBaudRate, 85
 - setDTR, 85
 - setFlowcontrol, 85
 - setParity, 85
 - setPort, 85
 - setRTS, 85
 - setStandardBaudRate, 85
 - setStopbits, 86
 - setTermios, 86
 - setTimeout, 86

- stopbits_, 87
- timeout_, 87
- waitByteTimes, 86
- waitForChange, 86
- waitReadable, 86
- waitfordata, 86
- write, 86
- write_mutex, 87
- writeLock, 86
- writeUnlock, 86
- xonxoff_, 87
- serial::Timeout, 89
 - inter_byte_timeout, 90
 - max, 90
 - read_timeout_constant, 90
 - read_timeout_multiplier, 90
 - simpleTimeout, 90
 - Timeout, 90
 - write_timeout_constant, 90
 - write_timeout_multiplier, 90
- serial::termios2, 87
 - c_cc, 87
 - c_cflag, 87
 - c_iflag, 87
 - c_ispeed, 87
 - c_lflag, 87
 - c_line, 87
 - c_oflag, 87
 - c_ospeed, 87
- serial_port
 - ydlidar::YDlidarDriver, 117
- SerialImpl
 - serial::Serial::SerialImpl, 84
- serialnum
 - device_info, 48
 - ydlidar_protocol.h, 143
- set
 - Event, 49
- set_common_props
 - serial, 23
- set_databits
 - serial, 23
- set_flowcontrol
 - serial, 23
- set_parity
 - serial, 23
- set_signal_handler
 - v8stdint.h, 134
- set_stopbits
 - serial, 23
- setAutoReconnect
 - ydlidar::YDlidarDriver, 106
- setBaudrate
 - serial::Serial, 77
 - serial::Serial::SerialImpl, 85
- setBreak
 - serial::Serial, 77
 - serial::Serial::SerialImpl, 85
- setBytesize
 - serial::Serial, 77
 - serial::Serial::SerialImpl, 85
- setCustomBaudRate
 - serial::Serial::SerialImpl, 85
- setDTR
 - serial::Serial, 78
 - serial::Serial::SerialImpl, 85
 - ydlidar::YDlidarDriver, 107
- setFlowcontrol
 - serial::Serial, 78
 - serial::Serial::SerialImpl, 85
- setIntensities
 - ydlidar::YDlidarDriver, 107
- setParity
 - serial::Serial, 78
 - serial::Serial::SerialImpl, 85
- setPort
 - serial::Serial, 78
 - serial::Serial::SerialImpl, 85
- setRTS
 - serial::Serial, 78
 - serial::Serial::SerialImpl, 85
- setSamplingRate
 - ydlidar::YDlidarDriver, 107
- setScanFrequencyAdd
 - ydlidar::YDlidarDriver, 107
- setScanFrequencyAddMic
 - ydlidar::YDlidarDriver, 108
- setScanFrequencyDis
 - ydlidar::YDlidarDriver, 108
- setScanFrequencyDisMic
 - ydlidar::YDlidarDriver, 109
- setStandardBaudRate
 - serial::Serial::SerialImpl, 85
- setStopbits
 - serial::Serial, 79
 - serial::Serial::SerialImpl, 86
- setTermios
 - serial::Serial::SerialImpl, 86
- setTimeout
 - serial::Serial, 79, 80
 - serial::Serial::SerialImpl, 86
- sevenbits
 - serial, 22
- shortest_angular_distance
 - angles, 18
- shortest_angular_distance_with_limits
 - angles, 18
- shutdownNow
 - ydlidar, 28
- signal_handler
 - v8stdint.h, 134
- signal_handler_t
 - v8stdint.h, 133
- simpleTimeout
 - serial::Timeout, 90
- sixbits

- serial, [22](#)
- size
 - cmd_packet, [29](#)
 - lidar_ans_header, [55](#)
 - ydlidar_protocol.h, [144](#)
- src/CYdLidar.cpp, [146](#)
- src/common.h, [145](#)
- src/impl/list_ports/list_ports_linux.cpp, [146](#)
- src/impl/list_ports/list_ports_osx.cpp, [146](#)
- src/impl/list_ports/list_ports_win.cpp, [146](#)
- src/impl/unix/list_ports_linux.cpp, [146](#)
- src/impl/unix/unix.h, [146](#)
- src/impl/unix/unix_serial.cpp, [147](#)
- src/impl/unix/unix_serial.h, [148](#)
- src/impl/unix/unix_timer.cpp, [149](#)
- src/impl/windows/list_ports_win.cpp, [146](#)
- src/impl/windows/win.h, [150](#)
- src/impl/windows/win_serial.cpp, [150](#)
- src/impl/windows/win_serial.h, [150](#)
- src/impl/windows/win_timer.cpp, [150](#)
- src/lock.c, [150](#)
- src/serial.cpp, [151](#)
- src/ydlidar_driver.cpp, [152](#)
- stamp
 - LaserScan, [54](#)
 - node_info, [58](#)
 - ydlidar_protocol.h, [144](#)
- startAutoScan
 - ydlidar::YDlidarDriver, [109](#)
- startMotor
 - ydlidar::YDlidarDriver, [110](#)
- startScan
 - ydlidar::YDlidarDriver, [110](#)
- state
 - function_state, [50](#)
 - ydlidar_protocol.h, [144](#)
- status
 - device_health, [47](#)
 - ydlidar_protocol.h, [144](#)
- stop
 - ydlidar::YDlidarDriver, [111](#)
- stopMotor
 - ydlidar::YDlidarDriver, [111](#)
- stopScan
 - ydlidar::YDlidarDriver, [111](#)
- stopbits_
 - serial::Serial::SerialImpl, [87](#)
- stopbits_one
 - serial, [23](#)
- stopbits_one_point_five
 - serial, [23](#)
- stopbits_t
 - serial, [22](#)
- stopbits_two
 - serial, [23](#)
- subType
 - lidar_ans_header, [55](#)
 - ydlidar_protocol.h, [144](#)
- sync_flag
 - node_info, [58](#)
 - ydlidar_protocol.h, [144](#)
- sync_quality
 - node_info, [58](#)
 - ydlidar_protocol.h, [144](#)
- syncByte
 - cmd_packet, [29](#)
 - ydlidar_protocol.h, [144](#)
- syncByte1
 - lidar_ans_header, [55](#)
 - ydlidar_protocol.h, [144](#)
- syncByte2
 - lidar_ans_header, [55](#)
 - ydlidar_protocol.h, [144](#)
- TCGETS2
 - unix_serial.cpp, [148](#)
- TCSETS2
 - unix_serial.cpp, [148](#)
- TIOCINQ
 - unix_serial.cpp, [148](#)
- TYPE_TOF
 - ydlidar_protocol.h, [141](#)
- TYPE_TRIANGLE
 - ydlidar_protocol.h, [141](#)
- TYPE_Tail
 - ydlidar_protocol.h, [141](#)
- terminate
 - Thread, [89](#)
- Thread, [88](#)
 - _func, [89](#)
 - _handle, [89](#)
 - _param, [89](#)
 - ~Thread, [88](#)
 - createThread, [88](#)
 - createThreadAux, [88](#)
 - getHandle, [88](#)
 - getParam, [89](#)
 - join, [89](#)
 - operator==, [89](#)
 - terminate, [89](#)
 - Thread, [88](#)
 - ThreadCreateObjectFunctor, [89](#)
- thread.h
 - CLASS_THREAD, [129](#)
 - UNUSED, [129](#)
- thread_proc_t
 - v8stdint.h, [133](#)
- ThreadCreateObjectFunctor
 - Thread, [89](#)
- time_increment
 - LaserConfig, [51](#)
- Timeout
 - serial::Timeout, [90](#)
- timeout_
 - serial::Serial::SerialImpl, [87](#)
- timer.h
 - BEGIN_STATIC_CODE, [130](#)

- delay, 130
- END_STATIC_CODE, 130
- getTime, 130
- getms, 130
- timespec_from_ms
 - serial, 23
- timespec_now
 - serial::MillisecondTimer, 57
- to_degrees
 - angles, 19
- trans_delay
 - ydlidar::YDlidarDriver, 117
- trigger_interrupt_guard_condition
 - v8stdint.h, 134
- turnOff
 - CYdLidar, 45
- turnOn
 - CYdLidar, 45
- two_pi_complement
 - angles, 19
- type
 - lidar_ans_header, 55
 - ydlidar_protocol.h, 144
- UNLOCK
 - lock.h, 124
- UNUSED
 - thread.h, 129
 - v8stdint.h, 133
- unix_serial.cpp
 - BOTHER, 148
 - SNCCS, 148
 - TCGETS2, 148
 - TCSETS2, 148
 - TIOCINQ, 148
- unlock
 - Locker, 56
- unlock_device
 - lock.h, 125
- utils.h
 - YDLIDAR_API, 131
- uucp_lock
 - lock.c, 151
 - lock.h, 125
- uucp_unlock
 - lock.c, 151
 - lock.h, 125
- v8stdint.h
 - __attribute__, 133
 - __small_endian, 133
 - __access, 133
 - DEVICE_DRIVER_TYPE_SERIALPORT, 133
 - DEVICE_DRIVER_TYPE_TCP, 133
 - g_signal_status, 134
 - INVALID_TIMESTAMP, 133
 - IS_FAIL, 133
 - IS_OK, 133
 - IS_TIMEOUT, 133
 - old_signal_handler, 134
 - RESULT_FAIL, 133
 - RESULT_OK, 133
 - RESULT_TIMEOUT, 133
 - result_t, 133
 - set_signal_handler, 134
 - signal_handler, 134
 - signal_handler_t, 133
 - thread_proc_t, 133
 - trigger_interrupt_guard_condition, 134
 - UNUSED, 133
- Valu8Tou16
 - ydlidar::YDlidarDriver, 117
- W1F6GNoise_W1F5SNoise_W1F4MotorCtl_W4F0←
 - SnYear
 - LaserDebug, 52
- W2F5Output2K4K5K_W5F0Date
 - LaserDebug, 52
- W3F4BoradHardVer_W4F0Moth
 - LaserDebug, 52
- W3F4CusMajor_W4F0CusMinor
 - LaserDebug, 52
- W3F4HardwareVer_W4F0FirewareMajor
 - LaserDebug, 52
- W4F3Model_W3F0DebugInfTranVer
 - LaserDebug, 52
- W7F0SnNumH
 - LaserDebug, 52
- W7F0SnNumL
 - LaserDebug, 52
- wait
 - Event, 49
- waitByteTimes
 - serial::Serial, 80
 - serial::Serial::SerialImpl, 86
- waitDevicePackage
 - ydlidar::YDlidarDriver, 112
- waitForChange
 - serial::Serial, 80
 - serial::Serial::SerialImpl, 86
- waitForData
 - ydlidar::YDlidarDriver, 112
- waitPackage
 - ydlidar::YDlidarDriver, 112
- waitReadable
 - serial::Serial, 80
 - serial::Serial::SerialImpl, 86
- waitResponseHeader
 - ydlidar::YDlidarDriver, 113
- waitScanData
 - ydlidar::YDlidarDriver, 113
- waitfordata
 - serial::Serial, 80
 - serial::Serial::SerialImpl, 86
- write
 - serial::Serial, 80, 81
 - serial::Serial::SerialImpl, 86
- write_

- serial::Serial, [81](#)
- write_mutex
 - serial::Serial::SerialImpl, [87](#)
- write_timeout_constant
 - serial::Timeout, [90](#)
- write_timeout_multiplier
 - serial::Timeout, [90](#)
- writeData
 - serial::Serial, [81](#)
- writeLock
 - serial::Serial::SerialImpl, [86](#)
- writeUnlock
 - serial::Serial::SerialImpl, [86](#)
- xonxoff_
 - serial::Serial::SerialImpl, [87](#)
- YDLIDAR_API
 - utils.h, [131](#)
- YDLIDAR_F2
 - ydlidar, [24](#)
- YDLIDAR_F4
 - ydlidar, [24](#)
- YDLIDAR_F4PRO
 - ydlidar, [24](#)
- YDLIDAR_G1
 - ydlidar, [25](#)
- YDLIDAR_G10
 - ydlidar, [25](#)
- YDLIDAR_G2A
 - ydlidar, [25](#)
- YDLIDAR_G2B
 - ydlidar, [25](#)
- YDLIDAR_G2C
 - ydlidar, [25](#)
- YDLIDAR_G4
 - ydlidar, [24](#)
- YDLIDAR_G4PRO
 - ydlidar, [24](#)
- YDLIDAR_G4B
 - ydlidar, [25](#)
- YDLIDAR_G4C
 - ydlidar, [25](#)
- YDLIDAR_G6
 - ydlidar, [25](#)
- YDLIDAR_MODLES
 - ydlidar, [24](#)
- YDLIDAR_R2
 - ydlidar, [24](#)
- YDLIDAR_RATE_10K
 - ydlidar, [25](#)
- YDLIDAR_RATE_4K
 - ydlidar, [25](#)
- YDLIDAR_RATE_8K
 - ydlidar, [25](#)
- YDLIDAR_RATE_9K
 - ydlidar, [25](#)
- YDLIDAR_RATE
 - ydlidar, [25](#)
- YDLIDAR_S2
 - ydlidar, [25](#)
- YDLIDAR_S4
 - ydlidar, [24](#)
- YDLIDAR_S4B
 - ydlidar, [25](#)
- YDLIDAR_T1
 - ydlidar, [24](#)
- YDLIDAR_TG15
 - ydlidar, [25](#)
- YDLIDAR_TG30
 - ydlidar, [25](#)
- YDLIDAR_TG50
 - ydlidar, [25](#)
- YDLIDAR_Tail
 - ydlidar, [25](#)
- YDLIDAR_X4
 - ydlidar, [24](#)
- YDlidarDriver
 - ydlidar::YDlidarDriver, [96](#)
- ydlidar, [23](#)
 - ConvertLidarToUserSmample, [25](#)
 - ConvertUserToLidarSmample, [25](#)
 - fileExists, [25](#)
 - hasIntensity, [25](#)
 - hasSampleRate, [25](#)
 - hasScanFrequencyCtrl, [26](#)
 - hasZeroAngle, [26](#)
 - init, [26](#)
 - isOctaveLidar, [26](#)
 - isOldVersionTOFLidar, [27](#)
 - isSerialNumbValid, [27](#)
 - isSupportLidar, [27](#)
 - isSupportMotorCtrl, [27](#)
 - isSupportScanFrequency, [27](#)
 - isTOFLidar, [27](#)
 - isValidSampleRate, [28](#)
 - isValidValue, [28](#)
 - isVersionValid, [28](#)
 - lidarModelDefaultSampleRate, [28](#)
 - lidarModelToString, [28](#)
 - ok, [28](#)
 - ParseLaserDebugInfo, [28](#)
 - shutdownNow, [28](#)
 - YDLIDAR_F2, [24](#)
 - YDLIDAR_F4, [24](#)
 - YDLIDAR_F4PRO, [24](#)
 - YDLIDAR_G1, [25](#)
 - YDLIDAR_G10, [25](#)
 - YDLIDAR_G2A, [25](#)
 - YDLIDAR_G2B, [25](#)
 - YDLIDAR_G2C, [25](#)
 - YDLIDAR_G4, [24](#)
 - YDLIDAR_G4PRO, [24](#)
 - YDLIDAR_G4B, [25](#)
 - YDLIDAR_G4C, [25](#)
 - YDLIDAR_G6, [25](#)
 - YDLIDAR_MODLES, [24](#)

- YDLIDAR_R2, 24
- YDLIDAR_RATE_10K, 25
- YDLIDAR_RATE_4K, 25
- YDLIDAR_RATE_8K, 25
- YDLIDAR_RATE_9K, 25
- YDLIDAR_RATE, 25
- YDLIDAR_S2, 25
- YDLIDAR_S4, 24
- YDLIDAR_S4B, 25
- YDLIDAR_T1, 24
- YDLIDAR_TG15, 25
- YDLIDAR_TG30, 25
- YDLIDAR_TG50, 25
- YDLIDAR_Tail, 25
- YDLIDAR_X4, 24
- ydliar::YDlidarDriver, 91
 - _dataEvent, 114
 - _lock, 114
 - _serial, 114
 - _serial_lock, 114
 - _thread, 114
 - ~YDlidarDriver, 96
 - ascendScanData, 96
 - async_size, 114
 - asyncRecvPos, 114
 - cacheScanData, 97
 - checkAutoConnecting, 97
 - checkDeviceInfo, 97
 - Checksum, 114
 - ChecksumCal, 114
 - ChecksumResult, 115
 - checkTransDelay, 97
 - clearDTR, 97
 - connect, 97
 - createThread, 99
 - DEFAULT_HEART_BEAT, 96
 - DEFAULT_TIMEOUT_COUNT, 96
 - DEFAULT_TIMEOUT, 96
 - disableDataGrabbing, 99
 - disconnect, 99
 - FirstSampleAngle, 115
 - flushSerial, 99
 - get_device_health_success, 115
 - get_device_info_success, 115
 - getData, 99
 - getDeviceInfo, 100
 - getHealth, 100
 - getSDKVersion, 101
 - getSamplingRate, 101
 - getScanFrequency, 101
 - getZeroOffsetAngle, 102
 - globalRecvBuffer, 115
 - grabScanData, 102
 - has_device_header, 115
 - has_package_error, 115
 - header_, 115
 - headerBuffer, 115
 - health_, 115
 - healthBuffer, 115
 - info_, 115
 - infoBuffer, 115
 - IntervalSampleAngle, 115
 - IntervalSampleAngle_LastPackage, 115
 - isAutoReconnect, 115
 - isAutoconnting, 115
 - isConnected, 115
 - isScanning, 115
 - isSupportMotorDtrCtrl, 116
 - isconnected, 103
 - isscanning, 103
 - last_device_byte, 116
 - LastSampleAngle, 116
 - LastSampleAngleCal, 116
 - lidarPortList, 103
 - m_baudrate, 116
 - m_intensities, 116
 - m_sampling_rate, 116
 - MAX_SCAN_NODES, 96
 - model, 116
 - package, 116
 - package_Sample_Index, 116
 - package_index, 116
 - PackageSampleBytes, 117
 - packages, 116
 - PropertyBuilderByName, 104, 105
 - reset, 105
 - retryCount, 117
 - sample_rate, 117
 - SampleNumIAndCTCal, 117
 - scan_frequence, 117
 - scan_node_buf, 117
 - scan_node_count, 117
 - sendCommand, 105
 - sendData, 106
 - serial_port, 117
 - setAutoReconnect, 106
 - setDTR, 107
 - setIntensities, 107
 - setSamplingRate, 107
 - setScanFrequencyAdd, 107
 - setScanFrequencyAddMic, 108
 - setScanFrequencyDis, 108
 - setScanFrequencyDisMic, 109
 - startAutoScan, 109
 - startMotor, 110
 - startScan, 110
 - stop, 111
 - stopMotor, 111
 - stopScan, 111
 - trans_delay, 117
 - Valu8Tou16, 117
 - waitDevicePackage, 112
 - waitForData, 112
 - waitPackage, 112
 - waitResponseHeader, 113
 - waitScanData, 113

- YDlidarDriver, 96
- ydliar_protocol.h
 - __attribute__, 142
 - _countof, 139
 - angle, 142
 - angle_q6_checkbit, 142
 - CT_Normal, 141
 - CT_RingStart, 141
 - CT_Tail, 141
 - checksum, 142
 - cmd_flag, 142
 - CT, 141
 - data, 142
 - debug_info, 142
 - distance_q2, 142
 - enable, 142
 - error_code, 142
 - exposure, 142
 - firmware_version, 142
 - flag, 142
 - frequency, 143
 - GLASSNOISEINTENSITY, 139
 - hardware_version, 143
 - INTENSITY_NORMAL_PACKAGE_SIZE, 139
 - index, 143
 - LIDAR_ANS_SYNC_BYTE1, 139
 - LIDAR_ANS_SYNC_BYTE2, 139
 - LIDAR_ANS_TYPE_DEVHEALTH, 139
 - LIDAR_ANS_TYPE_DEVINFO, 139
 - LIDAR_ANS_TYPE_MEASUREMENT, 139
 - LIDAR_CMD_ADD_EXPOSURE, 139
 - LIDAR_CMD_DIS_EXPOSURE, 139
 - LIDAR_CMD_DISABLE_CONST_FREQ, 139
 - LIDAR_CMD_DISABLE_LOW_POWER, 139
 - LIDAR_CMD_ENABLE_CONST_FREQ, 139
 - LIDAR_CMD_ENABLE_LOW_POWER, 139
 - LIDAR_CMD_FORCE_SCAN, 139
 - LIDAR_CMD_FORCE_STOP, 139
 - LIDAR_CMD_GET_AIMSPEED, 139
 - LIDAR_CMD_GET_DEVICE_HEALTH, 139
 - LIDAR_CMD_GET_DEVICE_INFO, 139
 - LIDAR_CMD_GET_EAI, 139
 - LIDAR_CMD_GET_OFFSET_ANGLE, 139
 - LIDAR_CMD_GET_SAMPLING_RATE, 139
 - LIDAR_CMD_RESET, 139
 - LIDAR_CMD_RUN_INVERSION, 140
 - LIDAR_CMD_RUN_POSITIVE, 140
 - LIDAR_CMD_SAVE_SET_EXPOSURE, 140
 - LIDAR_CMD_SCAN, 140
 - LIDAR_CMD_SET_AIMSPEED_ADDMIC, 140
 - LIDAR_CMD_SET_AIMSPEED_ADD, 140
 - LIDAR_CMD_SET_AIMSPEED_DISMIC, 140
 - LIDAR_CMD_SET_AIMSPEED_DIS, 140
 - LIDAR_CMD_SET_LOW_EXPOSURE, 140
 - LIDAR_CMD_SET_SAMPLING_RATE, 140
 - LIDAR_CMD_STATE_MODEL_MOTOR, 140
 - LIDAR_CMD_STOP, 140
 - LIDAR_CMD_SYNC_BYTE, 140
 - LIDAR_CMDFLAG_HAS_PAYLOAD, 140
 - LIDAR_RESP_MEASUREMENT_ANGLE_SAMP←
PLE_SHIFT, 140
 - LIDAR_RESP_MEASUREMENT_ANGLE_SHIFT, 140
 - LIDAR_RESP_MEASUREMENT_CHECKBIT, 140
 - LIDAR_RESP_MEASUREMENT_DISTANCE_S←
HIFT, 140
 - LIDAR_RESP_MEASUREMENT_QUALITY_SH←
IFT, 140
 - LIDAR_RESP_MEASUREMENT_SYNCBIT, 140
 - LIDAR_STATUS_ERROR, 140
 - LIDAR_STATUS_OK, 140
 - LIDAR_STATUS_WARNING, 140
 - LidarTypeID, 141
 - M_PI, 141
 - model, 143
 - NORMAL_PACKAGE_SIZE, 141
 - Node_Default_Quality, 141
 - Node_NotSync, 141
 - Node_Sync, 141
 - nowPackageNum, 143
 - package_CT, 143
 - package_Head, 143
 - packageFirstSampleAngle, 143
 - packageLastSampleAngle, 143
 - PackagePaidBytes, 141
 - packageSample, 143
 - packageSampleDistance, 143
 - PackageSampleMaxLngth, 141
 - PakageSampleDistance, 143
 - PakageSampleQuality, 143
 - PH, 141
 - PropertyBuilderByName, 141
 - rate, 143
 - rotation, 143
 - SUNNOISEINTENSITY, 141
 - scan_frequence, 143
 - serialnum, 143
 - size, 144
 - stamp, 144
 - state, 144
 - status, 144
 - subType, 144
 - sync_flag, 144
 - sync_quality, 144
 - syncByte, 144
 - syncByte1, 144
 - syncByte2, 144
 - TYPE_TOF, 141
 - TYPE_TRIANGLE, 141
 - TYPE_Tail, 141
 - type, 144