;; Auto-generated. Do not edit!

(when (boundp 'robotnik\_msgs\_new::LaserStatus)

(if (not (find-package "ROBOTNIK\_MSGS\_NEW"))

(make-package "ROBOTNIK\_MSGS\_NEW"))

(shadow 'LaserStatus (find-package "ROBOTNIK\_MSGS\_NEW")))

(unless (find-package "ROBOTNIK\_MSGS\_NEW::LASERSTATUS")

(make-package "ROBOTNIK\_MSGS\_NEW::LASERSTATUS"))

(in-package "ROS")

;;//! \htmlinclude LaserStatus.msg.html

(defclass robotnik\_msgs\_new::LaserStatus

:super ros::object

:slots (\_name \_detecting\_obstacles \_contaminated \_free\_warning \_warning\_zones ))

(defmethod robotnik\_msgs\_new::LaserStatus

(:init

(&key

((:name \_\_name) "")

((:detecting\_obstacles \_\_detecting\_obstacles) nil)

((:contaminated \_\_contaminated) nil)

((:free\_warning \_\_free\_warning) nil)

((:warning\_zones \_\_warning\_zones) (let (r) (dotimes (i 0) (push nil r)) r))

)

(send-super :init)

(setq \_name (string \_\_name))

(setq \_detecting\_obstacles \_\_detecting\_obstacles)

(setq \_contaminated \_\_contaminated)

(setq \_free\_warning \_\_free\_warning)

(setq \_warning\_zones \_\_warning\_zones)

self)

(:name

(&optional \_\_name)

(if \_\_name (setq \_name \_\_name)) \_name)

(:detecting\_obstacles

(&optional \_\_detecting\_obstacles)

(if \_\_detecting\_obstacles (setq \_detecting\_obstacles \_\_detecting\_obstacles)) \_detecting\_obstacles)

(:contaminated

(&optional \_\_contaminated)

(if \_\_contaminated (setq \_contaminated \_\_contaminated)) \_contaminated)

(:free\_warning

(&optional \_\_free\_warning)

(if \_\_free\_warning (setq \_free\_warning \_\_free\_warning)) \_free\_warning)

(:warning\_zones

(&optional \_\_warning\_zones)

(if \_\_warning\_zones (setq \_warning\_zones \_\_warning\_zones)) \_warning\_zones)

(:serialization-length

()

(+

;; string \_name

4 (length \_name)

;; bool \_detecting\_obstacles

1

;; bool \_contaminated

1

;; bool \_free\_warning

1

;; bool[] \_warning\_zones

(\* 1 (length \_warning\_zones)) 4

))

(:serialize

(&optional strm)

(let ((s (if strm strm

(make-string-output-stream (send self :serialization-length)))))

;; string \_name

(write-long (length \_name) s) (princ \_name s)

;; bool \_detecting\_obstacles

(if \_detecting\_obstacles (write-byte -1 s) (write-byte 0 s))

;; bool \_contaminated

(if \_contaminated (write-byte -1 s) (write-byte 0 s))

;; bool \_free\_warning

(if \_free\_warning (write-byte -1 s) (write-byte 0 s))

;; bool[] \_warning\_zones

(write-long (length \_warning\_zones) s)

(dotimes (i (length \_warning\_zones))

(if (elt \_warning\_zones i) (write-byte -1 s) (write-byte 0 s))

)

;;

(if (null strm) (get-output-stream-string s))))

(:deserialize

(buf &optional (ptr- 0))

;; string \_name

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setq \_name (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

;; bool \_detecting\_obstacles

(setq \_detecting\_obstacles (not (= 0 (sys::peek buf ptr- :char)))) (incf ptr- 1)

;; bool \_contaminated

(setq \_contaminated (not (= 0 (sys::peek buf ptr- :char)))) (incf ptr- 1)

;; bool \_free\_warning

(setq \_free\_warning (not (= 0 (sys::peek buf ptr- :char)))) (incf ptr- 1)

;; bool[] \_warning\_zones

(let (n)

(setq n (sys::peek buf ptr- :integer)) (incf ptr- 4)

(setq \_warning\_zones (make-list n))

(dotimes (i n)

(setf (elt \_warning\_zones i) (not (= 0 (sys::peek buf ptr- :char)))) (incf ptr- 1)

))

;;

self)

)

(setf (get robotnik\_msgs\_new::LaserStatus :md5sum-) "59f57d3a0c4aa9b97dcd8bd40152ebb4")

(setf (get robotnik\_msgs\_new::LaserStatus :datatype-) "robotnik\_msgs\_new/LaserStatus")

(setf (get robotnik\_msgs\_new::LaserStatus :definition-)

"string name

bool detecting\_obstacles

bool contaminated

bool free\_warning

# one input per each warning zone.

# first areas are closer to the robot (i.e. more restrictive)

bool[] warning\_zones

")

(provide :robotnik\_msgs\_new/LaserStatus "59f57d3a0c4aa9b97dcd8bd40152ebb4")