;; Auto-generated. Do not edit!

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

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

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

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

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

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

(in-package "ROS")

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

(intern "\*INIT\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*INIT\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*INIT\_STATE\* 100)

(intern "\*STANDBY\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*STANDBY\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*STANDBY\_STATE\* 200)

(intern "\*READY\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*READY\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*READY\_STATE\* 300)

(intern "\*EMERGENCY\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*EMERGENCY\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*EMERGENCY\_STATE\* 400)

(intern "\*FAILURE\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*FAILURE\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*FAILURE\_STATE\* 500)

(intern "\*SHUTDOWN\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*SHUTDOWN\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*SHUTDOWN\_STATE\* 600)

(intern "\*UNKOWN\_STATE\*" (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(shadow '\*UNKOWN\_STATE\* (find-package "ROBOTNIK\_MSGS\_NEW::STATE"))

(defconstant robotnik\_msgs\_new::State::\*UNKOWN\_STATE\* 700)

(defclass robotnik\_msgs\_new::State

:super ros::object

:slots (\_state \_desired\_freq \_real\_freq \_state\_description ))

(defmethod robotnik\_msgs\_new::State

(:init

(&key

((:state \_\_state) 0)

((:desired\_freq \_\_desired\_freq) 0.0)

((:real\_freq \_\_real\_freq) 0.0)

((:state\_description \_\_state\_description) "")

)

(send-super :init)

(setq \_state (round \_\_state))

(setq \_desired\_freq (float \_\_desired\_freq))

(setq \_real\_freq (float \_\_real\_freq))

(setq \_state\_description (string \_\_state\_description))

self)

(:state

(&optional \_\_state)

(if \_\_state (setq \_state \_\_state)) \_state)

(:desired\_freq

(&optional \_\_desired\_freq)

(if \_\_desired\_freq (setq \_desired\_freq \_\_desired\_freq)) \_desired\_freq)

(:real\_freq

(&optional \_\_real\_freq)

(if \_\_real\_freq (setq \_real\_freq \_\_real\_freq)) \_real\_freq)

(:state\_description

(&optional \_\_state\_description)

(if \_\_state\_description (setq \_state\_description \_\_state\_description)) \_state\_description)

(:serialization-length

()

(+

;; int32 \_state

4

;; float32 \_desired\_freq

4

;; float32 \_real\_freq

4

;; string \_state\_description

4 (length \_state\_description)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; int32 \_state

(write-long \_state s)

;; float32 \_desired\_freq

(sys::poke \_desired\_freq (send s :buffer) (send s :count) :float) (incf (stream-count s) 4)

;; float32 \_real\_freq

(sys::poke \_real\_freq (send s :buffer) (send s :count) :float) (incf (stream-count s) 4)

;; string \_state\_description

(write-long (length \_state\_description) s) (princ \_state\_description s)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; int32 \_state

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

;; float32 \_desired\_freq

(setq \_desired\_freq (sys::peek buf ptr- :float)) (incf ptr- 4)

;; float32 \_real\_freq

(setq \_real\_freq (sys::peek buf ptr- :float)) (incf ptr- 4)

;; string \_state\_description

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

;;

self)

)

(setf (get robotnik\_msgs\_new::State :md5sum-) "b3a486e7c5d573e5d1475cd7c02a518b")

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

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

"# constants

int32 INIT\_STATE = 100

int32 STANDBY\_STATE = 200

int32 READY\_STATE = 300

int32 EMERGENCY\_STATE = 400

int32 FAILURE\_STATE = 500

int32 SHUTDOWN\_STATE = 600

int32 UNKOWN\_STATE = 700

# state of the component

int32 state

# desired control loop frecuency

float32 desired\_freq

# real frecuency

float32 real\_freq

# Description of the state

string state\_description

")

(provide :robotnik\_msgs\_new/State "b3a486e7c5d573e5d1475cd7c02a518b")