;; Auto-generated. Do not edit!

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

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

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

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

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

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

(in-package "ROS")

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

(defclass robotnik\_msgs\_new::MotorStatus

:super ros::object

:slots (\_state \_status \_communicationstatus \_statusword \_driveflags \_activestatusword \_activedriveflags \_digitaloutputs \_digitalinputs \_averagecurrent \_analoginputs ))

(defmethod robotnik\_msgs\_new::MotorStatus

(:init

(&key

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

((:status \_\_status) "")

((:communicationstatus \_\_communicationstatus) "")

((:statusword \_\_statusword) "")

((:driveflags \_\_driveflags) "")

((:activestatusword \_\_activestatusword) (let (r) (dotimes (i 0) (push "" r)) r))

((:activedriveflags \_\_activedriveflags) (let (r) (dotimes (i 0) (push "" r)) r))

((:digitaloutputs \_\_digitaloutputs) 0)

((:digitalinputs \_\_digitalinputs) 0)

((:averagecurrent \_\_averagecurrent) 0.0)

((:analoginputs \_\_analoginputs) (make-array 0 :initial-element 0.0 :element-type :float))

)

(send-super :init)

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

(setq \_status (string \_\_status))

(setq \_communicationstatus (string \_\_communicationstatus))

(setq \_statusword (string \_\_statusword))

(setq \_driveflags (string \_\_driveflags))

(setq \_activestatusword \_\_activestatusword)

(setq \_activedriveflags \_\_activedriveflags)

(setq \_digitaloutputs (round \_\_digitaloutputs))

(setq \_digitalinputs (round \_\_digitalinputs))

(setq \_averagecurrent (float \_\_averagecurrent))

(setq \_analoginputs \_\_analoginputs)

self)

(:state

(&optional \_\_state)

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

(:status

(&optional \_\_status)

(if \_\_status (setq \_status \_\_status)) \_status)

(:communicationstatus

(&optional \_\_communicationstatus)

(if \_\_communicationstatus (setq \_communicationstatus \_\_communicationstatus)) \_communicationstatus)

(:statusword

(&optional \_\_statusword)

(if \_\_statusword (setq \_statusword \_\_statusword)) \_statusword)

(:driveflags

(&optional \_\_driveflags)

(if \_\_driveflags (setq \_driveflags \_\_driveflags)) \_driveflags)

(:activestatusword

(&optional \_\_activestatusword)

(if \_\_activestatusword (setq \_activestatusword \_\_activestatusword)) \_activestatusword)

(:activedriveflags

(&optional \_\_activedriveflags)

(if \_\_activedriveflags (setq \_activedriveflags \_\_activedriveflags)) \_activedriveflags)

(:digitaloutputs

(&optional \_\_digitaloutputs)

(if \_\_digitaloutputs (setq \_digitaloutputs \_\_digitaloutputs)) \_digitaloutputs)

(:digitalinputs

(&optional \_\_digitalinputs)

(if \_\_digitalinputs (setq \_digitalinputs \_\_digitalinputs)) \_digitalinputs)

(:averagecurrent

(&optional \_\_averagecurrent)

(if \_\_averagecurrent (setq \_averagecurrent \_\_averagecurrent)) \_averagecurrent)

(:analoginputs

(&optional \_\_analoginputs)

(if \_\_analoginputs (setq \_analoginputs \_\_analoginputs)) \_analoginputs)

(:serialization-length

()

(+

;; string \_state

4 (length \_state)

;; string \_status

4 (length \_status)

;; string \_communicationstatus

4 (length \_communicationstatus)

;; string \_statusword

4 (length \_statusword)

;; string \_driveflags

4 (length \_driveflags)

;; string[] \_activestatusword

(apply #'+ (mapcar #'(lambda (x) (+ 4 (length x))) \_activestatusword)) 4

;; string[] \_activedriveflags

(apply #'+ (mapcar #'(lambda (x) (+ 4 (length x))) \_activedriveflags)) 4

;; int32 \_digitaloutputs

4

;; int32 \_digitalinputs

4

;; float32 \_averagecurrent

4

;; float32[] \_analoginputs

(\* 4 (length \_analoginputs)) 4

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; string \_state

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

;; string \_status

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

;; string \_communicationstatus

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

;; string \_statusword

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

;; string \_driveflags

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

;; string[] \_activestatusword

(write-long (length \_activestatusword) s)

(dolist (elem \_activestatusword)

(write-long (length elem) s) (princ elem s)

)

;; string[] \_activedriveflags

(write-long (length \_activedriveflags) s)

(dolist (elem \_activedriveflags)

(write-long (length elem) s) (princ elem s)

)

;; int32 \_digitaloutputs

(write-long \_digitaloutputs s)

;; int32 \_digitalinputs

(write-long \_digitalinputs s)

;; float32 \_averagecurrent

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

;; float32[] \_analoginputs

(write-long (length \_analoginputs) s)

(dotimes (i (length \_analoginputs))

(sys::poke (elt \_analoginputs i) (send s :buffer) (send s :count) :float) (incf (stream-count s) 4)

)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; string \_state

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

;; string \_status

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

;; string \_communicationstatus

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

;; string \_statusword

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

;; string \_driveflags

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

;; string[] \_activestatusword

(let (n)

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

(setq \_activestatusword (make-list n))

(dotimes (i n)

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setf (elt \_activestatusword i) (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

))

;; string[] \_activedriveflags

(let (n)

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

(setq \_activedriveflags (make-list n))

(dotimes (i n)

(let (n) (setq n (sys::peek buf ptr- :integer)) (incf ptr- 4) (setf (elt \_activedriveflags i) (subseq buf ptr- (+ ptr- n))) (incf ptr- n))

))

;; int32 \_digitaloutputs

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

;; int32 \_digitalinputs

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

;; float32 \_averagecurrent

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

;; float32[] \_analoginputs

(let (n)

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

(setq \_analoginputs (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_analoginputs i) (sys::peek buf ptr- :float)) (incf ptr- 4)

))

;;

self)

)

(setf (get robotnik\_msgs\_new::MotorStatus :md5sum-) "bee7a4674a1d69703bedee2437db3d94")

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

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

"string state

string status

string communicationstatus

string statusword

string driveflags

string[] activestatusword

string[] activedriveflags

int32 digitaloutputs

int32 digitalinputs

float32 averagecurrent

float32[] analoginputs

")

(provide :robotnik\_msgs\_new/MotorStatus "bee7a4674a1d69703bedee2437db3d94")