;; Auto-generated. Do not edit!

(when (boundp 'ur\_msgs\_new::RobotStateRTMsg)

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

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

(shadow 'RobotStateRTMsg (find-package "UR\_MSGS\_NEW")))

(unless (find-package "UR\_MSGS\_NEW::ROBOTSTATERTMSG")

(make-package "UR\_MSGS\_NEW::ROBOTSTATERTMSG"))

(in-package "ROS")

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

(defclass ur\_msgs\_new::RobotStateRTMsg

:super ros::object

:slots (\_time \_q\_target \_qd\_target \_qdd\_target \_i\_target \_m\_target \_q\_actual \_qd\_actual \_i\_actual \_tool\_acc\_values \_tcp\_force \_tool\_vector \_tcp\_speed \_digital\_input\_bits \_motor\_temperatures \_controller\_timer \_test\_value \_robot\_mode \_joint\_modes ))

(defmethod ur\_msgs\_new::RobotStateRTMsg

(:init

(&key

((:time \_\_time) 0.0)

((:q\_target \_\_q\_target) (make-array 0 :initial-element 0.0 :element-type :float))

((:qd\_target \_\_qd\_target) (make-array 0 :initial-element 0.0 :element-type :float))

((:qdd\_target \_\_qdd\_target) (make-array 0 :initial-element 0.0 :element-type :float))

((:i\_target \_\_i\_target) (make-array 0 :initial-element 0.0 :element-type :float))

((:m\_target \_\_m\_target) (make-array 0 :initial-element 0.0 :element-type :float))

((:q\_actual \_\_q\_actual) (make-array 0 :initial-element 0.0 :element-type :float))

((:qd\_actual \_\_qd\_actual) (make-array 0 :initial-element 0.0 :element-type :float))

((:i\_actual \_\_i\_actual) (make-array 0 :initial-element 0.0 :element-type :float))

((:tool\_acc\_values \_\_tool\_acc\_values) (make-array 0 :initial-element 0.0 :element-type :float))

((:tcp\_force \_\_tcp\_force) (make-array 0 :initial-element 0.0 :element-type :float))

((:tool\_vector \_\_tool\_vector) (make-array 0 :initial-element 0.0 :element-type :float))

((:tcp\_speed \_\_tcp\_speed) (make-array 0 :initial-element 0.0 :element-type :float))

((:digital\_input\_bits \_\_digital\_input\_bits) 0.0)

((:motor\_temperatures \_\_motor\_temperatures) (make-array 0 :initial-element 0.0 :element-type :float))

((:controller\_timer \_\_controller\_timer) 0.0)

((:test\_value \_\_test\_value) 0.0)

((:robot\_mode \_\_robot\_mode) 0.0)

((:joint\_modes \_\_joint\_modes) (make-array 0 :initial-element 0.0 :element-type :float))

)

(send-super :init)

(setq \_time (float \_\_time))

(setq \_q\_target \_\_q\_target)

(setq \_qd\_target \_\_qd\_target)

(setq \_qdd\_target \_\_qdd\_target)

(setq \_i\_target \_\_i\_target)

(setq \_m\_target \_\_m\_target)

(setq \_q\_actual \_\_q\_actual)

(setq \_qd\_actual \_\_qd\_actual)

(setq \_i\_actual \_\_i\_actual)

(setq \_tool\_acc\_values \_\_tool\_acc\_values)

(setq \_tcp\_force \_\_tcp\_force)

(setq \_tool\_vector \_\_tool\_vector)

(setq \_tcp\_speed \_\_tcp\_speed)

(setq \_digital\_input\_bits (float \_\_digital\_input\_bits))

(setq \_motor\_temperatures \_\_motor\_temperatures)

(setq \_controller\_timer (float \_\_controller\_timer))

(setq \_test\_value (float \_\_test\_value))

(setq \_robot\_mode (float \_\_robot\_mode))

(setq \_joint\_modes \_\_joint\_modes)

self)

(:time

(&optional \_\_time)

(if \_\_time (setq \_time \_\_time)) \_time)

(:q\_target

(&optional \_\_q\_target)

(if \_\_q\_target (setq \_q\_target \_\_q\_target)) \_q\_target)

(:qd\_target

(&optional \_\_qd\_target)

(if \_\_qd\_target (setq \_qd\_target \_\_qd\_target)) \_qd\_target)

(:qdd\_target

(&optional \_\_qdd\_target)

(if \_\_qdd\_target (setq \_qdd\_target \_\_qdd\_target)) \_qdd\_target)

(:i\_target

(&optional \_\_i\_target)

(if \_\_i\_target (setq \_i\_target \_\_i\_target)) \_i\_target)

(:m\_target

(&optional \_\_m\_target)

(if \_\_m\_target (setq \_m\_target \_\_m\_target)) \_m\_target)

(:q\_actual

(&optional \_\_q\_actual)

(if \_\_q\_actual (setq \_q\_actual \_\_q\_actual)) \_q\_actual)

(:qd\_actual

(&optional \_\_qd\_actual)

(if \_\_qd\_actual (setq \_qd\_actual \_\_qd\_actual)) \_qd\_actual)

(:i\_actual

(&optional \_\_i\_actual)

(if \_\_i\_actual (setq \_i\_actual \_\_i\_actual)) \_i\_actual)

(:tool\_acc\_values

(&optional \_\_tool\_acc\_values)

(if \_\_tool\_acc\_values (setq \_tool\_acc\_values \_\_tool\_acc\_values)) \_tool\_acc\_values)

(:tcp\_force

(&optional \_\_tcp\_force)

(if \_\_tcp\_force (setq \_tcp\_force \_\_tcp\_force)) \_tcp\_force)

(:tool\_vector

(&optional \_\_tool\_vector)

(if \_\_tool\_vector (setq \_tool\_vector \_\_tool\_vector)) \_tool\_vector)

(:tcp\_speed

(&optional \_\_tcp\_speed)

(if \_\_tcp\_speed (setq \_tcp\_speed \_\_tcp\_speed)) \_tcp\_speed)

(:digital\_input\_bits

(&optional \_\_digital\_input\_bits)

(if \_\_digital\_input\_bits (setq \_digital\_input\_bits \_\_digital\_input\_bits)) \_digital\_input\_bits)

(:motor\_temperatures

(&optional \_\_motor\_temperatures)

(if \_\_motor\_temperatures (setq \_motor\_temperatures \_\_motor\_temperatures)) \_motor\_temperatures)

(:controller\_timer

(&optional \_\_controller\_timer)

(if \_\_controller\_timer (setq \_controller\_timer \_\_controller\_timer)) \_controller\_timer)

(:test\_value

(&optional \_\_test\_value)

(if \_\_test\_value (setq \_test\_value \_\_test\_value)) \_test\_value)

(:robot\_mode

(&optional \_\_robot\_mode)

(if \_\_robot\_mode (setq \_robot\_mode \_\_robot\_mode)) \_robot\_mode)

(:joint\_modes

(&optional \_\_joint\_modes)

(if \_\_joint\_modes (setq \_joint\_modes \_\_joint\_modes)) \_joint\_modes)

(:serialization-length

()

(+

;; float64 \_time

8

;; float64[] \_q\_target

(\* 8 (length \_q\_target)) 4

;; float64[] \_qd\_target

(\* 8 (length \_qd\_target)) 4

;; float64[] \_qdd\_target

(\* 8 (length \_qdd\_target)) 4

;; float64[] \_i\_target

(\* 8 (length \_i\_target)) 4

;; float64[] \_m\_target

(\* 8 (length \_m\_target)) 4

;; float64[] \_q\_actual

(\* 8 (length \_q\_actual)) 4

;; float64[] \_qd\_actual

(\* 8 (length \_qd\_actual)) 4

;; float64[] \_i\_actual

(\* 8 (length \_i\_actual)) 4

;; float64[] \_tool\_acc\_values

(\* 8 (length \_tool\_acc\_values)) 4

;; float64[] \_tcp\_force

(\* 8 (length \_tcp\_force)) 4

;; float64[] \_tool\_vector

(\* 8 (length \_tool\_vector)) 4

;; float64[] \_tcp\_speed

(\* 8 (length \_tcp\_speed)) 4

;; float64 \_digital\_input\_bits

8

;; float64[] \_motor\_temperatures

(\* 8 (length \_motor\_temperatures)) 4

;; float64 \_controller\_timer

8

;; float64 \_test\_value

8

;; float64 \_robot\_mode

8

;; float64[] \_joint\_modes

(\* 8 (length \_joint\_modes)) 4

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; float64 \_time

(sys::poke \_time (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

;; float64[] \_q\_target

(write-long (length \_q\_target) s)

(dotimes (i (length \_q\_target))

(sys::poke (elt \_q\_target i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_qd\_target

(write-long (length \_qd\_target) s)

(dotimes (i (length \_qd\_target))

(sys::poke (elt \_qd\_target i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_qdd\_target

(write-long (length \_qdd\_target) s)

(dotimes (i (length \_qdd\_target))

(sys::poke (elt \_qdd\_target i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_i\_target

(write-long (length \_i\_target) s)

(dotimes (i (length \_i\_target))

(sys::poke (elt \_i\_target i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_m\_target

(write-long (length \_m\_target) s)

(dotimes (i (length \_m\_target))

(sys::poke (elt \_m\_target i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_q\_actual

(write-long (length \_q\_actual) s)

(dotimes (i (length \_q\_actual))

(sys::poke (elt \_q\_actual i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_qd\_actual

(write-long (length \_qd\_actual) s)

(dotimes (i (length \_qd\_actual))

(sys::poke (elt \_qd\_actual i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_i\_actual

(write-long (length \_i\_actual) s)

(dotimes (i (length \_i\_actual))

(sys::poke (elt \_i\_actual i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_tool\_acc\_values

(write-long (length \_tool\_acc\_values) s)

(dotimes (i (length \_tool\_acc\_values))

(sys::poke (elt \_tool\_acc\_values i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_tcp\_force

(write-long (length \_tcp\_force) s)

(dotimes (i (length \_tcp\_force))

(sys::poke (elt \_tcp\_force i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_tool\_vector

(write-long (length \_tool\_vector) s)

(dotimes (i (length \_tool\_vector))

(sys::poke (elt \_tool\_vector i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64[] \_tcp\_speed

(write-long (length \_tcp\_speed) s)

(dotimes (i (length \_tcp\_speed))

(sys::poke (elt \_tcp\_speed i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64 \_digital\_input\_bits

(sys::poke \_digital\_input\_bits (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

;; float64[] \_motor\_temperatures

(write-long (length \_motor\_temperatures) s)

(dotimes (i (length \_motor\_temperatures))

(sys::poke (elt \_motor\_temperatures i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;; float64 \_controller\_timer

(sys::poke \_controller\_timer (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

;; float64 \_test\_value

(sys::poke \_test\_value (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

;; float64 \_robot\_mode

(sys::poke \_robot\_mode (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

;; float64[] \_joint\_modes

(write-long (length \_joint\_modes) s)

(dotimes (i (length \_joint\_modes))

(sys::poke (elt \_joint\_modes i) (send s :buffer) (send s :count) :double) (incf (stream-count s) 8)

)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; float64 \_time

(setq \_time (sys::peek buf ptr- :double)) (incf ptr- 8)

;; float64[] \_q\_target

(let (n)

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

(setq \_q\_target (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_q\_target i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_qd\_target

(let (n)

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

(setq \_qd\_target (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_qd\_target i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_qdd\_target

(let (n)

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

(setq \_qdd\_target (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_qdd\_target i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_i\_target

(let (n)

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

(setq \_i\_target (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_i\_target i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_m\_target

(let (n)

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

(setq \_m\_target (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_m\_target i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_q\_actual

(let (n)

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

(setq \_q\_actual (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_q\_actual i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_qd\_actual

(let (n)

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

(setq \_qd\_actual (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_qd\_actual i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_i\_actual

(let (n)

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

(setq \_i\_actual (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_i\_actual i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_tool\_acc\_values

(let (n)

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

(setq \_tool\_acc\_values (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_tool\_acc\_values i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_tcp\_force

(let (n)

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

(setq \_tcp\_force (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_tcp\_force i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_tool\_vector

(let (n)

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

(setq \_tool\_vector (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_tool\_vector i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64[] \_tcp\_speed

(let (n)

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

(setq \_tcp\_speed (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_tcp\_speed i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64 \_digital\_input\_bits

(setq \_digital\_input\_bits (sys::peek buf ptr- :double)) (incf ptr- 8)

;; float64[] \_motor\_temperatures

(let (n)

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

(setq \_motor\_temperatures (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_motor\_temperatures i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;; float64 \_controller\_timer

(setq \_controller\_timer (sys::peek buf ptr- :double)) (incf ptr- 8)

;; float64 \_test\_value

(setq \_test\_value (sys::peek buf ptr- :double)) (incf ptr- 8)

;; float64 \_robot\_mode

(setq \_robot\_mode (sys::peek buf ptr- :double)) (incf ptr- 8)

;; float64[] \_joint\_modes

(let (n)

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

(setq \_joint\_modes (instantiate float-vector n))

(dotimes (i n)

(setf (elt \_joint\_modes i) (sys::peek buf ptr- :double)) (incf ptr- 8)

))

;;

self)

)

(setf (get ur\_msgs\_new::RobotStateRTMsg :md5sum-) "ce6feddd3ccb4ca7dbcd0ff105b603c7")

(setf (get ur\_msgs\_new::RobotStateRTMsg :datatype-) "ur\_msgs\_new/RobotStateRTMsg")

(setf (get ur\_msgs\_new::RobotStateRTMsg :definition-)

"# Data structure for the realtime communications interface (aka Matlab interface)

# used by the Universal Robots controller

#

# This data structure is send at 125 Hz on TCP port 30003

#

# Dokumentation can be found on the Universal Robots Support Wiki

# (http://wiki03.lynero.net/Technical/RealTimeClientInterface?rev=9)

float64 time

float64[] q\_target

float64[] qd\_target

float64[] qdd\_target

float64[] i\_target

float64[] m\_target

float64[] q\_actual

float64[] qd\_actual

float64[] i\_actual

float64[] tool\_acc\_values

float64[] tcp\_force

float64[] tool\_vector

float64[] tcp\_speed

float64 digital\_input\_bits

float64[] motor\_temperatures

float64 controller\_timer

float64 test\_value

float64 robot\_mode

float64[] joint\_modes

")

(provide :ur\_msgs\_new/RobotStateRTMsg "ce6feddd3ccb4ca7dbcd0ff105b603c7")