;; Auto-generated. Do not edit!

(when (boundp 'ur\_dashboard\_msgs::SetModeActionGoal)

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

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

(shadow 'SetModeActionGoal (find-package "UR\_DASHBOARD\_MSGS")))

(unless (find-package "UR\_DASHBOARD\_MSGS::SETMODEACTIONGOAL")

(make-package "UR\_DASHBOARD\_MSGS::SETMODEACTIONGOAL"))

(in-package "ROS")

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

(if (not (find-package "ACTIONLIB\_MSGS"))

(ros::roseus-add-msgs "actionlib\_msgs"))

(if (not (find-package "STD\_MSGS"))

(ros::roseus-add-msgs "std\_msgs"))

(defclass ur\_dashboard\_msgs::SetModeActionGoal

:super ros::object

:slots (\_header \_goal\_id \_goal ))

(defmethod ur\_dashboard\_msgs::SetModeActionGoal

(:init

(&key

((:header \_\_header) (instance std\_msgs::Header :init))

((:goal\_id \_\_goal\_id) (instance actionlib\_msgs::GoalID :init))

((:goal \_\_goal) (instance ur\_dashboard\_msgs::SetModeGoal :init))

)

(send-super :init)

(setq \_header \_\_header)

(setq \_goal\_id \_\_goal\_id)

(setq \_goal \_\_goal)

self)

(:header

(&rest \_\_header)

(if (keywordp (car \_\_header))

(send\* \_header \_\_header)

(progn

(if \_\_header (setq \_header (car \_\_header)))

\_header)))

(:goal\_id

(&rest \_\_goal\_id)

(if (keywordp (car \_\_goal\_id))

(send\* \_goal\_id \_\_goal\_id)

(progn

(if \_\_goal\_id (setq \_goal\_id (car \_\_goal\_id)))

\_goal\_id)))

(:goal

(&rest \_\_goal)

(if (keywordp (car \_\_goal))

(send\* \_goal \_\_goal)

(progn

(if \_\_goal (setq \_goal (car \_\_goal)))

\_goal)))

(:serialization-length

()

(+

;; std\_msgs/Header \_header

(send \_header :serialization-length)

;; actionlib\_msgs/GoalID \_goal\_id

(send \_goal\_id :serialization-length)

;; ur\_dashboard\_msgs/SetModeGoal \_goal

(send \_goal :serialization-length)

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; std\_msgs/Header \_header

(send \_header :serialize s)

;; actionlib\_msgs/GoalID \_goal\_id

(send \_goal\_id :serialize s)

;; ur\_dashboard\_msgs/SetModeGoal \_goal

(send \_goal :serialize s)

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; std\_msgs/Header \_header

(send \_header :deserialize buf ptr-) (incf ptr- (send \_header :serialization-length))

;; actionlib\_msgs/GoalID \_goal\_id

(send \_goal\_id :deserialize buf ptr-) (incf ptr- (send \_goal\_id :serialization-length))

;; ur\_dashboard\_msgs/SetModeGoal \_goal

(send \_goal :deserialize buf ptr-) (incf ptr- (send \_goal :serialization-length))

;;

self)

)

(setf (get ur\_dashboard\_msgs::SetModeActionGoal :md5sum-) "0012d5645be9adc78c7732326d965640")

(setf (get ur\_dashboard\_msgs::SetModeActionGoal :datatype-) "ur\_dashboard\_msgs/SetModeActionGoal")

(setf (get ur\_dashboard\_msgs::SetModeActionGoal :definition-)

"# ====== DO NOT MODIFY! AUTOGENERATED FROM AN ACTION DEFINITION ======

Header header

actionlib\_msgs/GoalID goal\_id

SetModeGoal goal

================================================================================

MSG: std\_msgs/Header

# Standard metadata for higher-level stamped data types.

# This is generally used to communicate timestamped data

# in a particular coordinate frame.

#

# sequence ID: consecutively increasing ID

uint32 seq

#Two-integer timestamp that is expressed as:

# \* stamp.sec: seconds (stamp\_secs) since epoch (in Python the variable is called 'secs')

# \* stamp.nsec: nanoseconds since stamp\_secs (in Python the variable is called 'nsecs')

# time-handling sugar is provided by the client library

time stamp

#Frame this data is associated with

string frame\_id

================================================================================

MSG: actionlib\_msgs/GoalID

# The stamp should store the time at which this goal was requested.

# It is used by an action server when it tries to preempt all

# goals that were requested before a certain time

time stamp

# The id provides a way to associate feedback and

# result message with specific goal requests. The id

# specified must be unique.

string id

================================================================================

MSG: ur\_dashboard\_msgs/SetModeGoal

# ====== DO NOT MODIFY! AUTOGENERATED FROM AN ACTION DEFINITION ======

# This action is for setting the robot into a desired mode (e.g. RUNNING) and safety mode into a

# non-critical state (e.g. NORMAL or REDUCED), for example after a safety incident happened.

# goal

ur\_dashboard\_msgs/RobotMode target\_robot\_mode

# Stop program execution before restoring the target mode. Can be used together with 'play\_program'.

bool stop\_program

# Play the currently loaded program after target mode is reached.#

# NOTE: Requesting mode RUNNING in combination with this will make the robot continue the motion it

# was doing before. This might probably lead into the same problem (protective stop, EM-Stop due to

# faulty motion, etc.) If you want to be safe, set the 'stop\_program' flag below and manually play

# the program after robot state is returned to normal.

# This flag will only be used when requesting mode RUNNING

bool play\_program

================================================================================

MSG: ur\_dashboard\_msgs/RobotMode

int8 NO\_CONTROLLER=-1

int8 DISCONNECTED=0

int8 CONFIRM\_SAFETY=1

int8 BOOTING=2

int8 POWER\_OFF=3

int8 POWER\_ON=4

int8 IDLE=5

int8 BACKDRIVE=6

int8 RUNNING=7

int8 UPDATING\_FIRMWARE=8

int8 mode

")

(provide :ur\_dashboard\_msgs/SetModeActionGoal "0012d5645be9adc78c7732326d965640")