;; Auto-generated. Do not edit!

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

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

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

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

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

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

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

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

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

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

(in-package "ROS")

(intern "\*FUN\_SET\_DIGITAL\_OUT\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*FUN\_SET\_DIGITAL\_OUT\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*FUN\_SET\_DIGITAL\_OUT\* 1)

(intern "\*FUN\_SET\_FLAG\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*FUN\_SET\_FLAG\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*FUN\_SET\_FLAG\* 2)

(intern "\*FUN\_SET\_ANALOG\_OUT\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*FUN\_SET\_ANALOG\_OUT\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*FUN\_SET\_ANALOG\_OUT\* 3)

(intern "\*FUN\_SET\_TOOL\_VOLTAGE\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*FUN\_SET\_TOOL\_VOLTAGE\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*FUN\_SET\_TOOL\_VOLTAGE\* 4)

(intern "\*STATE\_OFF\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*STATE\_OFF\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*STATE\_OFF\* 0)

(intern "\*STATE\_ON\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*STATE\_ON\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*STATE\_ON\* 1)

(intern "\*STATE\_TOOL\_VOLTAGE\_0V\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*STATE\_TOOL\_VOLTAGE\_0V\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*STATE\_TOOL\_VOLTAGE\_0V\* 0)

(intern "\*STATE\_TOOL\_VOLTAGE\_12V\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*STATE\_TOOL\_VOLTAGE\_12V\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*STATE\_TOOL\_VOLTAGE\_12V\* 12)

(intern "\*STATE\_TOOL\_VOLTAGE\_24V\*" (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(shadow '\*STATE\_TOOL\_VOLTAGE\_24V\* (find-package "UR\_MSGS\_NEW::SETIOREQUEST"))

(defconstant ur\_msgs\_new::SetIORequest::\*STATE\_TOOL\_VOLTAGE\_24V\* 24)

(defclass ur\_msgs\_new::SetIORequest

:super ros::object

:slots (\_fun \_pin \_state ))

(defmethod ur\_msgs\_new::SetIORequest

(:init

(&key

((:fun \_\_fun) 0)

((:pin \_\_pin) 0)

((:state \_\_state) 0.0)

)

(send-super :init)

(setq \_fun (round \_\_fun))

(setq \_pin (round \_\_pin))

(setq \_state (float \_\_state))

self)

(:fun

(&optional \_\_fun)

(if \_\_fun (setq \_fun \_\_fun)) \_fun)

(:pin

(&optional \_\_pin)

(if \_\_pin (setq \_pin \_\_pin)) \_pin)

(:state

(&optional \_\_state)

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

(:serialization-length

()

(+

;; int8 \_fun

1

;; int8 \_pin

1

;; float32 \_state

4

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; int8 \_fun

(write-byte \_fun s)

;; int8 \_pin

(write-byte \_pin s)

;; float32 \_state

(sys::poke \_state (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))

;; int8 \_fun

(setq \_fun (sys::peek buf ptr- :char)) (incf ptr- 1)

(if (> \_fun 127) (setq \_fun (- \_fun 256)))

;; int8 \_pin

(setq \_pin (sys::peek buf ptr- :char)) (incf ptr- 1)

(if (> \_pin 127) (setq \_pin (- \_pin 256)))

;; float32 \_state

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

;;

self)

)

(defclass ur\_msgs\_new::SetIOResponse

:super ros::object

:slots (\_success ))

(defmethod ur\_msgs\_new::SetIOResponse

(:init

(&key

((:success \_\_success) nil)

)

(send-super :init)

(setq \_success \_\_success)

self)

(:success

(&optional \_\_success)

(if \_\_success (setq \_success \_\_success)) \_success)

(:serialization-length

()

(+

;; bool \_success

1

))

(:serialize

(&optional strm)

(let ((s (if strm strm

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

;; bool \_success

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

;;

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

(:deserialize

(buf &optional (ptr- 0))

;; bool \_success

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

;;

self)

)

(defclass ur\_msgs\_new::SetIO

:super ros::object

:slots ())

(setf (get ur\_msgs\_new::SetIO :md5sum-) "e1b580ccf43a938f2efbbb98bbe3e277")

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

(setf (get ur\_msgs\_new::SetIO :request) ur\_msgs\_new::SetIORequest)

(setf (get ur\_msgs\_new::SetIO :response) ur\_msgs\_new::SetIOResponse)

(defmethod ur\_msgs\_new::SetIORequest

(:response () (instance ur\_msgs\_new::SetIOResponse :init)))

(setf (get ur\_msgs\_new::SetIORequest :md5sum-) "e1b580ccf43a938f2efbbb98bbe3e277")

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

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

"# Service allows setting digital (DIO) and analog (AIO) IO, as well as flags

# and configuring tool voltage.

#

# This service has three request fields (see below).

#

# When setting DIO or AIO pins to new values:

#

# fun The function to perform

# pin Numeric ID of the pin to set

# state Desired pin state (signal level for analog or STATE\_(ON|OFF) for DIO and flags)

#

# When configuring tool voltage:

#

# fun Set to FUN\_SET\_TOOL\_VOLTAGE

# pin Ignored

# state Desired tool voltage (use STATE\_TOOL\_VOLTAGE constants)

# constants

# valid function values

#

# Note: 'fun' is short for 'function' (ie: the function the service should perform).

int8 FUN\_SET\_DIGITAL\_OUT = 1

int8 FUN\_SET\_FLAG = 2

int8 FUN\_SET\_ANALOG\_OUT = 3

int8 FUN\_SET\_TOOL\_VOLTAGE = 4

# valid values for 'state' when setting digital IO or flags

int8 STATE\_OFF = 0

int8 STATE\_ON = 1

# valid 'state' values when setting tool voltage

int8 STATE\_TOOL\_VOLTAGE\_0V = 0

int8 STATE\_TOOL\_VOLTAGE\_12V = 12

int8 STATE\_TOOL\_VOLTAGE\_24V = 24

# request fields

int8 fun

int8 pin

float32 state

---

bool success

")

(setf (get ur\_msgs\_new::SetIOResponse :md5sum-) "e1b580ccf43a938f2efbbb98bbe3e277")

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

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

"# Service allows setting digital (DIO) and analog (AIO) IO, as well as flags

# and configuring tool voltage.

#

# This service has three request fields (see below).

#

# When setting DIO or AIO pins to new values:

#

# fun The function to perform

# pin Numeric ID of the pin to set

# state Desired pin state (signal level for analog or STATE\_(ON|OFF) for DIO and flags)

#

# When configuring tool voltage:

#

# fun Set to FUN\_SET\_TOOL\_VOLTAGE

# pin Ignored

# state Desired tool voltage (use STATE\_TOOL\_VOLTAGE constants)

# constants

# valid function values

#

# Note: 'fun' is short for 'function' (ie: the function the service should perform).

int8 FUN\_SET\_DIGITAL\_OUT = 1

int8 FUN\_SET\_FLAG = 2

int8 FUN\_SET\_ANALOG\_OUT = 3

int8 FUN\_SET\_TOOL\_VOLTAGE = 4

# valid values for 'state' when setting digital IO or flags

int8 STATE\_OFF = 0

int8 STATE\_ON = 1

# valid 'state' values when setting tool voltage

int8 STATE\_TOOL\_VOLTAGE\_0V = 0

int8 STATE\_TOOL\_VOLTAGE\_12V = 12

int8 STATE\_TOOL\_VOLTAGE\_24V = 24

# request fields

int8 fun

int8 pin

float32 state

---

bool success

")

(provide :ur\_msgs\_new/SetIO "e1b580ccf43a938f2efbbb98bbe3e277")