emucharts_fcusoftware_MisraC

Generated by Doxygen 1.8.11

Contents

Chapter 1

Table of transitions

Transition name	Current state	Next state	Condition	Action
click_CLR	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_0	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_1	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE

Transition name	Current state	Next state	Condition	Action
click_digit_2	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_3	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_4	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_5	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_6	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE

Transition name	Current state	Next state	Condition	Action
click_digit_7	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_8	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_digit_9	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_editbox_↩ pressure	QNH	EDIT_PRESSURE	! st->editbox_↩ selected	st->editbox_↩ selected = TRUE st->elapse = MA↩ X_ELAPSE st->data_↩ entry = [object Object],[object Object],[object Object],[object Object]
click_ESC	EDIT_PRESSURE	QNH		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object] st->editbox_↩ selected = FALSE

Transition name	Current state	Next state	Condition	Action
click_hPa	QNH	QNH	st->data_entry.↩ units != st->hPa	st->data_entry.↩ units = st->hPa st->data_↩ entry.dispval = st->data_entry.↩ programmedValue ∗ 33.86f st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object],[object Object] st->data_entry.↩ programmedValue = st->data_entry.↩ programmedValue ∗ 33.86f
	EDIT_PRESSURE	EDIT_PRESSURE	st->data_entry.↩ units != st->hPa	st->data_entry.↩ units = st->hPa st->data_entry.↩ programmedValue = st->data_entry.↩ programmedValue ∗ 33.86f
	STD	STD		st->data_entry.↩ units = st->hPa st->data_entry.↩ programmedValue = st->STD_HPA st->data_entry.↩ dispval = st->ST↩ D_HPA st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object]

Transition name	Current state	Next state	Condition	Action
click_inHg	QNH	QNH	st->data_entry.↩ units != st->inHg	st->data_entry.↩ units = st->inHg st->data_↩ entry.dispval = st->data_entry.↩ programmedValue / 33.86f st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object],[object Object] st->data_entry.↩ programmedValue = st->data_entry.↩ programmedValue / 33.86f
	EDIT_PRESSURE	EDIT_PRESSURE	st->data_entry.↩ units != st->inHg	st->data_entry.↩ units = st->inHg st->data_entry.↩ programmedValue = st->data_entry.↩ programmedValue / 33.46f
	STD	STD		st->data_entry.↩ units = st->inHg st->data_entry.↩ programmedValue = st->STD_INHG st->data_entry.↩ dispval = st->ST↩ D_INHG st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object]
click_OK	EDIT_PRESSURE	QNH		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object] st->editbox_↩ selected = FALSE

Transition name	Current state	Next state	Condition	Action
click_point	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapse = MA↩ X_ELAPSE
click_QNH_RADIO	STD	QNH		st->data_↩ entry.dispval = st->data_entry.↩ programmedValue st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object]
click_STD_RADIO	QNH	STD	st->data_entry.↩ units == st->hPa	st->data_entry.↩ programmedValue = st->STD_HPA st->data_entry.↩ dispval = st->ST↩ D_HPA st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object]
	QNH	STD	st->data_entry.↩ units == st->inHg	st->data_entry.↩ programmedValue = st->STD_INHG st->data_entry.↩ dispval = st->ST↩ D_INHG st->data_entry.↩ display = [object Object],[object Object],[object Object],[object Object]
tick	EDIT_PRESSURE	QNH	st->elapse == 0	st->data_↩ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object]

Transition name	Current state	Next state	Condition	Action
	EDIT_PRESSURE	EDIT_PRESSURE	st->elapse > 0	st->elapse = st->elapse - 1u

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 state Struct Reference

Structure containing labelled states and variables.

`#include <emucharts_fcusoftware_MisraC.h>`

Public Attributes

- integer data_entry decimalDigits
- C_8 data_entry display [STRING_LENGTH]
- D_64 data_entry dispval
- nat data_entry integerDigits
- C_8 data_entry msg [STRING_LENGTH]
- UC_8 data_entry pointEntered
- D_64 data_entry programmedValue
- UnitsType data_entry units
- UC_8 editbox_selected
- UI_32 elapse
- MachineState current_state
- MachineState previous_state

4.1.1 Detailed Description

Structure containing labelled states and variables.

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

4.1.2 Member Data Documentation

4.1.2.1 MachineState state::current_state

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

4.1.2.2 integer data_entry state::decimalDigits

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.3 C_8 data_entry state::display[STRING_LENGTH]

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.4 D_64 data_entry state::dispval

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.5 UC_8 state::editbox_selected

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.6 UI_32 state::elapse

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.7 nat data_entry state::integerDigits

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.8 C_8 data_entry state::msg[STRING_LENGTH]

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.9 UC_8 data_entry state::pointEntered

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.10 MachineState state::previous_state

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

4.1.2.11 D_64 data_entry state::programmedValue

Examples:

emucharts_fcusoftware_MisraC.c.

4.1.2.12 UnitsType data_entry state::units

Examples:

emucharts_fcusoftware_MisraC.c.

The documentation for this struct was generated from the following file:

- emucharts_fcusoftware_MisraC.h

Chapter 5

File Documentation

5.1 Android_emucharts_fcusoftware_MisraC.c File Reference

```
#include "Android_emucharts_fcusoftware_MisraC.h"
#include <assert.h>
```

Functions

- void enter (const MachineState newStateLabel)
  'Enter' auxiliary function.
- void leave (const MachineState currentStateLabel)
  'Leave' auxiliary function.
- JNIEXPORT void JNICALL init (JNIEnv *env, jobject callingObject, jobject obj)
  Initialiser.
- JNIEXPORT void JNICALL click_CLR (JNIEnv *env, jobject callingObject, jobject obj)
  click_CLR transition function.
- JNIEXPORT void JNICALL click_digit_0 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_0 transition function.
- JNIEXPORT void JNICALL click_digit_1 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_1 transition function.
- JNIEXPORT void JNICALL click_digit_2 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_2 transition function.
- JNIEXPORT void JNICALL click_digit_3 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_3 transition function.
- JNIEXPORT void JNICALL click_digit_4 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_4 transition function.
- JNIEXPORT void JNICALL click_digit_5 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_5 transition function.
- JNIEXPORT void JNICALL click_digit_6 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_6 transition function.
- JNIEXPORT void JNICALL click_digit_7 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_7 transition function.
- JNIEXPORT void JNICALL click_digit_8 (JNIEnv *env, jobject callingObject, jobject obj)
  click_digit_8 transition function.

- JNIEXPORT void JNICALL click_digit_9 (JNIEnv ∗env, jobject callingObject, jobject obj)
  click_digit_9 transition function.
- JNIEXPORT void JNICALL click_editbox_pressure (JNIEnv ∗env, jobject callingObject, jobject obj)
  click_editbox_pressure transition function.
- JNIEXPORT void JNICALL click_ESC (JNIEnv ∗env, jobject callingObject, jobject obj)
  click_ESC transition function.
- JNIEXPORT void JNICALL click_hPa (JNIEnv ∗env, jobject callingObject, jobject obj)
  click_hPa transition function.

Variables

- const UI_32 MAX_ELAPSE = 60u
  Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

5.1.1 Function Documentation

5.1.1.1 JNIEXPORT void JNICALL click_CLR (JNIEnv ∗ *env,* jobject *callingObject,* jobject *obj*)

click_CLR transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.2 JNIEXPORT void JNICALL click_digit_0 (JNIEnv ∗ *env,* jobject *callingObject,* jobject *obj*)

click_digit_0 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.3 JNIEXPORT void JNICALL click_digit_1 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_1 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.4 JNIEXPORT void JNICALL click_digit_2 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_2 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.5 JNIEXPORT void JNICALL click_digit_3 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_3 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.6 JNIEXPORT void JNICALL click_digit_4 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_4 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.7 JNIEXPORT void JNICALL click_digit_5 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_5 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.8 JNIEXPORT void JNICALL click_digit_6 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_6 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.9 JNIEXPORT void JNICALL click_digit_7 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_7 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.10 JNIEXPORT void JNICALL click_digit_8 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_8 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.11 JNIEXPORT void JNICALL click_digit_9 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_9 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.12 JNIEXPORT void JNICALL click_editbox_pressure (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_editbox_pressure transition function.

It changes state from "QNH" to "EDIT_PRESSURE" when condition [! st->editbox_selected] holds.

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("QNH")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

main.c.

5.1.1.13 JNIEXPORT void JNICALL click_ESC (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_ESC transition function.

It changes state from "EDIT_PRESSURE" to "QNH".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH")

Examples:

main.c.

5.1.1.14 JNIEXPORT void JNICALL click_hPa (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_hPa transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "QNH" when condition [st->data_entry.units != st->hPa] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->data_entry.units != st->hPa] holds,

from "STD" to "STD"

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("QNH" or "EDIT_PRESSURE" or "STD")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE" or "STD")

C code generated using PVSio-web MisraCPrinter ver 1.0

Tool freely available at http://www.pvsioweb.org

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.1.1.15 void enter (const MachineState *newStateLabel*)

'Enter' auxiliary function.

Parameters

newStateLabel	label to update the current state.
st	state structure pointer

Returns

void

See also

leave()

5.1.1.16 JNIEXPORT void JNICALL init (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

Initialiser.

Parameters

st	state structure pointer

Returns

void

Examples:

main.c.

5.1.1.17 void leave (const MachineState *currentStateLabel*)

'Leave' auxiliary function.

Parameters

currentStateLabel	label to update the previous state.
st	state structure pointer

Returns

void

See also

enter()

5.1.2 Variable Documentation

5.1.2.1 const UI_32 MAX_ELAPSE = 60u

Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

constants variables

5.2 Android_emucharts_fcusoftware_MisraC.h File Reference

```
#include <string.h>
#include <jni.h>
```

Macros

- #define true 1
- #define false 0
- #define TRUE 1
- #define FALSE 0
- #define STRING_LENGTH 8

Typedefs

- typedef char C_8

  Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

- typedef double D_64
- typedef unsigned int UI_32
- typedef unsigned char UC_8

Enumerations

- enum MachineState {
  EDIT_PRESSURE, QNH, STD, EDIT_PRESSURE,
  QNH, STD }

  Enumeration of state labels.

Functions

- void enter (const MachineState newStateLabel)

  'Enter' auxiliary function.
- void leave (const MachineState currentStateLabel)

  'Leave' auxiliary function.
- JNIEXPORT void JNICALL init (JNIEnv ∗env, jobject callingObject, jobject obj)

  Initialiser.
- JNIEXPORT void JNICALL click_CLR (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_CLR transition function.
- JNIEXPORT void JNICALL click_digit_0 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_0 transition function.
- JNIEXPORT void JNICALL click_digit_1 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_1 transition function.
- JNIEXPORT void JNICALL click_digit_2 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_2 transition function.
- JNIEXPORT void JNICALL click_digit_3 (JNIEnv ∗env, jobject callingObject, jobject obj)

click_digit_3 transition function.

- JNIEXPORT void JNICALL click_digit_4 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_4 transition function.
- JNIEXPORT void JNICALL click_digit_5 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_5 transition function.
- JNIEXPORT void JNICALL click_digit_6 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_6 transition function.
- JNIEXPORT void JNICALL click_digit_7 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_7 transition function.
- JNIEXPORT void JNICALL click_digit_8 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_8 transition function.
- JNIEXPORT void JNICALL click_digit_9 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_9 transition function.
- JNIEXPORT void JNICALL click_editbox_pressure (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_editbox_pressure transition function.
- JNIEXPORT void JNICALL click_ESC (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_ESC transition function.
- JNIEXPORT void JNICALL click_hPa (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_hPa transition function.
- JNIEXPORT void JNICALL click_inHg (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_inHg transition function.
- JNIEXPORT void JNICALL click_OK (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_OK transition function.
- JNIEXPORT void JNICALL click_point (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_point transition function.
- JNIEXPORT void JNICALL click_QNH_RADIO (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_QNH_RADIO transition function.
- JNIEXPORT void JNICALL click_STD_RADIO (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_STD_RADIO transition function.
- JNIEXPORT void JNICALL tick (JNIEnv ∗env, jobject callingObject, jobject obj)

  tick transition function.
- JNIEXPORT jboolean JNICALL per_click_CLR (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_CLR permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_0 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_0 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_1 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_1 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_2 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_2 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_3 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_3 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_4 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_4 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_5 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_5 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_6 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_6 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_7 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_7 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_digit_8 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_8 permission function for transition.

- JNIEXPORT jboolean JNICALL per_click_digit_9 (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_digit_9 permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_editbox_pressure (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_editbox_pressure permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_ESC (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_ESC permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_hPa (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_hPa permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_inHg (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_inHg permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_OK (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_OK permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_point (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_point permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_QNH_RADIO (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_QNH_RADIO permission function for transition.
- JNIEXPORT jboolean JNICALL per_click_STD_RADIO (JNIEnv ∗env, jobject callingObject, jobject obj)

  click_STD_RADIO permission function for transition.
- JNIEXPORT jboolean JNICALL per_tick (JNIEnv ∗env, jobject callingObject, jobject obj)

  tick permission function for transition.
- JNIEXPORT integer JNICALL Get_1data_entry decimalDigits (JNIEnv ∗env, jobject callingObject)

  Get data_entry.decimalDigits function.
- JNIEXPORT C_8 JNICALL Get_1data_entry display (JNIEnv ∗env, jobject callingObject)

  Get data_entry.display function.
- JNIEXPORT D_64 JNICALL Get_1data_entry dispval (JNIEnv ∗env, jobject callingObject)

  Get data_entry.dispval function.
- JNIEXPORT nat JNICALL Get_1data_entry integerDigits (JNIEnv ∗env, jobject callingObject)

  Get data_entry.integerDigits function.
- JNIEXPORT C_8 JNICALL Get_1data_entry msg (JNIEnv ∗env, jobject callingObject)

  Get data_entry.msg function.
- JNIEXPORT UC_8 JNICALL Get_1data_entry pointEntered (JNIEnv ∗env, jobject callingObject)

  Get data_entry.pointEntered function.
- JNIEXPORT D_64 JNICALL Get_1data_entry programmedValue (JNIEnv ∗env, jobject callingObject)

  Get data_entry.programmedValue function.
- JNIEXPORT UnitsType JNICALL Get_1data_entry units (JNIEnv ∗env, jobject callingObject)

  Get data_entry.units function.
- JNIEXPORT UC_8 JNICALL Get_1editbox_selected (JNIEnv ∗env, jobject callingObject)

  Get editbox_selected function.
- JNIEXPORT UI_32 JNICALL Get_1elapse (JNIEnv ∗env, jobject callingObject)

  Get elapse function.

Variables

- const UI_32 MAX_ELAPSE

  constants variables

5.2.1 Macro Definition Documentation

5.2.1.1 #define false 0

5.2.1.2 #define FALSE 0

Examples:

emucharts_fcusoftware_MisraC.c.

5.2.1.3 #define STRING_LENGTH 8

5.2.1.4 #define true 1

5.2.1.5 #define TRUE 1

Examples:

emucharts_fcusoftware_MisraC.c.

5.2.2 Typedef Documentation

5.2.2.1 typedef char C_8

Model: emucharts_fcusoftware_MisraC Author: <author name>="""> <affiliation>

5.2.2.2 typedef double D_64

5.2.2.3 typedef unsigned char UC_8

5.2.2.4 typedef unsigned int UI_32

5.2.3 Enumeration Type Documentation

5.2.3.1 enum MachineState

Enumeration of state labels.

Enumerator

EDIT_PRESSURE
QNH
STD
EDIT_PRESSURE
QNH
STD

5.2.4 Function Documentation

5.2.4.1 JNIEXPORT void JNICALL click_CLR (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_CLR transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.2 JNIEXPORT void JNICALL click_digit_0 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_0 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.3 JNIEXPORT void JNICALL click_digit_1 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_1 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.4 JNIEXPORT void JNICALL click_digit_2 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_2 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.5 JNIEXPORT void JNICALL click_digit_3 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_3 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.6 JNIEXPORT void JNICALL click_digit_4 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_4 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.7 JNIEXPORT void JNICALL click_digit_5 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_5 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.8 JNIEXPORT void JNICALL click_digit_6 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_6 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.9 JNIEXPORT void JNICALL click_digit_7 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_7 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.10 JNIEXPORT void JNICALL click_digit_8 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_8 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.11 JNIEXPORT void JNICALL click_digit_9 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_9 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.12 JNIEXPORT void JNICALL click_editbox_pressure (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_editbox_pressure transition function.

It changes state from "QNH" to "EDIT_PRESSURE" when condition [! st->editbox_selected] holds.

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("QNH")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.2.4.13 JNIEXPORT void JNICALL click_ESC (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_ESC transition function.

It changes state from "EDIT_PRESSURE" to "QNH".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH")

5.2.4.14 JNIEXPORT void JNICALL click_hPa (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_hPa transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "QNH" when condition [st->data_entry.units != st->hPa] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->data_entry.units != st->hPa] holds,

from "STD" to "STD"

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("QNH" or "EDIT_PRESSURE" or "STD")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE" or "STD")

C code generated using PVSio-web MisraCPrinter ver 1.0

Tool freely available at http://www.pvsioweb.org

5.2.4.15 JNIEXPORT void JNICALL click_inHg (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_inHg transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "QNH" when condition [st->data_entry.units != st->inHg] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->data_entry.units != st->inHg] holds,

from "STD" to "STD"

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("QNH" or "EDIT_PRESSURE" or "STD")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE" or "STD")

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.16 JNIEXPORT void JNICALL click_OK (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_OK transition function.

It changes state from "EDIT_PRESSURE" to "QNH".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH")

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.17 JNIEXPORT void JNICALL click_point (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_point transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.18 JNIEXPORT void JNICALL click_QNH_RADIO (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_QNH_RADIO transition function.

It changes state from "STD" to "QNH".

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("STD")

Postcondition

function is moving to the right state ("QNH")

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.19 JNIEXPORT void JNICALL click_STD_RADIO (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_STD_RADIO transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "STD" when condition [st->data_entry.units == st->hPa] holds,

from "QNH" to "STD" when condition [st->data_entry.units == st->inHg] holds

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("QNH")

Postcondition

function is moving to the right state ("STD")

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.20 JNIEXPORT integer JNICALL Get_1data_entry decimalDigits (JNIEnv * *env,* jobject *callingObject*)

Get data_entry.decimalDigits function.

Returns

integer

5.2.4.21 JNIEXPORT C_8 JNICALL Get_1data_entry display (JNIEnv * *env,* jobject *callingObject*)

Get data_entry.display function.

Returns

C_8

5.2.4.22 JNIEXPORT D_64 JNICALL Get_1data_entry dispval (JNIEnv * *env,* jobject *callingObject*)

Get data_entry.dispval function.

Returns

D_64

5.2.4.23 void enter (const MachineState *newStateLabel*)

'Enter' auxiliary function.

Parameters

newStateLabel	label to update the current state.
st	state structure pointer

Returns

void

See also

leave()

5.2.4.24 JNIEXPORT UC_8 JNICALL Get_1editbox_selected (JNIEnv * *env,* jobject *callingObject*)

Get editbox_selected function.

Returns

UC_8

5.2.4.25 JNIEXPORT UI_32 JNICALL Get_1elapse (JNIEnv * *env,* jobject *callingObject*)

Get elapse function.

Returns

UI_32

5.2.4.26 JNIEXPORT void JNICALL init (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

Initialiser.

Parameters

st	state structure pointer

Returns

void

5.2.4.27 JNIEXPORT nat JNICALL Get_1data_entry integerDigits (JNIEnv * *env,* jobject *callingObject*)

Get data_entry.integerDigits function.

Returns

nat

5.2.4.28 void leave (const MachineState *currentStateLabel*)

'Leave' auxiliary function.

Parameters

currentStateLabel	label to update the previous state.
st	state structure pointer

Returns

void

See also

enter()

5.2.4.29 JNIEXPORT C_8 JNICALL Get_1data_entry msg (JNIEnv * *env*, jobject *callingObject*)

Get data_entry.msg function.

Returns

C_8

5.2.4.30 JNIEXPORT jboolean JNICALL per_click_CLR (JNIEnv * *env*, jobject *callingObject*, jobject *obj*)

click_CLR permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.31 JNIEXPORT jboolean JNICALL per_click_digit_0 (JNIEnv * *env*, jobject *callingObject*, jobject *obj*)

click_digit_0 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.32 JNIEXPORT jboolean JNICALL per_click_digit_1 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_1 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.33 JNIEXPORT jboolean JNICALL per_click_digit_2 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_2 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.34 JNIEXPORT jboolean JNICALL per_click_digit_3 (JNIEnv * *env*, jobject *callingObject*, jobject *obj*)

click_digit_3 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.35 JNIEXPORT jboolean JNICALL per_click_digit_4 (JNIEnv * *env*, jobject *callingObject*, jobject *obj*)

click_digit_4 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.36 JNIEXPORT jboolean JNICALL per_click_digit_5 (JNIEnv * *env*, jobject *callingObject*, jobject *obj*)

click_digit_5 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.37 JNIEXPORT jboolean JNICALL per_click_digit_6 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_6 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.38 JNIEXPORT jboolean JNICALL per_click_digit_7 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_7 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.39 JNIEXPORT jboolean JNICALL per_click_digit_8 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_8 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩
PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.40 JNIEXPORT jboolean JNICALL per_click_digit_9 (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_digit_9 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.41 JNIEXPORT jboolean JNICALL per_click_editbox_pressure (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_editbox_pressure permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.42 JNIEXPORT jboolean JNICALL per_click_ESC (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_ESC permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.43 JNIEXPORT jboolean JNICALL per_click_hPa (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_hPa permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH" or "EDIT_PRESSURE" or "STD")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.44 JNIEXPORT jboolean JNICALL per_click_inHg (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_inHg permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH" or "EDIT_PRESSURE" or "STD")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.45 JNIEXPORT jboolean JNICALL per_click_OK (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_OK permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.46 JNIEXPORT jboolean JNICALL per_click_point (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_point permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.47 JNIEXPORT jboolean JNICALL per_click_QNH_RADIO (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_QNH_RADIO permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "STD")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.48 JNIEXPORT jboolean JNICALL per_click_STD_RADIO (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

click_STD_RADIO permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.49 JNIEXPORT jboolean JNICALL per_tick (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

tick permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.50 JNIEXPORT UC_8 JNICALL Get_1data_entry pointEntered (JNIEnv * *env,* jobject *callingObject*)

Get data_entry.pointEntered function.

Returns

UC_8

5.2.4.51 JNIEXPORT D_64 JNICALL Get_1data_entry programmedValue (JNIEnv * *env,* jobject *callingObject*)

Get data_entry.programmedValue function.

Returns

D_64

5.2.4.52 JNIEXPORT void JNICALL tick (JNIEnv * *env,* jobject *callingObject,* jobject *obj*)

tick transition function.

This function is generated merging two or more triggers with the same name. It changes state from "EDIT_PRE↩SSURE" to "QNH" when condition [st->elapse == 0] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->elapse > 0] holds

Parameters

st	state structure pointer

Returns

void

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c, and main.c.

5.2.4.53 JNIEXPORT UnitsType JNICALL Get_1data_entry units (JNIEnv * *env*, jobject *callingObject*)

Get data_entry.units function.

Returns

UnitsType

5.2.5 Variable Documentation

5.2.5.1 const UI_32 MAX_ELAPSE

constants variables

constants variables

5.3 emucharts_fcusoftware_MisraC.c File Reference

```
#include "emucharts_fcusoftware_MisraC.h"
#include <assert.h>
```

Functions

- void enter (MachineState newStateLabel, state *st)
  'Enter' auxiliary function.
- void leave (MachineState currentStateLabel, state *st)
  'Leave' auxiliary function.
- state init (state *st)
  Initialiser.
- state click_CLR (state *st)
  click_CLR transition function.
- state click_digit_0 (state *st)
  click_digit_0 transition function.
- state click_digit_1 (state *st)
  click_digit_1 transition function.
- state click_digit_2 (state *st)
  click_digit_2 transition function.
- state click_digit_3 (state *st)
  click_digit_3 transition function.
- state click_digit_4 (state *st)
  click_digit_4 transition function.
- state click_digit_5 (state *st)
  click_digit_5 transition function.
- state click_digit_6 (state *st)
  click_digit_6 transition function.
- state click_digit_7 (state *st)
  click_digit_7 transition function.

- state click_digit_8 (state ∗st)

  click_digit_8 transition function.
- state click_digit_9 (state ∗st)

  click_digit_9 transition function.
- state click_editbox_pressure (state ∗st)

  click_editbox_pressure transition function.
- state click_ESC (state ∗st)

  click_ESC transition function.

Variables

- const UI_32 MAX_ELAPSE = 60u

  Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

5.3.1 Function Documentation

5.3.1.1 state click_CLR (state ∗ *st*)

click_CLR transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.2 state click_digit_0 (state ∗ *st*)

click_digit_0 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.3 state click_digit_1 (state * *st*)

click_digit_1 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.4 state click_digit_2 (state * *st*)

click_digit_2 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.5 state click_digit_3 (state * *st*)

click_digit_3 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.6 state click_digit_4 (state * *st*)

click_digit_4 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.7 state click_digit_5 (state * *st*)

click_digit_5 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.8 state click_digit_6 (state * *st*)

click_digit_6 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.9 state click_digit_7 (state * *st*)

click_digit_7 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.10 state click_digit_8 (state * *st*)

click_digit_8 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.11 state click_digit_9 (state * *st*)

click_digit_9 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.12 state click_editbox_pressure (state * *st*)

click_editbox_pressure transition function.

It changes state from "QNH" to "EDIT_PRESSURE" when condition [! st->editbox_selected] holds.

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("QNH")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.13 state click_ESC (state * *st*)

click_ESC transition function.

It changes state from "EDIT_PRESSURE" to "QNH".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH")

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.14 void enter (MachineState *newStateLabel,* state * *st*)

'Enter' auxiliary function.

Parameters

newStateLabel	label to update the current state.
st	state structure pointer

Returns

void

See also

leave()

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.15 state init (state * st)

Initialiser.

Parameters

st	state structure pointer

Returns

state structure

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.1.16 void leave (MachineState *currentStateLabel*, state * st)

'Leave' auxiliary function.

Parameters

currentStateLabel	label to update the previous state.
st	state structure pointer

Returns

void

See also

enter()

Examples:

emucharts_fcusoftware_MisraC.c.

5.3.2 Variable Documentation

5.3.2.1 const UI_32 MAX_ELAPSE = 60u

Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

constants variables

Examples:

emucharts_fcusoftware_MisraC.c.

5.4 emucharts_fcusoftware_MisraC.h File Reference

```
#include <string.h>
```

Classes

- struct state

  Structure containing labelled states and variables.

Macros

- #define true 1
- #define false 0
- #define TRUE 1
- #define FALSE 0
- #define STRING_LENGTH 8

Typedefs

- typedef char C_8

  Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

- typedef double D_64
- typedef unsigned int UI_32
- typedef unsigned char UC_8

Enumerations

- enum MachineState {
  EDIT_PRESSURE, QNH, STD, EDIT_PRESSURE,
  QNH, STD }

  Enumeration of state labels.

Functions

- void enter (MachineState newStateLabel, state *st)

  'Enter' auxiliary function.
- void leave (MachineState currentStateLabel, state *st)

  'Leave' auxiliary function.
- state init (state *st)

  Initialiser.
- state click_CLR (state *st)

  click_CLR transition function.
- state click_digit_0 (state *st)

  click_digit_0 transition function.
- state click_digit_1 (state *st)

  click_digit_1 transition function.
- state click_digit_2 (state *st)

  click_digit_2 transition function.
- state click_digit_3 (state *st)

  click_digit_3 transition function.
- state click_digit_4 (state *st)

  click_digit_4 transition function.
- state click_digit_5 (state *st)

  click_digit_5 transition function.
- state click_digit_6 (state *st)

  click_digit_6 transition function.
- state click_digit_7 (state *st)

  click_digit_7 transition function.
- state click_digit_8 (state *st)

  click_digit_8 transition function.
- state click_digit_9 (state *st)

  click_digit_9 transition function.
- state click_editbox_pressure (state *st)

  click_editbox_pressure transition function.
- state click_ESC (state *st)

  click_ESC transition function.
- state click_hPa (state *st)

  click_hPa transition function.
- state click_inHg (state *st)

click_inHg transition function.

- state click_OK (state *st)

  click_OK transition function.
- state click_point (state *st)

  click_point transition function.
- state click_QNH_RADIO (state *st)

  click_QNH_RADIO transition function.
- state click_STD_RADIO (state *st)

  click_STD_RADIO transition function.
- state tick (state *st)

  tick transition function.
- UC_8 per_click_CLR (const state *st)

  click_CLR permission function for transition.
- UC_8 per_click_digit_0 (const state *st)

  click_digit_0 permission function for transition.
- UC_8 per_click_digit_1 (const state *st)

  click_digit_1 permission function for transition.
- UC_8 per_click_digit_2 (const state *st)

  click_digit_2 permission function for transition.
- UC_8 per_click_digit_3 (const state *st)

  click_digit_3 permission function for transition.
- UC_8 per_click_digit_4 (const state *st)

  click_digit_4 permission function for transition.
- UC_8 per_click_digit_5 (const state *st)

  click_digit_5 permission function for transition.
- UC_8 per_click_digit_6 (const state *st)

  click_digit_6 permission function for transition.
- UC_8 per_click_digit_7 (const state *st)

  click_digit_7 permission function for transition.
- UC_8 per_click_digit_8 (const state *st)

  click_digit_8 permission function for transition.
- UC_8 per_click_digit_9 (const state *st)

  click_digit_9 permission function for transition.
- UC_8 per_click_editbox_pressure (const state *st)

  click_editbox_pressure permission function for transition.
- UC_8 per_click_ESC (const state *st)

  click_ESC permission function for transition.
- UC_8 per_click_hPa (const state *st)

  click_hPa permission function for transition.
- UC_8 per_click_inHg (const state *st)

  click_inHg permission function for transition.
- UC_8 per_click_OK (const state *st)

  click_OK permission function for transition.
- UC_8 per_click_point (const state *st)

  click_point permission function for transition.
- UC_8 per_click_QNH_RADIO (const state *st)

  click_QNH_RADIO permission function for transition.
- UC_8 per_click_STD_RADIO (const state *st)

  click_STD_RADIO permission function for transition.
- UC_8 per_tick (const state *st)

  tick permission function for transition.

Variables

- const UI_32 MAX_ELAPSE
  constants variables

5.4.1 Macro Definition Documentation

5.4.1.1 #define false 0

5.4.1.2 #define FALSE 0

5.4.1.3 #define STRING_LENGTH 8

5.4.1.4 #define true 1

5.4.1.5 #define TRUE 1

5.4.2 Typedef Documentation

5.4.2.1 typedef char C_8

Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

5.4.2.2 typedef double D_64

5.4.2.3 typedef unsigned char UC_8

5.4.2.4 typedef unsigned int UI_32

5.4.3 Enumeration Type Documentation

5.4.3.1 enum MachineState

Enumeration of state labels.

Enumerator

EDIT_PRESSURE
QNH
STD
EDIT_PRESSURE
QNH
STD

5.4.4 Function Documentation

5.4.4.1 state click_CLR (state * *st*)

click_CLR transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.2 state click_digit_0 (state * st)

click_digit_0 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.3 state click_digit_1 (state * st)

click_digit_1 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.4 state click_digit_2 (state * st)

click_digit_2 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.5 state click_digit_3 (state * st)

click_digit_3 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.6 state click_digit_4 (state * st)

click_digit_4 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.7 state click_digit_5 (state * st)

click_digit_5 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.8 state click_digit_6 (state * st)

click_digit_6 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.9 state click_digit_7 (state * st)

click_digit_7 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.10 state click_digit_8 (state * st)

click_digit_8 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.11 state click_digit_9 (state * st)

click_digit_9 transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.12 state click_editbox_pressure (state * st)

click_editbox_pressure transition function.

It changes state from "QNH" to "EDIT_PRESSURE" when condition [! st->editbox_selected] holds.

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("QNH")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.13 state click_ESC (state * st)

click_ESC transition function.

It changes state from "EDIT_PRESSURE" to "QNH".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH")

5.4.4.14 state click_hPa (state * st)

click_hPa transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "QNH" when condition [st->data_entry.units != st->hPa] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->data_entry.units != st->hPa] holds,

from "STD" to "STD"

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("QNH" or "EDIT_PRESSURE" or "STD")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE" or "STD")

5.4.4.15 state click_inHg (state * *st*)

click_inHg transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "QNH" when condition [st->data_entry.units != st->inHg] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->data_entry.units != st->inHg] holds,

from "STD" to "STD"

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("QNH" or "EDIT_PRESSURE" or "STD")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE" or "STD")

5.4.4.16 state click_OK (state * *st*)

click_OK transition function.

It changes state from "EDIT_PRESSURE" to "QNH".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH")

5.4.4.17 state click_point (state * st)

click_point transition function.

It changes state from "EDIT_PRESSURE" to "EDIT_PRESSURE".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("EDIT_PRESSURE")

5.4.4.18 state click_QNH_RADIO (state * st)

click_QNH_RADIO transition function.

It changes state from "STD" to "QNH".

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("STD")

Postcondition

function is moving to the right state ("QNH")

5.4.4.19 state click_STD_RADIO (state * *st*)

click_STD_RADIO transition function.

This function is generated merging two or more triggers with the same name. It changes state from "QNH" to "STD" when condition [st->data_entry.units == st->hPa] holds,

from "QNH" to "STD" when condition [st->data_entry.units == st->inHg] holds

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("QNH")

Postcondition

function is moving to the right state ("STD")

5.4.4.20 void enter (MachineState *newStateLabel,* state * *st*)

'Enter' auxiliary function.

Parameters

newStateLabel	label to update the current state.
st	state structure pointer

Returns

void

See also

leave()

5.4.4.21 state init (state * st)

Initialiser.

Parameters

st	state structure pointer

Returns

state structure

5.4.4.22 void leave (MachineState currentStateLabel, state * st)

'Leave' auxiliary function.

Parameters

currentStateLabel	label to update the previous state.
st	state structure pointer

Returns

void

See also

enter()

5.4.4.23 UC_8 per_click_CLR (const state * st)

click_CLR permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.24 UC_8 per_click_digit_0 (const state * st)

click_digit_0 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.25 UC_8 per_click_digit_1 (const state * st)

click_digit_1 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.26 UC_8 per_click_digit_2 (const state * st)

click_digit_2 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.27 UC_8 per_click_digit_3 (const state * st)

click_digit_3 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.28 UC_8 per_click_digit_4 (const state * st)

click_digit_4 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.29 UC_8 per_click_digit_5 (const state * st)

click_digit_5 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.30 UC_8 per_click_digit_6 (const state * st)

click_digit_6 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.31 UC_8 per_click_digit_7 (const state * st)

click_digit_7 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.32 UC_8 per_click_digit_8 (const state * st)

click_digit_8 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.33 UC_8 per_click_digit_9 (const state * st)

click_digit_9 permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.34 UC_8 per_click_editbox_pressure (const state * st)

click_editbox_pressure permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.35 UC_8 per_click_ESC (const state * st)

click_ESC permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.36 UC_8 per_click_hPa (const state * st)

click_hPa permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH" or "EDIT_PRESSURE" or "STD")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.37 UC_8 per_click_inHg (const state * st)

click_inHg permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH" or "EDIT_PRESSURE" or "STD")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.38 UC_8 per_click_OK (const state * st)

click_OK permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.39 UC_8 per_click_point (const state * st)

click_point permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.40 UC_8 per_click_QNH_RADIO (const state * st)

click_QNH_RADIO permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "STD")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.41 UC_8 per_click_STD_RADIO (const state * st)

click_STD_RADIO permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "QNH")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.42 UC_8 per_tick (const state * st)

tick permission function for transition.

Use to check if functions can be performed, it controls if the current state is eligible. (i.e., current state is "EDIT_↩ PRESSURE")

Parameters

st	state structure pointer

Returns

boolean

5.4.4.43 state tick (state * st)

tick transition function.

This function is generated merging two or more triggers with the same name. It changes state from "EDIT_PRE↩ SSURE" to "QNH" when condition [st->elapse == 0] holds,

from "EDIT_PRESSURE" to "EDIT_PRESSURE" when condition [st->elapse > 0] holds

Parameters

st	state structure pointer

Returns

state structure

Precondition

function is called from the right state ("EDIT_PRESSURE")

Postcondition

function is moving to the right state ("QNH" or "EDIT_PRESSURE")

5.4.5 Variable Documentation

5.4.5.1 const UI_32 MAX_ELAPSE

constants variables

constants variables

5.5 main.c File Reference

```
#include "emucharts_fcusoftware_MisraC.h"
#include <stdio.h>
#include <string.h>
```

Macros

- #define NUM_OF_FUNC 20

  Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

Functions

- int main ()

5.5.1 Macro Definition Documentation

5.5.1.1 #define NUM_OF_FUNC 20

Model: emucharts_fcusoftware_MisraC Author: <author name>=""> <affiliation>

Examples:

main.c.

5.5.2 Function Documentation

5.5.2.1 int main ( )

< Useful for printf()

Examples:

main.c.

Chapter 6

Example Documentation

6.1 emucharts_fcusoftware_MisraC.c

```
#include "emucharts_fcusoftware_MisraC.h"
#include <assert.h>

#ifdef DEBUG
#include <stdio.h>
#define debug_print(fmt, args...) \
        do { fprintf(stderr, "%s:%d:%s(): " fmt, \
                __FILE__, __LINE__, __FUNCTION__, ##args); } while (0)
#endif
/* constants variables */
const UI_32 MAX_ELAPSE = 60u;

/* definition of auxiliary functions */
void enter(MachineState newStateLabel, state *st) {
#ifdef DEBUG
    debug_print("Entering state nr. '%u'.\n", newStateLabel);
#endif
    st->current_state = newStateLabel;
}
void leave(MachineState currentStateLabel, state *st) {
#ifdef DEBUG
    debug_print("Leaving state nr. '%u'.\n", currentStateLabel);
#endif
    st->previous_state = currentStateLabel;
}

/* definition of initialisation function */
state init(state *st){
#ifdef DEBUG
    debug_print("Initialisation of state variables.\n");
#endif
    st->data_entry.decimalDigits = 0;
    strcpy(st->data_entry.display, "STD_INHG");
    st->data_entry.dispval = STD_INHG;
    st->data_entry.integerDigits = 0;
    strcpy(st->data_entry.msg, """");
    st->data_entry.pointEntered = FALSE;
    st->data_entry.programmedValue = STD_INHG;
    st->data_entry.units = inHg;
    st->editbox_selected = FALSE;
    st->elapse = MAX_ELAPSE;
    enter(STD, st);

    assert ( st->current_state == STD );
    return *st;
}

/* definition of transition functions */
state click_CLR(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
```

```
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_0(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_1(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_2(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_3(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}
```

```
state click_digit_4(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_5(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_6(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_7(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_8(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
```

```
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_digit_9(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_editbox_pressure(state *st) {
    assert( st->current_state == QNH);
    if (! st->editbox_selected) {
        leave(QNH, st);
        st->editbox_selected = TRUE;
#ifdef DEBUG
        debug_print("Action st->editbox_selected = TRUE issued.\n");
#endif
        st->elapse = MAX_ELAPSE;
#ifdef DEBUG
        debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif
        st->data_entry = [object Object],[object Object],[object Object],[object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object] issued.\n");
#endif
        enter(EDIT_PRESSURE, st);
    }
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_ESC(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->editbox_selected = FALSE;
#ifdef DEBUG
    debug_print("Action st->editbox_selected = FALSE issued.\n");
#endif

    enter(QNH, st);
    assert( st->current_state == QNH );
    return *st;
}

state click_hPa(state *st) {
    assert( st->current_state ==  QNH || st->current_state ==
      EDIT_PRESSURE || st->current_state ==  STD );

    if ( (st->current_state == QNH) && (st->data_entry.units != st->hPa) ) {
        leave(QNH, st);
        st->data_entry.units = st->hPa;
#ifdef DEBUG
        debug_print("Action st->data_entry.units = st->hPa issued.\n");
#endif
        st->data_entry.dispval = st->data_entry.programmedValue * 33.86f;
#ifdef DEBUG
        debug_print("Action st->data_entry.dispval = st->data_entry.programmedValue * 33.86f issued.\n");
#endif
        st->data_entry.display = [object Object],[object Object],[object Object],[object Object],[
      object Object],[object Object];
#ifdef DEBUG
```

```
        debug_print("Action st->data_entry.display = [object Object],[object Object],[object
       Object],[object Object],[object Object],[object Object] issued.\n");
#endif
        st->data_entry.programmedValue = st->data_entry.
      programmedValue * 33.86f;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->data_entry.programmedValue * 33.86f
       issued.\n");
#endif
        enter(QNH, st);
        assert( st->current_state == QNH );
        return *st;
    }
    if ( (st->current_state == EDIT_PRESSURE) && (st->data_entry.
      units != st->hPa) ) {
        leave(EDIT_PRESSURE, st);
        st->data_entry.units = st->hPa;
#ifdef DEBUG
        debug_print("Action st->data_entry.units = st->hPa issued.\n");
#endif
        st->data_entry.programmedValue = st->data_entry.
      programmedValue * 33.86f;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->data_entry.programmedValue * 33.86f
       issued.\n");
#endif
        enter(EDIT_PRESSURE, st);
        assert( st->current_state == EDIT_PRESSURE );
        return *st;
    }
    if (current_state == STD) {
        leave(STD, st);
        st->data_entry.units = st->hPa;
#ifdef DEBUG
        debug_print("Action st->data_entry.units = st->hPa issued.\n");
#endif
        st->data_entry.programmedValue = st->STD_HPA;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->STD_HPA issued.\n");
#endif
        st->data_entry.dispval = st->STD_HPA;
#ifdef DEBUG
        debug_print("Action st->data_entry.dispval = st->STD_HPA issued.\n");
#endif
        st->data_entry.display = [object Object],[object Object],[object Object],[object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry.display = [object Object],[object Object],[object
       Object],[object Object] issued.\n");
#endif
        enter(STD, st);
        assert( st->current_state == STD );
        return *st;
    return *st;
}

state click_inHg(state *st) {
    assert( st->current_state ==  QNH || st->current_state ==
      EDIT_PRESSURE || st->current_state ==  STD );

    if ( (st->current_state == QNH) && (st->data_entry.units != st->inHg) ) {
        leave(QNH, st);
        st->data_entry.units = st->inHg;
#ifdef DEBUG
        debug_print("Action st->data_entry.units = st->inHg issued.\n");
#endif
        st->data_entry.dispval = st->data_entry.programmedValue / 33.86f;
#ifdef DEBUG
        debug_print("Action st->data_entry.dispval = st->data_entry.programmedValue / 33.86f issued.\n");
#endif
        st->data_entry.display = [object Object],[object Object],[object Object],[object Object],[
      object Object],[object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry.display = [object Object],[object Object],[object
       Object],[object Object],[object Object],[object Object] issued.\n");
#endif
        st->data_entry.programmedValue = st->data_entry.
      programmedValue / 33.86f;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->data_entry.programmedValue / 33.86f
       issued.\n");
#endif
        enter(QNH, st);
        assert( st->current_state == QNH );
        return *st;
    }
    if ( (st->current_state == EDIT_PRESSURE) && (st->data_entry.
      units != st->inHg) ) {
```

```
        leave(EDIT_PRESSURE, st);
        st->data_entry.units = st->inHg;
#ifdef DEBUG
        debug_print("Action st->data_entry.units = st->inHg issued.\n");
#endif
        st->data_entry.programmedValue = st->data_entry.
      programmedValue / 33.46f;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->data_entry.programmedValue / 33.46f
       issued.\n");
#endif
        enter(EDIT_PRESSURE, st);
        assert( st->current_state == EDIT_PRESSURE );
        return *st;
    }
    if (current_state == STD) {
        leave(STD, st);
        st->data_entry.units = st->inHg;
#ifdef DEBUG
        debug_print("Action st->data_entry.units = st->inHg issued.\n");
#endif
        st->data_entry.programmedValue = st->STD_INHG;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->STD_INHG issued.\n");
#endif
        st->data_entry.dispval = st->STD_INHG;
#ifdef DEBUG
        debug_print("Action st->data_entry.dispval = st->STD_INHG issued.\n");
#endif
        st->data_entry.display = [object Object],[object Object],[object Object],[object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry.display = [object Object],[object Object],[object
       Object],[object Object] issued.\n");
#endif
        enter(STD, st);
        assert( st->current_state == STD );
        return *st;
    return *st;
}

state click_OK(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->editbox_selected = FALSE;
#ifdef DEBUG
    debug_print("Action st->editbox_selected = FALSE issued.\n");
#endif

    enter(QNH, st);
    assert( st->current_state == QNH );
    return *st;
}

state click_point(state *st) {
    assert( st->current_state == EDIT_PRESSURE);
    leave(EDIT_PRESSURE, st);
    st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
    debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
    st->elapse = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapse = MAX_ELAPSE issued.\n");
#endif

    enter(EDIT_PRESSURE, st);
    assert( st->current_state == EDIT_PRESSURE );
    return *st;
}

state click_QNH_RADIO(state *st) {
    assert( st->current_state == STD);
    leave(STD, st);
    st->data_entry.dispval = st->data_entry.programmedValue;
#ifdef DEBUG
    debug_print("Action st->data_entry.dispval = st->data_entry.programmedValue issued.\n");
#endif
    st->data_entry.display = [object Object],[object Object],[object Object],[object Object];
#ifdef DEBUG
```

```
    debug_print("Action st->data_entry.display = [object Object],[object Object],[object Object],[object
       Object] issued.\n");
#endif

    enter(QNH, st);
    assert( st->current_state == QNH );
    return *st;
}

state click_STD_RADIO(state *st) {
    assert( st->current_state ==  QNH );

    if ( (st->current_state == QNH) && (st->data_entry.units == st->hPa) ) {
        leave(QNH, st);
        st->data_entry.programmedValue = st->STD_HPA;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->STD_HPA issued.\n");
#endif
        st->data_entry.dispval = st->STD_HPA;
#ifdef DEBUG
        debug_print("Action st->data_entry.dispval = st->STD_HPA issued.\n");
#endif
        st->data_entry.display = [object Object],[object Object],[object Object],[object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry.display = [object Object],[object Object],[object
       Object],[object Object] issued.\n");
#endif
        enter(STD, st);
        assert( st->current_state == STD );
        return *st;
    }
    if ( (st->current_state == QNH) && (st->data_entry.units == st->inHg) ) {
        leave(QNH, st);
        st->data_entry.programmedValue = st->STD_INHG;
#ifdef DEBUG
        debug_print("Action st->data_entry.programmedValue = st->STD_INHG issued.\n");
#endif
        st->data_entry.dispval = st->STD_INHG;
#ifdef DEBUG
        debug_print("Action st->data_entry.dispval = st->STD_INHG issued.\n");
#endif
        st->data_entry.display = [object Object],[object Object],[object Object],[object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry.display = [object Object],[object Object],[object
       Object],[object Object] issued.\n");
#endif
        enter(STD, st);
        assert( st->current_state == STD );
        return *st;
    }
    return *st;
}

state tick(state *st) {
    assert( st->current_state ==  EDIT_PRESSURE );

    if ( (st->current_state == EDIT_PRESSURE) && (st->
      elapse == 0) ) {
        leave(EDIT_PRESSURE, st);
        st->data_entry = [object Object],[object Object],[object Object],[object Object],[object Object],[
      object Object];
#ifdef DEBUG
        debug_print("Action st->data_entry = [object Object],[object Object],[object Object],[object
       Object],[object Object],[object Object] issued.\n");
#endif
        enter(QNH, st);
        assert( st->current_state == QNH );
        return *st;
    }
    if ( (st->current_state == EDIT_PRESSURE) && (st->
      elapse > 0) ) {
        leave(EDIT_PRESSURE, st);
        st->elapse = st->elapse - 1u;
#ifdef DEBUG
        debug_print("Action st->elapse = st->elapse - 1u issued.\n");
#endif
        enter(EDIT_PRESSURE, st);
        assert( st->current_state == EDIT_PRESSURE );
        return *st;
    }
    return *st;
}


/* definition of permission function for transition functions */
UC_8 per_click_CLR(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
```

```
        return true;
    }
    return false;
}

UC_8 per_click_digit_0(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_1(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_2(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_3(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_4(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_5(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_6(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_7(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_8(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_digit_9(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_editbox_pressure(const state *st) {
    if (st->current_state == QNH){
        return true;
    }
    return false;
}

UC_8 per_click_ESC(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
```

```
}

UC_8 per_click_hPa(const state *st) {
    if (st->current_state ==  QNH || st->current_state ==
      EDIT_PRESSURE || st->current_state ==  STD ){
        return true;
    }
    return false;
}

UC_8 per_click_inHg(const state *st) {
    if (st->current_state ==  QNH || st->current_state ==
      EDIT_PRESSURE || st->current_state ==  STD ){
        return true;
    }
    return false;
}

UC_8 per_click_OK(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_point(const state *st) {
    if (st->current_state == EDIT_PRESSURE){
        return true;
    }
    return false;
}

UC_8 per_click_QNH_RADIO(const state *st) {
    if (st->current_state == STD){
        return true;
    }
    return false;
}

UC_8 per_click_STD_RADIO(const state *st) {
    if (st->current_state ==  QNH ){
        return true;
    }
    return false;
}

UC_8 per_tick(const state *st) {
    if (st->current_state ==  EDIT_PRESSURE ){
        return true;
    }
    return false;
}
```

6.2 main.c

An example to test the C code generated.

```
#include "emucharts_fcusoftware_MisraC.h"
#include <stdio.h>
#include <string.h>
#define NUM_OF_FUNC 20


int main(){
    UC_8 *MachineState[] = { (UC_8*)"EDIT_PRESSURE", (UC_8*)"QNH", (
      UC_8*)"STD" };   // Useful for printf()
    /*
     * At first instantiate state variable and call init() in order to initialise it as follow:
     */
    state s;
    init(&s);
    printf("Initialised, current state: %s \n-------\n", MachineState[s.
      current_state]);

    /*
     * Example generated in order to call all the functions
     * with the use of a menu to select the required function
```

```
 * (this example uses the standard output stream, it is illustrative purposes only)
 */

char *function_name[] = { "click_CLR", "click_digit_0", "click_digit_1", "click_digit_2", "
  click_digit_3", "click_digit_4", "click_digit_5", "click_digit_6", "click_digit_7", "click_digit_8", "click_digit_9", "
  click_editbox_pressure", "click_ESC", "click_hPa", "click_inHg", "click_OK", "click_point", "click_QNH_RADIO"
  , "click_STD_RADIO", "tick" };
int t, i;
printf("List of functions name\n");
for(i = 1; i <= NUM_OF_FUNC; i++){
    printf("%d - %s\n", i, function_name[i-1]);
}
 while(1){
    printf("Enter number of transition to be issued (enter 0 to exit): ");
    if( scanf("%d",&t) != 1 ){
        fprintf( stderr, "Expected a numbers as input\n");
        return 1;
    }
    t--;
    printf("\n");
    if ( (t < 0) || (t >= NUM_OF_FUNC) ){
        printf("Wrong input\n");
        return 1;
    }
    /*
    * It's recommended to call permission function before issuing the transition function as follow
    */
    else if (strcmp(function_name[t], "click_CLR") == 0){
        if (per_click_CLR(&s)){
            click_CLR(&s);
            printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
  s.current_state], MachineState[s.previous_state]);
        }
        else{
            printf("Permission to %s function denied\n", function_name[t]);
        }
    }
    else if (strcmp(function_name[t], "click_digit_0") == 0){
        if (per_click_digit_0(&s)){
            click_digit_0(&s);
            printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
  s.current_state], MachineState[s.previous_state]);
        }
        else{
            printf("Permission to %s function denied\n", function_name[t]);
        }
    }
    else if (strcmp(function_name[t], "click_digit_1") == 0){
        if (per_click_digit_1(&s)){
            click_digit_1(&s);
            printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
  s.current_state], MachineState[s.previous_state]);
        }
        else{
            printf("Permission to %s function denied\n", function_name[t]);
        }
    }
    else if (strcmp(function_name[t], "click_digit_2") == 0){
        if (per_click_digit_2(&s)){
            click_digit_2(&s);
            printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
  s.current_state], MachineState[s.previous_state]);
        }
        else{
            printf("Permission to %s function denied\n", function_name[t]);
        }
    }
    else if (strcmp(function_name[t], "click_digit_3") == 0){
        if (per_click_digit_3(&s)){
            click_digit_3(&s);
            printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
  s.current_state], MachineState[s.previous_state]);
        }
        else{
            printf("Permission to %s function denied\n", function_name[t]);
        }
    }
    else if (strcmp(function_name[t], "click_digit_4") == 0){
        if (per_click_digit_4(&s)){
            click_digit_4(&s);
            printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
  s.current_state], MachineState[s.previous_state]);
        }
        else{
            printf("Permission to %s function denied\n", function_name[t]);
        }
    }
```

```
  else if (strcmp(function_name[t], "click_digit_5") == 0){
      if (per_click_digit_5(&s)){
          click_digit_5(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_digit_6") == 0){
      if (per_click_digit_6(&s)){
          click_digit_6(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_digit_7") == 0){
      if (per_click_digit_7(&s)){
          click_digit_7(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_digit_8") == 0){
      if (per_click_digit_8(&s)){
          click_digit_8(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_digit_9") == 0){
      if (per_click_digit_9(&s)){
          click_digit_9(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_editbox_pressure") == 0){
      if (per_click_editbox_pressure(&s)){
          click_editbox_pressure(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_ESC") == 0){
      if (per_click_ESC(&s)){
          click_ESC(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_hPa") == 0){
      if (per_click_hPa(&s)){
          click_hPa(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
          printf("Permission to %s function denied\n", function_name[t]);
      }
  }
  else if (strcmp(function_name[t], "click_inHg") == 0){
      if (per_click_inHg(&s)){
          click_inHg(&s);
          printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
s.current_state], MachineState[s.previous_state]);
      }
      else{
```

```
                printf("Permission to %s function denied\n", function_name[t]);
            }
        }
        else if (strcmp(function_name[t], "click_OK") == 0){
            if (per_click_OK(&s)){
                click_OK(&s);
                printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
    s.current_state], MachineState[s.previous_state]);
            }
            else{
                printf("Permission to %s function denied\n", function_name[t]);
            }
        }
        else if (strcmp(function_name[t], "click_point") == 0){
            if (per_click_point(&s)){
                click_point(&s);
                printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
    s.current_state], MachineState[s.previous_state]);
            }
            else{
                printf("Permission to %s function denied\n", function_name[t]);
            }
        }
        else if (strcmp(function_name[t], "click_QNH_RADIO") == 0){
            if (per_click_QNH_RADIO(&s)){
                click_QNH_RADIO(&s);
                printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
    s.current_state], MachineState[s.previous_state]);
            }
            else{
                printf("Permission to %s function denied\n", function_name[t]);
            }
        }
        else if (strcmp(function_name[t], "click_STD_RADIO") == 0){
            if (per_click_STD_RADIO(&s)){
                click_STD_RADIO(&s);
                printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
    s.current_state], MachineState[s.previous_state]);
            }
            else{
                printf("Permission to %s function denied\n", function_name[t]);
            }
        }
        else if (strcmp(function_name[t], "tick") == 0){
            if (per_tick(&s)){
                tick(&s);
                printf("Press %s: current state: %s previous state: %s \n", function_name[t], MachineState[
    s.current_state], MachineState[s.previous_state]);
            }
            else{
                printf("Permission to %s function denied\n", function_name[t]);
            }
        }

        else{
            return 1;
        }
    }
    return 0;
}
```

Index