

emucharts\_fcusoftware\_MisraC

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Table of transitions</b>	<b>1</b>
<b>2</b>	<b>Class Index</b>	<b>9</b>
2.1	Class List . . . . .	9
<b>3</b>	<b>File Index</b>	<b>11</b>
3.1	File List . . . . .	11
<b>4</b>	<b>Class Documentation</b>	<b>13</b>
4.1	state Struct Reference . . . . .	13
4.1.1	Detailed Description . . . . .	13
4.1.2	Member Data Documentation . . . . .	13
4.1.2.1	current_state . . . . .	13
4.1.2.2	decimalDigits . . . . .	14
4.1.2.3	display . . . . .	14
4.1.2.4	dispval . . . . .	14
4.1.2.5	editbox_selected . . . . .	14
4.1.2.6	elapse . . . . .	14
4.1.2.7	integerDigits . . . . .	14
4.1.2.8	msg . . . . .	14
4.1.2.9	pointEntered . . . . .	15
4.1.2.10	previous_state . . . . .	15
4.1.2.11	programmedValue . . . . .	15
4.1.2.12	units . . . . .	15

<b>5 File Documentation</b>	<b>17</b>
5.1 Android_emucharts_fcusoftware_MisraC.c File Reference	17
5.1.1 Function Documentation	18
5.1.1.1 click_CLR(JNIEnv *env, jobject callingObject, jobject obj)	18
5.1.1.2 click_digit_0(JNIEnv *env, jobject callingObject, jobject obj)	18
5.1.1.3 click_digit_1(JNIEnv *env, jobject callingObject, jobject obj)	19
5.1.1.4 click_digit_2(JNIEnv *env, jobject callingObject, jobject obj)	19
5.1.1.5 click_digit_3(JNIEnv *env, jobject callingObject, jobject obj)	20
5.1.1.6 click_digit_4(JNIEnv *env, jobject callingObject, jobject obj)	20
5.1.1.7 click_digit_5(JNIEnv *env, jobject callingObject, jobject obj)	21
5.1.1.8 click_digit_6(JNIEnv *env, jobject callingObject, jobject obj)	21
5.1.1.9 click_digit_7(JNIEnv *env, jobject callingObject, jobject obj)	22
5.1.1.10 click_digit_8(JNIEnv *env, jobject callingObject, jobject obj)	22
5.1.1.11 click_digit_9(JNIEnv *env, jobject callingObject, jobject obj)	23
5.1.1.12 click_editbox_pressure(JNIEnv *env, jobject callingObject, jobject obj)	23
5.1.1.13 click_ESC(JNIEnv *env, jobject callingObject, jobject obj)	24
5.1.1.14 click_hPa(JNIEnv *env, jobject callingObject, jobject obj)	25
5.1.1.15 enter(const MachineState newStateLabel)	25
5.1.1.16 init(JNIEnv *env, jobject callingObject, jobject obj)	26
5.1.1.17 leave(const MachineState currentStateLabel)	26
5.1.2 Variable Documentation	26
5.1.2.1 MAX_ELAPSE	26
5.2 Android_emucharts_fcusoftware_MisraC.h File Reference	27
5.2.1 Macro Definition Documentation	30
5.2.1.1 false	30
5.2.1.2 FALSE	30
5.2.1.3 STRING_LENGTH	30
5.2.1.4 true	30
5.2.1.5 TRUE	30
5.2.2 Typedef Documentation	30

5.2.2.1	C_8 . . . . .	30
5.2.2.2	D_64 . . . . .	30
5.2.2.3	UC_8 . . . . .	30
5.2.2.4	UI_32 . . . . .	30
5.2.3	Enumeration Type Documentation . . . . .	30
5.2.3.1	MachineState . . . . .	30
5.2.4	Function Documentation . . . . .	30
5.2.4.1	click_CLR(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	30
5.2.4.2	click_digit_0(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	31
5.2.4.3	click_digit_1(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	31
5.2.4.4	click_digit_2(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	32
5.2.4.5	click_digit_3(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	32
5.2.4.6	click_digit_4(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	33
5.2.4.7	click_digit_5(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	33
5.2.4.8	click_digit_6(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	34
5.2.4.9	click_digit_7(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	34
5.2.4.10	click_digit_8(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	35
5.2.4.11	click_digit_9(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	35
5.2.4.12	click_editbox_pressure(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	35
5.2.4.13	click_ESC(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	36
5.2.4.14	click_hPa(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	36
5.2.4.15	click_inHg(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	37
5.2.4.16	click_OK(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	38
5.2.4.17	click_point(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	38
5.2.4.18	click_QNH_RADIO(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	39
5.2.4.19	click_STD_RADIO(JNIEnv *env, jobject callingObject, jobject obj) . . . . .	39
5.2.4.20	decimalDigits(JNIEnv *env, jobject callingObject) . . . . .	40
5.2.4.21	display(JNIEnv *env, jobject callingObject) . . . . .	40
5.2.4.22	dispval(JNIEnv *env, jobject callingObject) . . . . .	40
5.2.4.23	enter(const MachineState newStateLabel) . . . . .	40

5.2.4.24	<a href="#">Get_1editbox_selected(JNIEnv *env, jobject callingObject)</a>	41
5.2.4.25	<a href="#">Get_1elapse(JNIEnv *env, jobject callingObject)</a>	41
5.2.4.26	<a href="#">init(JNIEnv *env, jobject callingObject, jobject obj)</a>	41
5.2.4.27	<a href="#">integerDigits(JNIEnv *env, jobject callingObject)</a>	41
5.2.4.28	<a href="#">leave(const MachineState currentStateLabel)</a>	41
5.2.4.29	<a href="#">msg(JNIEnv *env, jobject callingObject)</a>	42
5.2.4.30	<a href="#">per_click_CLR(JNIEnv *env, jobject callingObject, jobject obj)</a>	42
5.2.4.31	<a href="#">per_click_digit_0(JNIEnv *env, jobject callingObject, jobject obj)</a>	42
5.2.4.32	<a href="#">per_click_digit_1(JNIEnv *env, jobject callingObject, jobject obj)</a>	43
5.2.4.33	<a href="#">per_click_digit_2(JNIEnv *env, jobject callingObject, jobject obj)</a>	43
5.2.4.34	<a href="#">per_click_digit_3(JNIEnv *env, jobject callingObject, jobject obj)</a>	44
5.2.4.35	<a href="#">per_click_digit_4(JNIEnv *env, jobject callingObject, jobject obj)</a>	44
5.2.4.36	<a href="#">per_click_digit_5(JNIEnv *env, jobject callingObject, jobject obj)</a>	44
5.2.4.37	<a href="#">per_click_digit_6(JNIEnv *env, jobject callingObject, jobject obj)</a>	45
5.2.4.38	<a href="#">per_click_digit_7(JNIEnv *env, jobject callingObject, jobject obj)</a>	45
5.2.4.39	<a href="#">per_click_digit_8(JNIEnv *env, jobject callingObject, jobject obj)</a>	45
5.2.4.40	<a href="#">per_click_digit_9(JNIEnv *env, jobject callingObject, jobject obj)</a>	47
5.2.4.41	<a href="#">per_click_editbox_pressure(JNIEnv *env, jobject callingObject, jobject obj)</a>	47
5.2.4.42	<a href="#">per_click_ESC(JNIEnv *env, jobject callingObject, jobject obj)</a>	48
5.2.4.43	<a href="#">per_click_hPa(JNIEnv *env, jobject callingObject, jobject obj)</a>	48
5.2.4.44	<a href="#">per_click_inHg(JNIEnv *env, jobject callingObject, jobject obj)</a>	48
5.2.4.45	<a href="#">per_click_OK(JNIEnv *env, jobject callingObject, jobject obj)</a>	49
5.2.4.46	<a href="#">per_click_point(JNIEnv *env, jobject callingObject, jobject obj)</a>	49
5.2.4.47	<a href="#">per_click_QNH_RADIO(JNIEnv *env, jobject callingObject, jobject obj)</a>	49
5.2.4.48	<a href="#">per_click_STD_RADIO(JNIEnv *env, jobject callingObject, jobject obj)</a>	50
5.2.4.49	<a href="#">per_tick(JNIEnv *env, jobject callingObject, jobject obj)</a>	50
5.2.4.50	<a href="#">pointEntered(JNIEnv *env, jobject callingObject)</a>	51
5.2.4.51	<a href="#">programmedValue(JNIEnv *env, jobject callingObject)</a>	51
5.2.4.52	<a href="#">tick(JNIEnv *env, jobject callingObject, jobject obj)</a>	51
5.2.4.53	<a href="#">units(JNIEnv *env, jobject callingObject)</a>	52

5.2.5	Variable Documentation	52
5.2.5.1	MAX_ELAPSE	52
5.3	emucharts_fcusoftware_MisraC.c File Reference	52
5.3.1	Function Documentation	53
5.3.1.1	click_CLR(state *st)	53
5.3.1.2	click_digit_0(state *st)	53
5.3.1.3	click_digit_1(state *st)	54
5.3.1.4	click_digit_2(state *st)	54
5.3.1.5	click_digit_3(state *st)	55
5.3.1.6	click_digit_4(state *st)	55
5.3.1.7	click_digit_5(state *st)	56
5.3.1.8	click_digit_6(state *st)	56
5.3.1.9	click_digit_7(state *st)	57
5.3.1.10	click_digit_8(state *st)	57
5.3.1.11	click_digit_9(state *st)	58
5.3.1.12	click_editbox_pressure(state *st)	58
5.3.1.13	click_ESC(state *st)	59
5.3.1.14	enter(MachineState newStateLabel, state *st)	59
5.3.1.15	init(state *st)	60
5.3.1.16	leave(MachineState currentStateLabel, state *st)	60
5.3.2	Variable Documentation	61
5.3.2.1	MAX_ELAPSE	61
5.4	emucharts_fcusoftware_MisraC.h File Reference	61
5.4.1	Macro Definition Documentation	64
5.4.1.1	false	64
5.4.1.2	FALSE	64
5.4.1.3	STRING_LENGTH	64
5.4.1.4	true	64
5.4.1.5	TRUE	64
5.4.2	Typedef Documentation	64

5.4.2.1	C_8 . . . . .	64
5.4.2.2	D_64 . . . . .	64
5.4.2.3	UC_8 . . . . .	64
5.4.2.4	UI_32 . . . . .	64
5.4.3	Enumeration Type Documentation . . . . .	64
5.4.3.1	MachineState . . . . .	64
5.4.4	Function Documentation . . . . .	64
5.4.4.1	click_CLR(state *st) . . . . .	64
5.4.4.2	click_digit_0(state *st) . . . . .	65
5.4.4.3	click_digit_1(state *st) . . . . .	65
5.4.4.4	click_digit_2(state *st) . . . . .	66
5.4.4.5	click_digit_3(state *st) . . . . .	66
5.4.4.6	click_digit_4(state *st) . . . . .	67
5.4.4.7	click_digit_5(state *st) . . . . .	67
5.4.4.8	click_digit_6(state *st) . . . . .	68
5.4.4.9	click_digit_7(state *st) . . . . .	68
5.4.4.10	click_digit_8(state *st) . . . . .	69
5.4.4.11	click_digit_9(state *st) . . . . .	69
5.4.4.12	click_editbox_pressure(state *st) . . . . .	69
5.4.4.13	click_ESC(state *st) . . . . .	70
5.4.4.14	click_hPa(state *st) . . . . .	70
5.4.4.15	click_inHg(state *st) . . . . .	71
5.4.4.16	click_OK(state *st) . . . . .	71
5.4.4.17	click_point(state *st) . . . . .	72
5.4.4.18	click_QNH_RADIO(state *st) . . . . .	72
5.4.4.19	click_STD_RADIO(state *st) . . . . .	73
5.4.4.20	enter(MachineState newStateLabel, state *st) . . . . .	73
5.4.4.21	init(state *st) . . . . .	74
5.4.4.22	leave(MachineState currentStateLabel, state *st) . . . . .	74
5.4.4.23	per_click_CLR(const state *st) . . . . .	74



5.4.4.24	<code>per_click_digit_0(const state *st)</code>	75
5.4.4.25	<code>per_click_digit_1(const state *st)</code>	75
5.4.4.26	<code>per_click_digit_2(const state *st)</code>	75
5.4.4.27	<code>per_click_digit_3(const state *st)</code>	76
5.4.4.28	<code>per_click_digit_4(const state *st)</code>	76
5.4.4.29	<code>per_click_digit_5(const state *st)</code>	76
5.4.4.30	<code>per_click_digit_6(const state *st)</code>	77
5.4.4.31	<code>per_click_digit_7(const state *st)</code>	77
5.4.4.32	<code>per_click_digit_8(const state *st)</code>	77
5.4.4.33	<code>per_click_digit_9(const state *st)</code>	78
5.4.4.34	<code>per_click_editbox_pressure(const state *st)</code>	78
5.4.4.35	<code>per_click_ESC(const state *st)</code>	78
5.4.4.36	<code>per_click_hPa(const state *st)</code>	79
5.4.4.37	<code>per_click_inHg(const state *st)</code>	79
5.4.4.38	<code>per_click_OK(const state *st)</code>	79
5.4.4.39	<code>per_click_point(const state *st)</code>	80
5.4.4.40	<code>per_click_QNH_RADIO(const state *st)</code>	80
5.4.4.41	<code>per_click_STD_RADIO(const state *st)</code>	80
5.4.4.42	<code>per_tick(const state *st)</code>	81
5.4.4.43	<code>tick(state *st)</code>	81
5.4.5	Variable Documentation	81
5.4.5.1	<code>MAX_ELAPSE</code>	81
5.5	<code>main.c</code> File Reference	82
5.5.1	Macro Definition Documentation	82
5.5.1.1	<code>NUM_OF_FUNC</code>	82
5.5.2	Function Documentation	82
5.5.2.1	<code>main()</code>	82
<b>6</b>	<b>Example Documentation</b>	<b>83</b>
6.1	<code>emucharts_fcusoftware_MisraC.c</code>	83
6.2	<code>main.c</code>	91
	<b>Index</b>	<b>95</b>



## 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] 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],[object Object] st->elapsed = MA↔ X_ELAPSE
click_digit_8	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↔ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapsed = MA↔ X_ELAPSE
click_digit_9	EDIT_PRESSURE	EDIT_PRESSURE		st->data_↔ entry = [object Object],[object Object],[object Object],[object Object],[object Object],[object Object] st->elapsed = MA↔ X_ELAPSE
click_editbox_↔ pressure	QNH	EDIT_PRESSURE	! st->editbox_↔ selected	st->editbox_↔ selected = TRUE st->elapsed = MA↔ X_ELAPSE st->data_↔ entry = [object Object],[object Object],[object Object],[object Object],[object Object] st->editbox_↔ selected = FALSE
click_ESC	EDIT_PRESSURE	QNH		st->data_↔ entry = [object Object],[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],[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],[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],[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->elapsed = 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->elapsed == 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:

<a href="#">state</a>	Structure containing labelled states and variables . . . . .	<a href="#">13</a>
-----------------------	--	--------------------



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Android_emucharts_fcusoftware_MisraC.c</a>	17
<a href="#">Android_emucharts_fcusoftware_MisraC.h</a>	27
<a href="#">emucharts_fcusoftware_MisraC.c</a>	52
<a href="#">emucharts_fcusoftware_MisraC.h</a>	61
<a href="#">main.c</a>	82



## 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::elapsed

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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>st</i>	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

<i>newStateLabel</i>	label to update the current state.
<i>st</i>	state structure pointer

## Returns

void

See also

[leave\(\)](#)

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

Initialiser.

Parameters

<i>st</i>	state structure pointer
-----------	-------------------------

Returns

void

Examples:

[main.c](#).

5.1.1.17 void leave ( const MachineState *currentStateLabel* )

'Leave' auxiliary function.

Parameters

<i>currentStateLabel</i>	label to update the previous state.
<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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**

<i>newStateLabel</i>	label to update the current state.
<i>st</i>	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\_1elapsed ( JNIEnv \* *env*, jobject *callingObject* )

Get elapsed function.

Returns

UI\_32

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

Initialiser.

Parameters

<i>st</i>	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

<i>currentStateLabel</i>	label to update the previous state.
<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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\_PRESSURE" to "QNH" when condition [st->elapse == 0] holds,

from "EDIT\_PRESSURE" to "EDIT\_PRESSURE" when condition [st->elapse > 0] holds

Parameters

<i>st</i>	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

<code>st</code>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>newStateLabel</i>	label to update the current state.
<i>st</i>	state structure pointer

## Returns

void

## See also

[leave\(\)](#)

## Examples:

[emucharts\\_fcusoftware\\_MisraC.c.](#)

**5.3.1.15 state init ( state \* st )**

Initialiser.

## Parameters

<i>st</i>	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

<i>currentStateLabel</i>	label to update the previous state.
<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>newStateLabel</i>	label to update the current state.
<i>st</i>	state structure pointer

**Returns**

void

**See also**

[leave\(\)](#)

**5.4.4.21 state init ( state \* st )**

Initialiser.

**Parameters**

<i>st</i>	state structure pointer
-----------	-------------------------

**Returns**

state structure

**5.4.4.22 void leave ( MachineState *currentStateLabel*, state \* st )**

'Leave' auxiliary function.

**Parameters**

<i>currentStateLabel</i>	label to update the previous state.
<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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**

<i>st</i>	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

<i>st</i>	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

<i>st</i>	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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
        debug_print("Action st->elapsed = 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->elapsed = MAX_ELAPSE;
#ifdef DEBUG
    debug_print("Action st->elapsed = 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],[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],[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
#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

Android\_emucharts\_fcusoftware\_MisraC.c, [17](#)

- [click\\_CLR, 18](#)
- [click\\_ESC, 24](#)
- [click\\_digit\\_0, 18](#)
- [click\\_digit\\_1, 19](#)
- [click\\_digit\\_2, 19](#)
- [click\\_digit\\_3, 20](#)
- [click\\_digit\\_4, 20](#)
- [click\\_digit\\_5, 21](#)
- [click\\_digit\\_6, 21](#)
- [click\\_digit\\_7, 22](#)
- [click\\_digit\\_8, 22](#)
- [click\\_digit\\_9, 23](#)
- [click\\_editbox\\_pressure, 23](#)
- [click\\_hPa, 24](#)
- [enter, 25](#)
- [init, 26](#)
- [leave, 26](#)
- [MAX\\_ELAPSE, 26](#)

Android\_emucharts\_fcusoftware\_MisraC.h, [27](#)

- [C\\_8, 30](#)
- [click\\_CLR, 30](#)
- [click\\_ESC, 36](#)
- [click\\_OK, 37](#)
- [click\\_QNH\\_RADIO, 38](#)
- [click\\_STD\\_RADIO, 39](#)
- [click\\_digit\\_0, 31](#)
- [click\\_digit\\_1, 31](#)
- [click\\_digit\\_2, 32](#)
- [click\\_digit\\_3, 32](#)
- [click\\_digit\\_4, 33](#)
- [click\\_digit\\_5, 33](#)
- [click\\_digit\\_6, 34](#)
- [click\\_digit\\_7, 34](#)
- [click\\_digit\\_8, 34](#)
- [click\\_digit\\_9, 35](#)
- [click\\_editbox\\_pressure, 35](#)
- [click\\_hPa, 36](#)
- [click\\_inHg, 37](#)
- [click\\_point, 38](#)
- [D\\_64, 30](#)
- [decimalDigits, 40](#)
- [display, 40](#)
- [dispval, 40](#)
- [EDIT\\_PRESSURE, 30](#)
- [enter, 40](#)
- [FALSE, 30](#)
- [false, 30](#)
- [Get\\_1editbox\\_selected, 41](#)

[Get\\_1elapsed, 41](#)

- [init, 41](#)
- [integerDigits, 41](#)
- [leave, 41](#)
- [MAX\\_ELAPSE, 52](#)
- [MachineState, 30](#)
- [msg, 42](#)
- [per\\_click\\_CLR, 42](#)
- [per\\_click\\_ESC, 47](#)
- [per\\_click\\_OK, 49](#)
- [per\\_click\\_QNH\\_RADIO, 49](#)
- [per\\_click\\_STD\\_RADIO, 50](#)
- [per\\_click\\_digit\\_0, 42](#)
- [per\\_click\\_digit\\_1, 43](#)
- [per\\_click\\_digit\\_2, 43](#)
- [per\\_click\\_digit\\_3, 43](#)
- [per\\_click\\_digit\\_4, 44](#)
- [per\\_click\\_digit\\_5, 44](#)
- [per\\_click\\_digit\\_6, 45](#)
- [per\\_click\\_digit\\_7, 45](#)
- [per\\_click\\_digit\\_8, 45](#)
- [per\\_click\\_digit\\_9, 47](#)
- [per\\_click\\_editbox\\_pressure, 47](#)
- [per\\_click\\_hPa, 48](#)
- [per\\_click\\_inHg, 48](#)
- [per\\_click\\_point, 49](#)
- [per\\_tick, 50](#)
- [pointEntered, 50](#)
- [programmedValue, 51](#)
- [QNH, 30](#)
- [STRING\\_LENGTH, 30](#)
- [STD, 30](#)
- [TRUE, 30](#)
- [tick, 51](#)
- [true, 30](#)
- [UC\\_8, 30](#)
- [UI\\_32, 30](#)
- [units, 51](#)

C\_8

- [Android\\_emucharts\\_fcusoftware\\_MisraC.h, 30](#)
- [emucharts\\_fcusoftware\\_MisraC.h, 64](#)

click\_CLR

- [Android\\_emucharts\\_fcusoftware\\_MisraC.c, 18](#)
- [Android\\_emucharts\\_fcusoftware\\_MisraC.h, 30](#)
- [emucharts\\_fcusoftware\\_MisraC.c, 53](#)
- [emucharts\\_fcusoftware\\_MisraC.h, 64](#)

click\_ESC

- [Android\\_emucharts\\_fcusoftware\\_MisraC.c, 24](#)
- [Android\\_emucharts\\_fcusoftware\\_MisraC.h, 36](#)

- emucharts\_fcusoftware\_MisraC.c, [59](#)
- emucharts\_fcusoftware\_MisraC.h, [70](#)
- click\_OK
  - Android\_emucharts\_fcusoftware\_MisraC.h, [37](#)
  - emucharts\_fcusoftware\_MisraC.h, [71](#)
- click\_QNH\_RADIO
  - Android\_emucharts\_fcusoftware\_MisraC.h, [38](#)
  - emucharts\_fcusoftware\_MisraC.h, [72](#)
- click\_STD\_RADIO
  - Android\_emucharts\_fcusoftware\_MisraC.h, [39](#)
  - emucharts\_fcusoftware\_MisraC.h, [73](#)
- click\_digit\_0
  - Android\_emucharts\_fcusoftware\_MisraC.c, [18](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [31](#)
  - emucharts\_fcusoftware\_MisraC.c, [53](#)
  - emucharts\_fcusoftware\_MisraC.h, [65](#)
- click\_digit\_1
  - Android\_emucharts\_fcusoftware\_MisraC.c, [19](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [31](#)
  - emucharts\_fcusoftware\_MisraC.c, [54](#)
  - emucharts\_fcusoftware\_MisraC.h, [65](#)
- click\_digit\_2
  - Android\_emucharts\_fcusoftware\_MisraC.c, [19](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [32](#)
  - emucharts\_fcusoftware\_MisraC.c, [54](#)
  - emucharts\_fcusoftware\_MisraC.h, [66](#)
- click\_digit\_3
  - Android\_emucharts\_fcusoftware\_MisraC.c, [20](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [32](#)
  - emucharts\_fcusoftware\_MisraC.c, [55](#)
  - emucharts\_fcusoftware\_MisraC.h, [66](#)
- click\_digit\_4
  - Android\_emucharts\_fcusoftware\_MisraC.c, [20](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [33](#)
  - emucharts\_fcusoftware\_MisraC.c, [55](#)
  - emucharts\_fcusoftware\_MisraC.h, [67](#)
- click\_digit\_5
  - Android\_emucharts\_fcusoftware\_MisraC.c, [21](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [33](#)
  - emucharts\_fcusoftware\_MisraC.c, [56](#)
  - emucharts\_fcusoftware\_MisraC.h, [67](#)
- click\_digit\_6
  - Android\_emucharts\_fcusoftware\_MisraC.c, [21](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [34](#)
  - emucharts\_fcusoftware\_MisraC.c, [56](#)
  - emucharts\_fcusoftware\_MisraC.h, [68](#)
- click\_digit\_7
  - Android\_emucharts\_fcusoftware\_MisraC.c, [22](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [34](#)
  - emucharts\_fcusoftware\_MisraC.c, [57](#)
  - emucharts\_fcusoftware\_MisraC.h, [68](#)
- click\_digit\_8
  - Android\_emucharts\_fcusoftware\_MisraC.c, [22](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [34](#)
  - emucharts\_fcusoftware\_MisraC.c, [57](#)
  - emucharts\_fcusoftware\_MisraC.h, [68](#)
- click\_digit\_9
  - Android\_emucharts\_fcusoftware\_MisraC.c, [23](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [35](#)
  - emucharts\_fcusoftware\_MisraC.c, [58](#)
  - emucharts\_fcusoftware\_MisraC.h, [69](#)
- click\_editbox\_pressure
  - Android\_emucharts\_fcusoftware\_MisraC.c, [23](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [35](#)
  - emucharts\_fcusoftware\_MisraC.c, [58](#)
  - emucharts\_fcusoftware\_MisraC.h, [69](#)
- click\_hPa
  - Android\_emucharts\_fcusoftware\_MisraC.c, [24](#)
  - Android\_emucharts\_fcusoftware\_MisraC.h, [36](#)
  - emucharts\_fcusoftware\_MisraC.h, [70](#)
- click\_inHg
  - Android\_emucharts\_fcusoftware\_MisraC.h, [37](#)
  - emucharts\_fcusoftware\_MisraC.h, [71](#)
- click\_point
  - Android\_emucharts\_fcusoftware\_MisraC.h, [38](#)
  - emucharts\_fcusoftware\_MisraC.h, [72](#)
- current\_state
  - state, [13](#)
- D\_64
  - Android\_emucharts\_fcusoftware\_MisraC.h, [30](#)
  - emucharts\_fcusoftware\_MisraC.h, [64](#)
- decimalDigits
  - Android\_emucharts\_fcusoftware\_MisraC.h, [40](#)
  - state, [13](#)
- display
  - Android\_emucharts\_fcusoftware\_MisraC.h, [40](#)
  - state, [14](#)
- dispal
  - Android\_emucharts\_fcusoftware\_MisraC.h, [40](#)
  - state, [14](#)
- EDIT\_PRESSURE
  - Android\_emucharts\_fcusoftware\_MisraC.h, [30](#)
  - emucharts\_fcusoftware\_MisraC.h, [64](#)
- editbox\_selected
  - state, [14](#)
- elapsed
  - state, [14](#)
- emucharts\_fcusoftware\_MisraC.c, [52](#)
  - click\_CLR, [53](#)
  - click\_ESC, [59](#)
  - click\_digit\_0, [53](#)
  - click\_digit\_1, [54](#)
  - click\_digit\_2, [54](#)
  - click\_digit\_3, [55](#)
  - click\_digit\_4, [55](#)
  - click\_digit\_5, [56](#)
  - click\_digit\_6, [56](#)
  - click\_digit\_7, [57](#)
  - click\_digit\_8, [57](#)
  - click\_digit\_9, [58](#)
  - click\_editbox\_pressure, [58](#)
  - enter, [59](#)
  - init, [60](#)
  - leave, [60](#)
  - MAX\_ELAPSE, [61](#)

emucharts\_fcusoftware\_MisraC.h, 61  
   C\_8, 64  
   click\_CLR, 64  
   click\_ESC, 70  
   click\_OK, 71  
   click\_QNH\_RADIO, 72  
   click\_STD\_RADIO, 73  
   click\_digit\_0, 65  
   click\_digit\_1, 65  
   click\_digit\_2, 66  
   click\_digit\_3, 66  
   click\_digit\_4, 67  
   click\_digit\_5, 67  
   click\_digit\_6, 68  
   click\_digit\_7, 68  
   click\_digit\_8, 68  
   click\_digit\_9, 69  
   click\_editbox\_pressure, 69  
   click\_hPa, 70  
   click\_inHg, 71  
   click\_point, 72  
   D\_64, 64  
   EDIT\_PRESSURE, 64  
   enter, 73  
   FALSE, 64  
   false, 64  
   init, 74  
   leave, 74  
   MAX\_ELAPSE, 81  
   MachineState, 64  
   per\_click\_CLR, 74  
   per\_click\_ESC, 78  
   per\_click\_OK, 79  
   per\_click\_QNH\_RADIO, 80  
   per\_click\_STD\_RADIO, 80  
   per\_click\_digit\_0, 75  
   per\_click\_digit\_1, 75  
   per\_click\_digit\_2, 75  
   per\_click\_digit\_3, 76  
   per\_click\_digit\_4, 76  
   per\_click\_digit\_5, 76  
   per\_click\_digit\_6, 77  
   per\_click\_digit\_7, 77  
   per\_click\_digit\_8, 77  
   per\_click\_digit\_9, 78  
   per\_click\_editbox\_pressure, 78  
   per\_click\_hPa, 79  
   per\_click\_inHg, 79  
   per\_click\_point, 80  
   per\_tick, 80  
   QNH, 64  
   STRING\_LENGTH, 64  
   STD, 64  
   TRUE, 64  
   tick, 81  
   true, 64  
   UC\_8, 64  
   UI\_32, 64

enter  
   Android\_emucharts\_fcusoftware\_MisraC.c, 25  
   Android\_emucharts\_fcusoftware\_MisraC.h, 40  
   emucharts\_fcusoftware\_MisraC.c, 59  
   emucharts\_fcusoftware\_MisraC.h, 73  
  
 FALSE  
   Android\_emucharts\_fcusoftware\_MisraC.h, 30  
   emucharts\_fcusoftware\_MisraC.h, 64  
  
 false  
   Android\_emucharts\_fcusoftware\_MisraC.h, 30  
   emucharts\_fcusoftware\_MisraC.h, 64  
  
 Get\_1editbox\_selected  
   Android\_emucharts\_fcusoftware\_MisraC.h, 41  
  
 Get\_1elapsed  
   Android\_emucharts\_fcusoftware\_MisraC.h, 41  
  
 init  
   Android\_emucharts\_fcusoftware\_MisraC.c, 26  
   Android\_emucharts\_fcusoftware\_MisraC.h, 41  
   emucharts\_fcusoftware\_MisraC.c, 60  
   emucharts\_fcusoftware\_MisraC.h, 74  
  
 integerDigits  
   Android\_emucharts\_fcusoftware\_MisraC.h, 41  
   state, 14  
  
 leave  
   Android\_emucharts\_fcusoftware\_MisraC.c, 26  
   Android\_emucharts\_fcusoftware\_MisraC.h, 41  
   emucharts\_fcusoftware\_MisraC.c, 60  
   emucharts\_fcusoftware\_MisraC.h, 74  
  
 MAX\_ELAPSE  
   Android\_emucharts\_fcusoftware\_MisraC.c, 26  
   Android\_emucharts\_fcusoftware\_MisraC.h, 52  
   emucharts\_fcusoftware\_MisraC.c, 61  
   emucharts\_fcusoftware\_MisraC.h, 81  
  
 MachineState  
   Android\_emucharts\_fcusoftware\_MisraC.h, 30  
   emucharts\_fcusoftware\_MisraC.h, 64  
  
 main  
   main.c, 82  
  
 main.c, 82  
   main, 82  
   NUM\_OF\_FUNC, 82  
  
 msg  
   Android\_emucharts\_fcusoftware\_MisraC.h, 42  
   state, 14  
  
 NUM\_OF\_FUNC  
   main.c, 82  
  
 per\_click\_CLR  
   Android\_emucharts\_fcusoftware\_MisraC.h, 42  
   emucharts\_fcusoftware\_MisraC.h, 74  
  
 per\_click\_ESC  
   Android\_emucharts\_fcusoftware\_MisraC.h, 47  
   emucharts\_fcusoftware\_MisraC.h, 78  
  
 per\_click\_OK

- Android\_emucharts\_fcusoftware\_MisraC.h, 49
- emucharts\_fcusoftware\_MisraC.h, 79
- per\_click\_QNH\_RADIO
  - Android\_emucharts\_fcusoftware\_MisraC.h, 49
  - emucharts\_fcusoftware\_MisraC.h, 80
- per\_click\_STD\_RADIO
  - Android\_emucharts\_fcusoftware\_MisraC.h, 50
  - emucharts\_fcusoftware\_MisraC.h, 80
- per\_click\_digit\_0
  - Android\_emucharts\_fcusoftware\_MisraC.h, 42
  - emucharts\_fcusoftware\_MisraC.h, 75
- per\_click\_digit\_1
  - Android\_emucharts\_fcusoftware\_MisraC.h, 43
  - emucharts\_fcusoftware\_MisraC.h, 75
- per\_click\_digit\_2
  - Android\_emucharts\_fcusoftware\_MisraC.h, 43
  - emucharts\_fcusoftware\_MisraC.h, 75
- per\_click\_digit\_3
  - Android\_emucharts\_fcusoftware\_MisraC.h, 43
  - emucharts\_fcusoftware\_MisraC.h, 76
- per\_click\_digit\_4
  - Android\_emucharts\_fcusoftware\_MisraC.h, 44
  - emucharts\_fcusoftware\_MisraC.h, 76
- per\_click\_digit\_5
  - Android\_emucharts\_fcusoftware\_MisraC.h, 44
  - emucharts\_fcusoftware\_MisraC.h, 76
- per\_click\_digit\_6
  - Android\_emucharts\_fcusoftware\_MisraC.h, 45
  - emucharts\_fcusoftware\_MisraC.h, 77
- per\_click\_digit\_7
  - Android\_emucharts\_fcusoftware\_MisraC.h, 45
  - emucharts\_fcusoftware\_MisraC.h, 77
- per\_click\_digit\_8
  - Android\_emucharts\_fcusoftware\_MisraC.h, 45
  - emucharts\_fcusoftware\_MisraC.h, 77
- per\_click\_digit\_9
  - Android\_emucharts\_fcusoftware\_MisraC.h, 47
  - emucharts\_fcusoftware\_MisraC.h, 78
- per\_click\_editbox\_pressure
  - Android\_emucharts\_fcusoftware\_MisraC.h, 47
  - emucharts\_fcusoftware\_MisraC.h, 78
- per\_click\_hPa
  - Android\_emucharts\_fcusoftware\_MisraC.h, 48
  - emucharts\_fcusoftware\_MisraC.h, 79
- per\_click\_inHg
  - Android\_emucharts\_fcusoftware\_MisraC.h, 48
  - emucharts\_fcusoftware\_MisraC.h, 79
- per\_click\_point
  - Android\_emucharts\_fcusoftware\_MisraC.h, 49
  - emucharts\_fcusoftware\_MisraC.h, 80
- per\_tick
  - Android\_emucharts\_fcusoftware\_MisraC.h, 50
  - emucharts\_fcusoftware\_MisraC.h, 80
- pointEntered
  - Android\_emucharts\_fcusoftware\_MisraC.h, 50
  - state, 14
- previous\_state
  - state, 15
- programmedValue
  - Android\_emucharts\_fcusoftware\_MisraC.h, 51
  - state, 15
- QNH
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- STRING\_LENGTH
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- STD
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- state, 13
  - current\_state, 13
  - decimalDigits, 13
  - display, 14
  - dispv, 14
  - editbox\_selected, 14
  - elapse, 14
  - integerDigits, 14
  - msg, 14
  - pointEntered, 14
  - previous\_state, 15
  - programmedValue, 15
  - units, 15
- TRUE
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- tick
  - Android\_emucharts\_fcusoftware\_MisraC.h, 51
  - emucharts\_fcusoftware\_MisraC.h, 81
- true
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- UC\_8
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- UI\_32
  - Android\_emucharts\_fcusoftware\_MisraC.h, 30
  - emucharts\_fcusoftware\_MisraC.h, 64
- units
  - Android\_emucharts\_fcusoftware\_MisraC.h, 51
  - state, 15