Window Lifter
Title: SW Component

History

Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
1.0	Draft 19-0ct-2017	Luis Archundia	Luis Archundia	Document inception
2.0	Draft 27-0ct-2017	Luis Archundia	Luis Archundia	Document Editing (Functional Description)
3.0	Draft 27-0ct-2017	Luis Archundia	Luis Archundia	Document Editing (Physical Block Diagram)
4.0	Draft 28-0ct-2017	Jorge Olvera	Jorge Olvera	Document Editing (Uses Case Diagram)
5.0	Draft 29-0ct-2017	Jorge Olvera	Jorge Olvera	Document Editing (Software Component Internal Breakdown)
6.0	Draft 29-0ct-2017	Jorge Olvera	Jorge Olvera	Document Editing (Functional Decomposition)
7.0	Draft 29-0ct-2017	Luis Archundia	Luis Archundia	Document Editing (Purpose, Realization, Constraints and Targets)
8.0	Draft 29-0ct-2017	Jorge Olvera	Jorge Olvera	Document Editing (Function description and dynamic behavior)
9.0	Draft 05-Nov-2017	Jorge Olvera	Jorge Olvera	Document Editing (Physical Block Diagram)
10	Draft 08-Nov-2017	Jorge Olvera	Jorge Olvera	Document Editing (Dynamic behavior)
11	Draft 23-Nov-2017	Luis Archundia	Luis Archundia	Document Editing (FSM added)
12	Draft 26-Nov-2017	Jorge Olvera	Jorge Olvera	Document Editing (Physical Block Diagram)
13	Draft 27-Nov-2017	Luis Archundia	Luis Archundia	Document Editing (FSM's element description)

Table of Contents

1 Purpose

To provide an embedded SW solution based on the ARM Cortex M4 Microcontroller along with a hardware functional prototype, capable of emulating the behavior of a car's window lifter module, using an LED bar and NXP's S32K144 Development Kit Platform, implementing basic movement functionalities as well as safety and comfort features. This embedded SW application shall emulate the control of the window's movement in two (2) directions; up and down.

2 Definitions and abbreviations

Definitions

T_UBYTE	Data type unsigned char
T_UWORD	*Data type unsigned int*
T_ULONG	*Data type unsigned long*
S_GPIO_Type	General purpose input output (GPIO) Structure
S_PORT_Type	PORT Structure
S_PCC_Type	Peripheral clock control PCC Structure
S_LPIT_Type	Low power interrup (LPIT) Structure
S_WDOG_Type	Watchdog (WDOG) Structure

Abbreviations

SW	Software
SDK	Software Development Kit (platform)
ms	Miliseconds
MAL	Microcontroller abstraction layer
HAL	Hardware abstraction layer
APP	Application layer
src	Source
io	Input /output
init	initialization
bc	buttonscontrol file .c /.h
wc	windowcontrol file .c / .h
ic	Indicatorcontrol .c / .h
UML	Unified modeling language

References

N°	Document name
1	WindowLifterRequirementsLetter
2	WindowLifter_TraceabilityMatrix
3	Test_Plan_WindowLifter

3 Realization constraints and targets

To create a SW solution along with a functional hardware prototype that is capable of emulating the behavior of a car's window movement along with the use of simple hardware components such as a 10 LED bar.
The solution provided must be developed using the NXP S32K144EVB Development Kit Platform provided.
The solution must consider implementing basic functionality such as upward/downward movement, automatic movement using the provided mechanical interfaces within the platform, as well as safety functionality such as an "Anti-Pinch" functionality and fail proven activation algorithms such as a de-bouncing mechanism (software de-bounce).
The project must also consider timing critical. It is mandatory that it comply within the time boundaries previously agreed in the requirements document.

Two buttons shall indicate window's movement button #1 for upward movement and button #2 for downward movement.

While the RED color of the LED bar will Indicate the window's movement and height. LED bar should turn its LEDs on and off in order to indicate movement. Upward movement (LEDs ON) means window closing, while downward movement (LEDs OFF) indicate window opening.

Well defined state changes should be implemented within the code considering the time between each state transition should be of 400 ms. Movement shall also be indicated with additional LEDs of different colors to indicate direction of movement: BLUE = upward movement and GREEN = downward movement.

An anti-bouncing algorithm shall also be implemented to avoid accidental activation of the module. This anti-bouncing mechanism should be implemented by software means and considering at least 10 ms in order to consider a valid press. Added to this feature, button combinations should be fail proven.

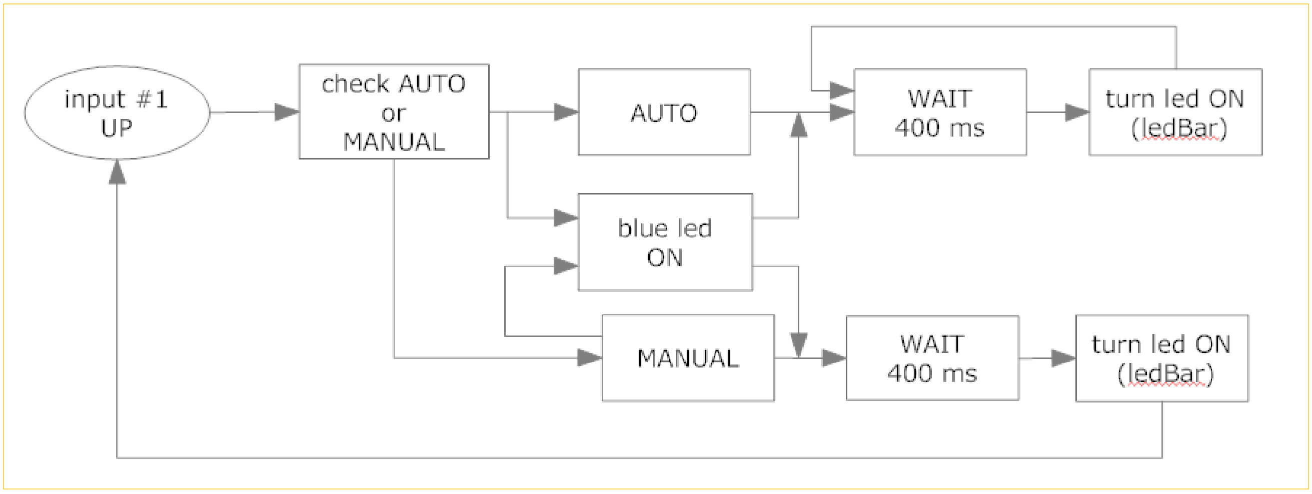
Different behaviors shall be invoked when a valid button press is detected, depending on the time the button was pressed.
Different behaviors to be invoked:

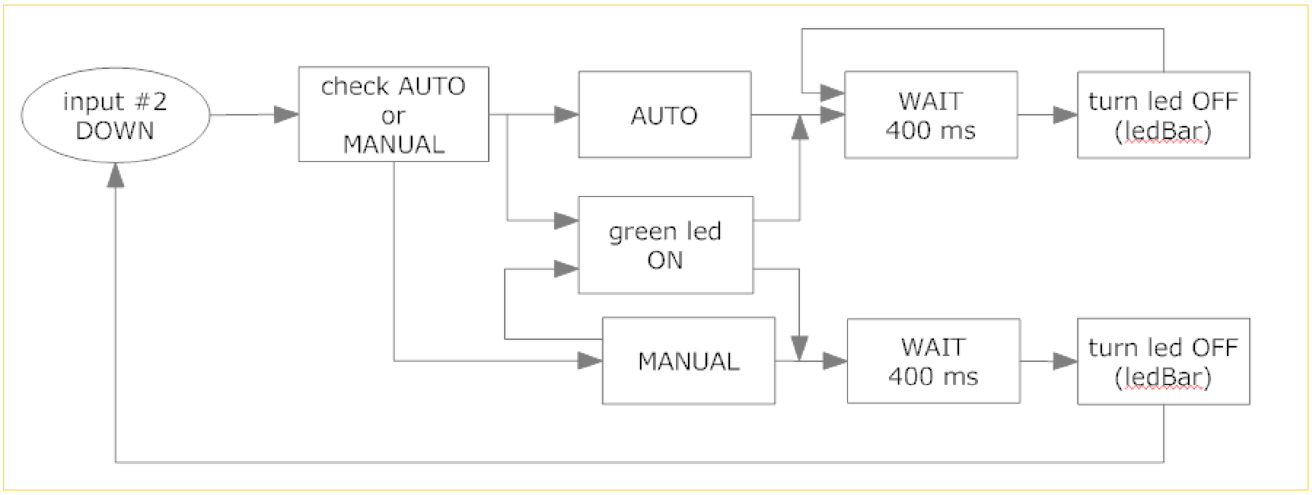
BUTTON # (INPUT)	**BEHAVIOR INVOKED**	**CONDITIONS TO ACTIVATE**
Button #1	Automatic UPWARD movement	10 ms > Button #1 pressed < 500 ms
Button #1	Manual UPWARD movement	Button #1 pressed >= 500 ms
Button #2	Automatic DOWNWARD movement	10 ms > Button #2 pressed < 500 ms
Button #2	Manual DOWNWARD movement	Button #2 pressed >= 500 ms.
Button #3	ANTI-PINCH Functionality	Button #3 pressed > 10 ms & UPWARD movement happening.

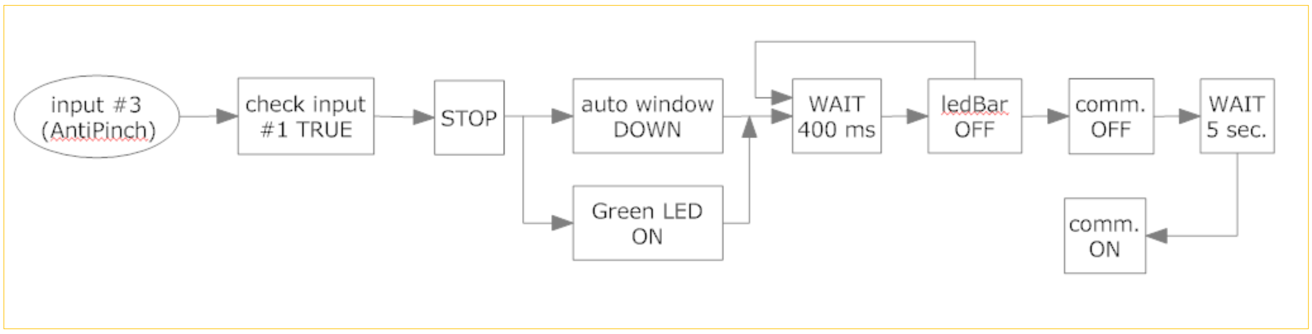
An added functionality shall also be implemented considering the user's safety. The ANTI-PINCH functionality shall be activated by pressing a third button (button #3) as long as the window is moving in upward direction. If the case is true, then the window shall stop movement, and move down automatically until completely open. Then the module shall stop any communication with the user for 5 seconds. After that, communication shall be restored and the module should behave normally.

4 SW Conceptual design

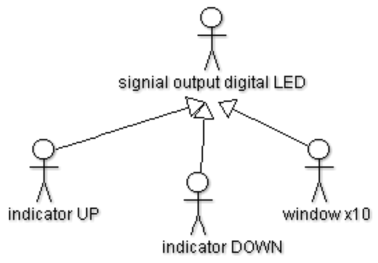
4.1 Block Diagram for conceptual design description





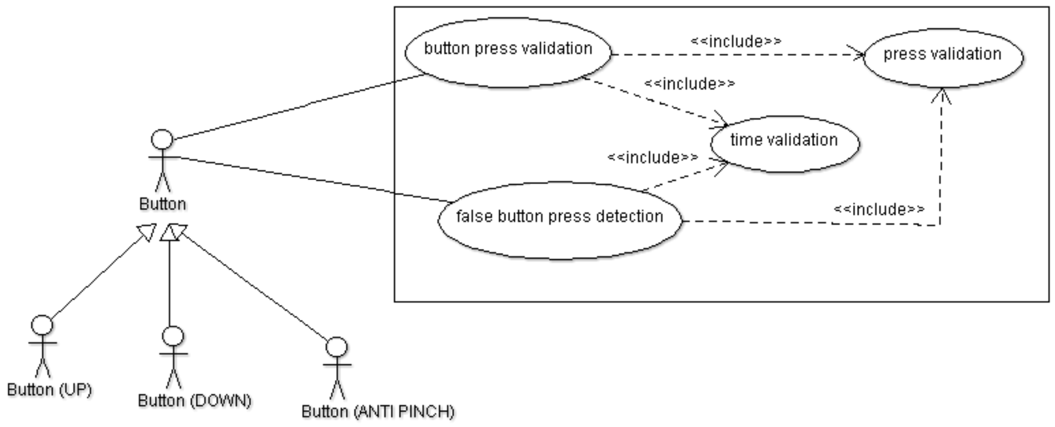


4.2 Uses Case Diagram for requirements analysis



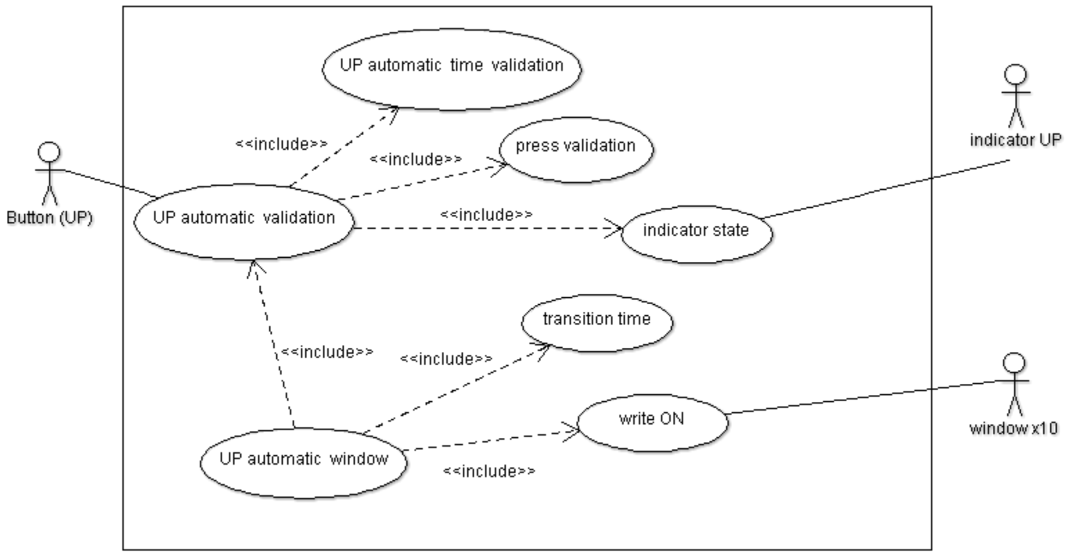
RF01, RF06

All indicator are LEDS
All indicators can have two states (High, Low)



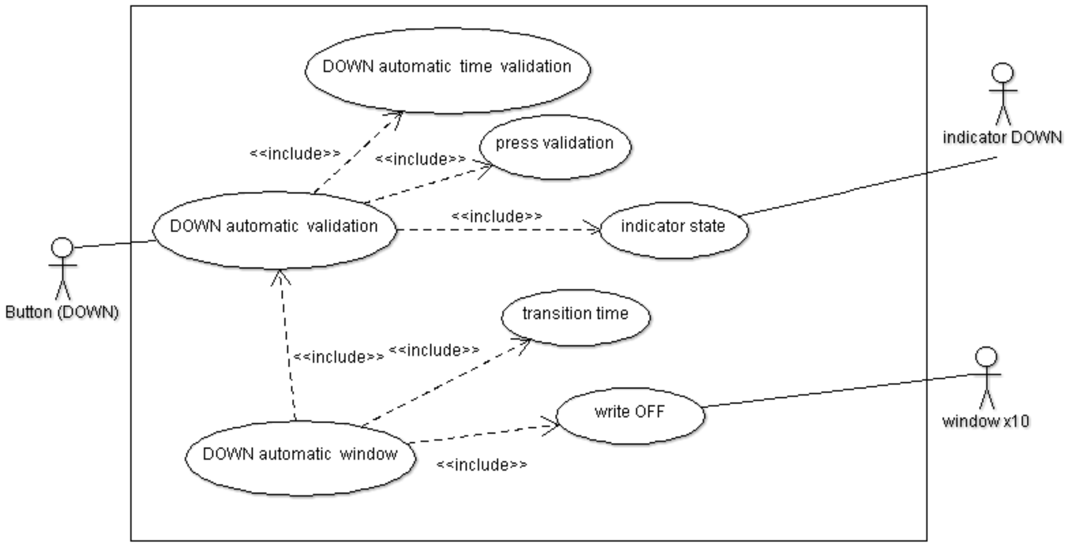
RF02, RF03, RF04, RF05

All pressed buttons have to be validated
All buttons have to be pressed at least 10 msec



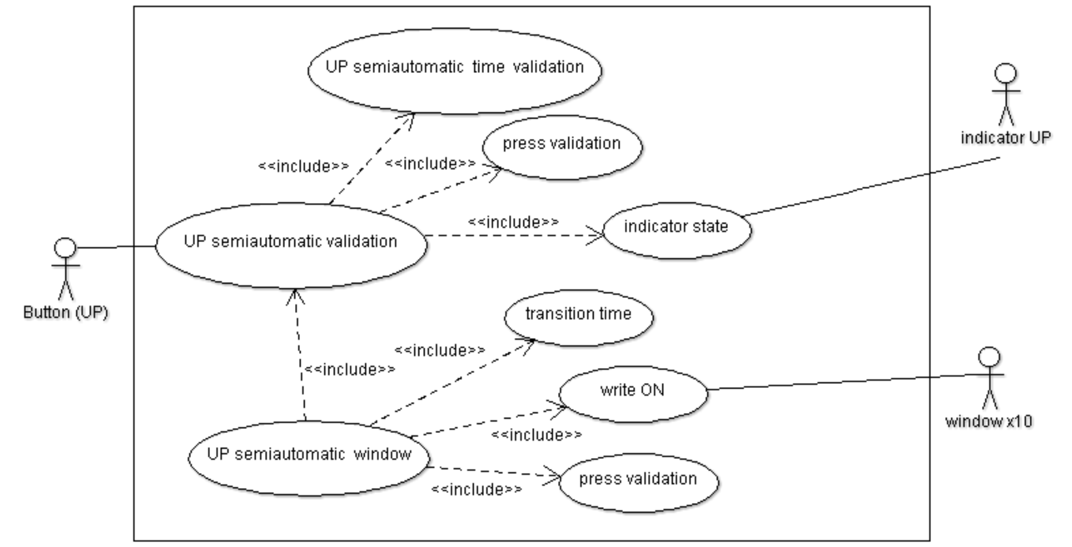
RF05, RF07

UP automatic have to be validated
Window closes automatically
The time between each transition time shall be 400 msc



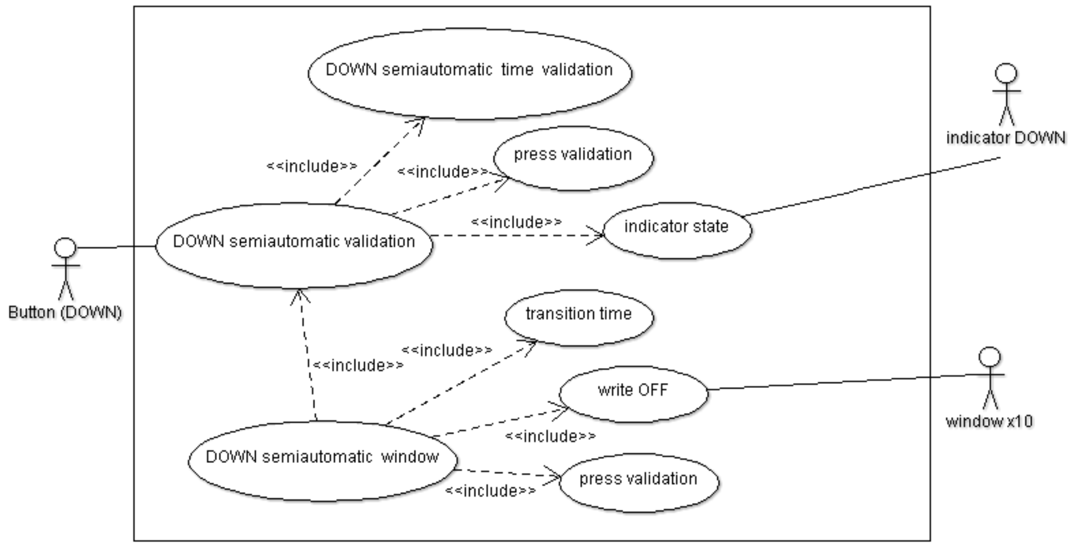
RF05, RF08

DOWN automatic have to be validated
Window opens automatically
The time between each transition time shall be 400 msc



RF05, RF09

UP semiautomatic have to be validated
Window closes semiautomatically
The time between each transition time shall be 400 msc



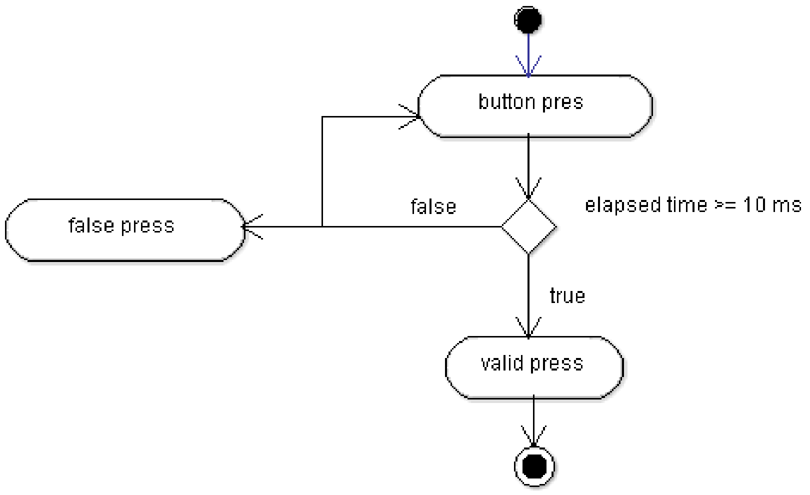
RF05, RF10

DOWN semiautomatic have to be validated
Window opens semiautomatically
The time between each transition time shall be 400 msc

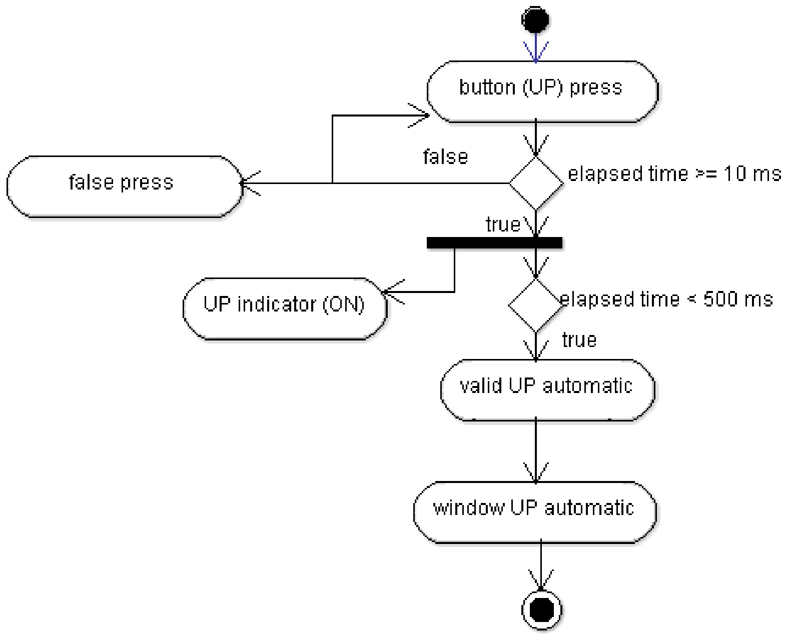
5 SW Component internal breakdown

5.1 Activity diagrams to identify software components

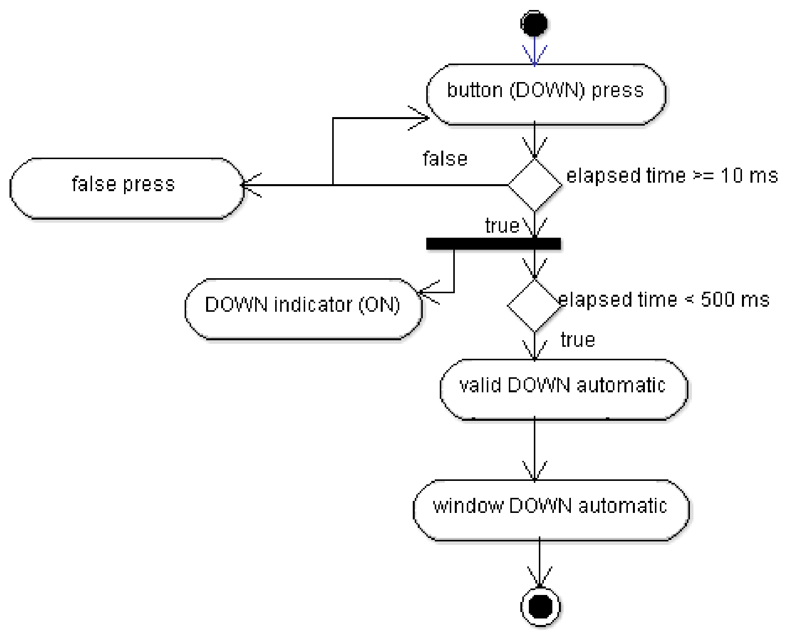
VALIDATE BUTTON PRESS



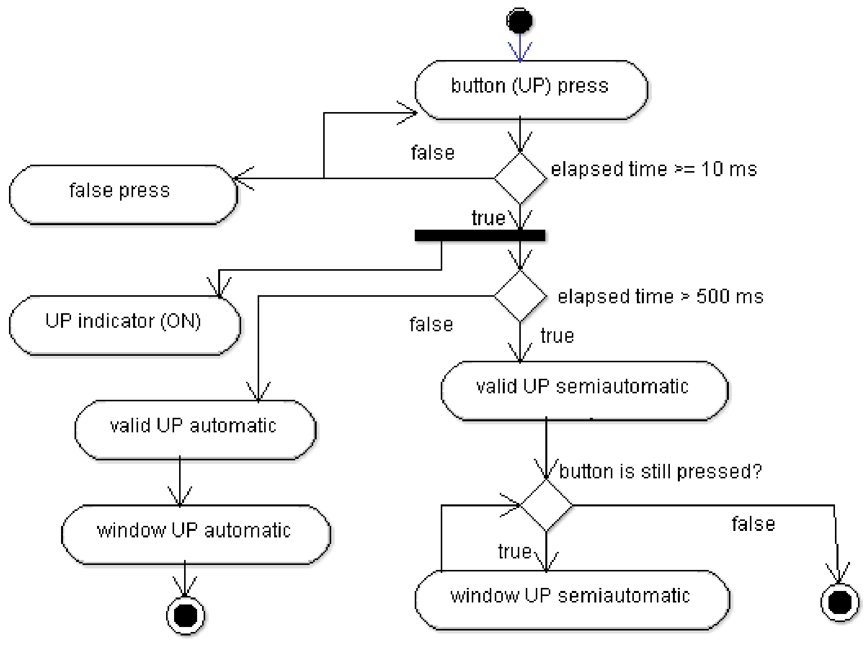
UP AUTOMATIC



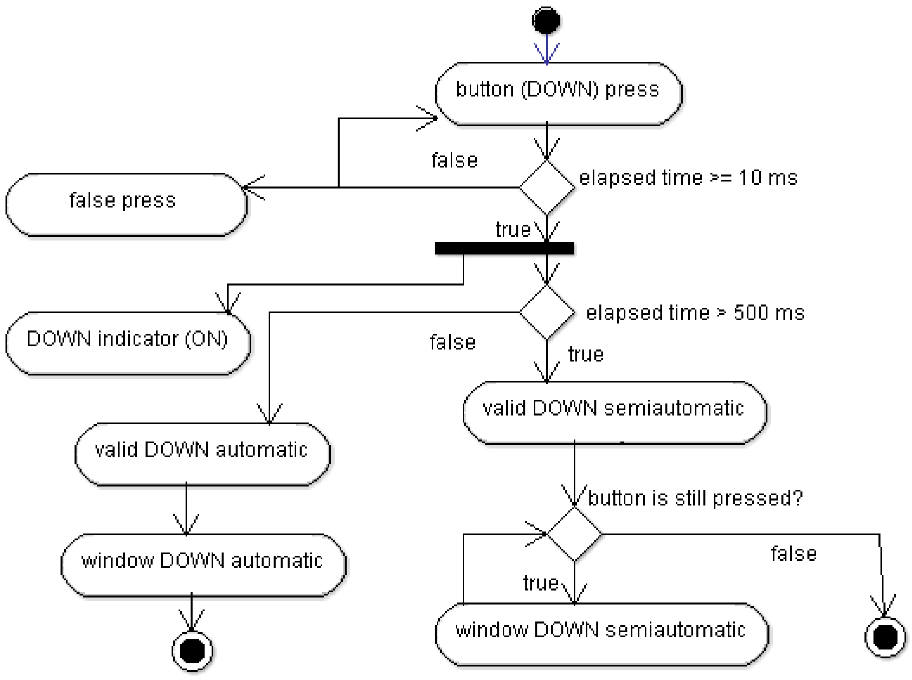
DOWN AUTOMATIC



UP SEMIAUTOMATIC



DOWN SEMIAUTOMATIC



5.2 Finite-State Machine

STATE	STATE NUMBER	DESCRIPTION
WAIT_FOR_A_PULSE	0	Initial state in which the FSM stands by until a signal is generated.
VALIDATE_A_PULSE	1	De-bouncing state. State in which code performs the de-bouncing feature, to get rid of accidental activation.
CHOOSE_TYPE_WORK	2	State in which the module, (based on the conditions provided), decides which feature to activate. Sets respective flags to perform specific functions in the code in addition to setting output indicators on.
AUTOMATIC_WORK	3	State in which code performs specifically the automatic up/down feature. Checks whether the flag set on the last state is either up or down. Then calls the necessary subsequent function(s) to activate the respective feature.
SEMIAUTOMATIC_WORK	4	State in which code performs specifically the semi-automatic up/down feature. Checks whether the flag set on the last state is activated as well as the last signal that set the flag. If both conditions are still true, then calls the necessary subsequent function(s) to activate the respective feature.
ANTIPINCH_WORK	5	This state checks whether the anti-pinch flag is activated. If true, then “wc_WindowDown(lpuw_CounterTime1ms);” function is called, in order to move the window down automatically. *Note that: the anti-pinch flag is only activated if either upward movement features are activated along with the anti-pinch button (sensor).
STOP_5_SECONDS	6	State in which an added safety feature is added as part of the anti-pinch functionality feature. After detecting the window in its lowest level. The module Breaks down communication with the user, remaining idle for 5 seconds. Module reestablishes communication with user after the 5 second idle time. Module returns to its initial state (state 0).

FLAGS	DESCRIPTION
bi1_flagAutomaticUp	Flag critical to validate the condition for the "Automatic upward motion" to be performed.
bi1_flagAutomaticDown	Flag critical to validate the condition for the "Automatic downward motion" to be performed.
bi1_flagSemiautomaticUp	Flag critical to validate the condition for the "Semi-Automatic upward motion" to be performed.
bi1_flagSemiautomaticDown	Flag critical to validate the condition for the "Semi-Automatic downward motion" to be performed.
bi1_flagUp	Flag critical to validate the condition to change from state #1 to state #2 (upward motion)
bi1_flagDown	Flag critical to validate the condition to change from state #1 to state #2 (downward motion)
bi1_flagAntipinch	Flag critical to validate the condition for the "Anti-pinch feature to be performed."

VARIABLES	DESCRIPTION
ruw_CounterTime1ms	Ram, UWORD (2 Bytes) variable to keep track of the time it takes to perform a functionality. Keeps being set and reset in order to comply with the required times provided by the specs.
rub_WindowLevel	Ram, UBYTE (1 Byte) variable to keep track of the window's height percentage, in order to know the highest/lowest level.

INPUTS	DESCRIPTION
bc_T_UBYTE_statusButt_Up()	Gets the button's value for the upward feature. Returns a bit: 1 == TRUE if pressed or 0 == FALSE if not pressed.
bc_T_UBYTE_statusButt_Down()	Gets the button's value for the downward feature. Returns a bit: 1 == TRUE if pressed or 0 == FALSE if not pressed.
bc_T_UBYTE_statusButt_AntiP()	Gets the button's value for the anti-pinch feature. Returns a bit: 1 == TRUE if pressed or 0 == FALSE if not pressed.

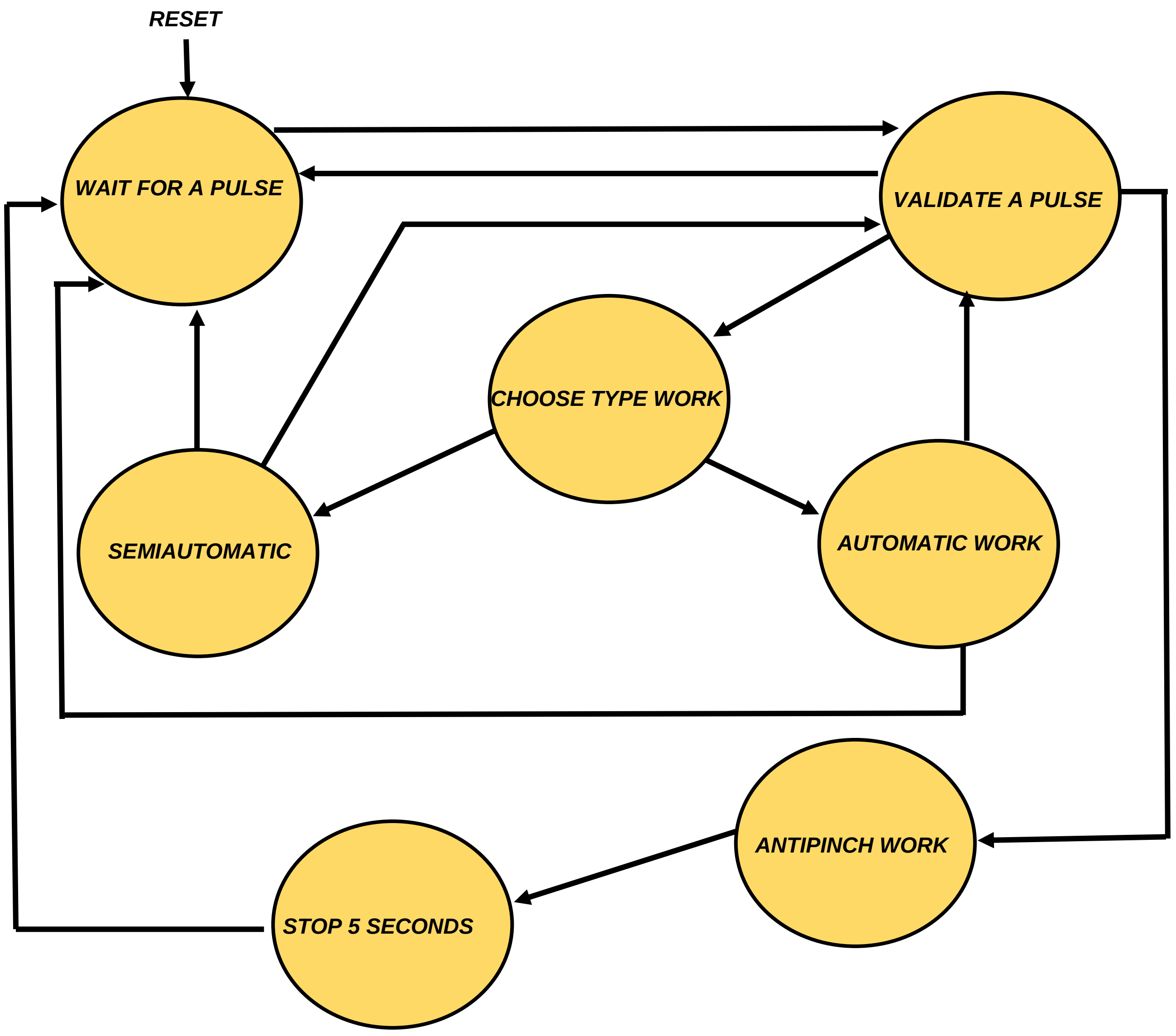
OUTPUTS	DESCRIPTION

ic_void_onIndicator_Up()
ic_void_offIndicator_Up()
	Sets or clears the output's value for the LED indicator (upward motion only) either ON or OFF. Returns a bit: 0 == TRUE if lit or 1 == FALSE if not lit.

ic_void_onIndicator_Down()
ic_void_offIndicator_Down()
	Sets or clears the output's value for the LED indicator (downward motion only) either ON or OFF. Returns a bit: 0 == TRUE if lit or 1 == FALSE if not lit.

wc_void_WindowDown(T_UWORD *lpub_Time)	Keeps track of the window's height and the time taken to move it downwards checks that it complies with the specified times while calling other function to perform the action.

`wc_void_WindowUp(T_UWORD *lpub_Time)`	Keeps track of the window's height and the time taken to move it upwards checks that it complies with the specified times while calling other function to perform the action.



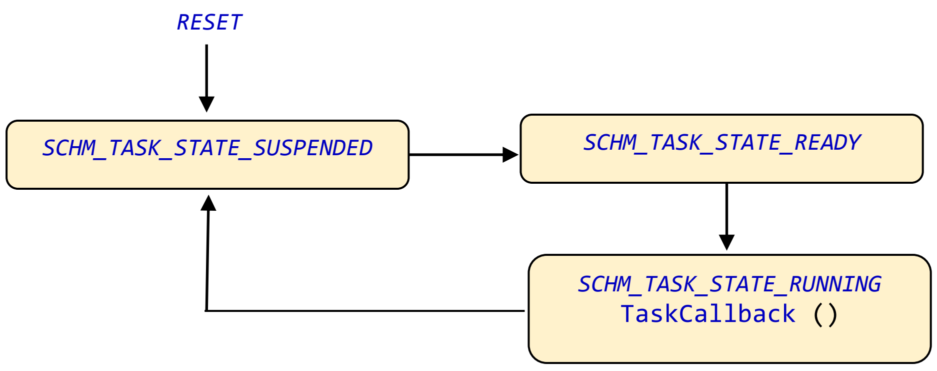
For more details on the code's implementation with FSM please click on the following link:

[Finite State Machine Diagram]

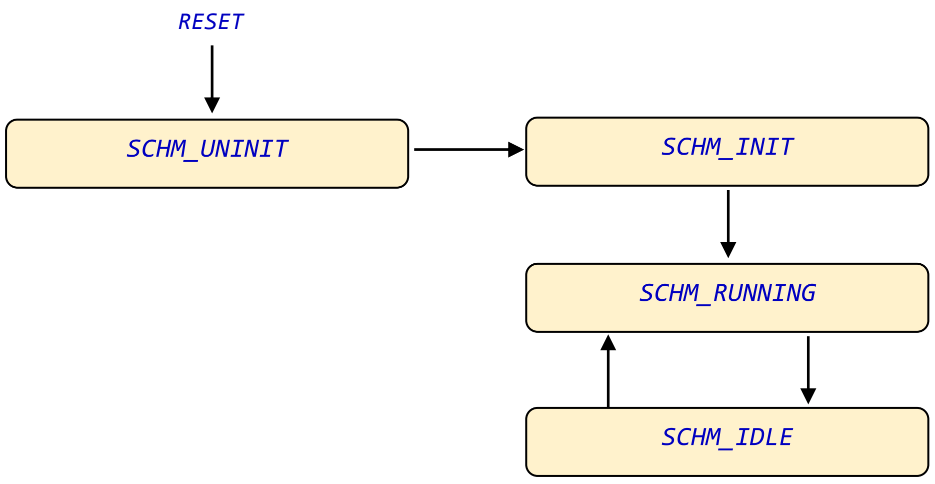
5.3 Binary Progression Scheduler

OSTYCK	TASKS	
500 us	**Task1 (1 millisecond)** : **Mask Task1**: **Offset Task1**:	This task implements the Window lifter´s FSM 0x01 0x00
	Task2 (2 milliseconds) : **Mask Task2**: **Offset Task2**:	This task implements a functions that turns on and turns off a LED 0x03 0x01

Task state and transition

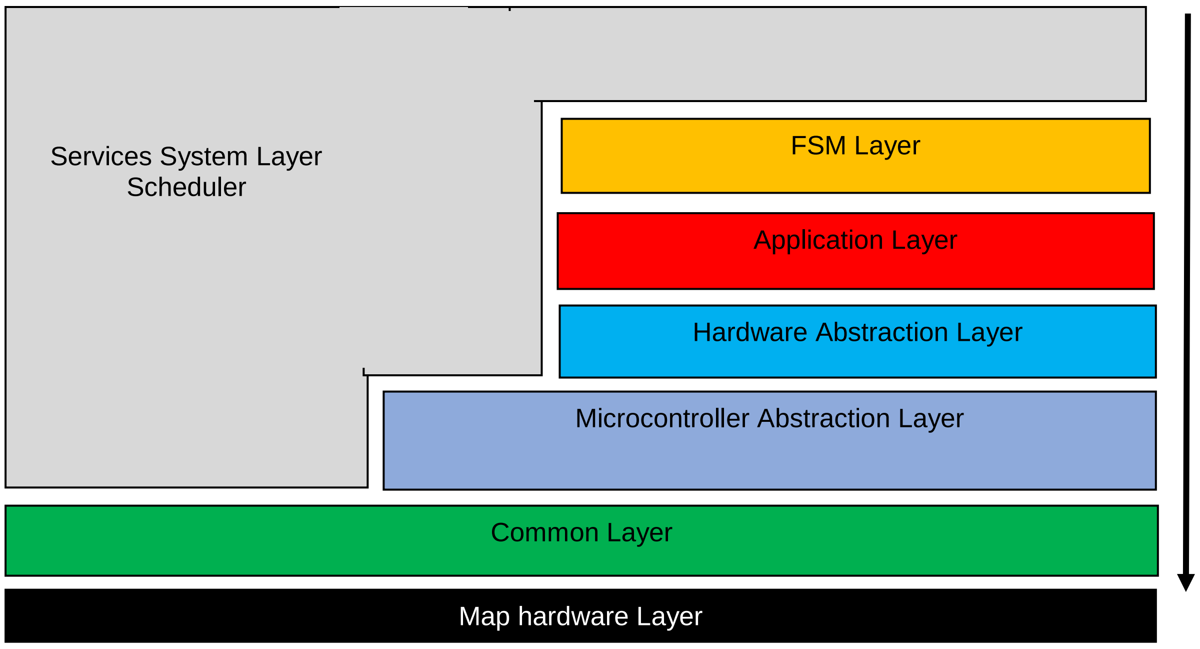


Scheduler state and transition

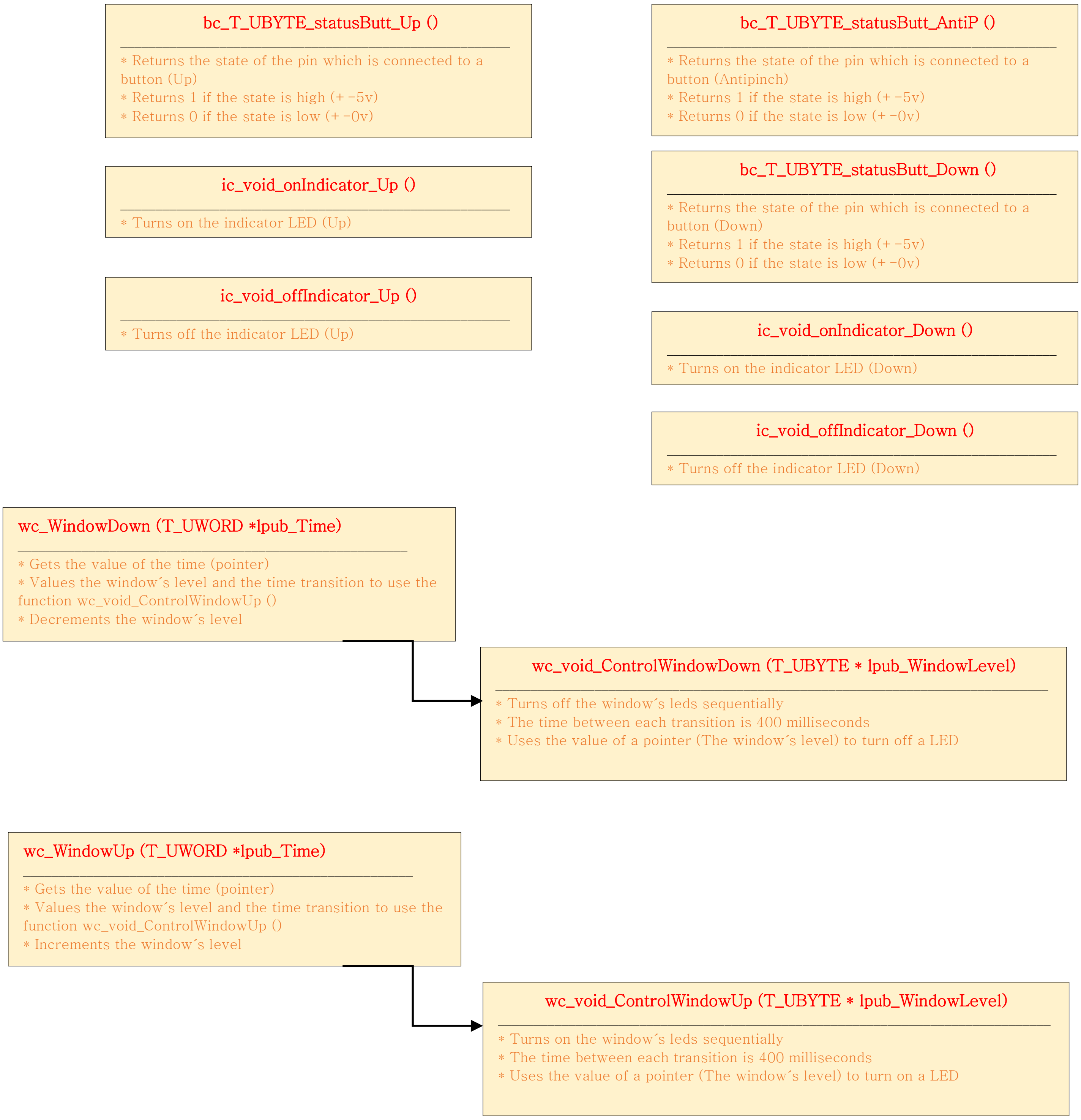


5.4 Physical Block Diagram for Design Description

⬛	Map hardware Layer	S32144k.h
🟩	Common Layer	General.h , Std_Types.h
🟦	Microcontroller Abstraction Layer	Io , Lpit , Mcu , Nvic , Port , Wdog
🟦	Hardware Abstraction Layer	hal_buttons , hal_indicators , hal_windowleds
⬜	Services System Layer (Scheduler)	SchM_Cfg , SchM_Tasks , SchM_Types , SchM
🟥	Application Layer	app_buttonscontrol , app_indicatorscontrol , app_windowcontrol
🟨	FSM Layer	mef_windowlifter



5.5 Functional Decomposition



bc_T_UBYTE_statusButt_Up ()
* Returns the state of the pin which is connected to a button (Up)
* Returns 1 if the state is high (+ -5v)
* Returns 0 if the state is low (+ -0v)

bc_T_UBYTE_statusButt_AntiP ()
* Returns the state of the pin which is connected to a button (Antipinch)
* Returns 1 if the state is high (+ -5v)
* Returns 0 if the state is low (+ -0v)

ic_void_onIndicator_Up ()
* Turns on the indicator LED (Up)

bc_T_UBYTE_statusButt_Down ()
* Returns the state of the pin which is connected to a button (Down)
* Returns 1 if the state is high (+ -5v)
* Returns 0 if the state is low (+ -0v)

ic_void_offIndicator_Up ()
* Turns off the indicator LED (Up)

ic_void_onIndicator_Down ()
* Turns on the indicator LED (Down)

ic_void_offIndicator_Down ()
* Turns off the indicator LED (Down)

wc_WindowDown (T_UWORD *lpub_Time)
* Gets the value of the time (pointer)
* Values the window´s level and the time transition to use the function wc_void_ControlWindowUp ()
* Decrements the window´s level

wc_void_ControlWindowDown (T_UBYTE * lpub_WindowLevel)
* Turns off the window´s leds sequentially
* The time between each transition is 400 milliseconds
* Uses the value of a pointer (The window´s level) to turn off a LED

wc_WindowUp (T_UWORD *lpub_Time)
* Gets the value of the time (pointer)
* Values the window´s level and the time transition to use the function wc_void_ControlWindowUp ()
* Increments the window´s level

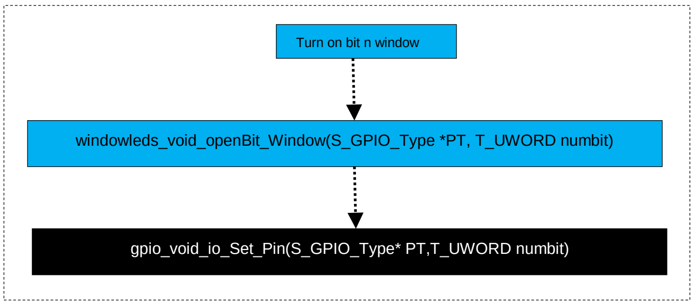
wc_void_ControlWindowUp (T_UBYTE * lpub_WindowLevel)
* Turns on the window´s leds sequentially
* The time between each transition is 400 milliseconds
* Uses the value of a pointer (The window´s level) to turn on a LED

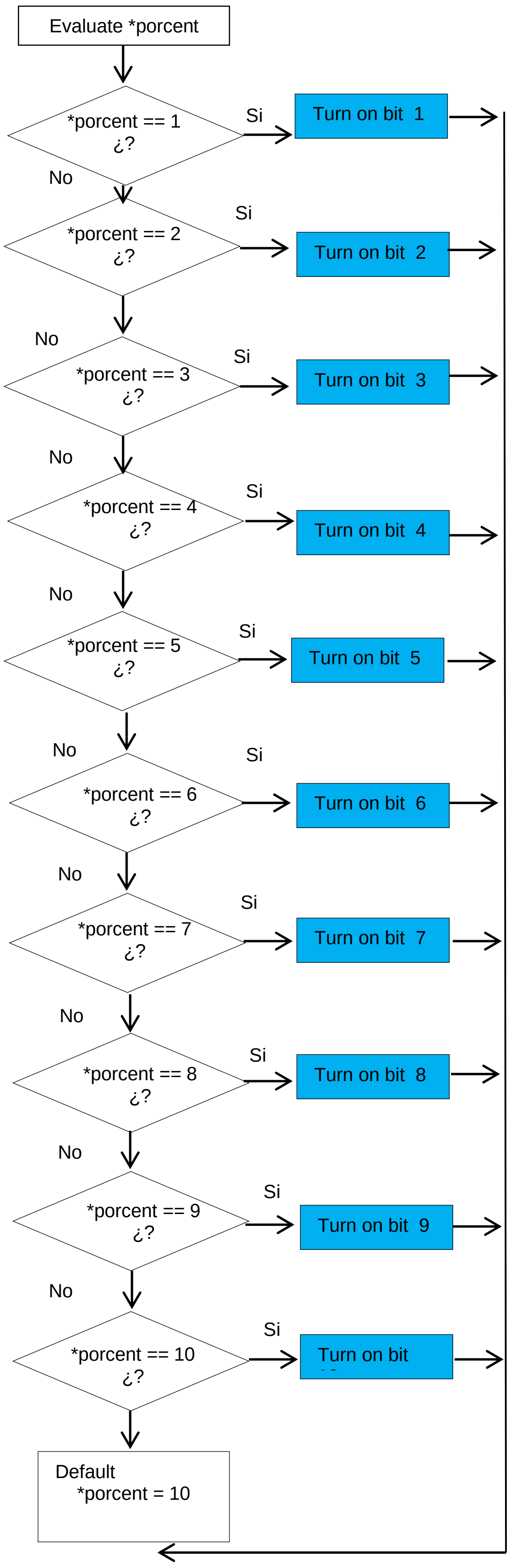
5.6 Function Description and Dynamic Behavior

5.6.1 void wc_void_ControlWindowUp (T_UBYTE * porcent)

Description	*This function controls the up movement, modifies the value of a pointer wich represents the position of the window*
Parameter 1	*Receives a pointer to integer to increment its value.*
Return Value	*void*
Precondition	*the window´s leds off, executed inside main*
Post condition	*the window´s leds on*

Dynamic Behavior

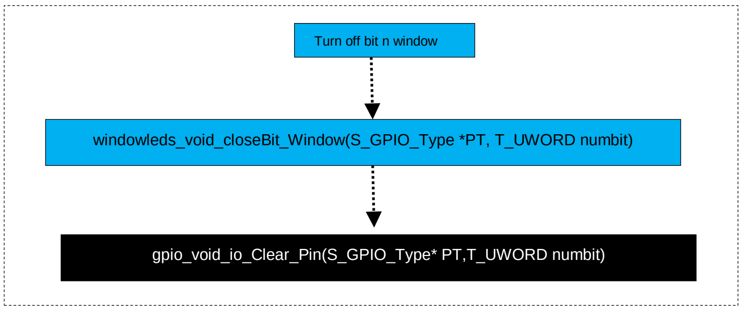

Turn on bit n window
windowleds_void_openBit_Window(S_GPIO_Type *PT, T_UWORD numbit)
gpio_void_io_Set_Pin(S_GPIO_Type* PT,T_UWORD numbit)

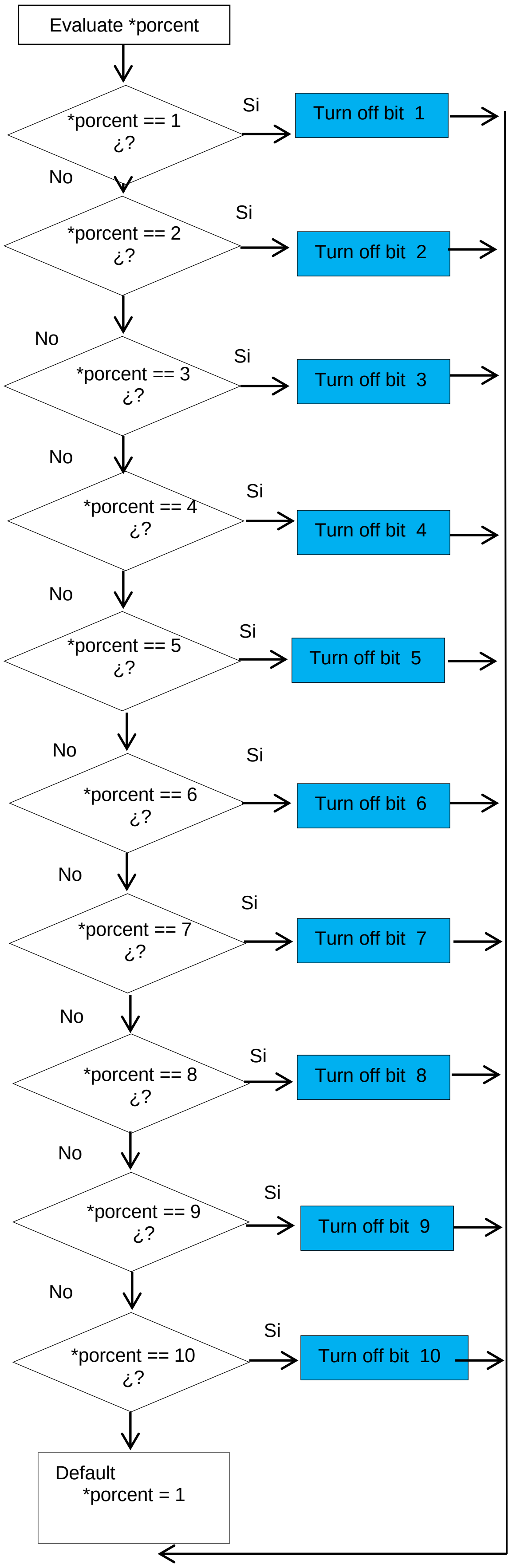

Evaluate *porcent
*porcent == 1 ¿?
Si
Turn on bit 1
No
*porcent == 2 ¿?
Si
Turn on bit 2
No
*porcent == 3 ¿?
Si
Turn on bit 3
No
*porcent == 4 ¿?
Si
Turn on bit 4
No
*porcent == 5 ¿?
Si
Turn on bit 5
No
*porcent == 6 ¿?
Si
Turn on bit 6
No
*porcent == 7 ¿?
Si
Turn on bit 7
No
*porcent == 8 ¿?
Si
Turn on bit 8
No
*porcent == 9 ¿?
Si
Turn on bit 9
No
*porcent == 10 ¿?
Si
Turn on bit
Default
*porcent = 10

5.6.2 void wc_void_ControlWindowDown (T_UBYTE * porcent)

Description	*This function controls the down movement, modifies the value of a pointer wich represents the position of the window*
Parameter 1	*Receives a pointer to integer to decrement its value.*
Return Value	*void*
Precondition	*the window´s leds on, executed inside main*
Post condition	*the window´s leds off*

Dynamic Behavior

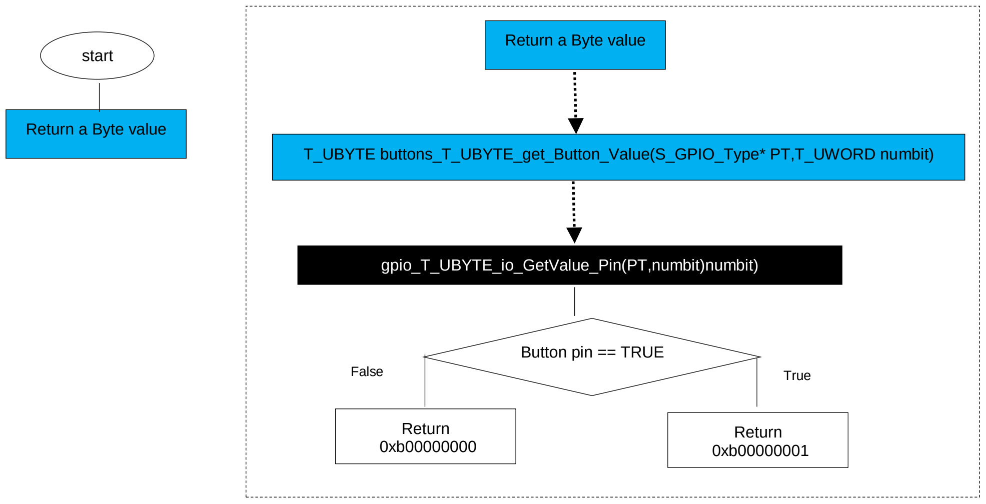

Turn off bit n window
windowleds_void_closeBit_Window(S_GPIO_Type *PT, T_UWORD numbit)
gpio_void_io_Clear_Pin(S_GPIO_Type* PT,T_UWORD numbit)


Evaluate *porcent
*porcent == 1 ¿?
Si
Turn off bit 1
No
*porcent == 2 ¿?
Si
Turn off bit 2
No
*porcent == 3 ¿?
Si
Turn off bit 3
No
*porcent == 4 ¿?
Si
Turn off bit 4
No
*porcent == 5 ¿?
Si
Turn off bit 5
No
*porcent == 6 ¿?
Si
Turn off bit 6
No
*porcent == 7 ¿?
Si
Turn off bit 7
No
*porcent == 8 ¿?
Si
Turn off bit 8
No
*porcent == 9 ¿?
Si
Turn off bit 9
No
*porcent == 10 ¿?
Si
Turn off bit 10
Default
*porcent = 1

5.6.3 T_UBYTE bc_ T_UBYTE _statusButt_Up ()

Description	*This function returns the value 1 (Byte) if the pin receives a high voltage signial*
Parameter 1	*void*
Return Value	*T_UBYTE*
Precondition	*Low voltage singial, executed inside main, called function, connected pin to Up button*
Post condition	*Value 1(Byte) if the signial is high or value 0 (Byte) if the signial is Low*

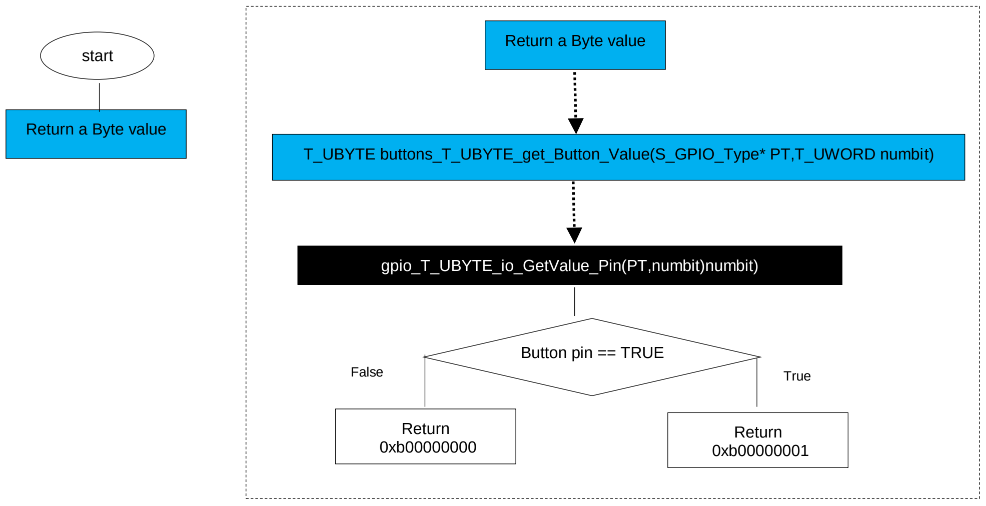
Dynamic Behavior


start
Return a Byte value
Return a Byte value
T_UBYTE buttons_T_UBYTE_get_Button_Value(S_GPIO_Type* PT,T_UWORD numbit)
gpio_T_UBYTE_io_GetValue_Pin(PT,numbit)numbit)
Button pin == TRUE
False
True
Return 0xb00000000
Return 0xb00000001

5.6.4 T_UBYTE bc_ T_UBYTE _statusButt_Down ()

Description	*This function returns the value 1 (Byte) if the pin receives a high voltage signial*
Parameter 1	*void*
Return Value	*T_UBYTE*
Precondition	*Low voltage singial, executed inside main, called function, connected pin to Down button*
Post condition	*Value 1(Byte) if the signial is high or value 0 (Byte) if the signial is Low*

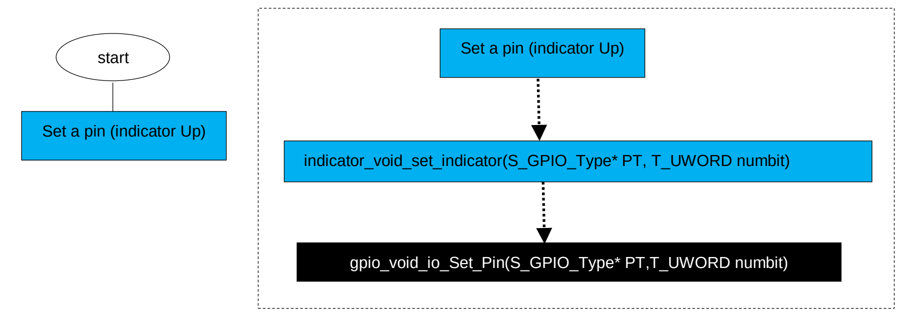
Dynamic Behavior


start
Return a Byte value
Return a Byte value
T_UBYTE buttons_T_UBYTE_get_Button_Value(S_GPIO_Type* PT,T_UWORD numbit)
gpio_T_UBYTE_io_GetValue_Pin(PT,numbit)numbit)
Button pin == TRUE
False
True
Return 0xb00000000
Return 0xb00000001

5.6.5 void ic_void_onIndicator_Up ()

Description	*This function turns on the Up indicator*
Parameter 1	*void*
Return Value	*void*
Precondition	*turned off up indicator,executed inside main*
Post condition	*turned on up indicator*

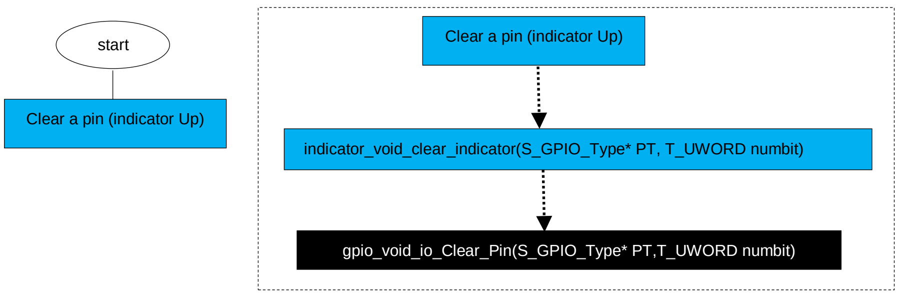
Dynamic Behavior


start
Set a pin (indicator Up)
Set a pin (indicator Up)
indicator_void_set_indicator(S_GPIO_Type* PT, T_UWORD numbit)
gpio_void_io_Set_Pin(S_GPIO_Type* PT,T_UWORD numbit)

5.6.6 void ic_void_offIndicator_Up ()

Description	*This function turns off the Up indicator*
Parameter 1	*void*
Return Value	*void*
Precondition	*turned on up indicator, executed inside main*
Post condition	*turned off up indicator*

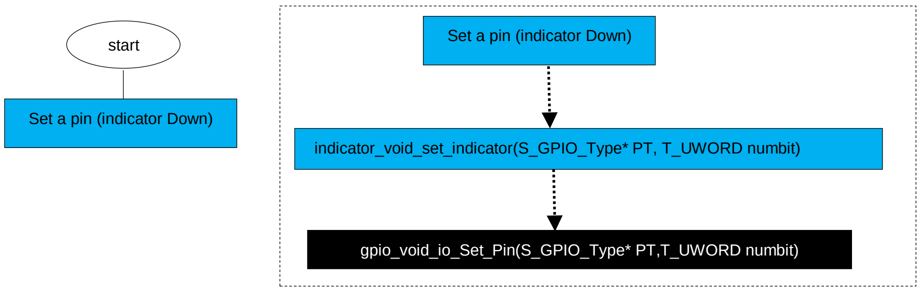
Dynamic Behavior


start
Clear a pin (indicator Up)
Clear a pin (indicator Up)
indicator_void_clear_indicator(S_GPIO_Type* PT, T_UWORD numbit)
gpio_void_io_Clear_Pin(S_GPIO_Type* PT,T_UWORD numbit)

5.6.7 void ic_void_onIndicator_Down ()

Description	*This function turns on the Down indicator*
Parameter 1	*void*
Return Value	*void*
Precondition	*turned off Down indicator, executed inside main*
Post condition	*turned on Down indicator*

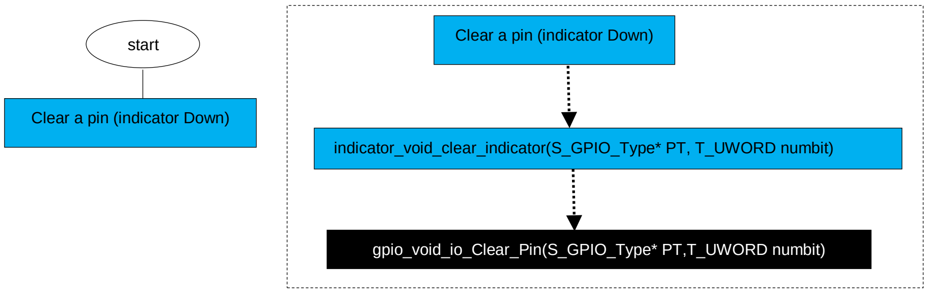
Dynamic Behavior


start
Set a pin (indicator Down)
Set a pin (indicator Down)
indicator_void_set_indicator(S_GPIO_Type* PT, T_UWORD numbit)
gpio_void_io_Set_Pin(S_GPIO_Type* PT,T_UWORD numbit)

5.6.8 void ic_void_offIndicator_Down ()

Description	*This function turns off the Down indicator*
Parameter 1	*void*
Return Value	*void*
Precondition	*turned on Down indicator, executed inside main*
Post condition	*turned off Down indicator*

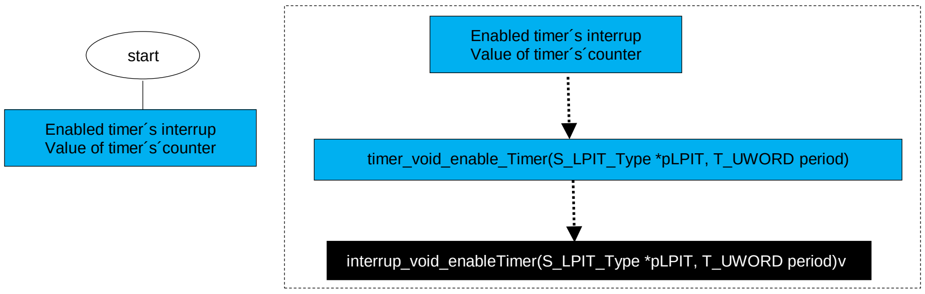
Dynamic Behavior


start
Clear a pin (indicator Down)
Clear a pin (indicator Down)
indicator_void_clear_indicator(S_GPIO_Type* PT, T_UWORD numbit)
gpio_void_io_Clear_Pin(S_GPIO_Type* PT,T_UWORD numbit)

5.6.9 void tmr1ms_void_EnableTimer1ms ()

Description	*This function enables the timer´s interrup and assigns a value to the timer´s counter*
Parameter 1	*void*
Return Value	*void*
Precondition	*Disabled interrup, executed inside main*
Post condition	*Enabled interrup, counter decrements its value*

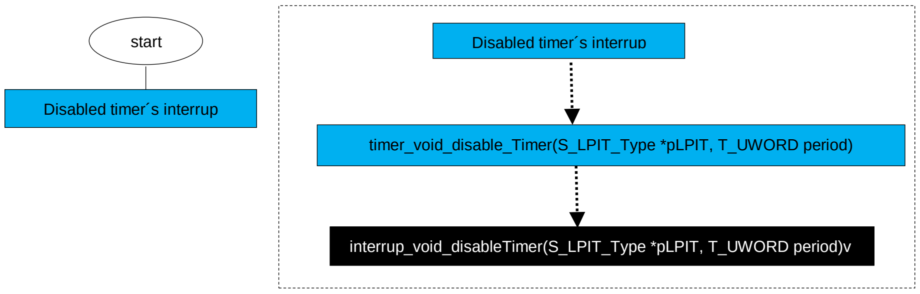
Dynamic Behavior


start
Enabled timer´s interrup
Value of timer´s´counter
Enabled timer´s interrup
Value of timer´s´counter
timer_void_enable_Timer(S_LPIT_Type *pLPIT, T_UWORD period)
interrup_void_enableTimer(S_LPIT_Type *pLPIT, T_UWORD period)v

5.6.10 void tmr1ms_void_DisableTimer1ms ()

Description	*This function disables the timer´s interrup*
Parameter 1	*void*
Return Value	*void*
Precondition	*Enabled interrup, executed inside main*
Post condition	*Disabled interrup*

Dynamic Behavior


start
Disabled timer´s interrup
Disabled timer´s interrup
timer_void_disable_Timer(S_LPIT_Type *pLPIT, T_UWORD period)
interrup_void_disableTimer(S_LPIT_Type *pLPIT, T_UWORD period)v