Title: Window lifter
SW Component WiLi

History				
Issue status (Index)	**Maturity/Date** (draft/invalid/valid) (dd-mmm-yyyy)	**Author** Department	**Check/Release** Department	**Description**
1.0	Draft 27-0ct-15	Edgar Escayola	Edgar Escayola	Creation of the document
1.1	Draft 31-10-15	Adrián Zacarías	Adrián Zacarías	Changes in purpose and constrains.
1.2	Draft 01-11-15	Edgar Escayola	Edgar Escayola	Functions description.
1.3	Draft 01-11-15	Edgar Escayola	Adrián Zacarías	Diagrams
1.4	Draft 16-11-15	Edgar Escayola	Adrián Zacarías	Updated diagrams: State diagram, architecture, use case and deployment.
1.5	Draft 18-11-15	Adrián Zacarías	Edgar Escayola	Changes in realization constrains, definitions and architecture.
1.6	Draft 19-11-15	Adrián Zacarías	Edgar Escayola	Creation of the sequence diagram.
1.7	Draft 20-11-15	Edgar Escayola	Adrián Zacarías	Update functions and requirements id.
1.8	Draft 22-11-15	Adrián Zacarías	Edgar Escayola	Functions and diagrams updated.
1.9	Draft 23-11-15	Edgar Escayola	Adrián Zacarías	Review changes implemented: Changes in purpose and functional decomposition diagram.

Table of Contents

1 Purpose

This document describes the design that should be implemented to fulfill the requirements of the project Window Lifter (WiLi).

Window lifter is the module responsible to control the window movement. It is controlled by two switches that indicate the direction of the window movement.

Req. id: 1.0, 1.1, 1.2

2 Definitions and abbreviations

ADC	–	Analog to Digital Converter
CAN	–	Controller Area Network
DC	–	Direct Current
JTAG	–	Joint Test Action Group
LED	–	Light Emitting Diode
LIN	–	Local Interconnect Network
LQFP	–	Low Profile Quad Flat Pack
MCU	–	Microcontroller Unit
msec	–	Microsecond (1E-3 seconds)
OSBDM	–	Open Source Background Debug Mode

References

N°	Document name	Reference	Revision
1	Window lifter requirements	WiLi/Documents/V-Cycle Process/1.0 Requirements/Window lifter requirements.doc	1.0
2	Traceability Matrix Template	WiLi/Documents/V-Cycle Process/1.0 Requirements/Traceability Matrix Template.xls	1.0
3	MPC5607B Microcontroller Reference Manual	WiLi/Documents/MPC5607BRM_Reference_Manual.pdf	7.2
4	Quick Start Guide TRK-MPC5606B	WiLi/Documents/Quick_Start_Guide.pdf	3

3 Realization constraints and targets

3.1 TRK-MPC5606B's features

- MPC5606B MCU (144-pin LQFP).
- On-board JTAG connection via open source OSBDM circuit using the MPC9S08JM MCU.
- MCZ3390S5EK system basis chip with advanced power management and integrated CAN transceiver and LIN 2.0 interface.
- CAN interface.
- LIN interface.
- Analog interface with potentiometer.
- High-efficency LEDs.
- 4 PushButtons.
- Serial communication interface.
- External power 9V DC to 12V DC regulated down to 5V DC.

3.2 External components

- 1 - 10 red LEDs bar.
- 10 - 470 Ohms resistors.
- 1 - 5mm RGB LED.
- 14 - Jumpers Male – Female.
- 1 - Protoboard

Req. Id. 2.2.

3.3 General behavior

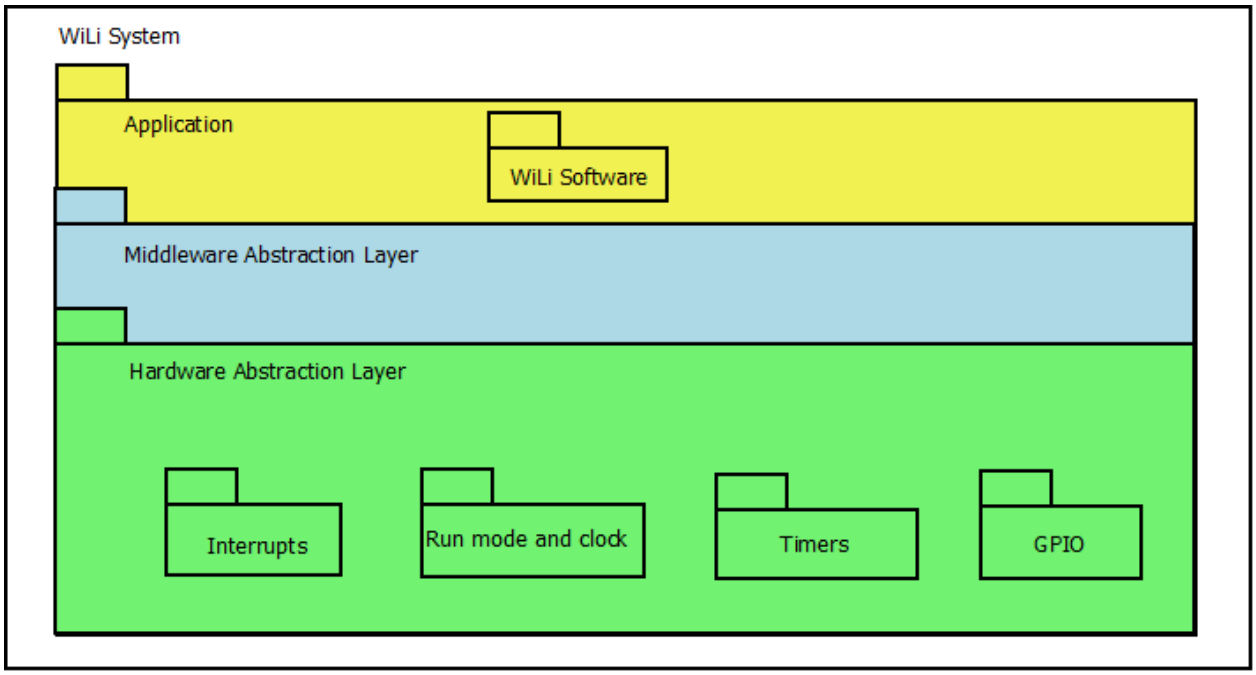
Window behavior: The window movements (UP and DOWN) are emulated using a 10 led bar. Req. Id. 2.0, 2.1, 2.6

Button Behavior: The button must be pressed for at least 10 msec in order to be a valid. Req. Id. 3.2

Anti pinch functionality: Anti pinch is a feature than prevents accidents between window and some human body parts like arms, hands, and head. Req. Id. 4.1

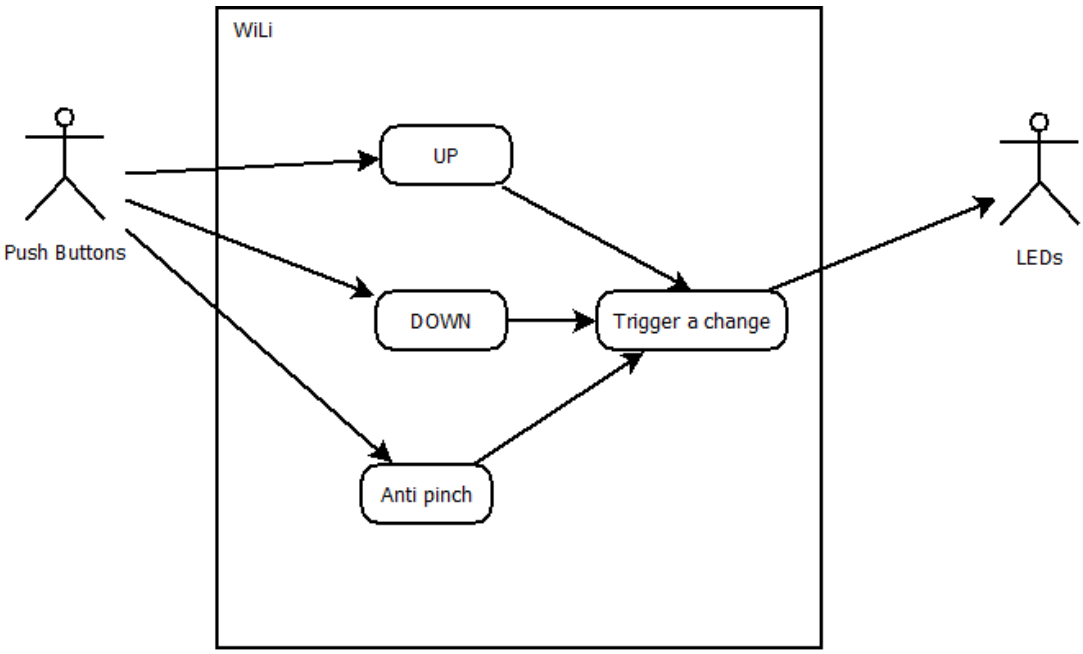
4 SW Conceptual design

4.1 Architecture design



The architecture diagram of the design shows the layers in which the system is divided. In the Hardware Abstraction Layer, there are the modules where the interrupts, clocks, timers and GPIO are handled. In the Middleware Abstraction Layer, there are not modules. In the Application Layer, the main software of the Window Lifter project is executed.

4.2 Use case diagram

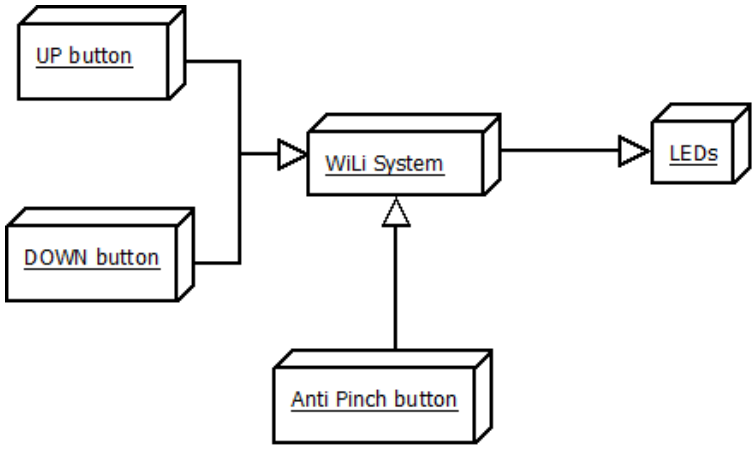


Req. Id. 1.0, 1.1, 2.3, 2.6, 2.7.

Users	Description
Push Buttons	There are three physical buttons that generates signals, which are read by the WiLi system.
LEDs	There are two LEDs and a LED bar. These are activated by the system.

Case	Description
UP	This case triggers the window to go UP and evaluates if it should go in the automatic or manual mode.
DOWN	This case triggers the window to go DOWN and evaluates if it should go in the automatic or manual mode.
Anti pinch	This case evaluates if the window goes UP in order to stop it and bring it down completely.
Trigger a change	This case executes the change evaluated by the cases before in order to send a signal to the LEDs.

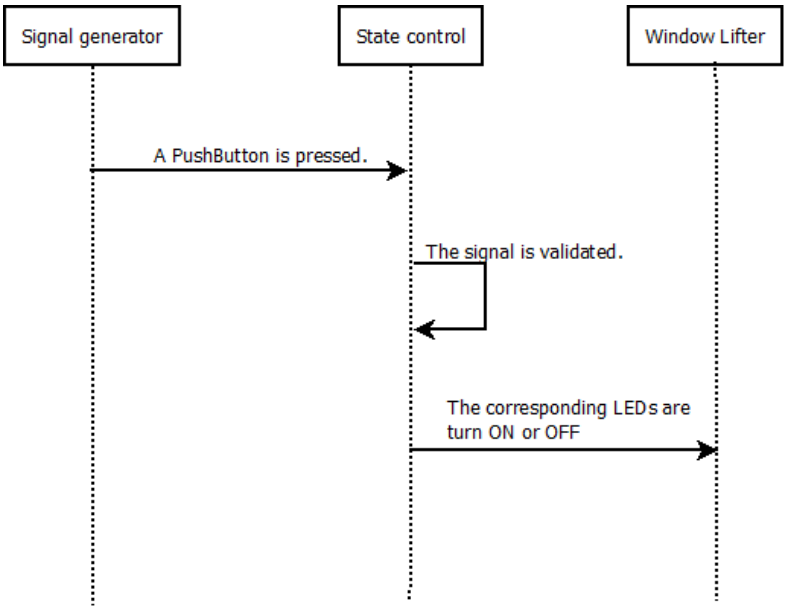
4.3 Deployment Diagram



Req. Id. 1.2, 4.3.

The deployment diagram shows the interaction of the system. There are three inputs: UP button, DOWN button, and Anti Pinch button. These modules generate changes that are shown directly in the output. The only outputs of this system are the signals that activate the different LEDs in the system.

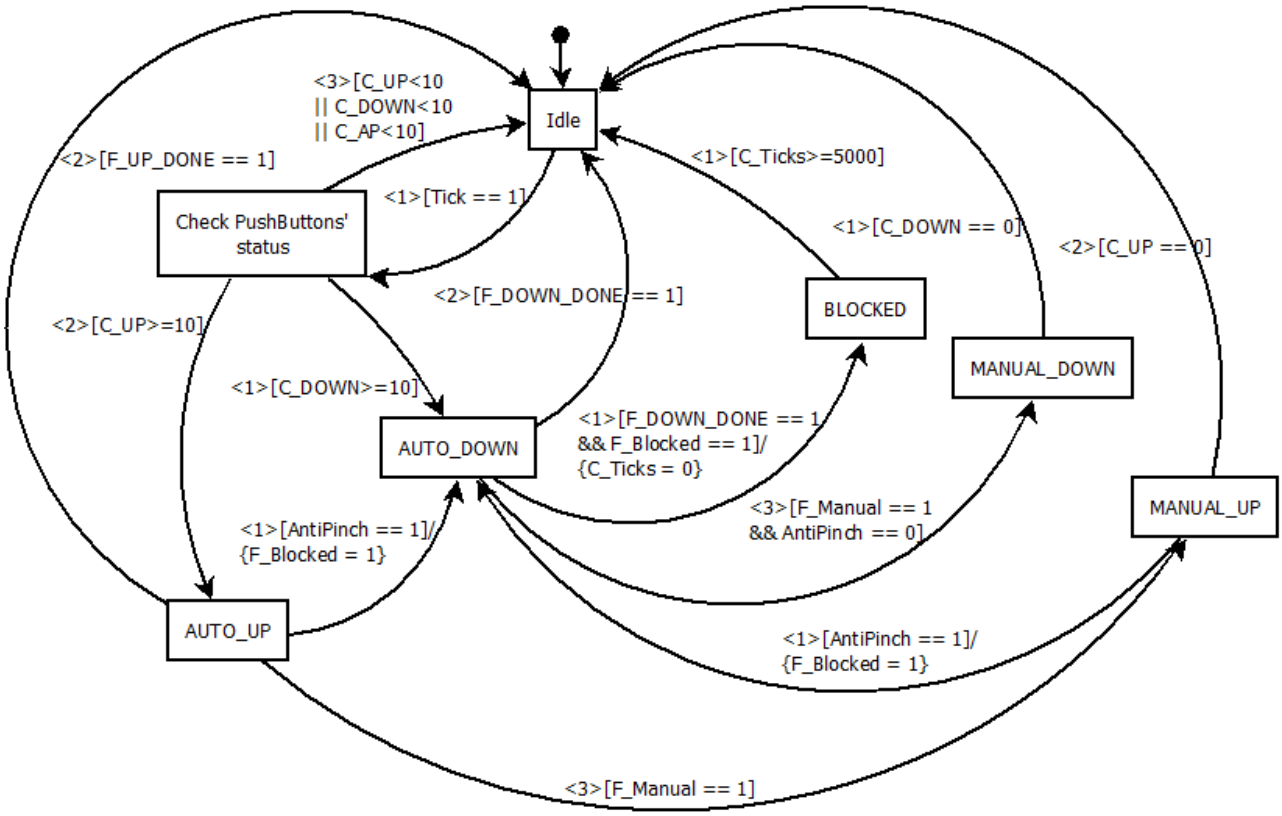
4.4 Sequence diagram



Req. Id. 4.3

This diagram illustrates the chronological order in which the signals are created, evaluated and implemented. First the different signals are created through the three different pushbuttons (UP, DOWN, and AntiPinch). Afterwards, the signal is validated. Finally, the system determines which LEDs should change state to control de window's level.

4.5 State diagram

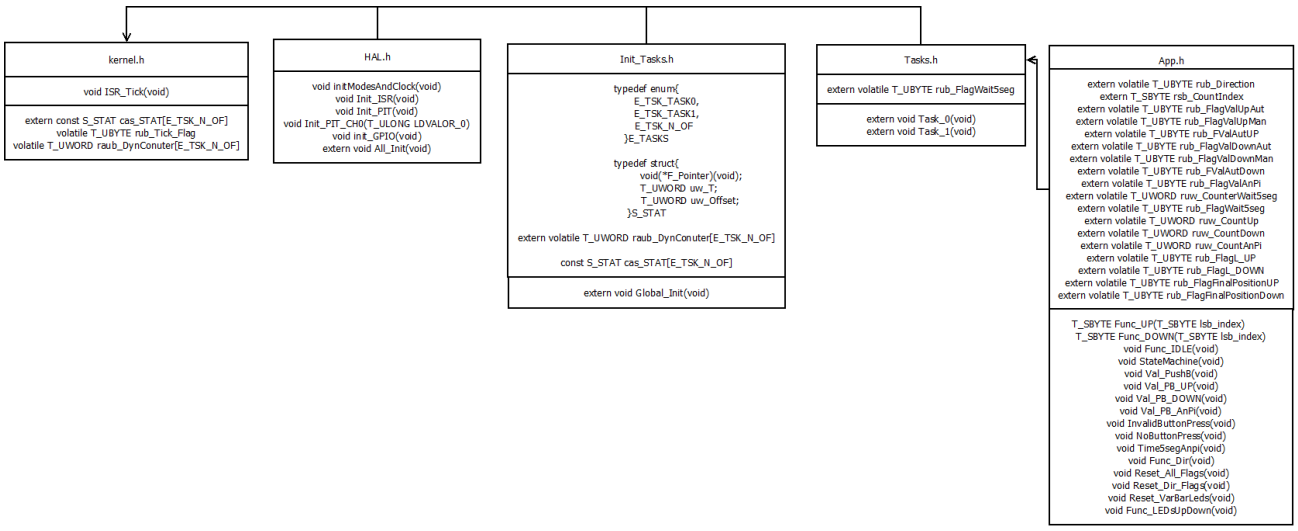


Req. Id. 2.6, 3.0, 3.1, 4.4, 4.5 4.6.

State	Description
Idle	State in which the software does not do a specific task. It only waits for changes to be made.
Check PushButtons' status	The status of the pushbuttons (up, down, and anti-pinch) is checked in order to keep track of the time that they have been pressed for.
AUTO_UP	The system shall keep going UP every 400 msec if the pushbutton up is released after being pressed for less than 500 msec.
AUTO_DOWN	The system shall keep going DOWN every 400 msec if the pushbutton down is released after being pressed for less than 500 msec.
MANUAL_UP	The system shall keep going UP every 400 msec if the pushbutton up is being pressed.
MANUAL_DOWN	The system shall keep going DOWN every 400 msec if the pushbutton down is being pressed.
BLOCKED	The system will be blocked for 5 seconds when the anti-pinch function is activated.

5 SW Component internal breakdown

5.1 Functional Decomposition



This diagram illustrates the way the code is divided into files. The following table describes the general function of each of the files.

File	Description
Kernel	It contains the main functions of the scheduler. It is in charge of creating the ticks that controls the timing of the whole program.
HAL	It contains the functions that correspond to the Hardware Abstraction Layer: Initialization of timers, GPIOs and interrupts.
Init_Tasks	It contains the initialization of the main tasks used in the Kernel file.
Tasks	It contains the implementation of the main tasks used in the Kernel file.
App	The functions of the main application are implemented in this file.

Function Description and Dynamic Behavior

5.2 Kernel

5.2.1 Function: void ISR_Tick(void)

Description	This function is the one that handles the clock Ticks in order to trigger the tasks. This interrupt runs periodically every millisecond according to the configuration of the PIT.
Parameter 1	Void
Return Value	Void
Precondition	This function is called when an interrupt is generated.
Post condition	It will literally interrupt the flow of the program to implement its code.
Error Conditions	

5.3 HAL

5.3.1 Function: void initModesAndClock(void)

Description	It initializes the mode of operation and clocks.
Parameter 1	*Void*
Parameter 2	*Does not apply.*
Return Value	*Void*
Precondition	This should be the first function called in the main program.
Post condition	It will be possible to use the oscillator with the configuration done.
Error Conditions	*Does not apply.*

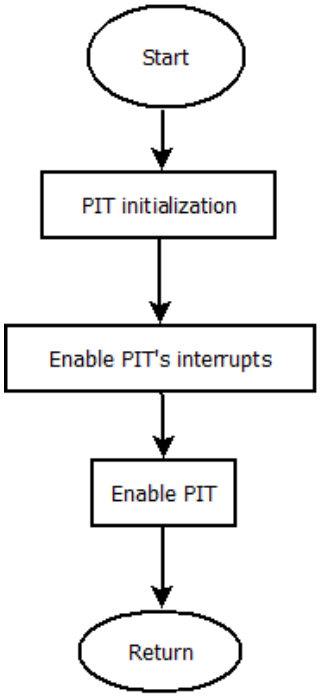
5.3.2 Function: void Init_ISR(void)

Description	Interrupts Configuration for timer PIT.
Parameter 1	Void
Return Value	Void
Precondition	This function is called in the beginning of the main program to initialize the interrupts for the PIT timer.
Post condition	The interrupts will be generated.
Error Conditions	Does not apply.

5.3.3 Function: void Init_PIT(void)

Description	*PIT is initialized.*
Parameter 1 <input| output| inout>	Value to generate an interrupt for the PIT 0
Parameter 2	Value to generate an interrupt for the PIT 1
Return Value	Void
Precondition	This function is called in the beginning of the main program to initialize the PIT.
Post condition	The interrupts will be generated.
Error Conditions	Does not apply.

Dynamic Behavior
Activity diagram



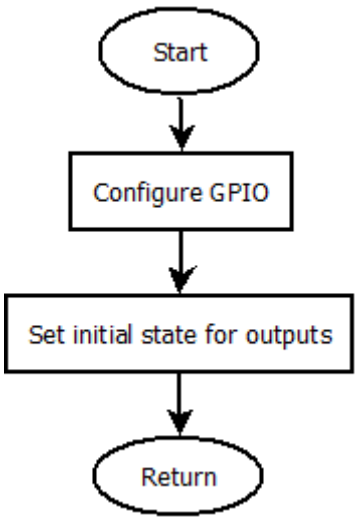
5.3.4 Function: void Init_PIT_CH0 (T_ULONG LDVALOR_0)

Description	This function configures the channel 0 of the PIT timer.
Parameter 1	T_ULONG
Parameter 2..n	Does not apply
Return Value	Void
Precondition	This function should be called in the beginning of the main application.
Post condition	The channel 0 of the PIT will be ready to use.
Error Conditions	Does not apply.

5.3.5 Function: void init_GPIO (void)

Description	This function configures the GPIO and initializes the outputs.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	This function should be called in the beginning of the main application.
Post condition	The ports can be used as configured.
Error Conditions	Does not apply.

Dynamic Behavior
Activity diagram



5.3.6 Function: extern void All_Init (void)

Description	This function initializes all the peripherals.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	This function should be called in the beginning of the main application.
Post condition	The pheriferals can be used.
Error Conditions	

5.4 Init_Tasks

5.4.1 Function: Void Global_Init(void)

Description	This function initializes the scheduler.
Parameter 1	Void
Return Value	Void
Precondition	This function should be called first in the main program.
Post condition	The scheduler's functionalities can be used.
Error Conditions	Does not apply.

5.5 Tasks

5.5.1 Function: Extern void task_0(void)

Description	This task is executed every millisecond. Its main function is to validate the presses of the push buttons. They should be longer than 10 msec. It also validates when the push buttons are pressed in the range of 10 to 500 msec in order to determine whether it should go to the automatic or manual mode.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	This function is called by the scheduler every millisecond.
Post condition	Does not apply.
Error Conditions	Does not apply.

Req. Id: 3.1

5.5.2 Function: Extern void task_1(void)

Description	This task is executed every 400 milliseconds. Its main function is to call the state machine that handles the movements in order to determine the next state of the LED bar and the direction indicators.
Parameter 1	Void
Return Value	Void
Precondition	This function is called by the scheduler every 400 milliseconds.
Post condition	Does not apply.
Error Conditions	

Req. Id. 2.4

5.6 App

5.6.1 Function: T_SBYTE Func_UP(T_SBYTE lsb_index)

Description	Function which generates the UP movement.
Parameter 1	T_SBYTE
Return Value	T_SBYTE
Precondition	The GPIOs should be already initialized.
Post condition	Does not apply.
Error Conditions	Does not apply.

Req. Id: 3.5

5.6.2 Function: T_SBYTE Func_DOWN(T_SBYTE lsb_index)

Description	Function which generates the DOWN movement.
Parameter 1	T_SBYTE
Return Value	T_SBYTE
Precondition	The GPIOs should be already initialized.
Post condition	Does not apply.
Error Conditions	Does not apply.

Req. Id: 3.5

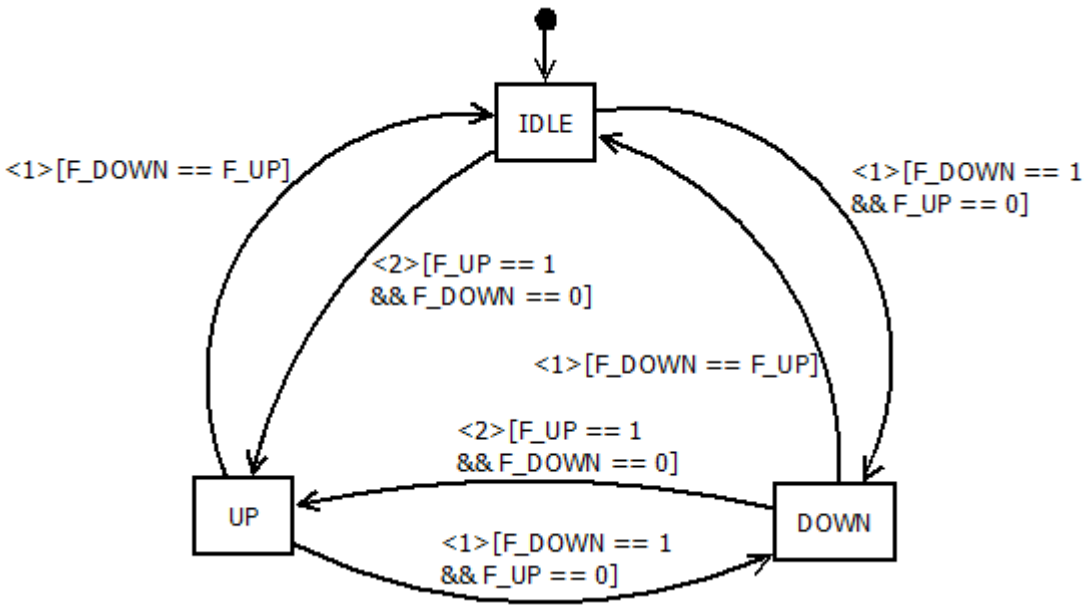
5.6.3 Function: void Func_IDLE(void)

Description	This function stops any signal showing movement.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.6.4 Function: void StateMachine(void)

Description	Function that holds the main state machine of the application. It chooses in which state it should be.
Parameter 1	Void
Return Value	Void
Precondition	The application should be already initialized.
Post condition	Does not apply.
Error Conditions	Does not apply.

Dynamic Behavior
State diagram

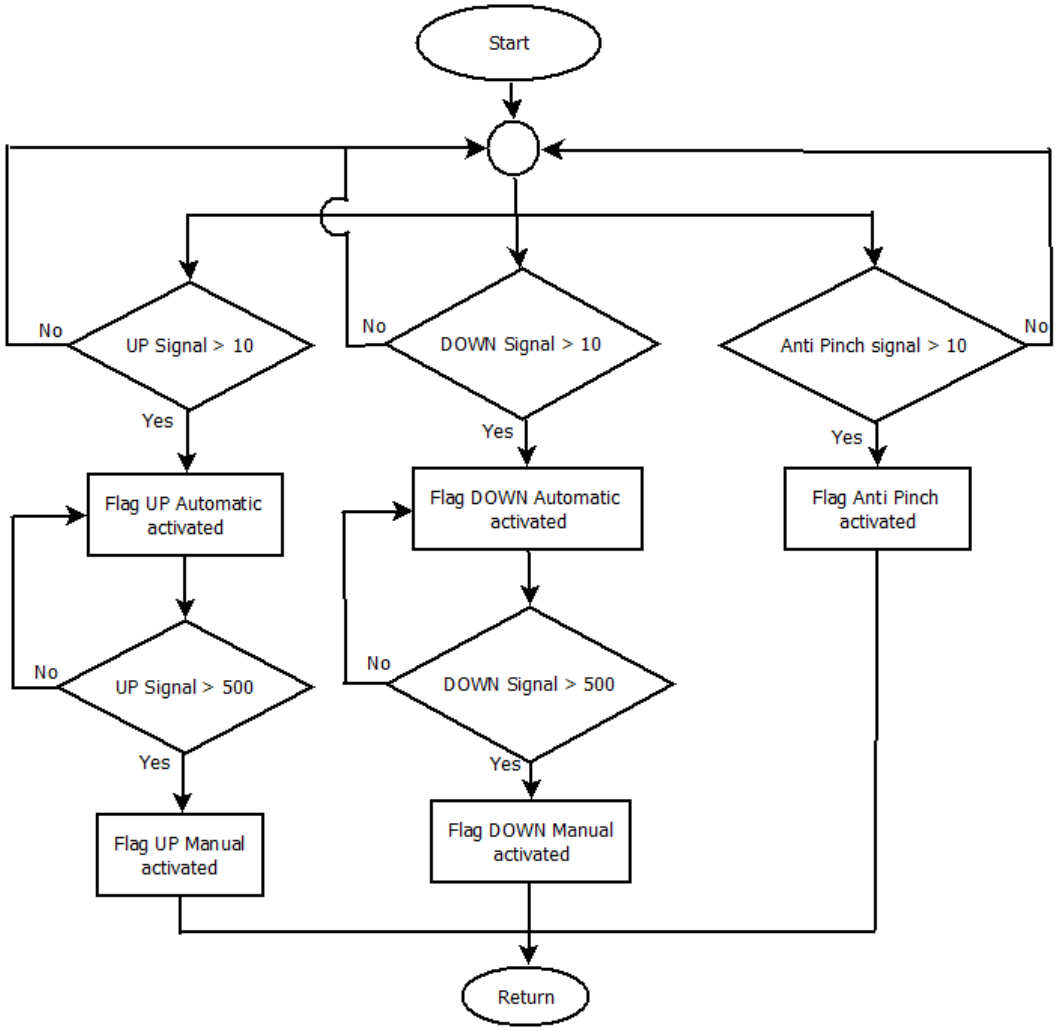


5.6.5 Function: void Val_PushB(void)

Description	This function validates the status of each push button. It is called by a task that runs every millisecond. It checks the status of the push buttons in order to consider it a valid press, and then it could run the functions that will change the status of the application.
Parameter 1	Void
Return Value	Void
Precondition	Does not apply.
Post condition	Does not apply.
Error Conditions	Does not apply.

Req. Id: 3.3, 3.4, 3.5

Dynamic Behavior
Activity diagram



5.6.6 Function: void Val_PB_UP(void)

Description	Function that sets and evaluates de counters in order to know for how long the push button UP has been pressed.
Parameter 1	Void
Return Value	Void
Precondition	The push button UP should be pressed.
Post condition	Does not apply.
Error Conditions	Does not apply.

Req. Id: 3.3, 3.4.

5.6.7 Function: void Val_PB_DOWN(void)

Description	Function that sets and evaluates de counters in order to know for how long the push button DOWN has been pressed.
Parameter 1	Void
Return Value	Void
Precondition	The push button DOWN should be pressed.
Post condition	Does not apply
Error Conditions	Does not apply

Req. Id: 3.0, 3.1, 3.2, 3.3, 3.4.

5.6.8 Function: void Val_PB_AnPi(void)

Description	Function that sets and evaluates de counters in order to know for how long the push button of the signal Anti Pinch has been pressed.
Parameter 1	Void
Return Value	Void
Precondition	The push button Anti Pinch should be pressed.
Post condition	Does not apply
Error Conditions	Does not apply

Req. Id: 3.0, 3.1, 3.2, 4.2.

5.6.9 Function: void InvalidButtonPress(void)

Description	This function clears the flags and sets the values necessary to declare an invalid button press.
Parameter 1	Void
Return Value	Void
Precondition	A push button signal was evaluated and considered invalid.
Post condition	Does not apply
Error Conditions	Does not apply

Req. Id: 3.0, 3.1, 3.2, 3.3, 4.2.

5.6.10 Function: void NoButtonPress(void)

Description	This function sets the flags to indicate that a button is not being pressed.
Parameter 1	Void
Return Value	Void
Precondition	No push button should be pressed.
Post condition	Does not apply
Error Conditions	Does not apply

5.6.11 Function: void Time5segAnpi(void)

Description	Function that decrements the value of the variable used to represent the 5 seconds delay. It also sets the flags so that the code will freeze after a Pinch.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	A valid Pinch signal should be recognized.
Post condition	The application will freeze for 5 seconds.
Error Conditions	Does not apply.

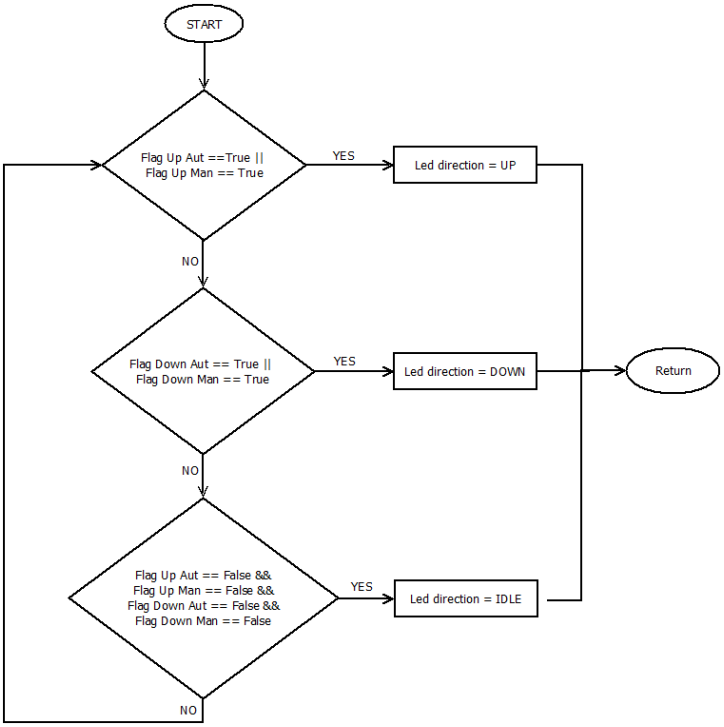
Req. Id. 4.6.

5.6.12 Function: void Func_Dir(void)

Description	Function which sets the direction of the movement according to the direction flags.
Parameter 1	Void
Parameter 2..n	Does not apply.
Return Value	Void
Precondition	Does not apply.
Post condition	Does not apply.
Error Conditions	Does not apply.

Req. Id: 2.7

Dynamic Behavior
Activity diagram



5.6.13 Function: void Reset_All_Flags(void)

Description	Function that resets all the flags that have to do with movement.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	Does not apply.
Post condition	Does not apply.
Error Conditions	Does not apply.

5.6.14 Function: void Reset_Dir_Flags(void)

Description	Function that resets all the flags that have to do with the direction movement.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	Does not apply.
Post condition	Does not apply.
Error Conditions	Does not apply.

5.6.15 Function: void Reset_VarBarLeds(void)

Description	Function that clears flags and counters generated by the push buttons.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

5.6.16 Function: void Func_LEDsUpDown(void)

Description	Function that sets the LED indicators that show the direction of the movement.
Parameter 1	Void
Parameter 2..n	Does not apply
Return Value	Void
Precondition	Does not apply
Post condition	Does not apply
Error Conditions	Does not apply

Dynamic Behavior
Activity diagram