Window Lifter
Title: Window Lifter Requirements

History				
Issue status (Index)	**Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)**	**Author Department**	**Check/Release Department**	**Description**
1.0	Draft 27-0ct-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Creation of the document.

History				
Issue status (Index)	**Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)**	**Author Department**	**Check/Release Department**	**Description**
2.0	Draft 28-0ct-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Added: Purpose, Definitions and abbreviations.

History				
Issue status (Index)	**Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)**	**Author Department**	**Check/Release Department**	**Description**
3.0	Draft 30-0ct-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Added: Realization constraints and targets.

History				
Issue status (Index)	**Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)**	**Author Department**	**Check/Release Department**	**Description**
4.0	Draft 31-0ct-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Added: SW Conceptual design.

History				
Issue status (Index)	**Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)**	**Author Department**	**Check/Release Department**	**Description**
5.0	Draft 1-Nov-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Added: SW Component internal breakdown.

History				
Issue status (Index)	**Maturity/Date (draft/invalid/valid) (dd-mmm-yyyy)**	**Author Department**	**Check/Release Department**	**Description**
6.0	Release 1-Nov-15	Francisco Quirarte y David Díaz	Francisco Quirarte	Added: General corrections.

Table of Contents

1. Purpose

Window lifter is a module that controlees the movement of a window. The module works with two switches that indicate the direction of the movement.

The window will be emulated using a 10 led bar. The time between each transition shall be 400 msec. Each window movement has to be indicated trough a led color. Depending on movement each led has to be turn on: **UP-BLUE DOWN-GREEN**.

In order to consider a validate button press; the button has to be pressed at least 10 msec. The module has to be able to detect fail button press. In case the button is pressed less than 10 msec or a button combination it will to be considered as invalid.

The module will have an antipinch function that will be emulated with a push button. This signal just can be considered as valid when the movement is UP. If this signal is valid then the module has to stop the UP Movement and then DOWN the window until the window get totally OPEN. After window is totally OPEN the module has to ignore during 5 seconds all button press. After this time the module has to recognize every button press.

2. Definitions and abbreviations

Definitions

Acronym	Definition
config_timer();	Initializes and configures the timers.
initModesAndClock();	Initializes and configures the interruptions.
INTC_InstallINTCInterruptHandler(led_azul,30,1);	When a button its pressed the interruption goes to the next value, calls the vector number 30 and calls the function void led_azul().
void led_azul(void);	Principal Function where its decided what rutine will be executed by the leds.

Abbreviations

Acronym	Definition
STM	System Timer Module
GPIO	General Purpose Input Output
CRT	Conversion Timing registers
IPM	Input Period Measurement
CIR	Channel Interrupt Register
SIUL	System Integration Unit Lite
STM CMP	STM Compare Register

References

N°	Document name
1	Traceability Matrix Template.xls
2	MPC5604B/C Microcontroller Reference Manual.pdf
3	Test_template.xls
4	http://www.freescale.com/products/power-architecture-processors/mpc5xxx-5xxx-32-bit-mcus/mpc56xx-mcus/mpc5606b-startertrak-development-kit:TRK-MPC5606B
5	http://www.freescale.com/products/power-architecture-processors/mpc5xxx-5xxx-32-bit-mcus/mpc56xx-mcus/ultra-reliable-mpc56xb-mcu-for-automotive-industrial-general-purpose:MPC560xB#pspFeatures
6	SW_C_Code_Review_Template.docx
7	Window-lifter-requirements.docx

3. Realization constraints and targets

The system will be running in the MPC 5606B. With the following specifications:

- MPC5606B MCU in a 144LQFP package.
- 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.
- CAN and LIN interface.
- Analog interface with potentiometer.
- High-efficiency LEDs.
- SCI serial communication interface.

TRK-MPC5606B: MPC5606B StarterTRAK (Development Kit)

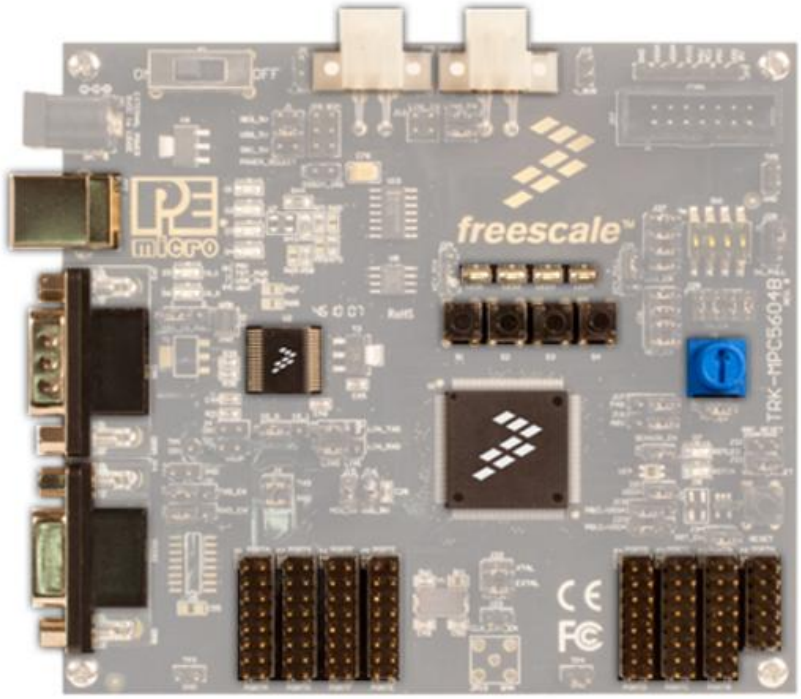

Figure 1: MPC5606B board.

- Operating Frequency (Max): 64 MHz.
- Total DMA Channels 16.
- Internal Flash (KB): 512
- GPIOs: 149.
- EEPROM: 64 KB DataFlash®
- RAM: Up to 96 KB
- Timer: 16 bits up to 64 channels

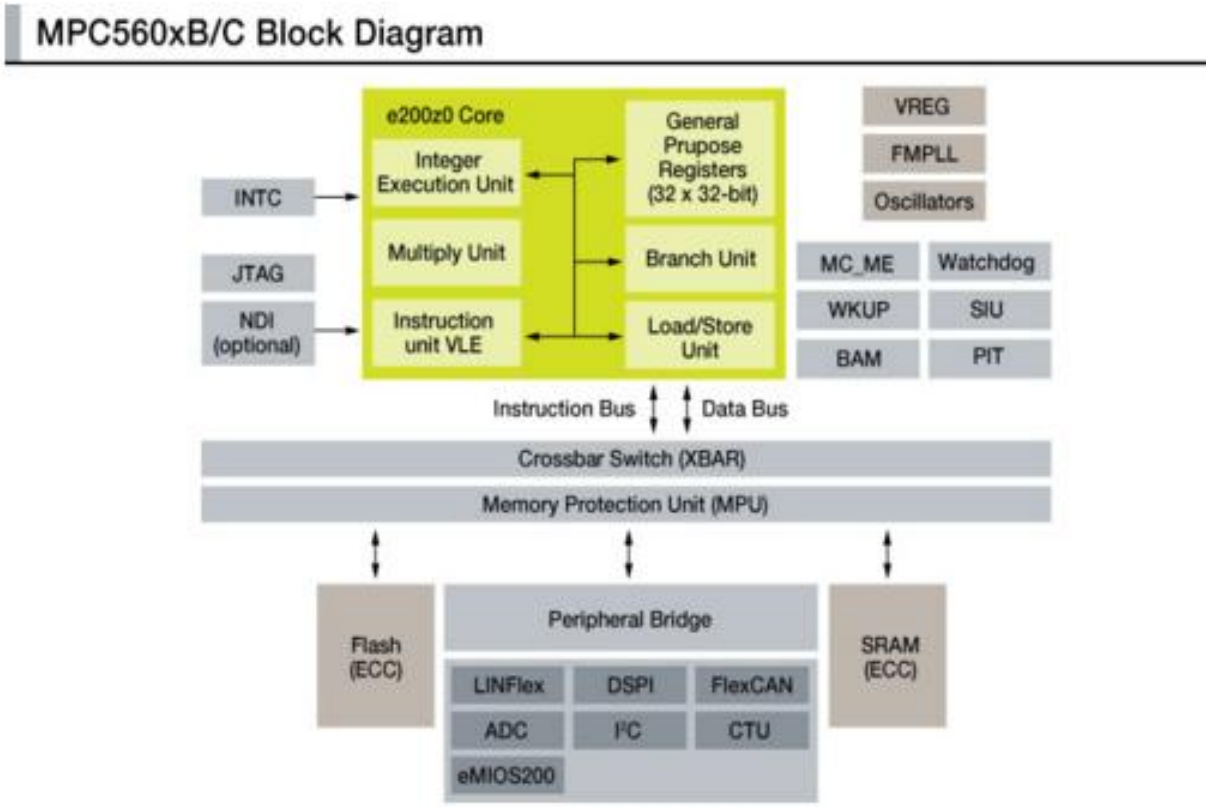

Figure 2: MPC56X B/C Boolero Arquitecture Family.

- ADC:
- 10 bits up to 36 channels
- 12 bits up to 16 channels
- Up to six CAN
- Up to six SPI
- Up to 10 LINFlex

4. SW Conceptual design

a) Use Case

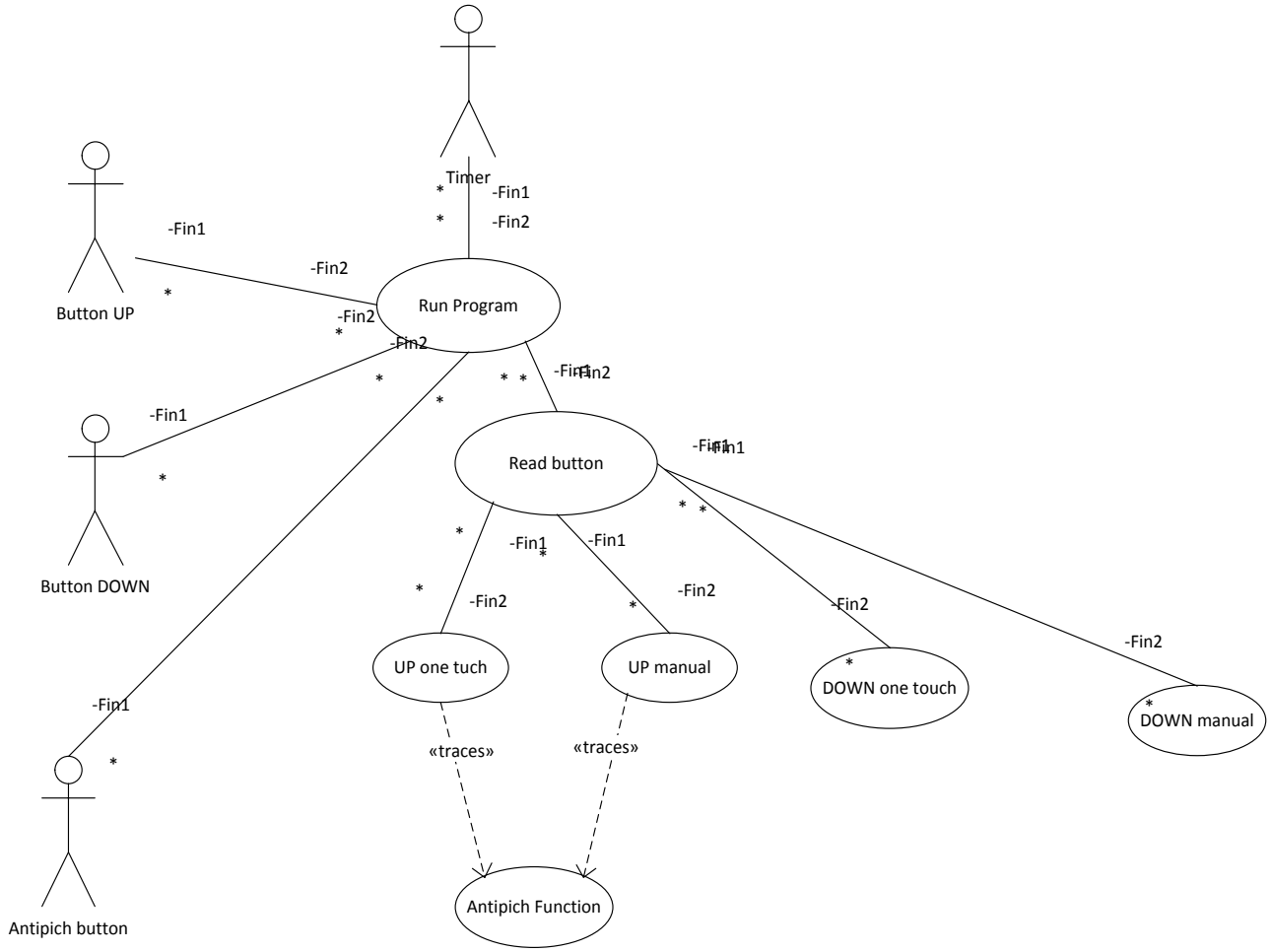

Figure 3: Use Case Diagram. In this diagram whe see the interaction between the user and the software.

b) Deployment

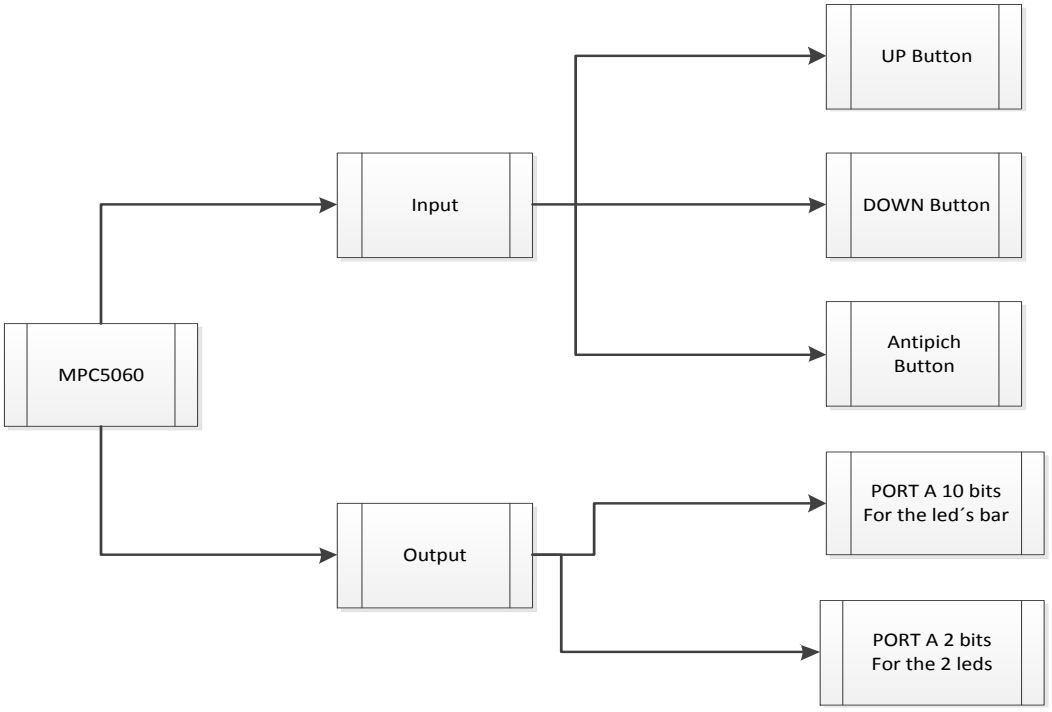

Figure 4: Deployment Diagram. Represents the physical configurations of software and hardware items.

c) Component

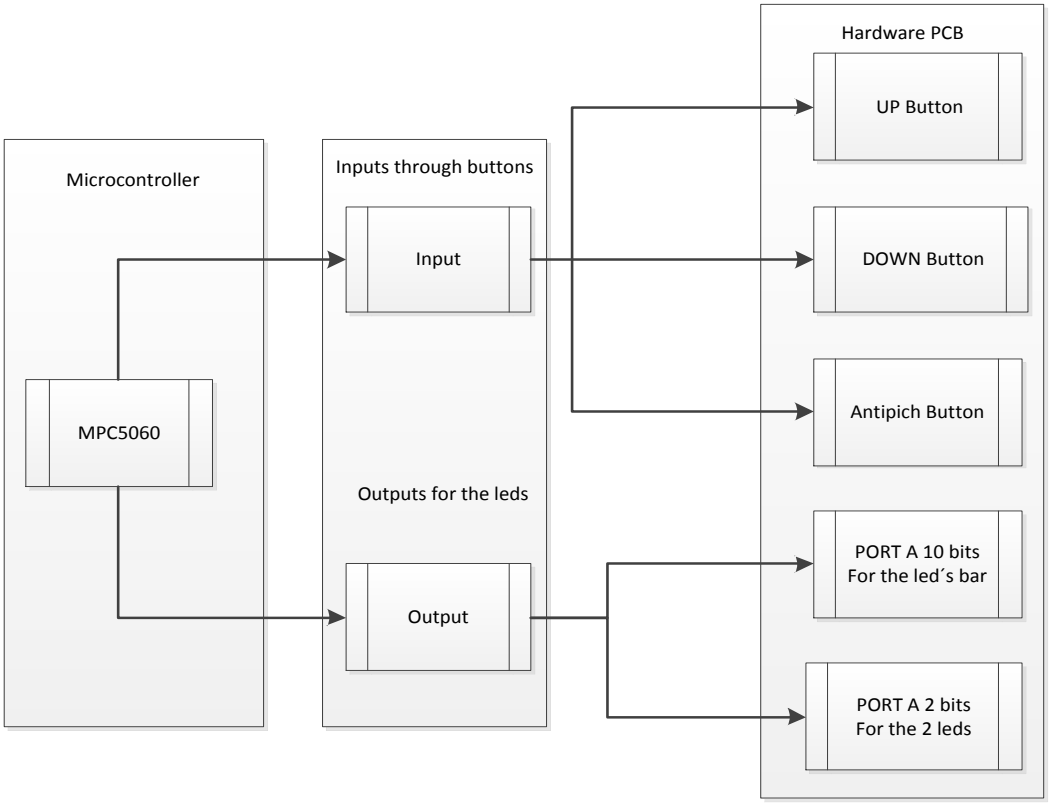

Figure 5: Component Diagram. Represents the structure of physical components**.**

d) Activity

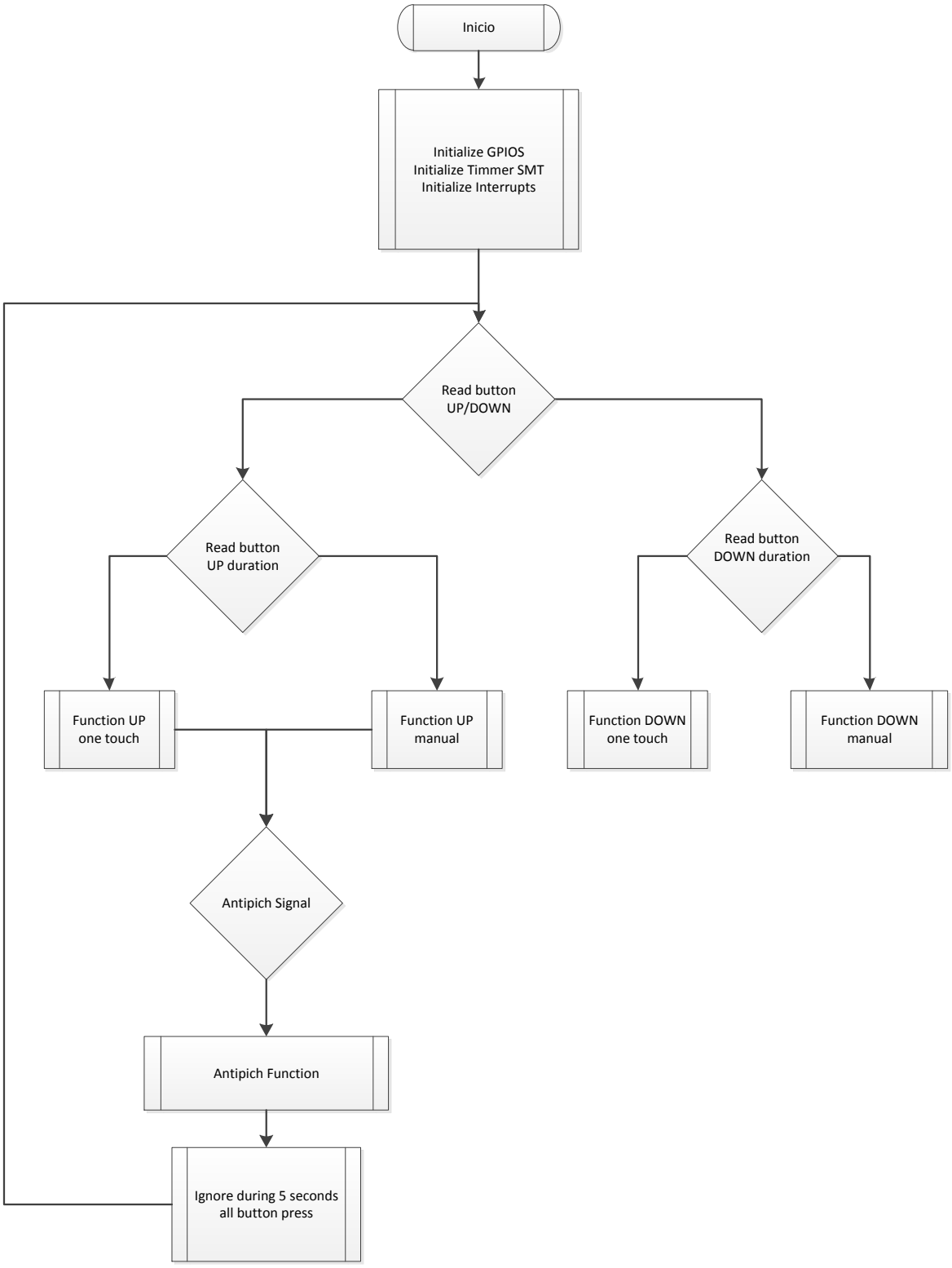

Figure 6: Activity Diagram. Describes the behavior of the Window Lifter program**.**

e) Class

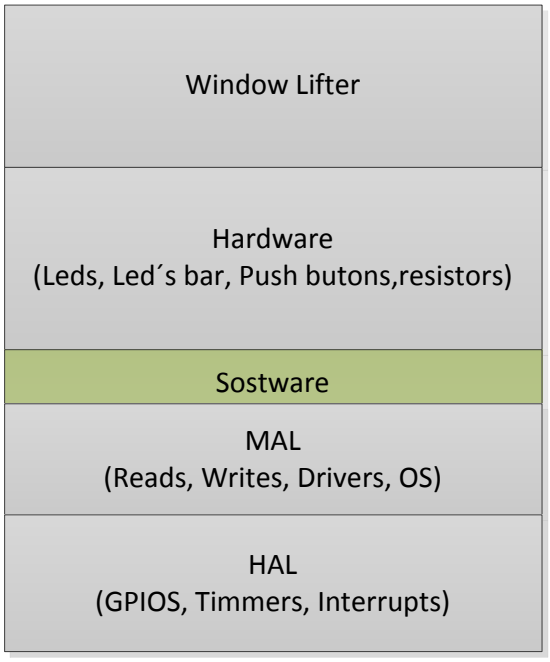

Figure 7: Class Diagram. Represents the different classes of the software.

f) Sequence

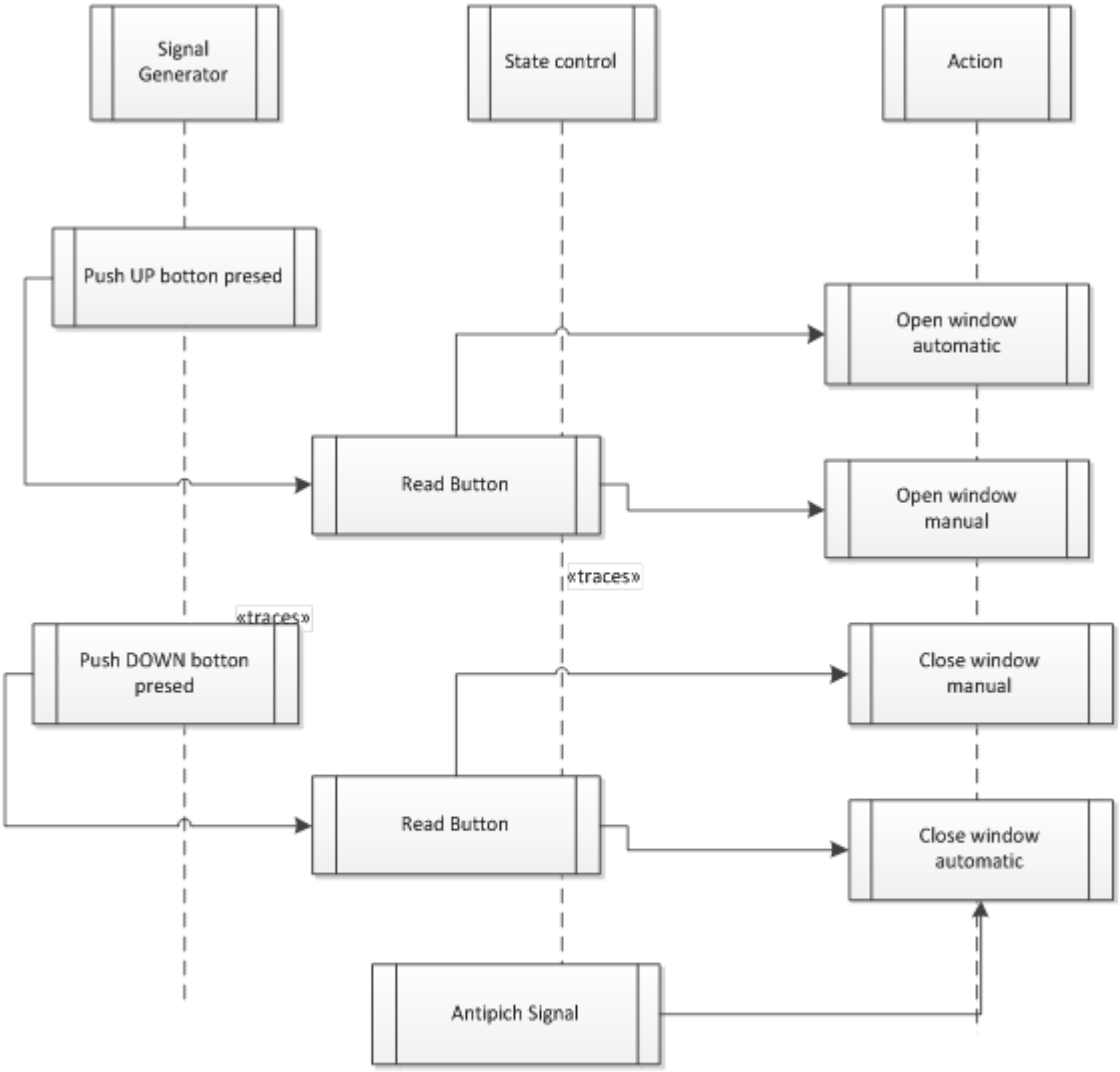

Figure 8: Sequence. Pattern of interaction among the objects, arranged in a chronological order lifelines.

5. SW Component internal breakdown

1. A header which contains all the libraries to perform the main system was used with the name **DRIVERS.h.**

2. A file called **STM_config.c** contains the initialization and configuration of the System Timer Module (STM). And will be included at the library DRIVERS.h.

3. A file called **init_clock_sw_leds.c** contains the initialization and configuration for the microcontroler, portA (used as output for the leds) and switches.

4. A file called **reset_clean.c** contains 4 functions (one for each channel) that reset the STM timer counter and clean the configured flags.

5.1 Functional Decomposition

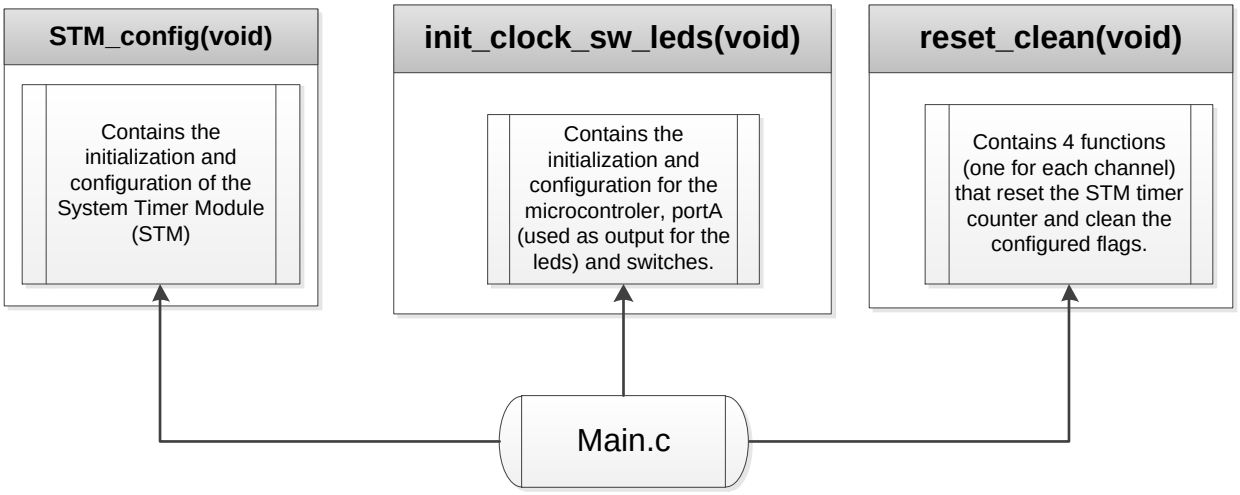

Figure 9: Functional Decomposition Diagram

5.2 Function Description and Dynamic Behavior

5.2.1 Function <*void*> ><*STM_config (void)*>

Description	Configures and initializes the SMT timers
Parameter 1 <*output*>	Configures the time of the comparator in each channel.
Parameter 2 <*output*>	Initialization of the flag of STM timer.
Parameter 3 <*output*>	Turns on timmer.
Return Value	None
Precondition	None
Post condition	None
Error Conditions	None

Dynamic Behavior

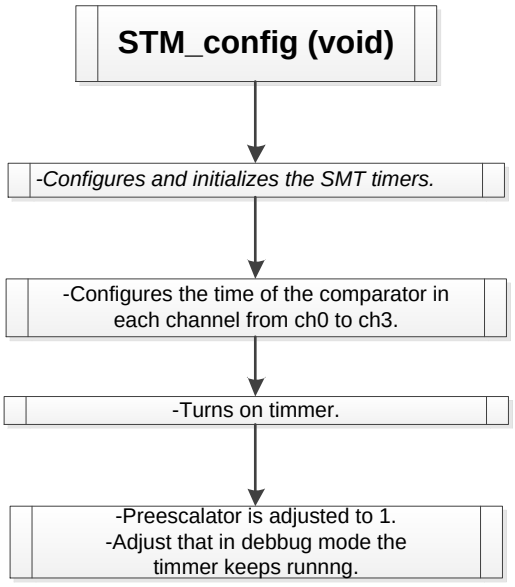

Figure 10: Flow Chart of function STM_config (void)

5.2.2 Function <void> <*init_clock_sw_leds (void)*>

Description	Contains the initialization and configuration for the microcontroler, portA and switches.
Parameter 1 <*output*>	Configures and initializes :RUN0, SIUL, ME.
Parameter 2 <*output*>	Configures and initializes the board switches (64 to 67).
Parameter 3 <*output*>	Configures and initializes the board switches (68 to 71).
Parameter 4 <*output*>	Configures and initializes PortA as output.
Return Value	None
Precondition	None
Post condition	None
Error Conditions	None

Dynamic Behavior

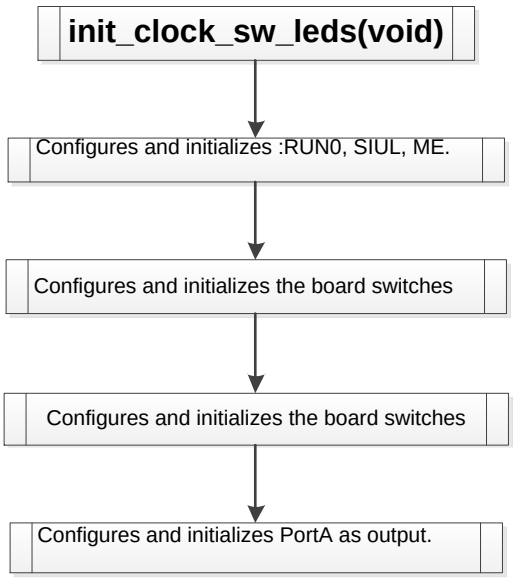

Figure 11: Flow Chart of function *init_clock_sw_leds(void)*

5.2.3 Function <void> *reset_clean(void)*

Description	Contains 4 functions (one for each channel) that reset the STM timer counter and clean the configured flags.
Parameter 1 <output>	Reset the STM timer comparator for channel 0 and clean the configured flags.
Parameter 2 <output>	Reset the STM timer comparator for channel 1 and clean the configured flags.
Parameter 3 <output>	Reset the STM timer comparator for channel 2 and clean the configured flags.
Parameter 4 <output>	Reset the STM timer comparator for channel 3 and clean the configured flags.
Return Value	None
Precondition	None
Post condition	None
Error Conditions	None

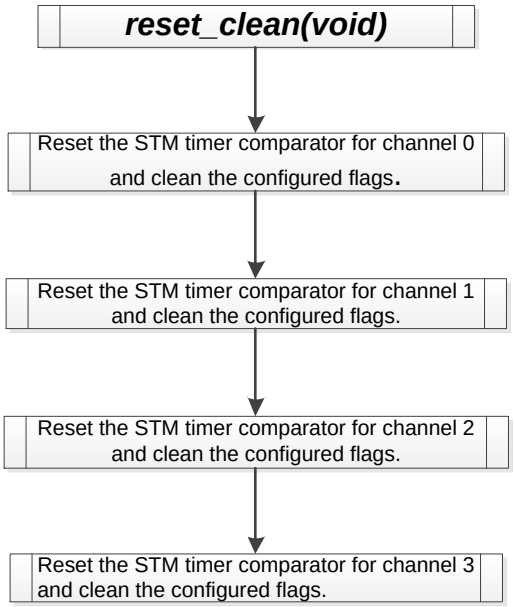

Figure 12: Flow Chart of function *reset_clean(void)*

5.2.4 **Function main <*void*> <*main(void)*>**

Description	*This function active all conditions and rules of the system*
Parameter 1 <input>	*Return all variables of the system*
Return Value	*None*
Precondition	e.g. if all conditions and libraries is OK the function is run
Error Conditions	*If in the parameter void change for another type of date maybe the function is brake*

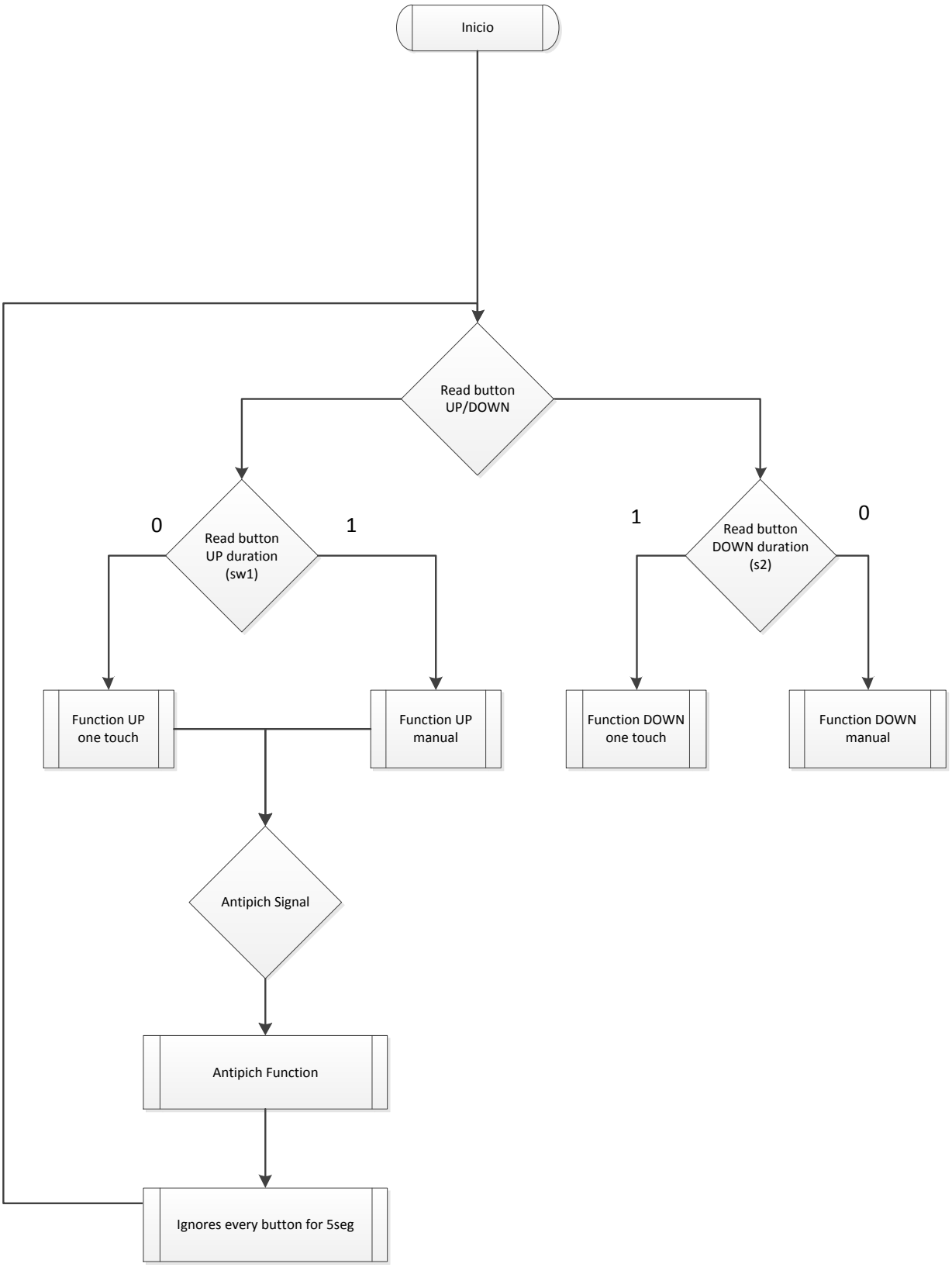

Figure 13: Flow Chart of function *main(void)*