Proyecto de Formación Dual

Detail Design

1 Purpose

This document has been written with the purpose of describing the detailed design of the Dual Training project application layer, based on the project architecture, listing and describing how the different functions will work in the system.

2 Definitions and abbreviations

Definitions

FDual: Formation Dual

3 References

Material proporcionado por Juan Luis Garcia Camacho

S. Johnson, Software Architecture Document generated using Rational SoDA template and Rational Rose model https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm#Architectural%20Representation

4 Realization constraints and targets

.

This document contains all the information related to the documentation of Detailed design for the Application layer of the project. This layer contains all the algorithms corresponding to the project goals.

5 SW Conceptual design

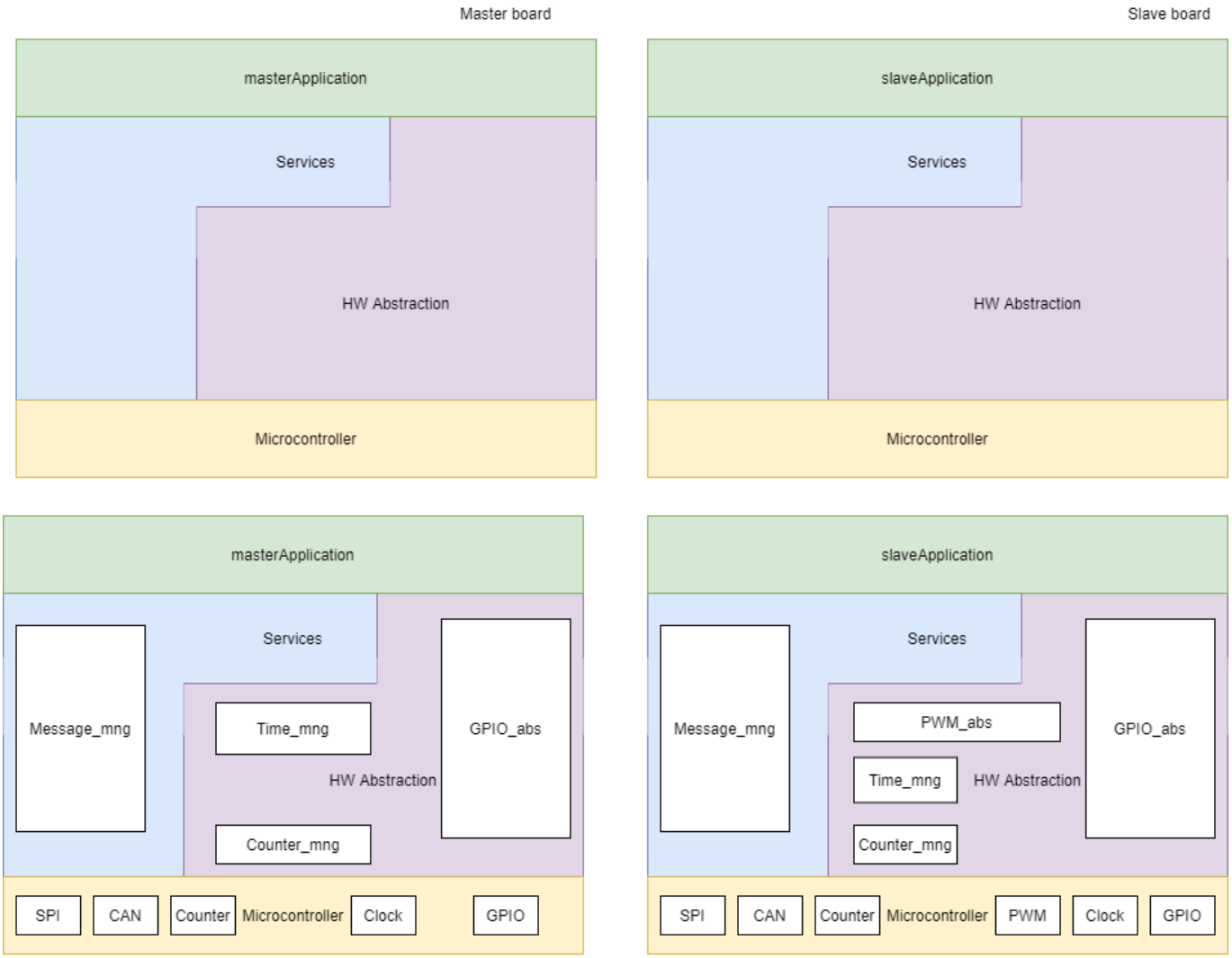
Link a nuestro documento de arquitectura.
https://proyecto-formacion-dual-team-acgf.github.io/ProyectoFDConti/SwArchitecture_TeamACGF.pdf

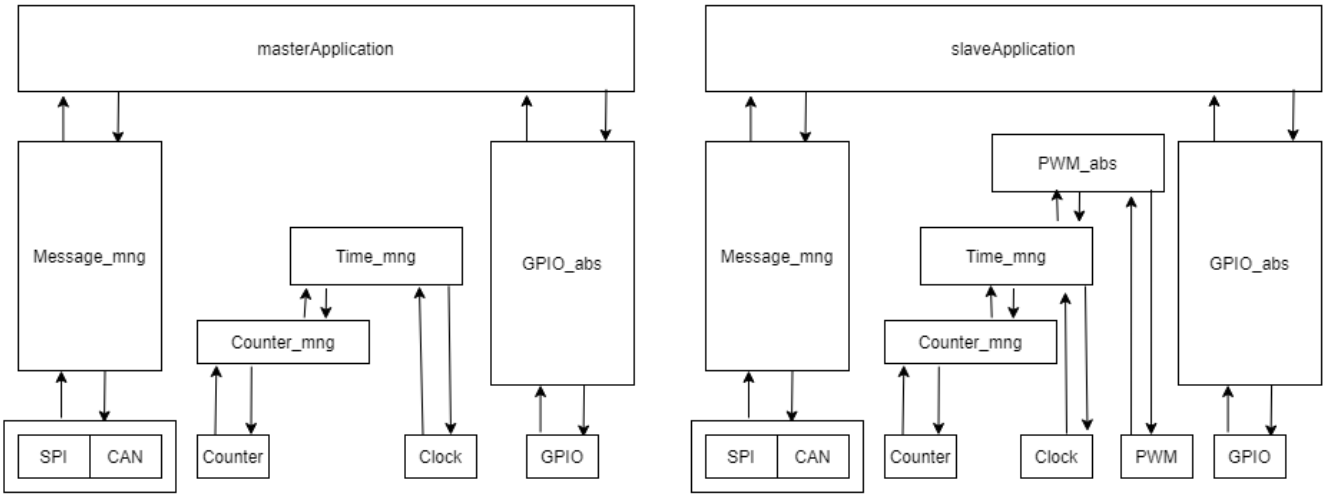
6 SW Components internal breakdown

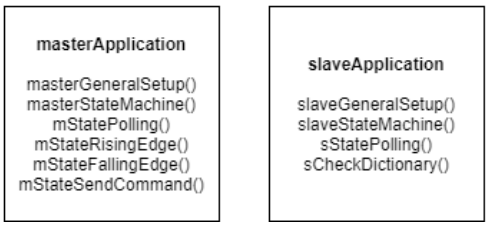
6.1 Application

The application layer interacts with all the other layers and components, and uses functions from the other layers to achieve the functionality of the project.

Architectural Description


Master board
masterApplication
Services
HW Abstraction
Microcontroller
Slave board
slaveApplication
Services
HW Abstraction
Microcontroller
masterApplication
Message_mng
Services
Time_mng
GPIO_abs
HW Abstraction
Counter_mng
SPI
CAN
Counter
Microcontroller
Clock
GPIO
slaveApplication
Message_mng
Services
PWM_abs
GPIO_abs
Time_mng
HW Abstraction
Counter_mng
SPI
CAN
Counter
Microcontroller
PWM
Clock
GPIO

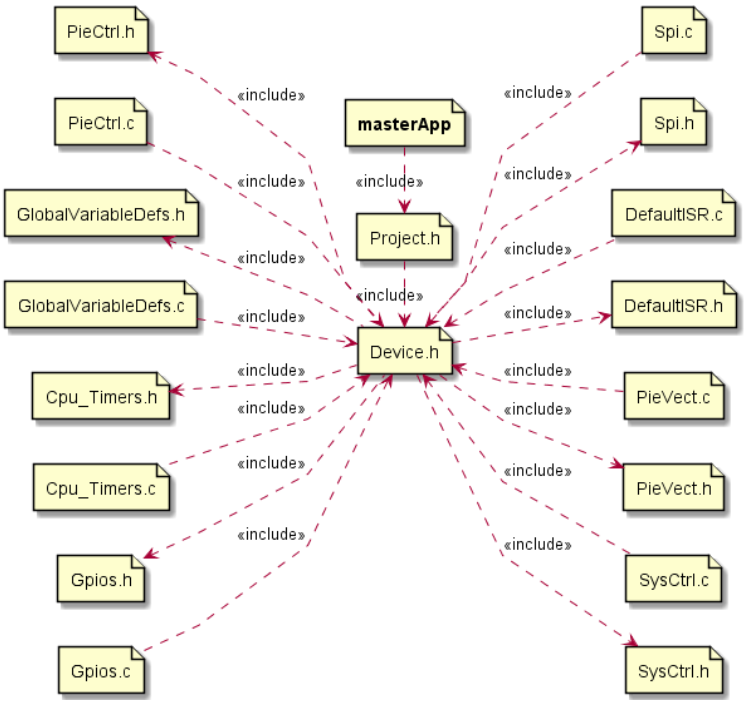




6.1.1 File structure

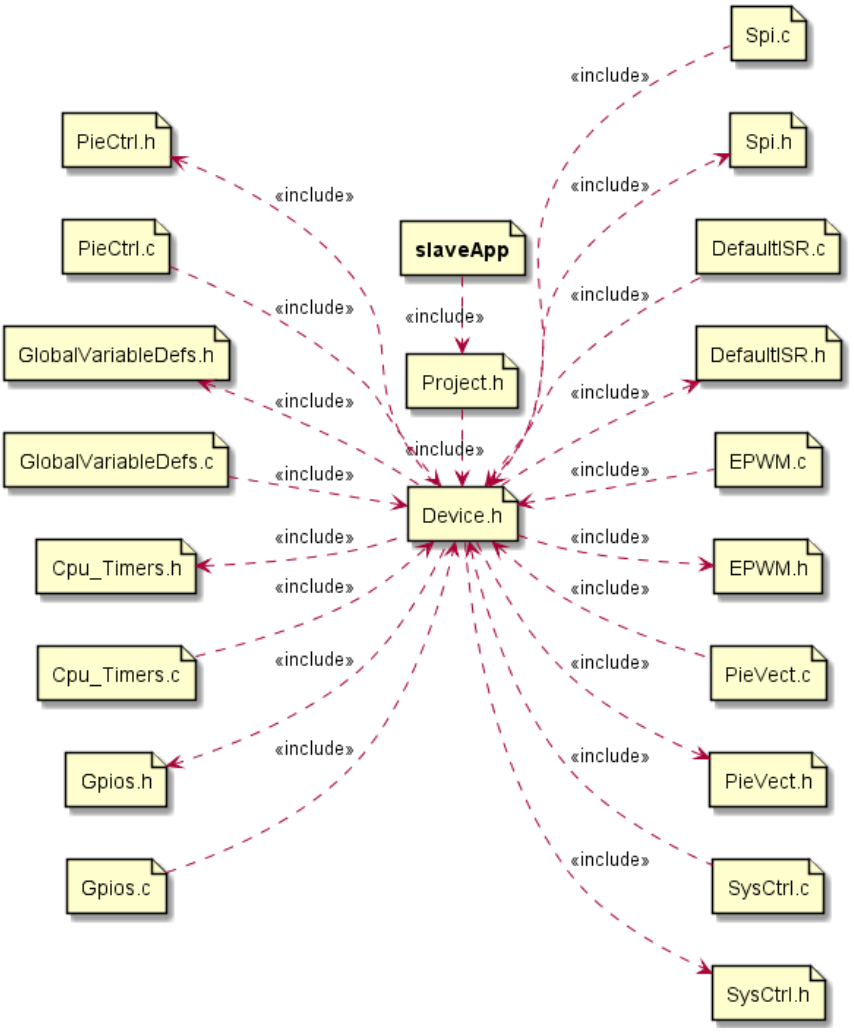
Master's Files Structure

The file include structure for Master shall be as follows:



Slave's Files Structure

The file include structure for Slave shall be as follows



6.1.2 Defines

(Para todo el componente, es decir, para toda la appl)

#define	**COP_FIRST_BYTE_OF_REFRESH** (0x55U) First byte of refresh sequence.
#define	**COP_SECOND_BYTE_OF_REFRESH** (0xAAU) Second byte of refresh sequence.

6.1.3 Enumerations types

MasterStates

Name:	**MasterStates**	
Type:	**Enumeration**	
Range:	State_Polling	Polling state
	State_FallingEdge	Falling Edge state
	State_RisingEdge	Rising Edge state
	State_SendCommand	Send Command state
Description	Data type that contains labels to reference states in state machine of master board	

SlaveStates

Name:	**SlaveStates**	
Type:	Enumeration	
Range:	State_polling	Polling state
	State_setPWM	Set_PWM state
	State_CommandReceived	State_commanReceived
Description	Data type that contains labels to reference states in state machine of slave board	

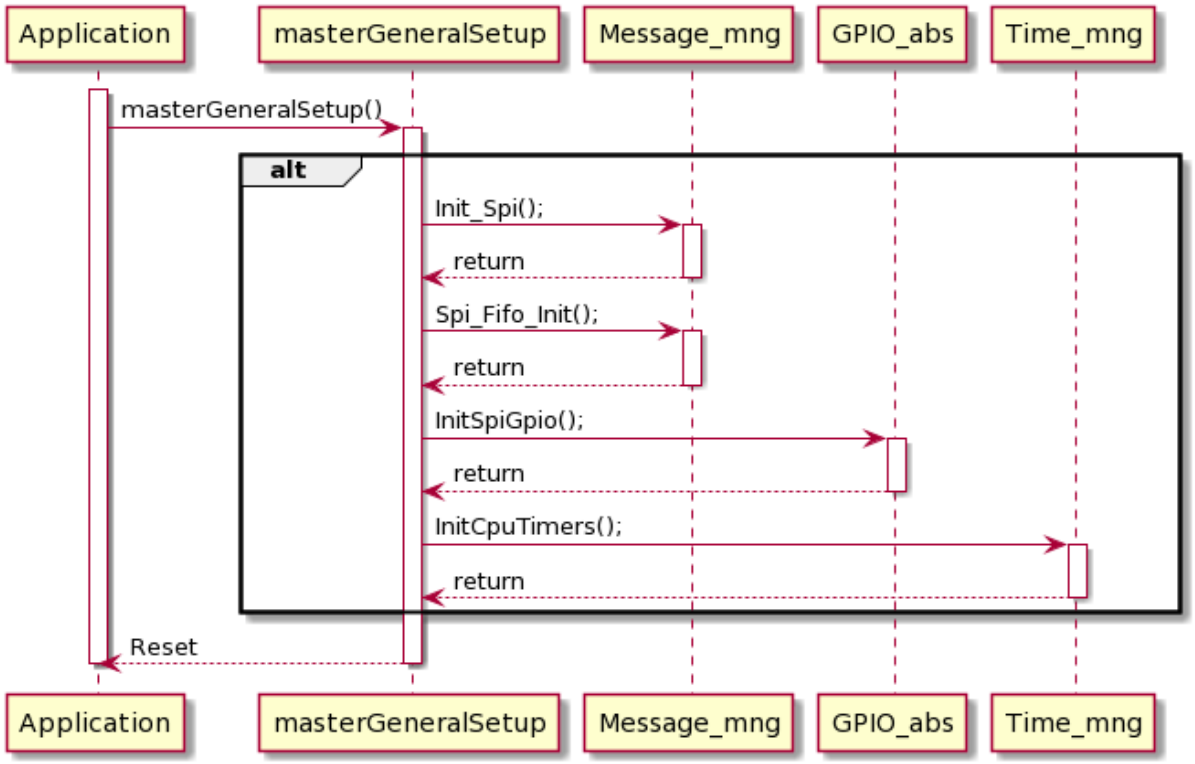
MASTER BOARD

6.1.4 Function masterGeneralSetup()

Name	**masterGeneralSetup()**

Description	Sets the different aspects of the Master Board for its correct operation, calling different function from the abstraction and services layers, the aspects that it will configure are: - InitSPI() - Spi_Fifo_Init() - InitSpiGpio() - InitCpuTimers()
Syntax	void masterGeneralSetup(void)
Parameter 1 [input]	void
Return Value	null
Precondition	none
Post condition	none

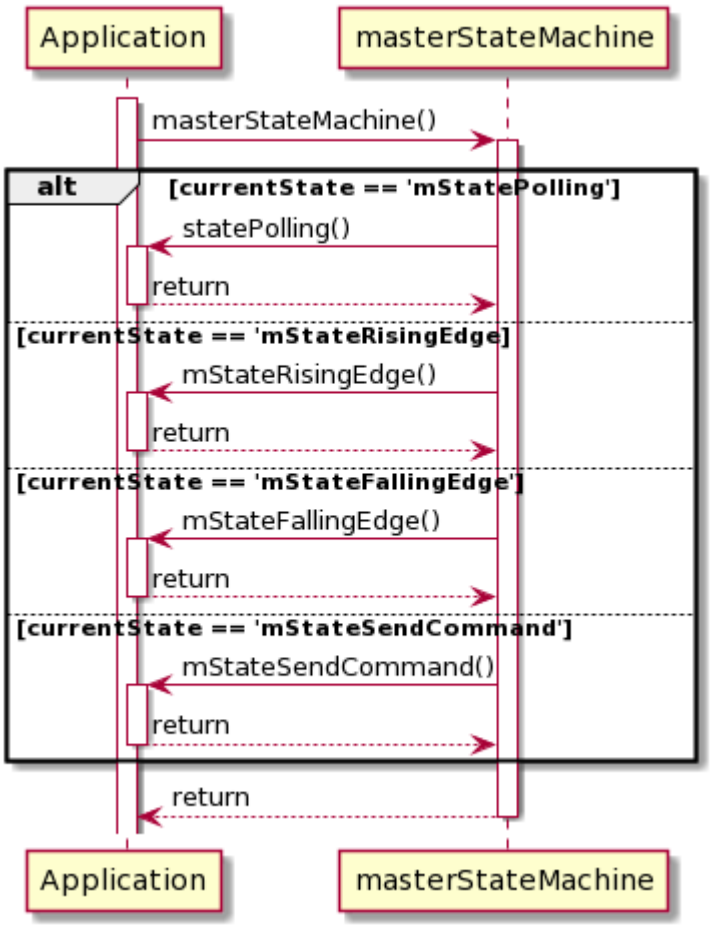
GeneralSetupsMaster() Dynamic Behaviour



6.1.5 Function masterStateMachine()

Name	**masterStateMachine()**
Description	masterStateMachine, Application Component currentState can be taken as the actual state of the machine currentState = mStatePolling (), when the button has not been pressed, waiting for the button to be pressed currentState = mStateRisingEdge (), when button was pressed Basetimer and PulseTimer started currentState = mStateFallingEdge () when button was released, restart pulsetimer
Syntax	void masterStateMachine(void)
Parameter 1 [input]	void
Return Value	null
Precondition	masterGeneralSetup() must be called at least once
Post condition	masterStateMachine() remains in the same state until the button is pushed again.

MasterStateMachine() Dynamic Behaviour


Application
masterStateMachine
masterStateMachine()
alt
[currentState == 'mStatePolling']
statePolling()
return
[currentState == 'mStateRisingEdge]
mStateRisingEdge()
return
[currentState == 'mStateFallingEdge']
mStateFallingEdge()
return
[currentState == 'mStateSendCommand']
mStateSendCommand()
return
return
Application
masterStateMachine

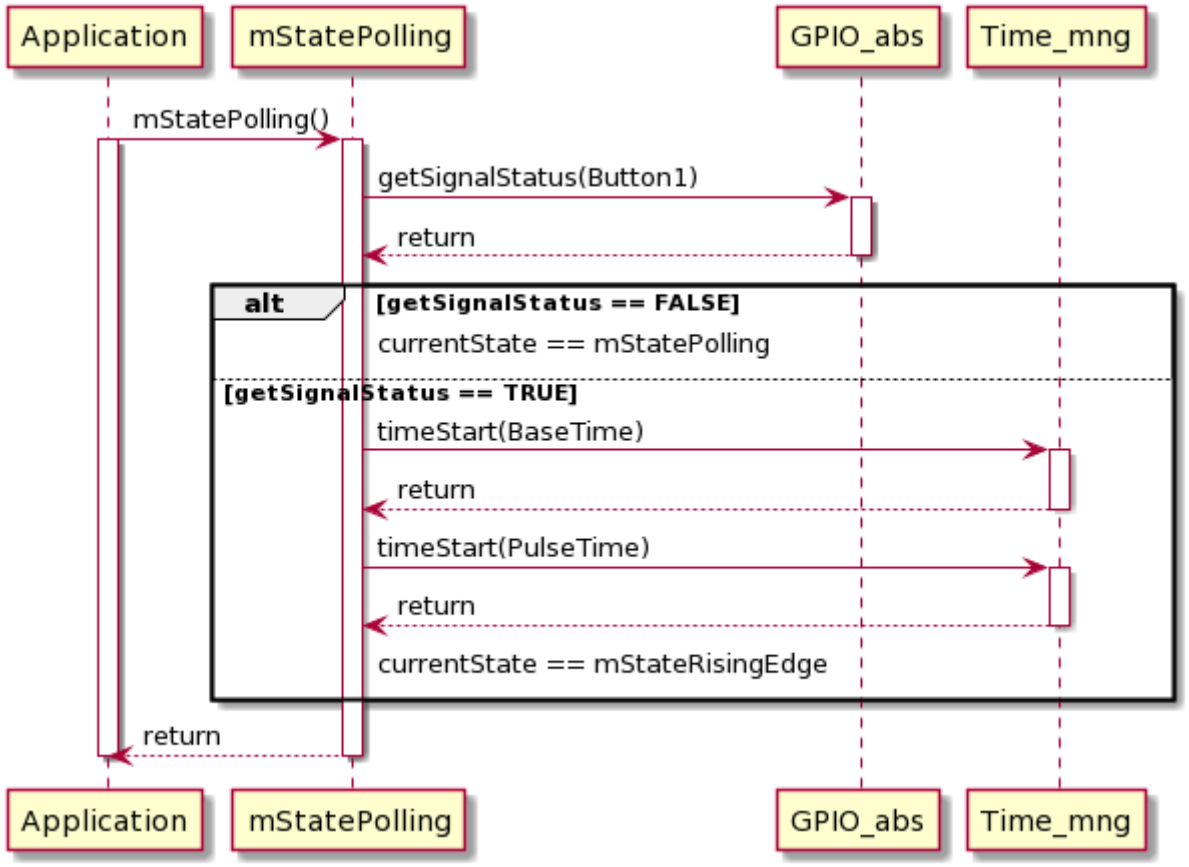
6.1.6 Function mStatePolling()

Name	StatePolling()
Description	Llama una función de la capa de abstracción para leer el estado de una señal de entrada llamada boton1. Mientras boton1 esté en bajo, mantiene CurrentState como State_Polling. Cuando boton1 esté en alto, llama las funciones de la capa de servicios para iniciar 2 times, una vez iniciado los Times, cambia CurrentState a State_RisingEdge. Time1: BaseTime para contar desde la primera vez que se apretó un botón Time2: PulseTime para contar el tiempo de la señal pulsada.
Syntax	void MasterStateMachine(void)
Parameter 1 [input]	Void
Return Value	Void
Precondition	GeneralSetUps debió haber sido llamada al menos una vez
Post condition	En caso de haber detectado un valor en alto de la señal boton1, CurrentState toma el valor del nuevo estado (State_RisingEdge)

Name	**mStatePolling()**
Description	Constantly reads the button input signal. Wait for an interrupt from Rising edge.
Syntax	void mStatePolling(void)
Parameter 1 [input]	void
Return Value	null

Precondition	masterGeneralSetup() must be called at least once
Post condition	masterStateMachine() remains in the same state until the button is pushed

Dynamic Behavior

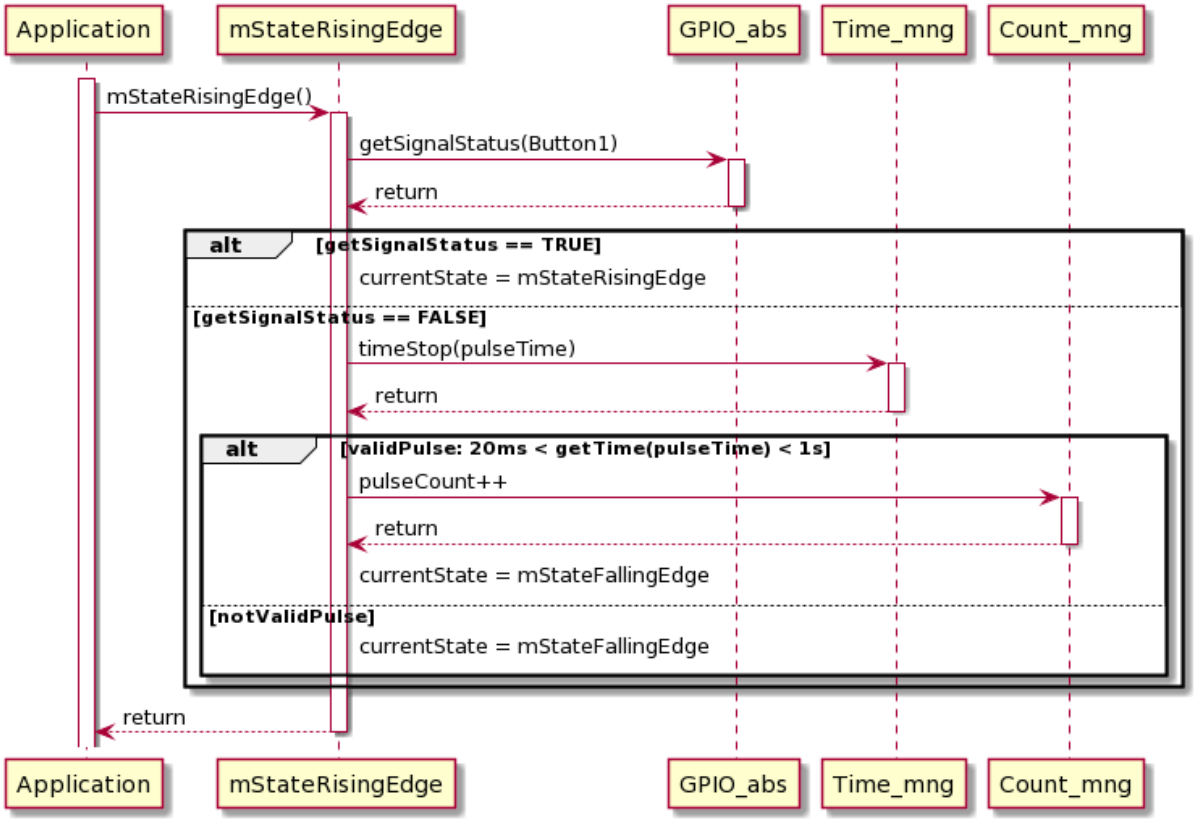


6.1.7 Function mStateRisingEdge()

Name	**mStateRisingEdge()**

Description	Waits for Rising edge detection at the GPIO interrupt corresponding to button input in pull-up configuration
Syntax	void mStateRisingEdge(void)
Parameter 1 [input]	void
Return Value	null
Precondition	masterGeneralSetup() must be called at least once
Post condition	masterStateMachine() waits for button low

mStateRisingEdge() Dynamic Behaviour

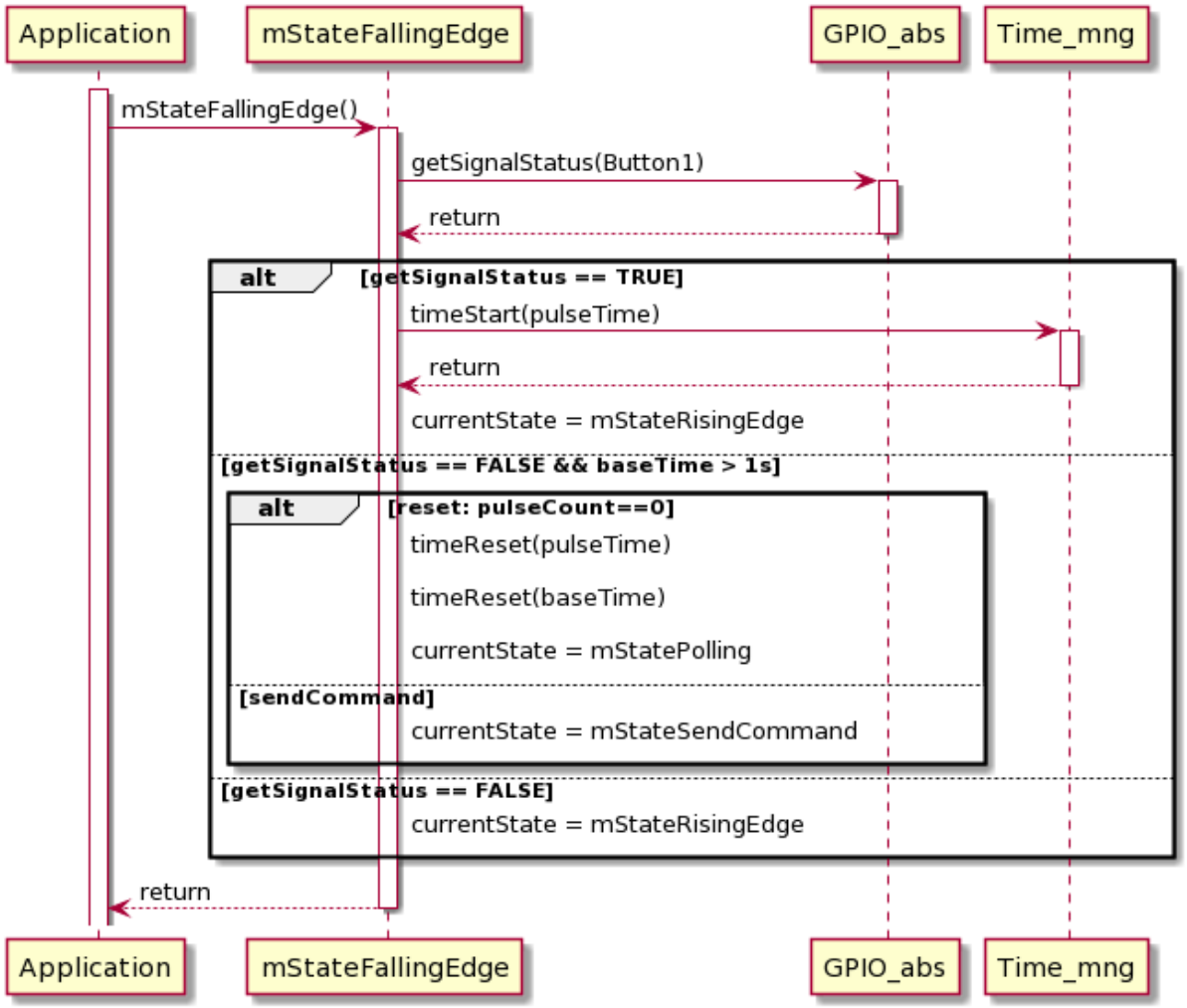


6.1.8 Function mStateFallingEdge()

Name	**mStateFallingEdge()**
Description	Waits for Falling edge detection at the GPIO interrupt corresponding to button input in pull-up configuration.
Syntax	void mStateFallingEdge(void)
Parameter 1 [input]	void
Return Value	null

Precondition	mStateRisingEdge()
Post condition	masterStateMachine() change state

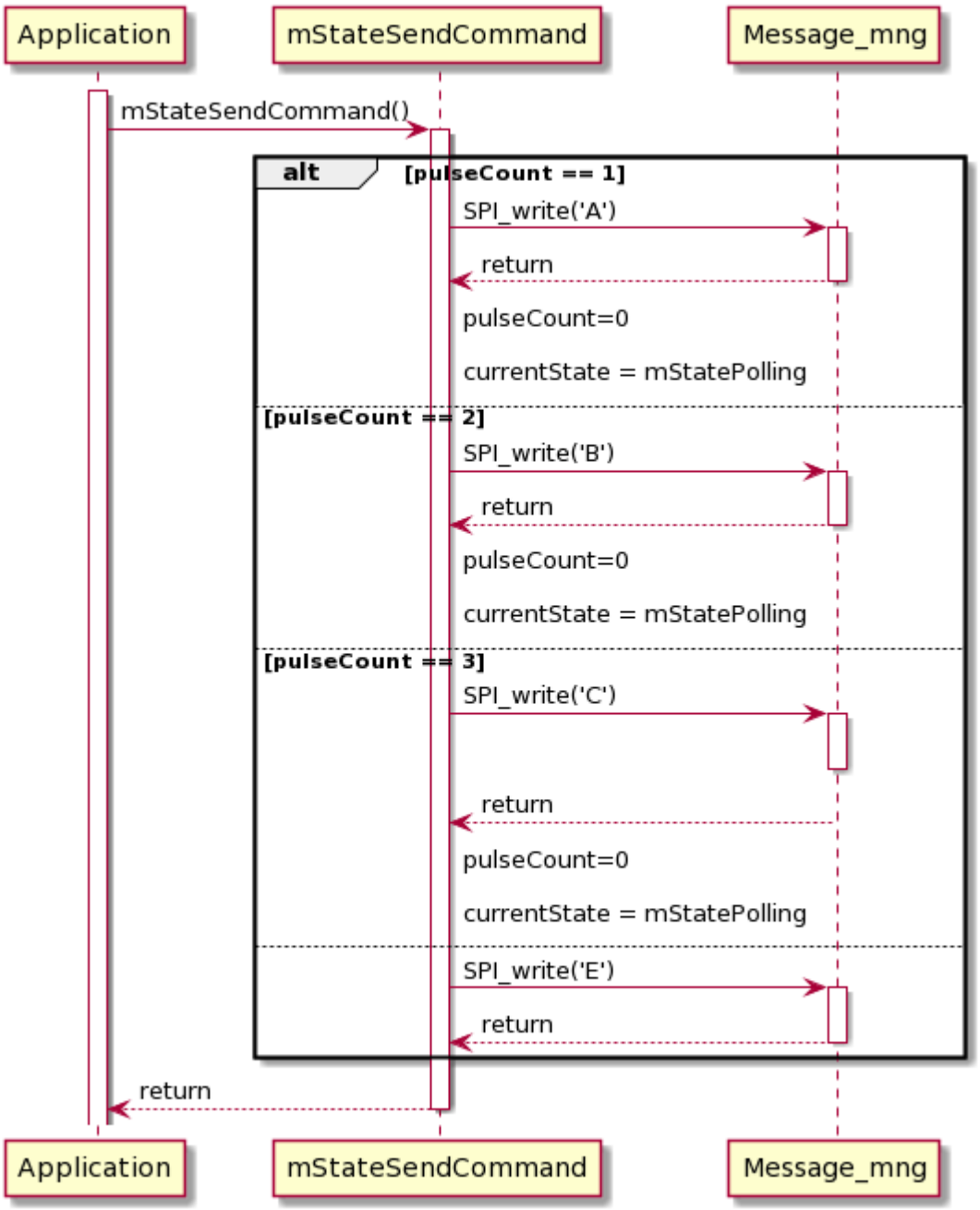
mStateFallingEdge() Dynamic Behaviour



6.1.9 Function mStateSendCommand()

Name	**mStateSendCommand()**
Description	The command is sent through the communication module of the Services layer, either CAN or SPI
Syntax	void mStateSendCommand(void)
Parameter 1 [input]	void
Return Value	null
Precondition	masterGeneralSetup() must be called at least once
Post condition	masterStateMachine() to state polling

mStateSendCommand() Dynamic Behaviour



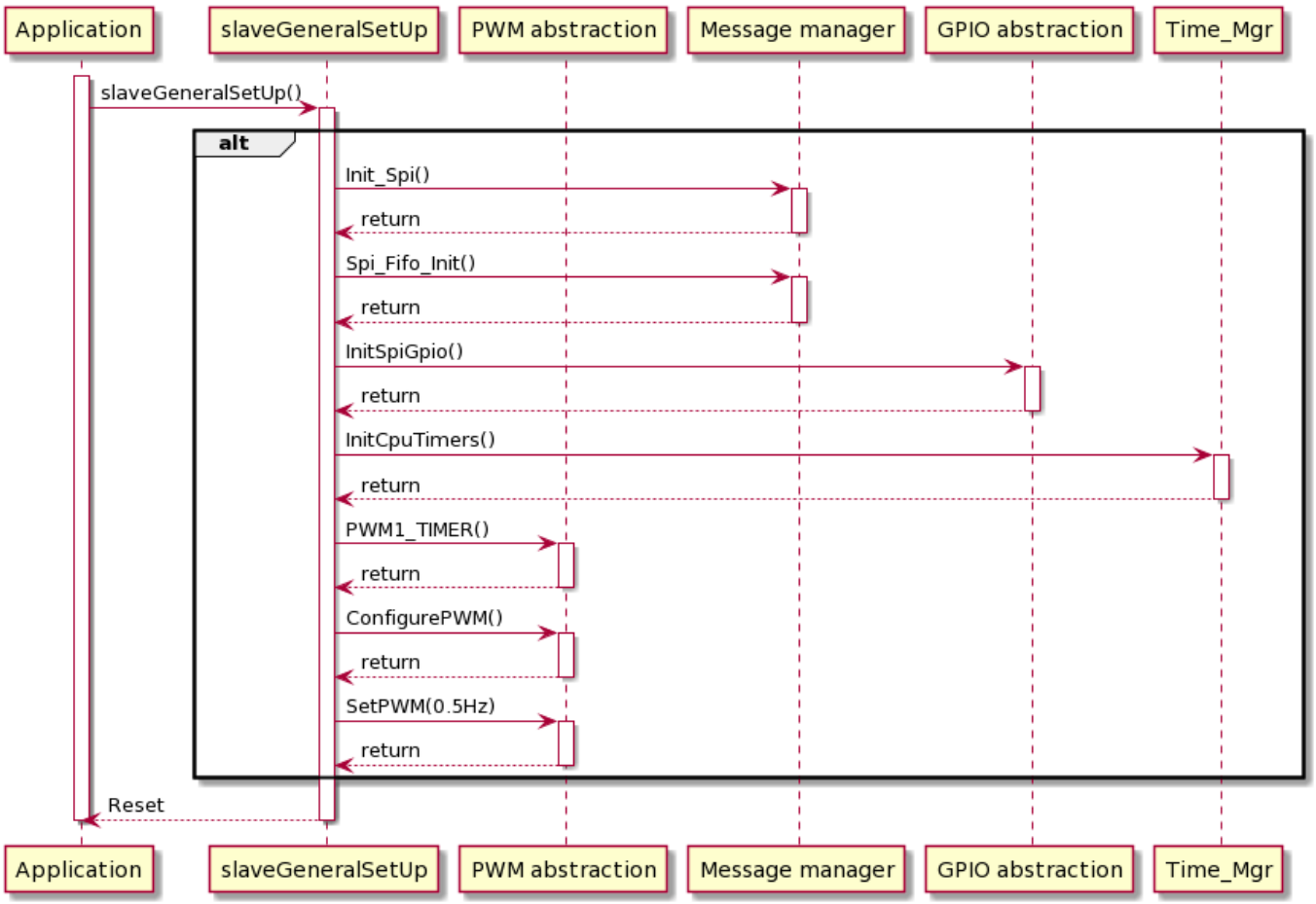
SLAVE BOARD

6.1.10 Function slaveGeneralSetup()

Name	slaveGeneralSetup()

Description	Sets the different aspects of the Slave Board for its correct operation, calling different function from the abstraction and services layers, the aspects that it will configure are: - InitSPI() - Spi_Fifo_Init() - InitSpiGpio() - InitCpuTimers() - PWM1_Timers() -ConfigurePWM()
Syntax	void slaveGeneralSetup(void)
Parameter 1 [input]	void
Return Value	null
Precondition	none
Post condition	none

slaveGeneralSetup() Dynamic Behaviour

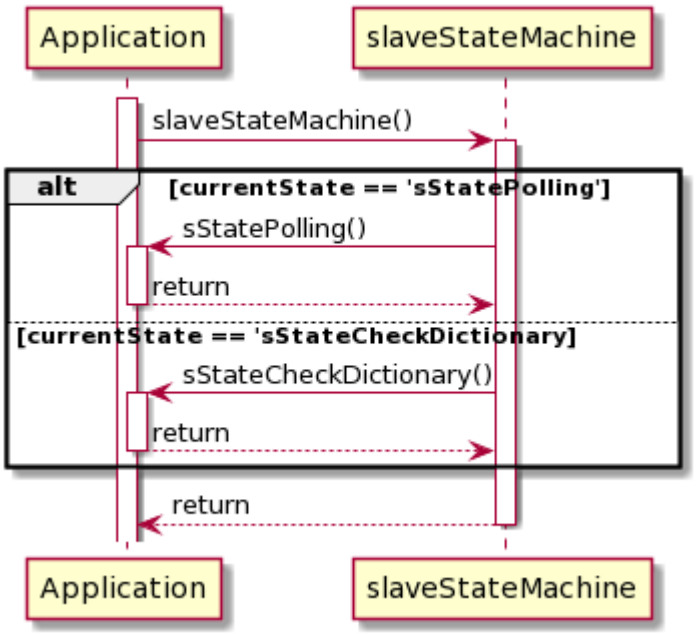


6.1.11 Function slaveStateMachine()

Name	slaveStateMachine()

Description	Slave board state machine, application component Current_state can be taken like actual machine state CurrentState = Polling, when at least 0.5hz PWM signal has been setted CurrentState = Set_PWM, after check command have a true condition to generate pwm a signal CurrenState = Check_Dicktionary after receive a msg sent by master
Syntax	Void slaveStateMachine(void)
Parameter 1 [input]	Void
Return Value	null
Precondition	slaveGeneralSetup() must be called at least once.
Post condition	sStatePolling() remains in the same state until another msg is received from the master board

SlaveStateMachine() Dynamic Behaviour

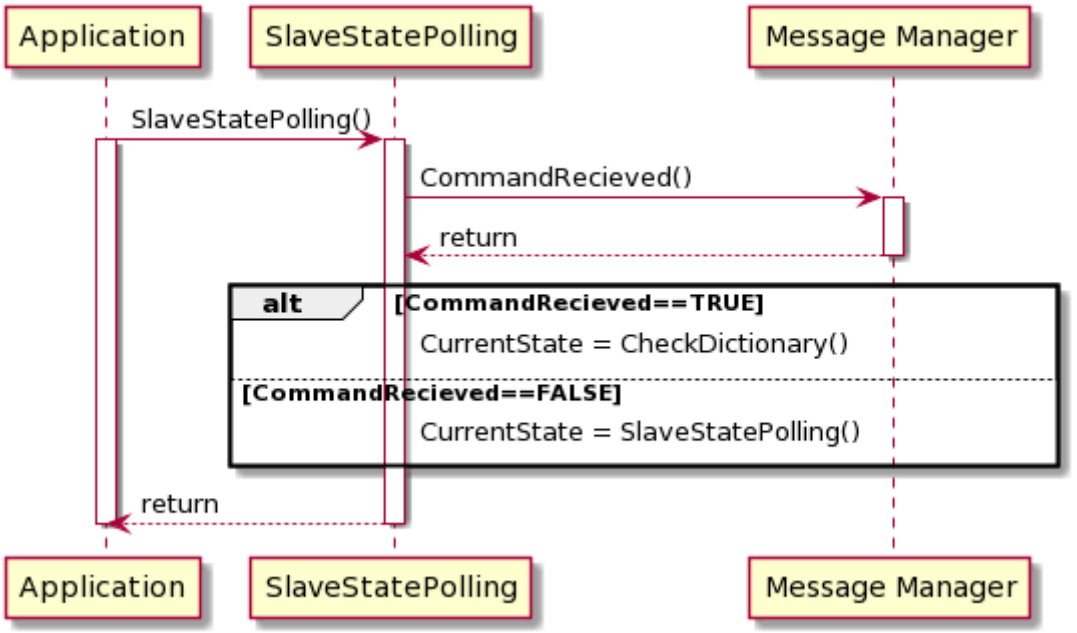


6.1.12 Function sStatePolling()

Name	**sStatePolling()**
Description	Calls a message manager function to check if a message has been received from the master board. **SlaveStatePolling()** changes its Current_State according to the operation that is running, Current_State remains in **SlaveStatePolling()** when no message is received or the system has been rebooted. Current_State will change in these cases: · When the system starts/has been rebooted, Current_State will change to the **SetPWMHZ_0.5z_State** to set the initial PWM signal to 0.5hz. · If a command has been received, Current_State changes to **MessageReceived_State** when it will be compared against the different conditions.

Syntax	void SlaveStatePolling(void)
Parameter 1 [input]	Void
Return Value	null
Precondition	SetPWMHZ () to 0.5hz and GeneralSetupSlave () must be called before the Slave State Polling () call
Post condition	SlaveStatePolling()remains in the same state until receiving another message from master

Dynamic Behaviour SlaveStatePolling()

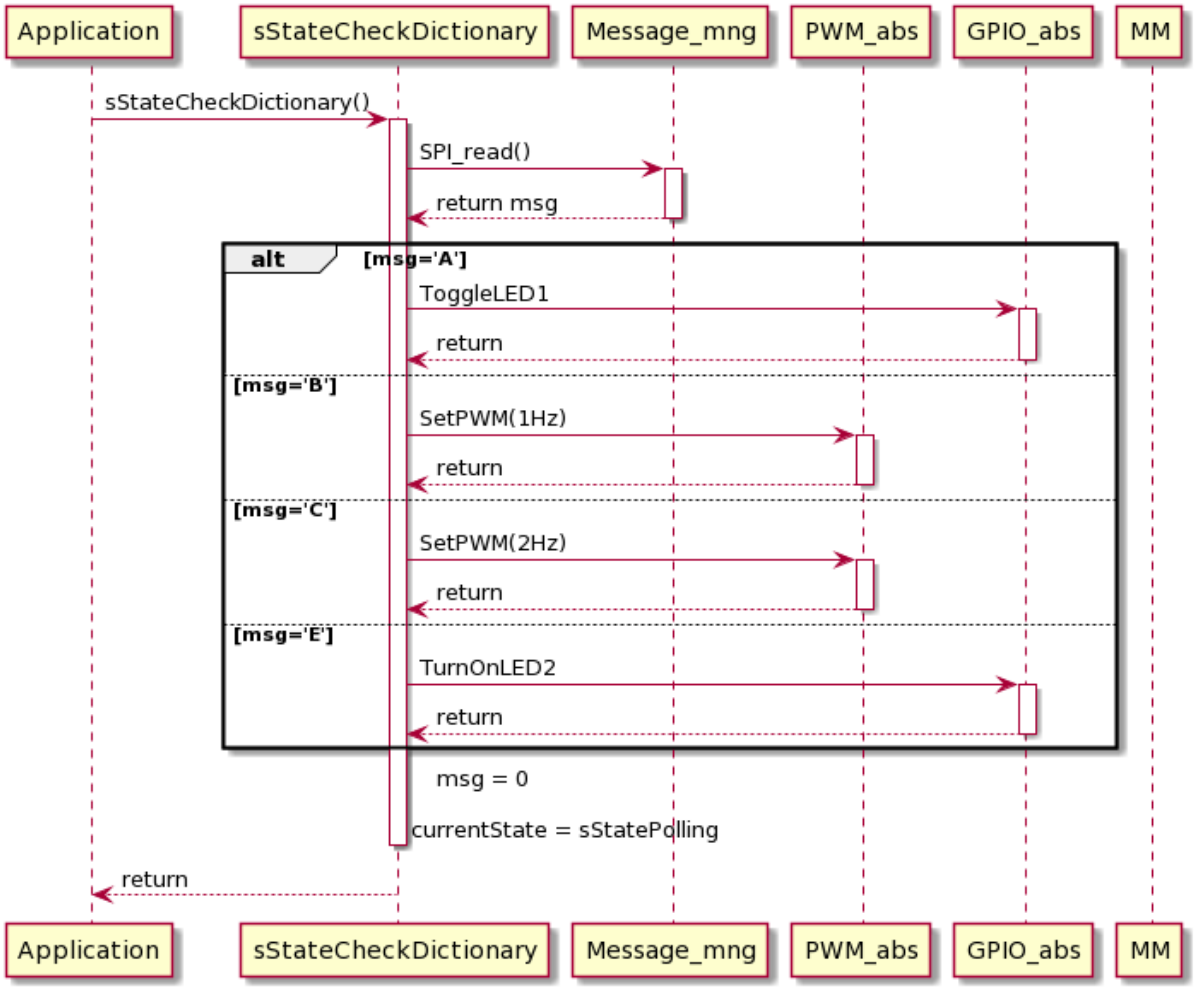


6.1.13 Function sStateCheckDictionary()

Name	**sStateCheckDictionary()**

Description	When a command is received from the SPI port read function, this function is called to interpret the same command. ● If a letter 'A' is received, the function will execute the function to make the Toggle on the first led (ToggleLED1). ● If the letter 'B' is received, then the function to be executed will be that of the setPWM with a frequency of 2Hz. ● If a letter 'C' is received, then the function to be executed will be that of the setPWM with a frequency of 1Hz. ● If a letter 'E' is received, then it is interpreted as an error and the LEd2 would be turned on with the TurnOnLED2 function. ● Once the status of the received command is set, the function returns to the StatePolling() function to wait for the next command from the master card.
Syntax	int CheckDictionary (int spiCommand)
Parameter 1 [input]	spiCommand
Return Value	null
Precondition	A command must be read in the SPI port and storaged in spiCommand variable.
Post condition	After executing this function, it will return to State_Polling to wait for another command from the master

Dynamic Behaviour


Application
sStateCheckDictionary
Message_mng
PWM_abs
GPIO_abs
MM
sStateCheckDictionary()
SPI_read()
return msg
alt
[msg='A']
ToggleLED1
return
[msg='B']
SetPWM(1Hz)
return
[msg='C']
SetPWM(2Hz)
return
[msg='E']
TurnOnLED2
return
msg = 0
currentState = sStatePolling
return
Application
sStateCheckDictionary
Message_mng
PWM_abs
GPIO_abs
MM