Smart Home project report

Submitted by:

Group Leader: **Samir Mosaad**

Group ***24***:

- **Marwan Atef**
- **Moustafa Ibrahim**
- **Zyad Mohamed**
- **Tarek Khaled**
- **Tarek Mohammed**
- **Moustafa Ahmed Emam**
- **AbdElJalil Nasser**
- **AbdElRahman Shawqy**

Submitted to:

Eng/ Mohamed Khaled

Table of Contents

Introduction:

This report, about the Smart Home project, includes documentation for project structure, different drivers implemented, their functionality and how they communicate with each other.

Project structure:

I. Header files:

Include global declarations and function prototypes for each driver to be implemented.

1. DIO
2. LCD
3. PWM
4. PORT
5. ADC
6. STEPPER
7. UART
8. SYSTICK
9. MACROS

II. C files:

Include functions implementations for each driver; initialization & operations.

1. DIO
2. LCD
3. PWM
4. PORT
5. ADC
6. STEPPER
7. UART
8. SYSTICK

III. Main files:

2 main files for each Tiva Launchpad, organize drivers' execution, function calls, and boards communication to deliver the required functionality.

Header files

1. DIO

A. Type Definitions:

i. Dio_LevelType

Type	*Enum*
Values	STD_LOW STD_HIGH

B. Function Definitions:

i. DIO_ReadPort

Input	**uint8** port_index **uint8** pins_mask
Return	**uint8** pins_level
Description	Return the value of the pins selected by pins_masks in the port selected by port_index.

ii. DIO_WritePort

Input	**uint8** port_index **uint8** pins_mask **Dio_LevelType** pins_level
Return	Void
Description	Change the value of the pins selected by pins_masks in the port selected by port_index to input pins_level.

iii. DIO_FlipPort

Input	**uint8** port_index **uint8** pins_mask
Return	Void
Description	Toggle the values of the pins selected by pins_masks in the port selected by port_index.

2. LCD

A. Function Definitions:

i. LCD_init

Input	Void
Return	Void
Description	Used to configure PORTA & PORTB to be connected to LCD and make it ready to display

ii. LCD_ sendCommand

Input	**uint8** command
Return	Void
Description	Used to send Commands to LCD like clear, move cursor ..etc.

iii. LCD_displayCharacter

Input	**uint8** data
Return	Void
Description	Used to display character on LCD

iv. LCD_integerToString

Input	**int** data
Return	Void
Description	Used to display intger on LCD

v. LCD_displayString

Input	**Const int** *data
Return	Void
Description	Used to display string on LCD

3. PWM

A. Type Definitions:

i. Pwm_TimerNumber

Type	*Enum*
Values	TIMER_0 TIMER_1 TIMER_2 TIMER_3 TIMER_4 TIMER_5

ii. Pwm_TimerMode

Type	*Enum*
Values	PWM ONE_SHOT REAL_TIME INPUT_EDGE

iii. Pwm_TimerA

Type	*Enum*
Values	TIMER_A_DISABLED TIMER_A_ENABLED

iv. Pwm_TimerB

Type	*Enum*
Values	TIMER_B_DISABLED TIMER_B_ENABLED

v. Pwm_TimerInversion

Type	*Enum*
Values	NON_INVERTED INVERTED

vi. Pwm_TimerConcatenate

Type	*Enum*
Values	CONCATENATE NO_CONCATENATION=4

vii. Pwm_TimerConfigStruct

Type	*Struct*
Members	**PWM_TimerNumber** PWM_TN **PWM_TimerMode** PWM_TM **PWM_TimerInversion** *PWM_TI **PWM_TimerConcatenate** PWM_TC **PWM_TimerA** PWM_TA **PWM_TimerB** PWM_TB **uint32_t** *PWM_PreScalar

B. Function Definitions:

iv. TIMER_init

Input	**const PWM_TimerConfigStruct ***
Return	Void
Description	Initializes the timer module

v. Timer_PwmOut

Input	**uint16_t**
Return	Void
Description	Map the ADC 12-Bit value to the right Duty Cycle value

4. PORT

A. Type Definitions:

i. Port_PinDirectionTyp

Type	*Enum*
Values	PORT_PIN_IN PORT_PIN_OUT

B. Function Definitions:

i. Port_Init

Input	**uint8** port_index
Return	Void
Description	Initialize port based on selected port_index [0:5] by enabling the clock, unlocking the port, and making the selected mode digital.

ii. Port_SetPinDirection

Input	**uint8** port_index **uint8** pins_mask **Port_PinDirectionType** pins_direction
Return	Void
Description	Change the direction of the selected pins by pins_mask in the port selected by port_index.

iii. Port_SetPinPullUp

Input	**uint8** port_index **uint8** pins_mask **uint8** enable
Return	Void
Description	If enable is 1, the selected pins by pins_mask in the port selected by port_index will be connected to internal pull-up resistor. If enable is 0, the selected pins by pins_mask in the port

	selected by port_index will be not be connected to internal pull-up resistor.

iv. Port_setPinPullDown

Input	**uint8** port_index **uint8** pins_mask **uint8** enable
Return	Void
Description	If enable is 1, the selected pins by pins_mask in the port selected by port_index will be connected to internal pulldown resistor. If enable is 0, the selected pins by pins_mask in the port selected by port_index will be not be connected to internal pull-down resistor.

5. SYSTICK

A. Type Definitions:

i. SysTick_clockSource

Type	*Enum*
Values	F_CPU_4 F_CPU

ii. SysTick_interrupt

Type	*Enum*
Values	INTERRUPT_DISABLED INTERRUPT_ENABLED

iii. SysTick_configure

Type	*Struct*
Members	**SysTick_clockSource** Clock **SysTick_interrupt** Interrupt

B. Function Definitions:

i. Systick_init

Input	**const SysTick_Configure ***
Return	Void
Description	Initialize systock module.

ii. Systick_delay

Input	**const uint32_t**
Return	Void
Description	Generate a delay in milliseconds.

6. UART

A. Type Definitions:

i. UART_Number

Type	*Enum*
Values	UART_0 UART_1 UART_2 UART_3 UART_4 UART_5 UART_6 UART_7

ii. UART_WordLength

Type	*Enum*
Values	BIT_5_ BIT_6_ BIT_7_ BIT_8_

iii. UART_FIFO

Type	*Enum*
Values	FIFO_DISABLED FIFO_ENABLED

iv. UART_StopBit

Type	*Enum*
Values	BIT_1_ BIT_2_

v. UART_ParityEnable

Type	*Enum*
Values	PARITY_DISABLED PARITY_ENABLED

vi. UART_ParitySelect

Type	*Enum*
Values	ODD_PARITY EVEN_PARITY

vii. UART_InterruptSelect

Type	*Enum*
Values	NO_INTERRUPT INTERRUPT

viii. UART_ConfigureStruct

Type	*Struct*
Members	**UART_Number** UN **UART_WordLength** UWL **UART_ParityEnable** UPE **UART_InterruptSelect** UIS **UART_ParitySelect** UPS **UART_StopBit** USB **UART_FIFO** UF

B. Function Definitions:

i. UART_init

Input	**UART_ConfigureStruct** *configure_pointer
Return	Void
Description	Initialize UART modules and enable system clock and GPIO clock.

ii. UART_sendByte

Input	**UART_Number** uNumber **const uint8_t** jOneChar
Return	Void
Description	Send one byte from selected UART channel.

iii. UART_sendString

Input	**UART_Number** uNumber **const uint8_t** *jOneWord
Return	Void
Description	Send one string from selected UART channel.

iv. UART_receiveByte

Input	**UART_Number** uNumber
Return	**uint8_t**
Description	Receive one byte from selected UART channel.

v. UART_receiveString

Input	**UART_Number** uNumber
Return	**uint8_t*** Word
Description	Receive one byte from selected UART channel.

vi. UART_setTransmitCallBack

Input	**UART_Number** uNumber
Return	Void
Description	Set the call back function which will be called in the interrupt service routine of the UART when transmission occurs.

vii. UART_setReceiveCallBack

Input	**UART_Number** uNumber
Return	Void
Description	Set the call back function which will be called in the interrupt service routine of the UART when receive occurs.

7. ADC

A. Type Definitions:

i. ADC_Number

Type	*Enum*
Values	ADC_0 ADC_1

ii. ADC_InterruptSelect

Type	*Enum*
Values	ADC_INTERRUPT_DISABLED ADC_INTERRUPT_ENABLED

iii. ADC_Sequencer

Type	*Enum*
Values	SEQUENCER_0 SEQUENCER_1 SEQUENCER_2 SEQUENCER_3

iv. ADC_EndOfConversion

Type	*Enum*
Values	SAMPLE_1 SAMPLE_2 SAMPLE_3 SAMPLE_4 SAMPLE_5 SAMPLE_6 SAMPLE_7 SAMPLE_8

v. ADC_TempSenseOrNormal

Type	*Enum*
Values	NORMAL_SELECT TEMP_SENSOR

vi. ADC_Sample

Type	*Struct*
Members	**suint8_t** SampleNumber **uint8_t** AnalogInput **uint8_t** LastSample **ADC_Sequencer** SequencerNumber **ADC_TempSenseOrNormal** T_OR_N **ADC_InterruptSelect** IS

i. ADC_ConfigureStruct

Type	*Struct*
Members	**ADC_Sample** * Samples **ADC_Number** AN **uint8_t** InterruptSelect_Mask **uint8_t** ActiveSequencer_Mask

B. Function Definitions:

i. ADC_init

Input	**ADC_ConfigureStruct** *configure_pointer
Return	Void
Description	Initiate any of the 2 ADC modules in the controller.

ii. ADC_readChannel

Input	**ADC_Number** AN
Return	**uint16_t**
Description	Read the Sequencer Result FIFO which contains the converted data.

iii. ADC_setISRCallBack

Input	**ADC_Number** AN
Return	void
Description	Set the call back function which will be called in the interrupt handler routine.

8. STEPPER

A. Type Definitions:

i. STEPPER_Pins

Type	**Enum**
Values	PIN_0 PIN_1 PIN_2 PIN_3 PIN_4 PIN_5 PIN_6 PIN_7

ii. STEPPER_ConfigStructure

Type	**Struct**
Members	**uint8_t** Port_Number **uint32_t** Por **STEPPER_Pins** Pins[4]

B. Function Definitions:

i. STEPPER_init

Input	**STEPPER_ConfigStructure ***
Return	Void
Description	Set the GPIO settings for the STEPPER motor.

ii. STEPPER_clockWise

Input	**const uint8_t**
Return	Void
Description	Rotate the STEPPER motor with any degree in clockwise direction.

iii. STEPPER_counterClockWise

Input	**const uint8_t**
Return	Void
Description	Rotate the STEPPER motor with any degree in counter clockwise direction.

Main files

In the main files came all modules initialization, function declarations and ports configuration whereas;

Tiva 1:

- Receive the value via UART0 and PWM the LED with the required value.
- Receive the character via UART1 and rotate the stepper motor with 30 in the required direction.
- Read the value of the temperature sensor and send it via UART2.

Tiva 2:

- Read the value of the potentiometer and send its ADC value via UART0.
- Check which button is pressed and send the associated char via UART1.
- Receive the value via UART2 and display it on the LCD.

References

A link for the github repository for the project:

https://github.com/shawqy/SmartHome_ARM