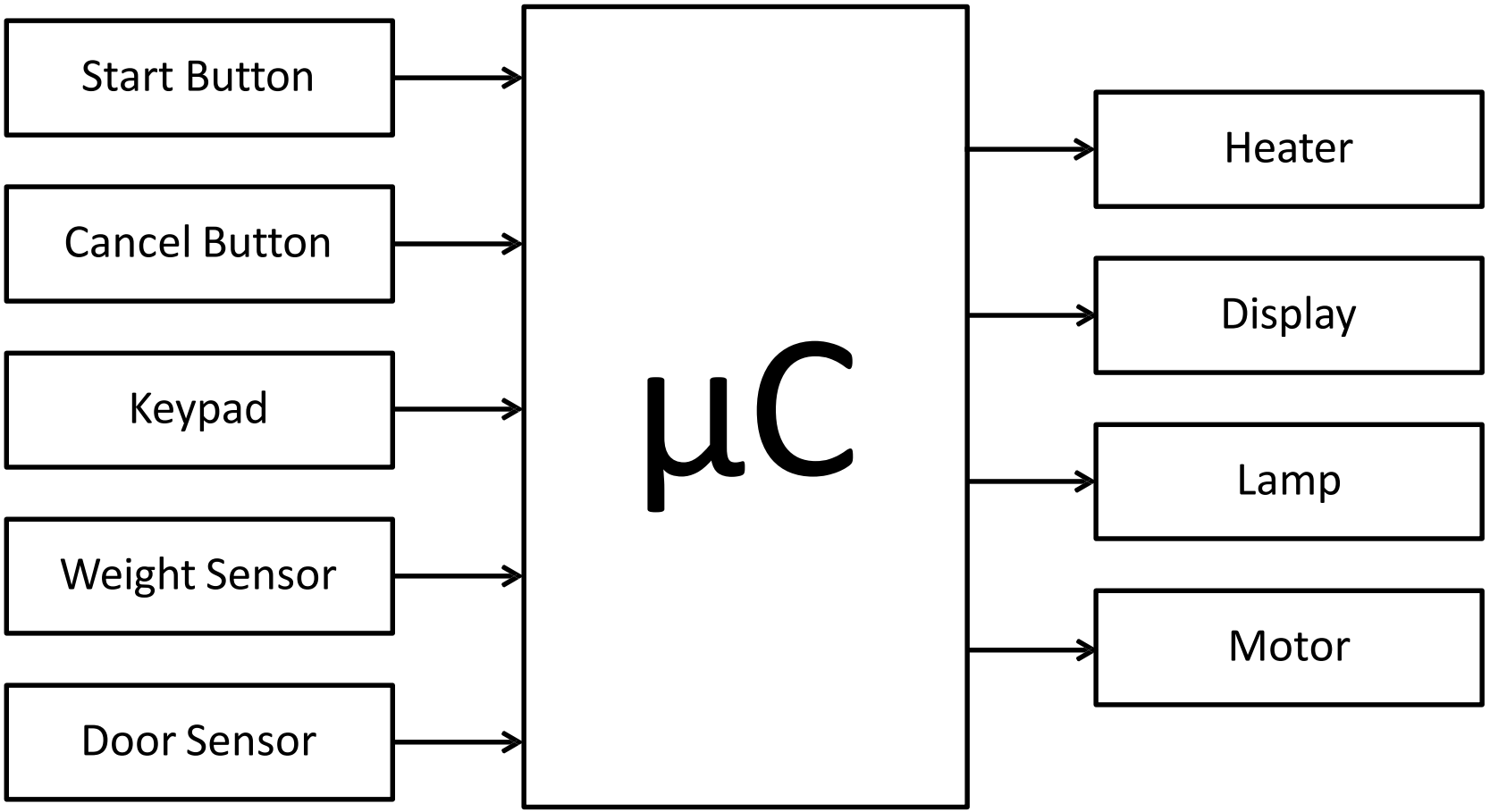
Microwave machine firmware

Swift Act challenge

By : Ayman Elhaddad

Overview


Start Button
Cancel Button
Keypad
Weight Sensor
Door Sensor
µC
Heater
Display
Lamp
Motor

Hardware

- Use PICSimLab v0.7
- Board: PICGenios, PIC18F452 controller , with :
  - Keypad
    [KEY_* to cancel / pause]
    [KEY_# to start / continue]
  - LCD
  - Two LEDs as lamps [B7 , B6]
  - Fan as motor.
  - Push buttons for door sensor [RB3] , and weight sensor [RB4] to toggle the sensors state (opened to closed, …)
  - Heater
- Using MPLAB IDE

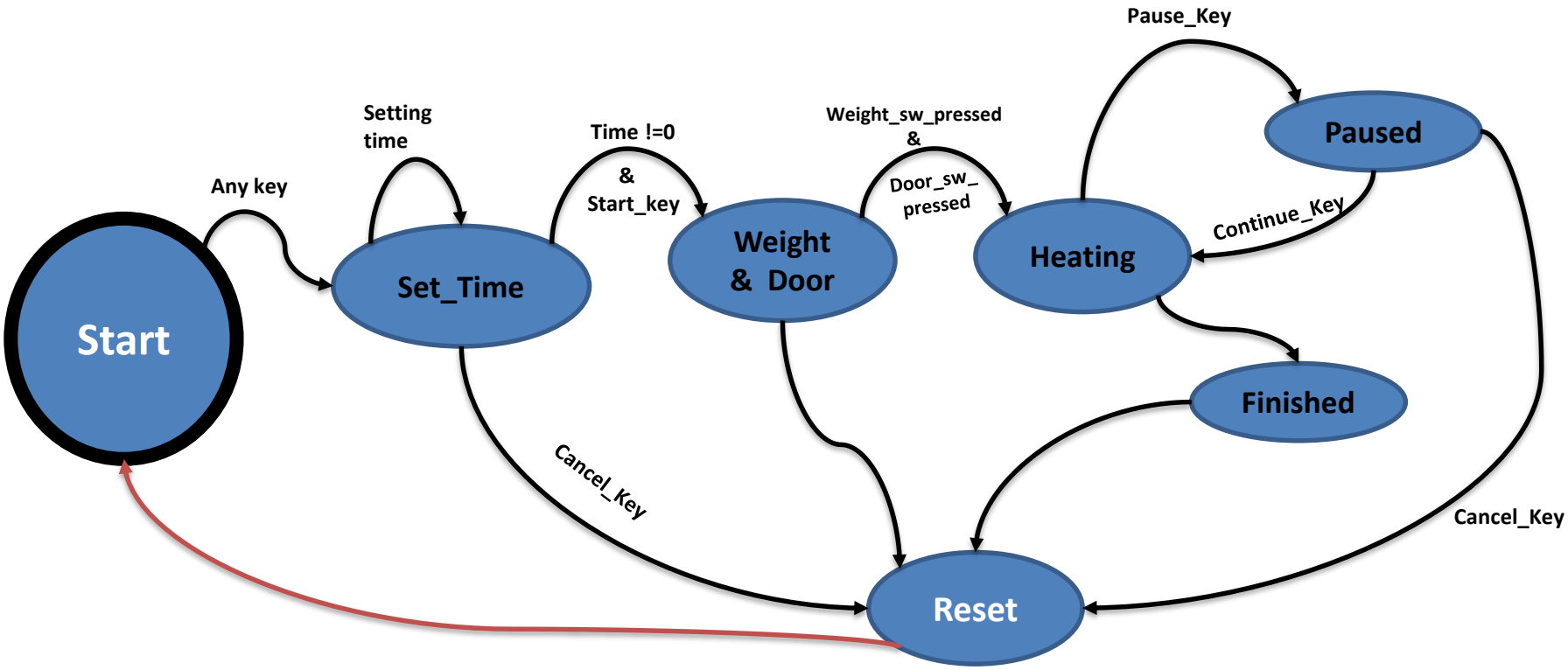
Specifications

- **Start button**
  - starts / Continue heating if:
    1. Time is set
    2. Door is closed
    3. Food is in microwave
- **Cancel button**
  - Pause heating if:
    1. Microwave is heating
  - Cancel heating if:
    1. Setting is not finished , “or” heating is paused .
- **Keypad** is used to enter the time of heating
- **LCD Display** displays time remaining if microwave is heating or displays [time setting or a message] if microwave is not heating.

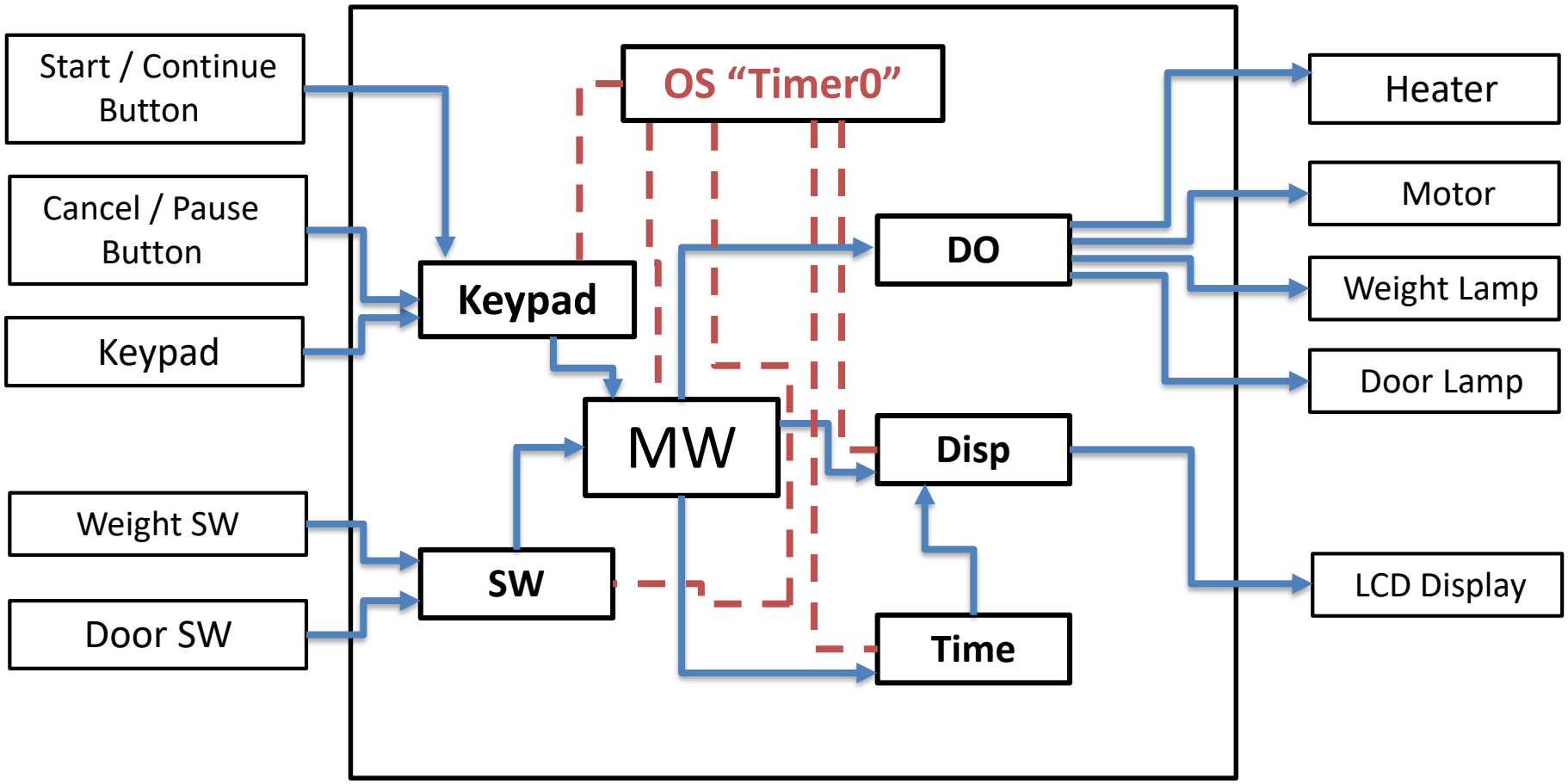
Specifications

- **Two Buttons** are used to toggle the state of Weight & Door sensors.
- **Two LEDs** are used to indicate the state of Weight & Door sensors , or toggling during the heating.
- When microwave in **heating** mode:
  - Two Lamps are Toggling
  - Heater is ON
  - Motor is ON
  - LCD Display shows remaining time
- When microwave in [**paused / cancel / finished**] mode:
  - Two Lamps are OFF
  - Heater is OFF
  - Motor is OFF
  - LCD Display shows a message, or time setting

State Machine



Context-Diagram



SW : Module

Function	Type
void SW_Init(tSW);	**Initialization** - Sets the sw pin direction and initial state
void SW_Update(void);	**Periodic Task** - Updates sw state based on scanned samples for each sw
tSW_State SW_GetState(tSW sw);	**Global function** - Returns the current sw state

Keypad : Module

Function	Type
void Keypad_Init (void);	**Initialization** - Sets the Keypad pins direction
void Keypad_Update (void);	**Periodic Task** - Calls the Keypad_Scan() function to indicate the new pressed key - Adds the new pressed key to the buffer if it exists
Keypad_Get_Data_From_buffer(tByte *pKey);	**Global function** - Returns the current pressed keys in buffer
static tByte Keypad_Scan (tByte *pKey);	**Private function** - Enables and disables rows pins , scans Columns to indicate the pressed key. - Returns if there is a new pressed key or not.

MW : Module

Function	Type
void MW_Update (void);	**Periodic Task** - Updates the microwave state , based on the inputs from keypad and sw modules - Enable/Disable the DO outputs (heater , motor , lamps) , based on the pressed key and the current state.

Disp : Module

Function	Type
void Set_Disp_State (tMW_State);	**Global function** - Sets the Disp state
void Disp_Update (void);	**Periodic Task** - Updates Disp_Time() function with the current MW_state - Receive time from time module
void Set_Disp_Current_Time_State (tCount_State);	**Global function** - Sets the Disp current count state
static void Disp_Time (tMW_State Current_MW_State);	**Private function** - Pass the time in the suitable form to the LCD to be displayed And call the blinking function based on the current count state
static void Disp_and_Blink_Time (tByte Min_Ten , tByte Min_Unit , tByte Sec_Ten , tByte Sec_Unit ,tBlink_State Blink_State);	**Private function** - Operates the blinking process , which is different based on the count state

Time : Module

Function	Type
void TIM_Init(void);	**Initialization** - Initialize time counters with initial values
void TIM_Update(void);	**Periodic Task** - Updates the time based on the current count state
void TIM_GetTime(tTIM_Time * time);	**Global function** - Returns the time
void Set_TIM_Current_Count_State (tCount_State);	**Global function** - Sets the TIM current count state
tByte Is_Finished (void);	**Global function** - Returns if the counting is finished or not
void Set_Time(tByte minutes_tens , tByte minutes_units , tByte seconds_tens , tByte seconds_units);	**Global function** - Sets the Time by the required values to start descending counting from them

LCD : Module

Function	Type
void LCD_Init(void);	**Initialization** - Set pins directions and initial values - Perform the 4-bit initialization steps and select the suitable display modes
void LCD_Update (void);	**Periodic Task** - Read the data stored in the buffer and assign it to LCD_SendChar() function
void LCD_SendChar(tByte character, tByte line, tByte column);	**Global function** - Displays a character to a specific place.
void LCD_SendString(tByte line, tByte * str , tByte col);	**Global function** - Displays a string starting from a specific place in the LCD.
void Set_LCD_BUFFER1 (tByte *BUF_1);	**Global functions** - To store string into the buffer , to be read by LCD_Update() and displayed.
void Set_LCD_BUFFER2 (tByte *BUF_2);	**Global functions** - To store string into the buffer , to be read by LCD_Update() and displayed.

LCD : Module

Function	Type
static void LCD_SendData(tByte character);	**Private function** - Used to cut the data in half to be sent in two processes , to be suitable to the 4-bit mode
static void LCD_GoTo(tByte line, tByte column);	**Private function** - Calls a specific command , which describes the needed position in the LCD.
static void LCD_Port(tByte Data);	**Private function** - Assigns data to the port pins
static void LCD_SendCommand(tByte command);	**Private function** - Sends commands to LCD.
void LCD_Clear(void);	**Global functions** - Clears The LCD.

Timer0 : Module

Function	Type
void TMR_Init(void);	**Initialization** - Initialize time counters with initial values
void TIM_Update(void);	**Periodic Task** - Updates the time based on the current count state
void __interrupt() TMR_Update(void) ;	**ISR Function** - Interrupt service routine , occurs every time tick . Used as OS
void TMR_Start(void);	**Global function** - Starts the timer0 , by updating the register with the required tick
tByte TMR_CheckOverFlow(void);	**Global function** - Returns the overflow flag state.
void TMR_Stop(void);	**Global function** - Disables the Timer0.

DO : Module

Function	Type
void DO_Init(void);	**Initialization** - Sets pins directions and initial values
void DO_SetState(tDO device ,tDO_State state);	**Global function** - Sets the state of every Digital Output , as it can be ON / OFF states
void Toggle_Lamps (void);	**Global function** - Toggles the two Lamps during heating state

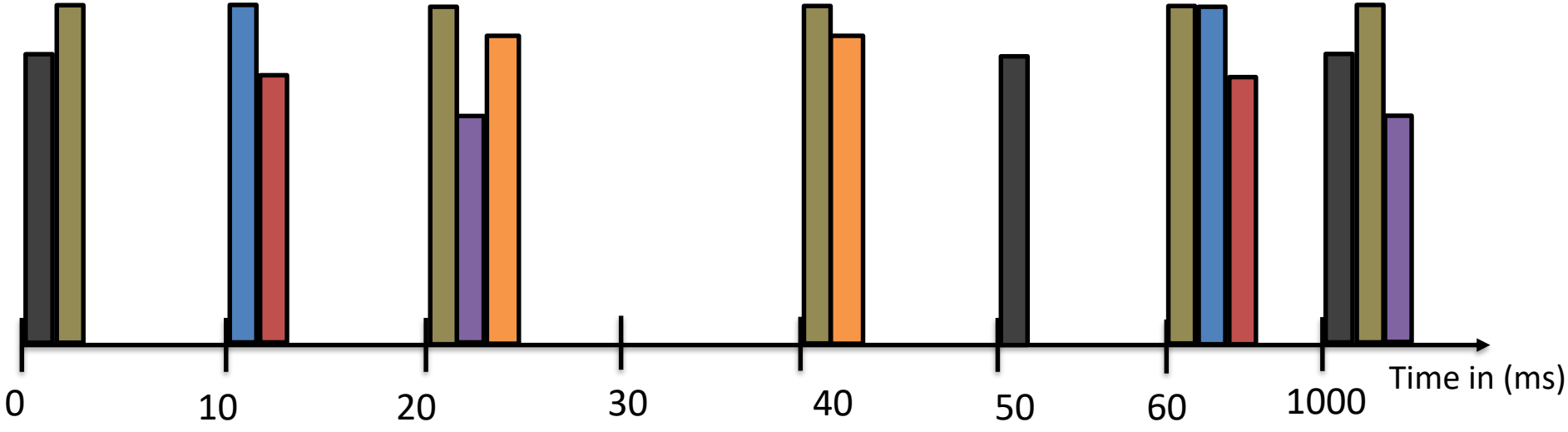
Dynamic Design

Using **Time Triggered** OS , with 10 ms period

	Task	Periodic Actions	Period of periodic actions	Period of task (ms)
Inputs	**SW**	- Samples - State	20 20	20
Inputs	**Keypad**	- Key	50	50
Processing	**MW**	- State	50	50
Processing	**Time**	- Seconds	1000	1000
Processing	**Disp**	- State	50	50
Outputs →	**LCD**	- Buffer	30	30

The common factor = 10 ms

Timeline of periodic tasks



SW

MW

Keypad

Disp

LCD

Time


thank you
danke
рахмат
謝謝
ngiyabonga
teşekkür ederim
Баярлалаа
спасибо
gracias
tapadh leat
obrigado
dziękuję
bedankt
sukriya
kop khun krap
go raibh maith agat
mochchakkeram
merci
arigatô
dakujem
grazie
terima kasih
감사합니다
ευχαριστώ
мерси
хвала
dank je
hvala
kiitos
dankie
asante
tack
mersi
xiexie
ありがとう