### **Air Conditioning System**

### Team 2

- 1- Mohab Ahmed
- 2- Anas Mahmoud
- 3- Mustafa Mohammed
- 4- Omar Taha

### Table of Contents

1-Description	3
1.1 Hardware components	3
1.2 Software requirements	3
2-Layered architecture	4
3-System Flow Chart	5
4-Schematic Capture	6
5-Drivers Description	7
5.1 DIO Driver	7
5.2 Timer Driver	7
5.3 ADC Driver	7
5.4 Keypad Driver	7
5.5 LCD Driver	8
5.6 Application Driver	8
6-API's	9
6.1 DIO Driver	9
6.2 Timer0 Driver	9
6.3 Timer2 Driver	9
6.4 ADC Driver	10
6.5 Keypad Driver	10
6.6 LCD Driver	10
6.7 APP Driver	11
7- API's Flow Chart	12
7.1 LCD	12
7.2 Keypad	25
7.3 ADC	27
7.4 buzzer	29
7.5 Timer 2	31
7.6 App	36

### 1-Description

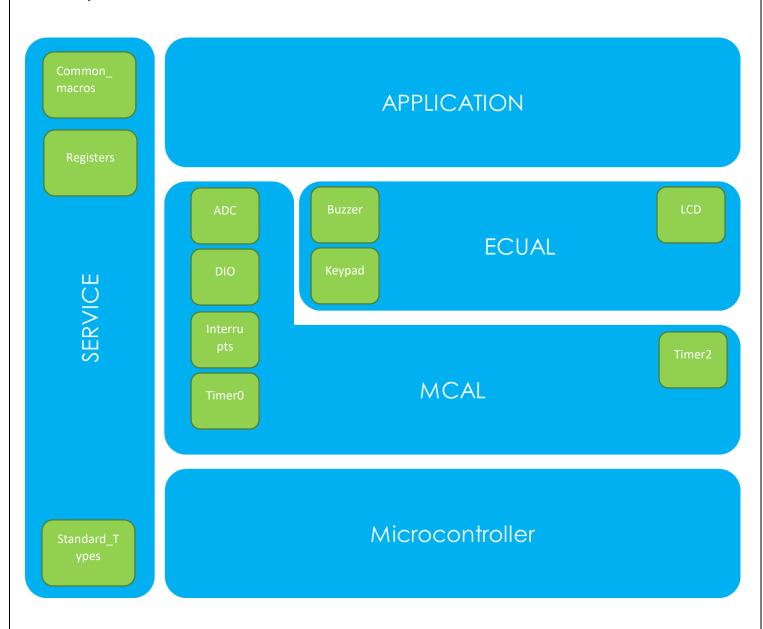
### 1.1 Hardware components

- 1. LCD (2\*16)
- 2. Keypad (3 \* 3) (Note: 4 buttons will be used)
- 3. Temperature sensor (LM35)
- 4. Buzzer

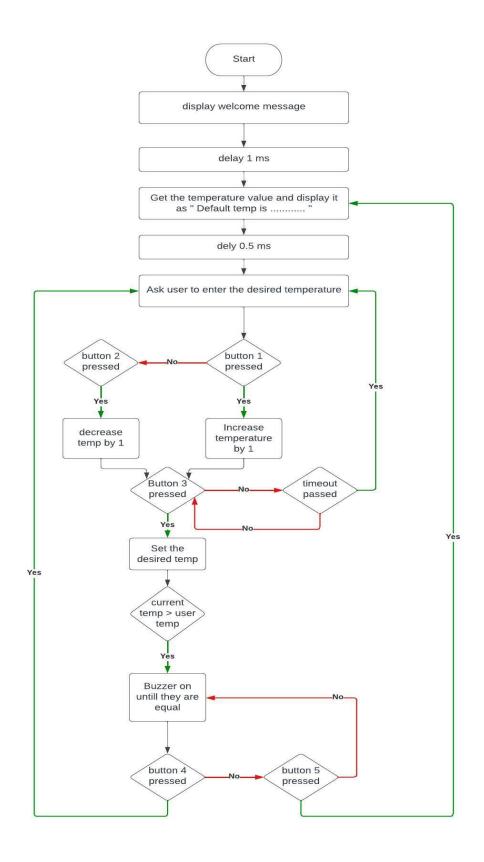
### 1.2 Software requirements

- 1. The system starts initially by welcoming the user
- 2. Then inform the user about the default temp and ask for desired one
- 3. Display range of temperature starts from min 18 and to max 35 C
- 4. To enter the desired temperature user has access through two buttons button one to increase and button two to decrease above or below the default temperature
- 5. Once button three is pressed the desired temperature will be set
- 6. If the desired temperature is greater than the default one buzzer will be on until both are equal
- 7. If button 4 is pressed at any time that will take the user back to set another desired temperature
- 8. Button 5 will be used to reset the system to start again

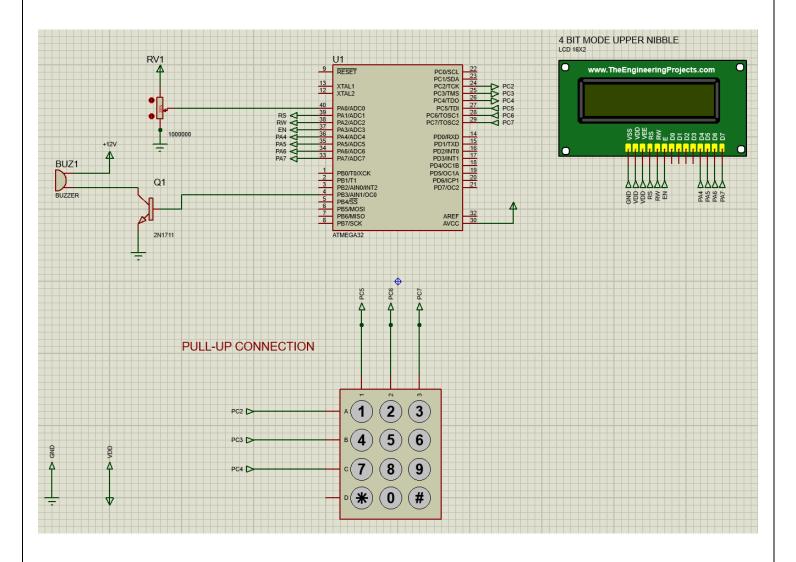
### 2-Layered architecture



### 3-System Flow Chart



### 4-Schematic Capture



### 5-Drivers Description

### 5.1 DIO Driver

Configuration: Consist of 4 API's

Location: MCAL

Function: used to set pin direction (input or output), pin value (high or low) or read a value

from a pin or toggle a pin

### 5.2 Timer Driver

Configuration: Consist of 5 API's

**Location: MCAL** 

Function: used to set a time delay

### 5.3 ADC Driver

Configuration: Consist of 2 API's

**Location: MCAL** 

Function: used to initialize ADC, read the value of ADC

### 5.4 Keypad Driver

Configuration: Consist of 2 API's

Location: HAL

Function: used to initialize the keypad, get pressed key

### 5.5 LCD Driver

Configuration: Consist of 14 API's

Location: HAL

Function: used to initialize the LCD, send command to LCD & display character or string to LCD & jump to specific position on LCD & to clear the LCD & to wright integer or float number on the LCD

### 5.6 Application Driver

Configuration: Consist of 9 API's

**Location: App** 

Function: combine between the drivers API's to meet the requirement

### 6-API's

### 6.1 DIO Driver

```
1- PinDirection_t DIO_setpindir (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pinid);
2- PinValue_t DIO_setpinvalue (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t u8_a_pinid);
3- PinRead_t DIO_readpin (uint8_t u8_a_portid, uint8_t u8_a_pinid, uint8_t* u8_a_val);
4- PinRead_t DIO_togglepin (uint8_t u8_a_portid, uint8_t u8_a_pinid);
```

### 6.2 Timer Driver

```
1- TMR0_init_error TMR0_init(void);
2- TMR0_start_error TMR0_start(void);
3- TMR0_stop_error TMR0_stop(void);
4- TMR0_delay_error TMR0_delayms(uint32_t u32_a_delayms);
5- TMR0_delay_error TMR0_delaymicos(uint32_t u32_a_delaymicros);
```

### 6.3 Timer2 Driver

```
1- err_state TIMER2_normalMode(void);
2- err_state TIMER2_initialValue(uint8_t value);
3- err_state TIMER2_perscalerMode(unsigned int prescaler);
4- err_state TIMER2_delay(float f_a_delayInMillis);
5- unsigned int TIMER2_getInitialValue(float f_a_delayInMillis);
```

```
6.4 ADC Driver
1- ADC initstatus ADC_Init(void);
2- uint16 t ADC read(void);
6.5 Keypad Driver
1- void KEYPAD init(void);
2- uint8 t KEYPAD getpressedkey(void);
6.6 LCD Driver
1- LCD init error LCD 8 bit init (void);
2- LCD_sendCommand_error LCD_8_bit_sendCommand(uint8_t u8_a_command);
3- LCD_sendChar_error LCD_8_bit_sendChar(uint8_t u8_a_char);
4- LCD init error LCD 4 bit init(void);
5- LCD sendCommand error LCD 4 bit sendCommand(uint8 t u8 a command);
6- LCD sendChar error LCD 4 bit sendChar(uint8 t u8 a char);
7- LCD_sendString_error LCD_sendString(uint8_t *u8_a_string);
8- void LCD goTo(uint8 t u8 a row, uint8 t u8 a column);
9- void LCD createCustomCharacter(uint8 t *u8 a bitMap,uint8 t u8 a location)
10- LCD init error LCD init(void);
11- LCD_sendCommand_error LCD_sendCommand(uint8 t u8 a command);
12- LCD sendChar error LCD sendChar(uint8 t u8 a char);
13- LCD_sendChar_error LCD_sendFloat(float f_a_number);
14- LCD_sendChar_error LCD_sendInteger(uint16_t u16_a_number);
```

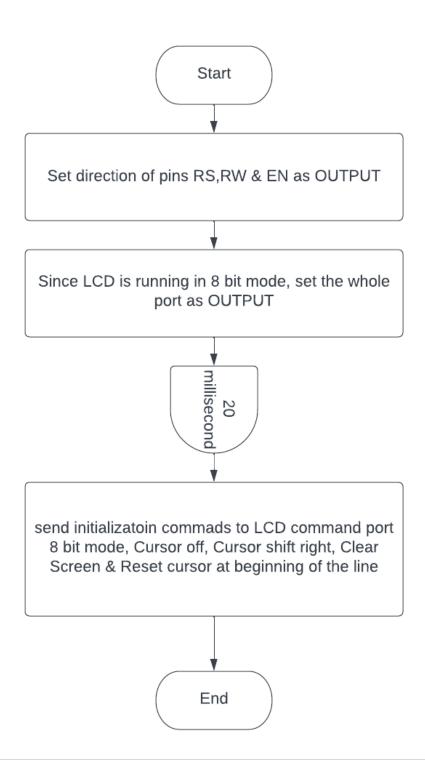
### 6.7 APP Driver

```
1- APP_initError APP_init(void);
2-void APP_printString(uint8_t u8_a_row, uint8_t u8_a_column, uint8_t *u8_a_str);
3-void APP_printChar(uint8_t u8_a_row, uint8_t u8_a_column, uint8_t u8_a_char);
4-void APP_printInteger(uint8_t u8_a_row, uint8_t u8_a_column, uint16_t u16_a_integer);
5-void APP_setTemp();
6-void APP_decrementBar();
7-void APP_incrementBar();
8-void APP_defaultView(void);
9- void get_current_overflow(void);
```

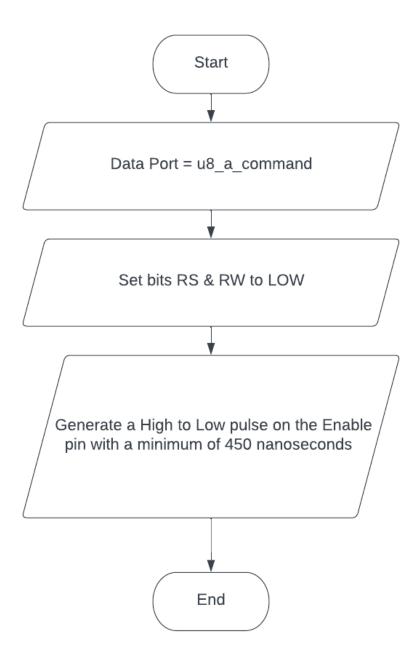
### 7- API's Flow Chart

### 7.1 LCD

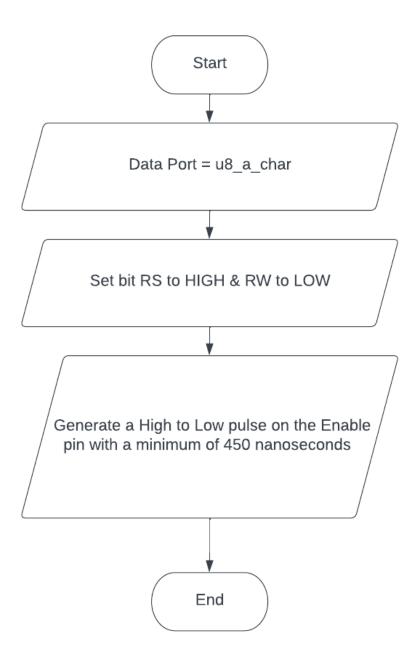
### LCD\_init\_error LCD\_8\_bit\_init(void);



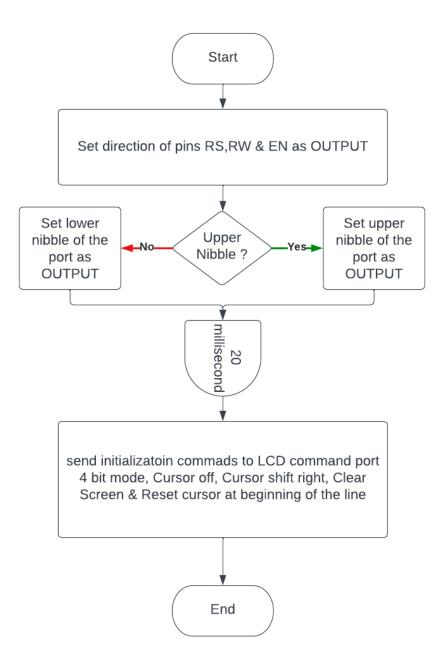
### LCD\_sendCommand\_error LCD\_8\_bit\_sendCommand(uint8\_t u8\_a\_command);



### LCD\_sendChar\_error LCD\_8\_bit\_sendChar(uint8\_t u8\_a\_char);



### LCD\_init\_error LCD\_4\_bit\_init(void);



### LCD\_sendCommand\_error LCD\_4\_bit\_sendCommand(uint8\_t u8\_a\_command); Start Upper LCD\_Data\_Port = (LCD\_Data\_Port & 0xF0) LCD\_Data\_Port = (LCD\_Data\_Port & 0x0F) | No-Yes-Nibble? | ((u8\_a\_command & 0xF0) >> 4) ((u8\_a\_command & 0xF0)) Set bits RS & RW to LOW Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds **Upper** LCD\_Data\_Port = (LCD\_Data\_Port & 0xF0) LCD\_Data\_Port = (LCD\_Data\_Port & 0x0F) | Nibble? ((u8\_a\_command & 0x0F)) ((u8\_a\_command & 0x0F) << 4) Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds End

### LCD\_sendChar\_error LCD\_4\_bit\_sendChar(uint8\_t u8\_a\_char); Start Upper LCD\_Data\_Port = (LCD\_Data\_Port & 0xF0) LCD\_Data\_Port = (LCD\_Data\_Port & 0x0F) | Nibble? | ((u8\_a\_char & 0xF0) >> 4) ((u8\_a\_char & 0xF0)) Set bits RS to HIGH & RW to LOW Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds **Upper** LCD\_Data\_Port = (LCD\_Data\_Port & 0xF0) LCD\_Data\_Port = (LCD\_Data\_Port & 0x0F) | No-Nibble? ((u8\_a\_char & 0x0F)) ((u8\_a\_char & 0x0F) << 4) Generate a High to Low pulse on the Enable pin with a minimum of 450 nanoseconds End

# LCD\_sendString\_error LCD\_sendString(uint8\_t \*u8\_a\_string); Start 4 bit mode? Yes-LCD\_4\_bit\_sendChar(u8\_a\_string[u16\_l\_charCount]) LCD\_8\_bit\_sendChar(u8\_a\_string[u16\_l\_charCount]) u8\_a\_string[u16\_l\_charCount!=null Νo End

# void LCD\_goTo(uint8\_t u8\_a\_row,uint8\_t u8\_a\_column); Start uint8\_t[]={Cursor\_Reset\_Line1,Cursor\_Reset\_Line2} LCD\_8\_bit\_sendCommand(u8\_I\_rows[u8\_a\_row]+u8\_a\_coloumn) 4 bit mode? LCD\_4\_bit\_sendCommand(u8\_I\_rows[u8\_a\_row]+u8\_a\_coloumn) End **19** | Page

# void LCD\_createCustomCharacter(uint8\_t \*u8\_a\_bitMap,uint8\_t u8\_a\_location); Start u8\_I\_count=0 4 bit mode? LCD\_8\_bit\_sendCommand( CGRAM + (u8\_a\_location\*8) ) LCD\_4\_bit\_sendCommand(CGRAM+(u8\_a\_location\*8)); u8\_I\_count<8? u8\_I\_count<8? -No--No-Yes Yes LCD\_8\_bit\_sendChar(u8\_a\_bitMap[u8\_l\_count]) End

# LCD\_init\_error LCD\_init(void); Start 4 bit mode? End **21** | Page

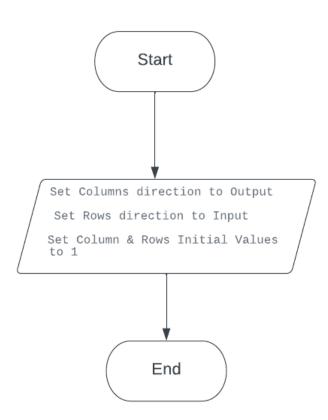
# LCD\_sendCommand\_error LCD\_sendCommand(uint8\_t u8\_a\_command); Start 4 bit mode? LCD\_8\_bit\_sendCommand(u8\_a\_command) -Yes► LCD\_4\_bit\_sendCommand(u8\_a\_command) End **22 |** Page

# LCD\_sendChar\_error LCD\_sendChar(uint8\_t u8\_a\_char); Start 4 bit mode? LCD\_4\_bit\_sendChar(u8\_a\_char) -Yes**⊳** End **23** | Page

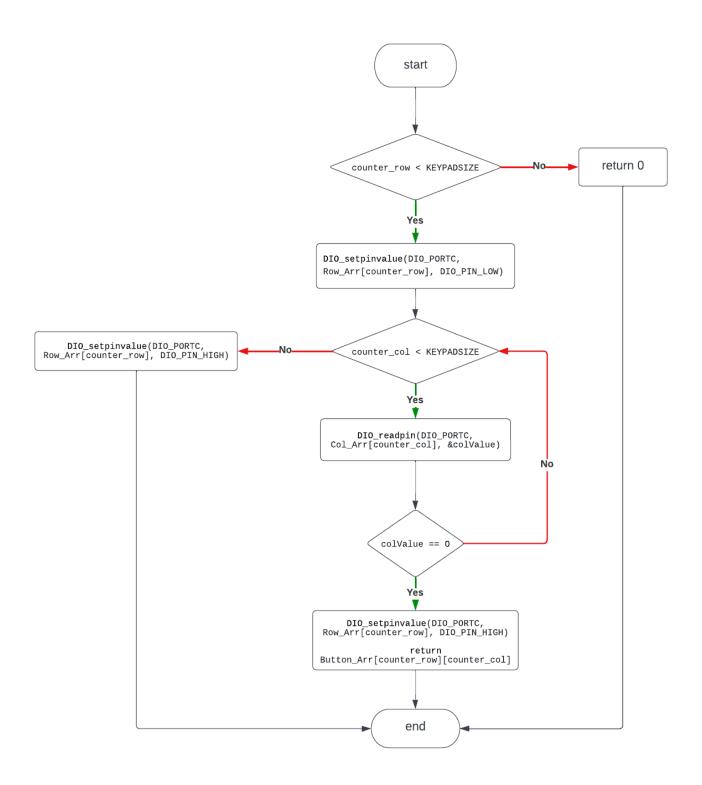
## sendChar\_error LCD\_sendInteger(uint16\_t u16\_a\_number); Start uint16\_t u\_l\_toString[10]; uint8\_t i =0; $uint8\_t j = 0;$ LCD\_sendChar('0'); -u16\_a\_number==0? No j>0 (u16\_a\_number>0)-Noj=i Yes u\_l\_toString[i]=(u16\_a\_number % 10) +48; u16\_a\_number/=10; Νo End

### 7.2 Keypad

### void KEYPAD\_init(void)

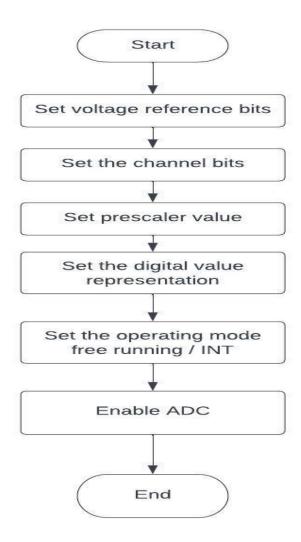


### uint8\_t KEYPAD\_getpressedkey(void)

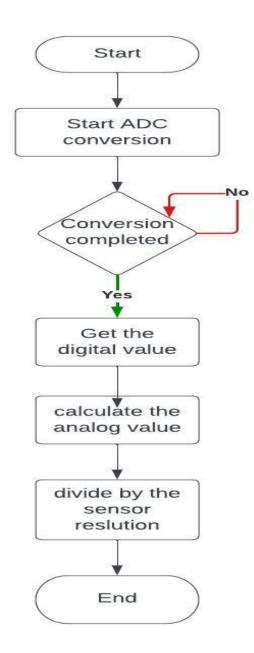


### 7.3 ADC

### ADC\_initstatus ADC\_init(void);

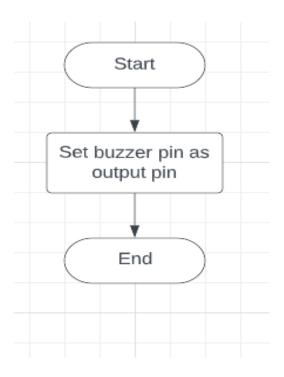


### uint32\_t ADC\_read(void);

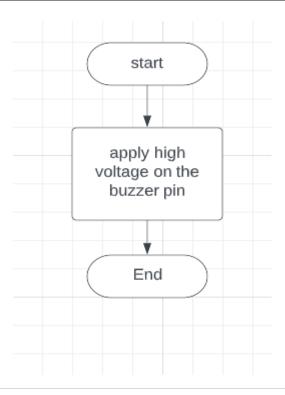


### 7.4 buzzer

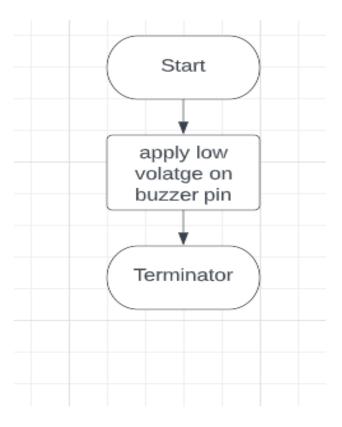
### void BUZZ\_init();



### void BUZZ\_on();

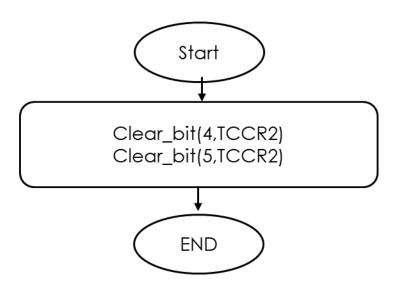


### void BUZZ\_off();

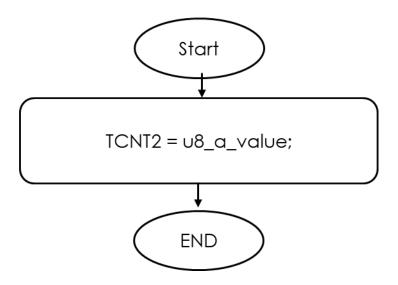


### 7.5 Timer 2

err\_state TIMER2\_normalMode(void);



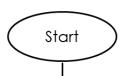
### err\_state TIMER2\_initialValue(uint8\_t value);



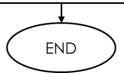
## err\_state TIMER2\_perscalerMode(unsigned int u16\_a\_prescaler); Start switch(u16\_a\_prescaler) TCCR2 | = 0x00 TCCR2 | = 0X01 TCCR2 | = 0X02 TCCR2 | = 0X03 END Return SUCCESS e 64 e 256 TCCR2 | = 0X04 TCCR2 | = 0X05 el0 TCCR2 | = 0X06 TCCR2 | = 0X07 e 12 Return Fail **33 |** Page

## err\_state TIMER2\_delay(float f\_a\_delayInMillis); Start U16\_I\_numberOfOVF= TIMER2\_getInitialValue(d\_a\_delayIn Millis) No u16\_l\_count < u16\_l\_numberOfO END TCCR2 = 0;TCNT0 = u16\_gs\_t2\_initial\_value; TIMER2\_prescalerMode(PRESCALER\_M QDE); ( TIFR & (1<<6) ) == 0 Yes TCCR2 = 0;Set\_Bit(6,TIFR); u16\_l\_count++;

### unsigned int TIMER2\_getInitialValue(float f\_a\_delayInMillis);

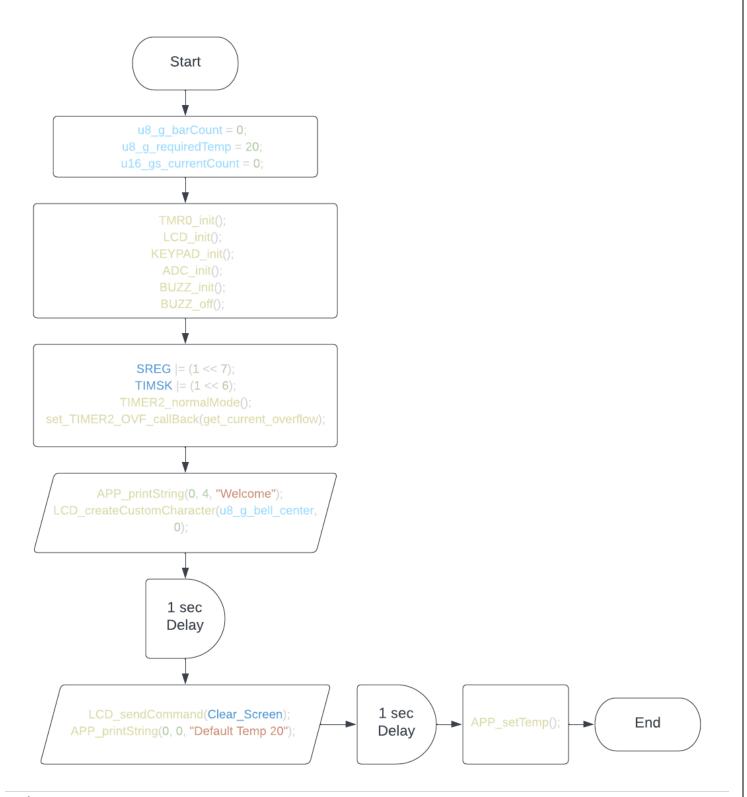


 $f\_a\_delayInMillis = f\_a\_delayInMillis* (0.001); \\ u16\_l\_numberOfOVF = ceil(f\_a\_delayInMillis / t2\_max\_delay ); \\ u16\_gs\_t2\_initial\_value = 256.0 - ceil( (f\_a\_delayInMillis/t\_tick) / u16\_l\_numberOfOVF ); \\ return u16\_l\_numberOfOVF; \\ \end{cases}$ 

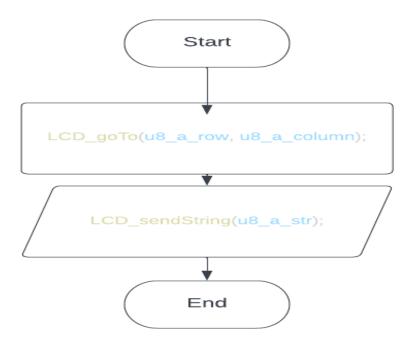


### 7.6 App

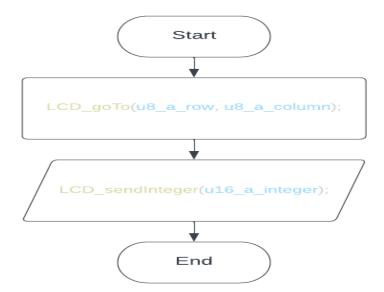
### initError APP\_init(void);



### void APP\_printString(uint8\_t u8\_a\_row, uint8\_t u8\_a\_column, uint8\_t \*u8\_a\_str);

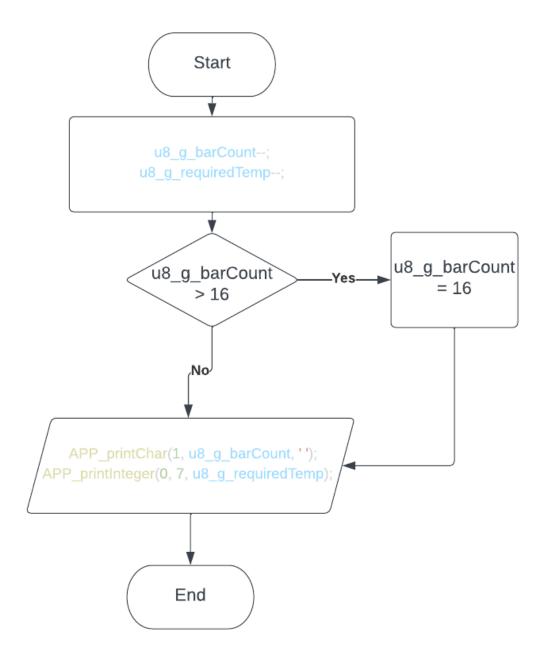


### printInteger(uint8\_t u8\_a\_row, uint8\_t u8\_a\_column, uint16\_t u16\_a\_integer);



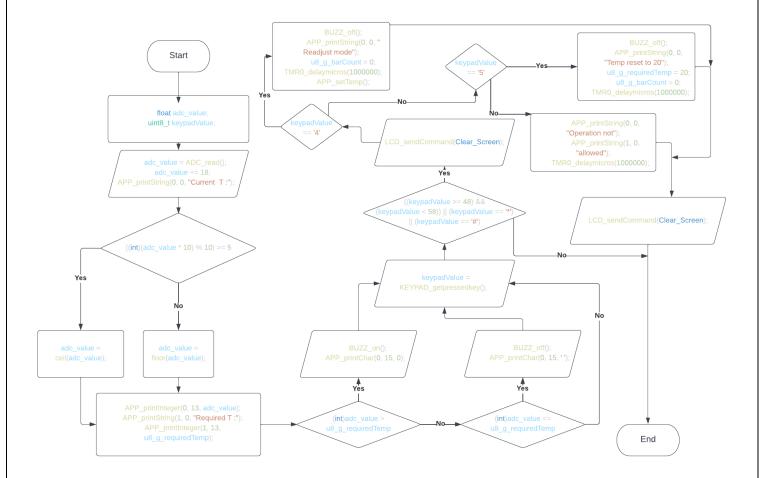
# void APP\_incrementBar(); Start APP\_printChar(1, u8\_g\_barCount, '|'); u8\_g\_barCount++; u8\_g\_requiredTemp++; u8\_g\_barCount APP\_printInteger(0, 7, u8\_g\_barCount = 16 u8\_g\_requiredTemp); > 16 Nο End

### void APP\_decrementBar();



### void APP\_setTemp(); Start . Yes uint16\_t u16\_l\_ovf; uint8\_t keypadValue; == '1' == '1' LCD\_sendCommand(Clear\_Screen); APP\_printString(0, 0, "Set Temp") 0.5 sec Delay ¥ Yes \_CD\_sendCommand(Clear\_Screen); APP\_printString(0, 0, "Min=18"); (keypadValue != '3') && u8\_g\_requiredTemp); APP\_printString(0, 0, "Temp set to APP\_printString(0, 10, "Max=35") TIMER2\_perscalerMode(TIMER\_OFF\_); > "); TCNT2 = 0; APP\_printInteger(0, 14, TIMER2\_perscalerMode(PRESCALER\_MODE) 18) 1 Sec Delay Yes u8\_g\_barCount++; End ${\color{red} LCD\_sendCommand(Clear\_Screen);}$

### void APP\_defaultView(void);



# Thank You!