Water Heater

1.0.0

Generated by Doxygen 1.8.18

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 data Struct Reference

sTask data struct store all data associate with each task

`#include <SCH_int.h>`

Data Fields

- void(∗ pTask)(void)
- uint16_t Delay
- uint16_t Period
- uint8_t RunMe

3.1.1 Detailed Description

sTask data struct store all data associate with each task

pTasks: pointer to task function. Delay: Delay (ticks) until the function will (next) be run Period: Interval (ticks) between subsequent runs.. RunMe: Incremented (by scheduler) when task is due to execute.

3.1.2 Field Documentation

3.1.2.1 Delay

`uint16_t Delay`

3.1.2.2 Period

```
uint16_t Period
```

3.1.2.3 pTask

```
void(* pTask) (void)
```

3.1.2.4 RunMe

```
uint8_t RunMe
```

Chapter 4

File Documentation

4.1 ADC_int.h File Reference

Macros

- #define ADC_CHANNEL_0 0x00
- #define ADC_CHANNEL_1 0x01
- #define ADC_CHANNEL_2 0x02
- #define ADC_CHANNEL_3 0x03
- #define ADC_CHANNEL_4 0x04
- #define ADC_CHANNEL_5 0x05
- #define ADC_CHANNEL_6 0x06
- #define ADC_CHANNEL_7 0x07

Functions

- void ADC_vidInit (void)
- uint16_t ADC_GetAdValue (uint8_t ChannelId)

4.1.1 Macro Definition Documentation

4.1.1.1 ADC_CHANNEL_0

```
#define ADC_CHANNEL_0 0x00
```

4.1.1.2 ADC_CHANNEL_1

```
#define ADC_CHANNEL_1 0x01
```

4.1.1.3 ADC_CHANNEL_2

```
#define ADC_CHANNEL_2 0x02
```

4.1.1.4 ADC_CHANNEL_3

```
#define ADC_CHANNEL_3 0x03
```

4.1.1.5 ADC_CHANNEL_4

```
#define ADC_CHANNEL_4 0x04
```

4.1.1.6 ADC_CHANNEL_5

```
#define ADC_CHANNEL_5 0x05
```

4.1.1.7 ADC_CHANNEL_6

```
#define ADC_CHANNEL_6 0x06
```

4.1.1.8 ADC_CHANNEL_7

```
#define ADC_CHANNEL_7 0x07
```

4.1.2 Function Documentation

4.1.2.1 ADC_GetAdValue()

```
uint16_t ADC_GetAdValue (
            uint8_t ChannelId )
```

4.1.2.2 ADC_vidInit()

```
void ADC_vidInit (
            void  )
```

4.2 ADC_Prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "BIT_MATH.h"
#include "ADC_int.h"
```

Macros

- #define _XTAL_FREQ 8000000

Functions

- void ADC_vidInit (void)
- uint16_t ADC_GetAdValue (uint8_t ChannelId)

4.2.1 Macro Definition Documentation

4.2.1.1 _XTAL_FREQ

```
#define _XTAL_FREQ 8000000
```

4.2.2 Function Documentation

4.2.2.1 ADC_GetAdValue()

```
uint16_t ADC_GetAdValue (
            uint8_t ChannelId )
```

4.2.2.2 ADC_vidInit()

```
void ADC_vidInit (
             void  )
```

4.3 App_int.h File Reference

Macros

- #define I2C_SPEED 50000
- #define HERE_WE_ARE_AGAIN 0x97
- #define EEPROM_HERE_BFORE_ADDRESS 0x30
- #define EEPROM_SAVE_TEMP_ADDRESS 0x00
- #define TEMP_THRSH 5
- #define NUM_READING 10
- #define LOCK 1
- #define UNLOCK 0
- #define SET_TEMP_TIME_OUT 5000
- #define INIT_TEMP 60
- #define MIN_TEMP 35
- #define MAX_TEMP 75

Functions

- void APP_Init (void)

  Initialization of all MCU peripherals, OS scheduler and crate the tasks.
- void APP_DeInit (void)

  DE-initialization of all MCU peripherals.
- void Set_Temp (void)

  Set Temp task run every 10 'ms', check if the user press the '+/'-' buttons.
- void Update_Temp (void)

  Update temp task run every 100 'ms', Read the LM35 temp with ADC and scale the value.
- void Control_Temp (void)

  Control Temp task run every 1000 'ms', Controlling the heater and cooler.

Variables

- uint8_t Goal_temp
- uint16_t Current_temp

4.3.1 Macro Definition Documentation

4.3.1.1 EEPROM_HERE_BFORE_ADDRESS

```
#define EEPROM_HERE_BFORE_ADDRESS 0x30
```

4.3.1.2 EEPROM_SAVE_TEMP_ADDRESS

```
#define EEPROM_SAVE_TEMP_ADDRESS 0x00
```

4.3.1.3 HERE_WE_ARE_AGAIN

```
#define HERE_WE_ARE_AGAIN 0x97
```

4.3.1.4 I2C_SPEED

```
#define I2C_SPEED 50000
```

4.3.1.5 INIT_TEMP

```
#define INIT_TEMP 60
```

4.3.1.6 LOCK

```
#define LOCK 1
```

4.3.1.7 MAX_TEMP

```
#define MAX_TEMP 75
```

4.3.1.8 MIN_TEMP

```
#define MIN_TEMP 35
```

4.3.1.9 NUM_READING

```
#define NUM_READING 10
```

4.3.1.10 SET_TEMP_TIME_OUT

```
#define SET_TEMP_TIME_OUT 5000
```

4.3.1.11 TEMP_THRSH

```
#define TEMP_THRSH 5
```

4.3.1.12 UNLOCK

```
#define UNLOCK 0
```

4.3.2 Function Documentation

4.3.2.1 APP_DeInit()

```
void APP_DeInit (
             void  )
```

DE-initialization of all MCU peripherals.

Parameters

void.	

Returns

void.

4.3.2.2 APP_Init()

```
void APP_Init (
            void  )
```

Initialization of all MCU peripherals, OS scheduler and crate the tasks.

Check for last value saved in EEPROM. MCU peripherals : GPIO pins, ADC, EXTI, TIM0, I2C. EXT peripherals : Seven segment display.

Parameters

void.	

Returns

void.

4.3.2.3 Control_Temp()

```
void Control_Temp (
            void  )
```

Control Temp task run every 1000 'ms', Controlling the heater and cooler.

Heater-ON/Cooler-OFF if : AVG_TEMP <= SET_TEMP - 5. Heater-OFF/Cooler-ON if : AVG_TEMP >= SET_T↩ EMP + 5.

Parameters

void.	

Returns

void.

4.3.2.4 Set_Temp()

```
void Set_Temp (
            void  )
```

Set Temp task run every 10 'ms', check if the user press the '+/'-' buttons.

if True : Take the new set temp and save it to EEPROM.

Parameters

void.	

Returns

void.

4.3.2.5 Update_Temp()

```
void Update_Temp (
            void  )
```

Update temp task run every 100 'ms', Read the LM35 temp with ADC and scale the value.

Max LM35 Output = 1500 'mv' when the temp is 150 C corresponding to 307 in ADC Range. Current_temp = (ADC_Value $*$ 150) / 307 .

Parameters

void.	

Returns

void.

4.3.3 Variable Documentation

4.3.3.1 Current_temp

```
uint16_t Current_temp
```

4.3.3.2 Goal_temp

```
uint8_t Goal_temp
```

4.4 APP_prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "BIT_MATH.h"
#include "DIO_int.h"
#include "SCH_int.h"
#include "SS_int.h"
#include "ADC_int.h"
#include "EXTI_int.h"
#include "I2C_int.h"
#include "EEPROM_int.h"
#include "App_int.h"
```

Macros

- #define _XTAL_FREQ 8000000

Functions

- void APP_Init (void)

  Initialization of all MCU peripherals, OS scheduler and crate the tasks.
- void APP_DeInit (void)

  DE-initialization of all MCU peripherals.
- void Update_Temp (void)

  Update temp task run every 100 'ms', Read the LM35 temp with ADC and scale the value.
- void Control_Temp (void)

  Control Temp task run every 1000 'ms', Controlling the heater and cooler.
- void Set_Temp (void)

  Set Temp task run every 10 'ms', check if the user press the '+/'-' buttons.

Variables

- uint8_t Goal_temp = INIT_TEMP
- uint16_t Current_temp = 0
- uint16_t Avg_temp = 0
- uint8_t Heater_Led_state = 0
- uint8_t Heater_state = 0
- uint16_t time_out = SET_TEMP_TIME_OUT
- uint8_t SS_Locked = UNLOCK

4.4.1 Macro Definition Documentation

4.4.1.1 _XTAL_FREQ

```
#define _XTAL_FREQ 8000000
```

4.4.2 Function Documentation

4.4.2.1 APP_DeInit()

```
void APP_DeInit (
            void  )
```

DE-initialization of all MCU peripherals.

Parameters

void.	

Returns

void.

4.4.2.2 APP_Init()

```
void APP_Init (
            void  )
```

Initialization of all MCU peripherals, OS scheduler and crate the tasks.

Check for last value saved in EEPROM. MCU peripherals : GPIO pins, ADC, EXTI, TIM0, I2C. EXT peripherals : Seven segment display.

Parameters

void.	

Returns

void.

4.4.2.3 Control_Temp()

```
void Control_Temp (
            void  )
```

Control Temp task run every 1000 'ms', Controlling the heater and cooler.

Heater-ON/Cooler-OFF if : AVG_TEMP <= SET_TEMP - 5. Heater-OFF/Cooler-ON if : AVG_TEMP >= SET_T↩ EMP + 5.

Parameters

void.	

Returns

void.

4.4.2.4 Set_Temp()

```
void Set_Temp (
             void  )
```

Set Temp task run every 10 'ms', check if the user press the '+/'-' buttons.

if True : Take the new set temp and save it to EEPROM.

Parameters

void.	

Returns

void.

4.4.2.5 Update_Temp()

```
void Update_Temp (
             void  )
```

Update temp task run every 100 'ms', Read the LM35 temp with ADC and scale the value.

Max LM35 Output = 1500 'mv' when the temp is 150 C corresponding to 307 in ADC Range. Current_temp = (ADC_Value $*$ 150) / 307 .

Parameters

void.	

Returns

void.

4.4.3 Variable Documentation

4.4.3.1 Avg_temp

```
uint16_t Avg_temp = 0
```

4.4.3.2 Current_temp

```
uint16_t Current_temp = 0
```

4.4.3.3 Goal_temp

```
uint8_t Goal_temp = INIT_TEMP
```

4.4.3.4 Heater_Led_state

```
uint8_t Heater_Led_state = 0
```

4.4.3.5 Heater_state

```
uint8_t Heater_state = 0
```

4.4.3.6 SS_Locked

```
uint8_t SS_Locked = UNLOCK
```

4.4.3.7 time_out

```
uint16_t time_out = SET_TEMP_TIME_OUT
```

4.5 BIT_MATH.h File Reference

Macros

- #define SET_BIT(VAR, BITNO) (VAR) |= (1 << (BITNO))
- #define CLR_BIT(VAR, BITNO) (VAR) &= ~(1 << (BITNO))
- #define TOG_BIT(VAR, BITNO) (VAR) ^= (1 << (BITNO))
- #define GET_BIT(VAR, BITNO) (((VAR) >> (BITNO)) & 0x01)

4.5.1 Macro Definition Documentation

4.5.1.1 CLR_BIT

```
#define CLR_BIT(
              VAR,
              BITNO ) (VAR) &= ~(1 << (BITNO))
```

4.5.1.2 GET_BIT

```
#define GET_BIT(
              VAR,
              BITNO ) (((VAR) >> (BITNO)) & 0x01)
```

4.5.1.3 SET_BIT

```
#define SET_BIT(
              VAR,
              BITNO ) (VAR) |= (1 << (BITNO))
```

4.5.1.4 TOG_BIT

```
#define TOG_BIT(
              VAR,
              BITNO ) (VAR) ^= (1 << (BITNO))
```

4.6 config.h File Reference

Macros

- #define _XTAL_FREQ 8000000

4.6.1 Macro Definition Documentation

4.6.1.1 _XTAL_FREQ

```
#define _XTAL_FREQ 8000000
```

4.7 DIO_int.h File Reference

Macros

- #define GPIOA 0
- #define GPIOB 1
- #define GPIOC 2
- #define GPIOD 3
- #define PIN0 0
- #define PIN1 1
- #define PIN2 2
- #define PIN3 3
- #define PIN4 4
- #define PIN5 5
- #define PIN6 6
- #define PIN7 7
- #define INPUT 0xff
- #define OUTPUT 0x00
- #define HIGH 0xff
- #define LOW 0x00

Functions

- uint8_t DIO_uint8_tGetPinValue (uint8_t PortId, uint8_t PinNumber)
- void DIO_VidSetPinValue (uint8_t PortId, uint8_t Pin, uint8_t Value)
- void DIO_VidSetPinDirection (uint8_t PortId, uint8_t Pin, uint8_t Value)
- uint8_t DIO_uint8_tGetPortValue (uint8_t PortId)
- void DIO_VidSetPortValue (uint8_t PortId, uint8_t Value)
- void DIO_VidSetPortDirection (uint8_t PortId, uint8_t Value)

4.7.1 Macro Definition Documentation

4.7.1.1 GPIOA

```
#define GPIOA 0
```

4.7.1.2 GPIOB

```
#define GPIOB 1
```

4.7.1.3 GPIOC

```
#define GPIOC 2
```

4.7.1.4 GPIOD

```
#define GPIOD 3
```

4.7.1.5 HIGH

```
#define HIGH 0xff
```

4.7.1.6 INPUT

```
#define INPUT 0xff
```

4.7.1.7 LOW

```
#define LOW 0x00
```

4.7.1.8 OUTPUT

```
#define OUTPUT 0x00
```

4.7.1.9 PIN0

```
#define PIN0 0
```

4.7.1.10 PIN1

```
#define PIN1 1
```

4.7.1.11 PIN2

```
#define PIN2 2
```

4.7.1.12 PIN3

```
#define PIN3 3
```

4.7.1.13 PIN4

```
#define PIN4 4
```

4.7.1.14 PIN5

```
#define PIN5 5
```

4.7.1.15 PIN6

```
#define PIN6 6
```

4.7.1.16 PIN7

```
#define PIN7 7
```

4.7.2 Function Documentation

4.7.2.1 DIO_uint8_tGetPinValue()

```
uint8_t DIO_uint8_tGetPinValue (
            uint8_t PortId,
            uint8_t PinNumber )
```

4.7.2.2 DIO_uint8_tGetPortValue()

```
uint8_t DIO_uint8_tGetPortValue (
            uint8_t PortId )
```

4.7.2.3 DIO_VidSetPinDirection()

```
void DIO_VidSetPinDirection (
            uint8_t PortId,
            uint8_t Pin,
            uint8_t Value )
```

4.7.2.4 DIO_VidSetPinValue()

```
void DIO_VidSetPinValue (
            uint8_t PortId,
            uint8_t Pin,
            uint8_t Value )
```

4.7.2.5 DIO_VidSetPortDirection()

```
void DIO_VidSetPortDirection (
            uint8_t PortId,
            uint8_t Value )
```

4.7.2.6 DIO_VidSetPortValue()

```
void DIO_VidSetPortValue (
            uint8_t PortId,
            uint8_t Value )
```

4.8 DIO_prog.c File Reference

```
#include <xc.h>
#include "BIT_MATH.h"
#include "stdint.h"
#include "DIO_reg.h"
#include "DIO_int.h"
```

Functions

- void DIO_VidSetPortDirection (uint8_t PortId, uint8_t Value)
- void DIO_VidSetPortValue (uint8_t PortId, uint8_t Value)
- void DIO_VidSetPinDirection (uint8_t PortId, uint8_t Pin, uint8_t Value)
- void DIO_VidSetPinValue (uint8_t PortId, uint8_t Pin, uint8_t Value)
- uint8_t DIO_uint8_tGetPortValue (uint8_t PortId)
- uint8_t DIO_uint8_tGetPinValue (uint8_t PortId, uint8_t PinNumber)

4.8.1 Function Documentation

4.8.1.1 DIO_uint8_tGetPinValue()

```
uint8_t DIO_uint8_tGetPinValue (
            uint8_t PortId,
            uint8_t PinNumber )
```

4.8.1.2 DIO_uint8_tGetPortValue()

```
uint8_t DIO_uint8_tGetPortValue (
            uint8_t PortId )
```

4.8.1.3 DIO_VidSetPinDirection()

```
void DIO_VidSetPinDirection (
             uint8_t PortId,
             uint8_t Pin,
             uint8_t Value )
```

4.8.1.4 DIO_VidSetPinValue()

```
void DIO_VidSetPinValue (
             uint8_t PortId,
             uint8_t Pin,
             uint8_t Value )
```

4.8.1.5 DIO_VidSetPortDirection()

```
void DIO_VidSetPortDirection (
             uint8_t PortId,
             uint8_t Value )
```

4.8.1.6 DIO_VidSetPortValue()

```
void DIO_VidSetPortValue (
             uint8_t PortId,
             uint8_t Value )
```

4.9 DIO_reg.h File Reference

Macros

- #define PORTA_Reg PORTA
- #define DDRA_Reg TRISA
- #define PORTB_Reg PORTB
- #define DDRB_Reg TRISB
- #define PORTC_Reg PORTC
- #define DDRC_Reg TRISC
- #define PORTD_Reg PORTD
- #define DDRD_Reg TRISD

4.9.1 Macro Definition Documentation

4.9.1.1 DDRA_Reg

```
#define DDRA_Reg TRISA
```

4.9.1.2 DDRB_Reg

```
#define DDRB_Reg TRISB
```

4.9.1.3 DDRC_Reg

```
#define DDRC_Reg TRISC
```

4.9.1.4 DDRD_Reg

```
#define DDRD_Reg TRISD
```

4.9.1.5 PORTA_Reg

```
#define PORTA_Reg PORTA
```

4.9.1.6 PORTB_Reg

```
#define PORTB_Reg PORTB
```

4.9.1.7 PORTC_Reg

```
#define PORTC_Reg PORTC
```

4.9.1.8 PORTD_Reg

```
#define PORTD_Reg PORTD
```

4.10 EEPROM_int.h File Reference

Macros

- #define EEPROM_Address_R 0xA1
- #define EEPROM_Address_W 0xA0

Functions

- void EEPROM_Write (uint8_t add, uint8_t data)

  Function to write single byte of data to EEPROM.
- uint8_t EEPROM_Read (uint8_t add)

  Function to read single byte from Address location in EEPROM.

4.10.1 Macro Definition Documentation

4.10.1.1 EEPROM_Address_R

```
#define EEPROM_Address_R 0xA1
```

4.10.1.2 EEPROM_Address_W

```
#define EEPROM_Address_W 0xA0
```

4.10.2 Function Documentation

4.10.2.1 EEPROM_Read()

```
uint8_t EEPROM_Read (
            uint8_t add )
```

Function to read single byte from Address location in EEPROM.

Parameters

Address	of location to read the data.

Returns

void.

4.10.2.2 EEPROM_Write()

```
void EEPROM_Write (
            uint8_t add,
            uint8_t data )
```

Function to write single byte of data to EEPROM.

Parameters

Address	of location to save the data.
Data	to save.

Returns

void.

4.11 EEPROM_prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "I2C_int.h"
#include "EEPROM_int.h"
```

Functions

- void EEPROM_Write (uint8_t add, uint8_t data)

  Function to write single byte of data to EEPROM.
- uint8_t EEPROM_Read (uint8_t add)

  Function to read single byte from Address location in EEPROM.

4.11.1 Function Documentation

4.11.1.1 EEPROM_Read()

```
uint8_t EEPROM_Read (
            uint8_t add )
```

Function to read single byte from Address location in EEPROM.

Parameters

Address	of location to read the data.

Returns

void.

4.11.1.2 EEPROM_Write()

```
void EEPROM_Write (
            uint8_t add,
            uint8_t data )
```

Function to write single byte of data to EEPROM.

Parameters

Address	of location to save the data.
Data	to save.

Returns

void.

4.12 EXTI_int.h File Reference

Functions

- void EXTI_Init (void)
  Function to Initialize PIN0 in PORTB as EXTI pin.

4.12.1 Function Documentation

4.12.1.1 EXTI_Init()

```
void EXTI_Init (
            void  )
```

Function to Initialize PIN0 in PORTB as EXTI pin.

Parameters

void.	

Returns

void.

4.13 EXTI_prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "BIT_MATH.h"
#include "EXTI_int.h"
```

Functions

- void EXTI_Init (void)

  Function to Initialize PIN0 in PORTB as EXTI pin.

4.13.1 Function Documentation

4.13.1.1 EXTI_Init()

```
void EXTI_Init (
            void  )
```

Function to Initialize PIN0 in PORTB as EXTI pin.

Parameters

void.	

Returns

void.

4.14 I2C_int.h File Reference

Functions

- void I2C_Master_Init (const uint32_t baud)

- void I2C_Master_Wait ()
- void I2C_Master_Start ()
- void I2C_Master_RepeatedStart ()
- void I2C_Master_Stop ()
- void I2C_ACK ()
- void I2C_NACK ()
- uint8_t I2C_Master_Write (uint8_t)
- uint8_t I2C_Read_Byte (void)

4.14.1 Function Documentation

4.14.1.1 I2C_ACK()

```
void I2C_ACK ( )
```

4.14.1.2 I2C_Master_Init()

```
void I2C_Master_Init (
            const uint32_t baud )
```

4.14.1.3 I2C_Master_RepeatedStart()

```
void I2C_Master_RepeatedStart ( )
```

4.14.1.4 I2C_Master_Start()

```
void I2C_Master_Start ( )
```

4.14.1.5 I2C_Master_Stop()

```
void I2C_Master_Stop ( )
```

4.14.1.6 I2C_Master_Wait()

```
void I2C_Master_Wait ( )
```

4.14.1.7 I2C_Master_Write()

```
uint8_t I2C_Master_Write (
            uint8_t  )
```

4.14.1.8 I2C_NACK()

```
void I2C_NACK ( )
```

4.14.1.9 I2C_Read_Byte()

```
uint8_t I2C_Read_Byte (
            void  )
```

4.15 I2C_prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "I2C_int.h"
```

Macros

- #define _XTAL_FREQ 8000000

Functions

- void I2C_Master_Init (const uint32_t baud)
- void I2C_Master_Wait ()
- void I2C_Master_Start ()
- void I2C_Master_RepeatedStart ()
- void I2C_Master_Stop ()
- uint8_t I2C_Master_Write (uint8_t data)
- uint8_t I2C_Read_Byte (void)
- void I2C_ACK (void)
- void I2C_NACK (void)

4.15.1 Macro Definition Documentation

4.15.1.1 _XTAL_FREQ

```
#define _XTAL_FREQ 8000000
```

4.15.2 Function Documentation

4.15.2.1 I2C_ACK()

```
void I2C_ACK (
             void  )
```

4.15.2.2 I2C_Master_Init()

```
void I2C_Master_Init (
             const uint32_t baud )
```

4.15.2.3 I2C_Master_RepeatedStart()

```
void I2C_Master_RepeatedStart ( )
```

4.15.2.4 I2C_Master_Start()

```
void I2C_Master_Start ( )
```

4.15.2.5 I2C_Master_Stop()

```
void I2C_Master_Stop ( )
```

4.15.2.6 I2C_Master_Wait()

```
void I2C_Master_Wait ( )
```

4.15.2.7 I2C_Master_Write()

```
uint8_t I2C_Master_Write (
            uint8_t data )
```

4.15.2.8 I2C_NACK()

```
void I2C_NACK (
            void  )
```

4.15.2.9 I2C_Read_Byte()

```
uint8_t I2C_Read_Byte (
            void  )
```

4.16 main.c File Reference

```
#include "config.h"
#include <xc.h>
#include <stdint.h>
#include "SCH_int.h"
#include "App_int.h"
#include "SS_int.h"
```

Functions

- int main (void)

4.16.1 Function Documentation

4.16.1.1 main()

```
int main (
             void  )
```

4.17 SCH_int.h File Reference

Data Structures

- struct data

  sTask data struct store all data associate with each task

Macros

- #define SCH_MAX_TASKS (3) /∗ Number of tasks ∗/
- #define ERROR_SCH_TOO_MANY_TASKS (1)
- #define ERROR_SCH_CANNOT_DELETE_TASK (2)
- #define ERROR_SCH_WAITING_FOR_SLAVE_TO_ACK (0xAA)
- #define ERROR_SCH_WAITING_FOR_START_COMMAND_FROM_MASTER (0xAA)
- #define ERROR_SCH_ONE_OR_MORE_SLAVES_DID_NOT_START (0xA0)
- #define ERROR_SCH_LOST_SLAVE (0x80)
- #define RETURN_NORMAL 0
- #define RETURN_ERROR 1
- #define SCH_RUNNING 1
- #define SCH_STOP 0

Typedefs

- typedef struct data sTask

  sTask data struct store all data associate with each task

Functions

- void SCH_Init_T0 (void)

  SCH_Init_T0() Scheduler initialization function.
- void SCH_Dispatch_Tasks (void)

  SCH_Dispatch_Tasks() This is the 'dispatcher' function.
- void SCH_Start (void)

  SCH_Start() Start the scheduler by enable the TIM0 interrupt.
- void SCH_Stop (void)

  SCH_Stop() Stop the scheduler by disable the TIM0 interrupt.
- uint8_t SCH_Add_Task (void(∗)(void), const uint16_t, const uint16_t)

  SCH_Add_Task() Causes a task (function) to be executed at regular intervals or after a user-defined delay.
- uint8_t SCH_Delete_Task (const uint8_t)

  SCH_Delete_Task() Removes a task from the scheduler.
- void TIM0_Init ()

Variables

- uint8_t SCH_state
- uint32_t SYS_TICK

4.17.1 Macro Definition Documentation

4.17.1.1 ERROR_SCH_CANNOT_DELETE_TASK

```
#define ERROR_SCH_CANNOT_DELETE_TASK (2)
```

4.17.1.2 ERROR_SCH_LOST_SLAVE

```
#define ERROR_SCH_LOST_SLAVE (0x80)
```

4.17.1.3 ERROR_SCH_ONE_OR_MORE_SLAVES_DID_NOT_START

```
#define ERROR_SCH_ONE_OR_MORE_SLAVES_DID_NOT_START (0xA0)
```

4.17.1.4 ERROR_SCH_TOO_MANY_TASKS

```
#define ERROR_SCH_TOO_MANY_TASKS (1)
```

4.17.1.5 ERROR_SCH_WAITING_FOR_SLAVE_TO_ACK

```
#define ERROR_SCH_WAITING_FOR_SLAVE_TO_ACK (0xAA)
```

4.17.1.6 ERROR_SCH_WAITING_FOR_START_COMMAND_FROM_MASTER

```
#define ERROR_SCH_WAITING_FOR_START_COMMAND_FROM_MASTER (0xAA)
```

4.17.1.7 RETURN_ERROR

```
#define RETURN_ERROR 1
```

4.17.1.8 RETURN_NORMAL

```
#define RETURN_NORMAL 0
```

4.17.1.9 SCH_MAX_TASKS

```
#define SCH_MAX_TASKS (3) /* Number of tasks */
```

4.17.1.10 SCH_RUNNING

```
#define SCH_RUNNING 1
```

4.17.1.11 SCH_STOP

```
#define SCH_STOP 0
```

4.17.2 Typedef Documentation

4.17.2.1 sTask

```
typedef struct data sTask
```

sTask data struct store all data associate with each task

pTasks: pointer to task function. Delay: Delay (ticks) until the function will (next) be run Period: Interval (ticks) between subsequent runs.. RunMe: Incremented (by scheduler) when task is due to execute.

4.17.3 Function Documentation

4.17.3.1 SCH_Add_Task()

```
uint8_t SCH_Add_Task (
            void(*)(void) ,
            const uint16_t ,
            const uint16_t  )
```

SCH_Add_Task() Causes a task (function) to be executed at regular intervals or after a user-defined delay.

Parameters

Pointer	to function.
Delay	time.
periodic	time.

Returns

Error state.

4.17.3.2 SCH_Delete_Task()

```
uint8_t SCH_Delete_Task (
            const uint8_t  )
```

SCH_Delete_Task() Removes a task from the scheduler.

Note that this does *not* delete the associated function from memory: it simply means that it is no longer called by the scheduler.

Parameters

TASK_INDEX	- The task index. Provided by SCH_Add_Task().

Returns

RETURN_ERROR or RETURN_NORMAL.

4.17.3.3 SCH_Dispatch_Tasks()

```
void SCH_Dispatch_Tasks (
            void  )
```

SCH_Dispatch_Tasks() This is the 'dispatcher' function.

When a task (function) is due to run, SCH_Dispatch_Tasks() will run it. This function must be called (repeatedly) from the main loop.

Parameters

void.	

Returns

void.

4.17.3.4 SCH_Init_T0()

```
void SCH_Init_T0 (
            void  )
```

SCH_Init_T0() Scheduler initialization function.

Prepares scheduler data structures and sets up timer interrupts at required rate. Must call this function before using the scheduler.

Parameters

void.	

Returns

void.

4.17.3.5 SCH_Start()

```
void SCH_Start (
            void  )
```

SCH_Start() Start the scheduler by enable the TIM0 interrupt.

Parameters

void.	

Returns

void.

4.17.3.6 SCH_Stop()

```
void SCH_Stop (
            void  )
```

SCH_Stop() Stop the scheduler by disable the TIM0 interrupt.

Parameters

void.	

Returns

void.

4.17.3.7 TIM0_Init()

```
void TIM0_Init ( )
```

4.17.4 Variable Documentation

4.17.4.1 SCH_state

```
uint8_t SCH_state
```

4.17.4.2 SYS_TICK

```
uint32_t SYS_TICK
```

4.18 SCH_Prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "BIT_MATH.h"
#include "SCH_int.h"
#include "App_int.h"
#include "EEPROM_int.h"
#include "DIO_int.h"
```

Macros

- #define NUM_OVF 8
- #define TIM0_PRELOAD_Value 48

Functions

- void TIM0_Init (void)
- uint8_t SCH_Add_Task (void(∗pFunction)(), const uint16_t DELAY, const uint16_t PERIOD)

  SCH_Add_Task() Causes a task (function) to be executed at regular intervals or after a user-defined delay.
- void SCH_Init_T0 (void)

  SCH_Init_T0() Scheduler initialization function.
- uint8_t SCH_Delete_Task (const uint8_t TASK_INDEX)

  SCH_Delete_Task() Removes a task from the scheduler.
- void SCH_Dispatch_Tasks (void)

  SCH_Dispatch_Tasks() This is the 'dispatcher' function.
- void SCH_Start (void)

  SCH_Start() Start the scheduler by enable the TIM0 interrupt.
- void SCH_Stop (void)

  SCH_Stop() Stop the scheduler by disable the TIM0 interrupt.
- void __interrupt () ISR(void)

Variables

- uint8_t num_ovf = 0
- sTask SCH_tasks_G [SCH_MAX_TASKS]
- uint8_t SCH_state = SCH_STOP
- uint32_t SYS_TICK = 0

4.18.1 Macro Definition Documentation

4.18.1.1 NUM_OVF

```
#define NUM_OVF 8
```

4.18.1.2 TIM0_PRELOAD_Value

```
#define TIM0_PRELOAD_Value 48
```

4.18.2 Function Documentation

4.18.2.1 __interrupt()

```
void __interrupt ( )
```

4.18.2.2 SCH_Add_Task()

```
uint8_t SCH_Add_Task (
            void(*)() pFunction,
            const uint16_t DELAY,
            const uint16_t PERIOD )
```

SCH_Add_Task() Causes a task (function) to be executed at regular intervals or after a user-defined delay.

Parameters

Pointer	to function.
Delay	time.
periodic	time.

Returns

sum of `values`, or 0.0 if `values` is empty.

4.18.2.3 SCH_Delete_Task()

```
uint8_t SCH_Delete_Task (
            const uint8_t  )
```

SCH_Delete_Task() Removes a task from the scheduler.

Note that this does *not* delete the associated function from memory: it simply means that it is no longer called by the scheduler.

Parameters

TASK_INDEX	- The task index. Provided by SCH_Add_Task().

Returns

RETURN_ERROR or RETURN_NORMAL.

4.18.2.4 SCH_Dispatch_Tasks()

```
void SCH_Dispatch_Tasks (
            void  )
```

SCH_Dispatch_Tasks() This is the 'dispatcher' function.

When a task (function) is due to run, SCH_Dispatch_Tasks() will run it. This function must be called (repeatedly) from the main loop.

Parameters

void.	

Returns

void.

4.18.2.5 SCH_Init_T0()

```
void SCH_Init_T0 (
            void  )
```

SCH_Init_T0() Scheduler initialization function.

Prepares scheduler data structures and sets up timer interrupts at required rate. Must call this function before using the scheduler.

Parameters

void.	

Returns

void.

4.18.2.6 SCH_Start()

```
void SCH_Start (
            void  )
```

SCH_Start() Start the scheduler by enable the TIM0 interrupt.

Parameters

void.	

Returns

void.

4.18.2.7 SCH_Stop()

```
void SCH_Stop (
            void  )
```

SCH_Stop() Stop the scheduler by disable the TIM0 interrupt.

Parameters

void.	

Returns

void.

4.18.2.8 TIM0_Init()

```
void TIM0_Init (
            void  )
```

4.18.3 Variable Documentation

4.18.3.1 num_ovf

```
uint8_t num_ovf = 0
```

4.18.3.2 SCH_state

```
uint8_t SCH_state = SCH_STOP
```

4.18.3.3 SCH_tasks_G

```
sTask SCH_tasks_G[SCH_MAX_TASKS]
```

4.18.3.4 SYS_TICK

```
uint32_t SYS_TICK = 0
```

4.19 SS_int.h File Reference

Macros

- #define SS_Data_PORT GPIOD
- #define SS_Control_PORT GPIOA
- #define SS_Conrol_PIN0 PIN4
- #define SS_Conrol_PIN1 PIN5

Functions

- void SS_Init ()

  Control Temp task run every 1000 'ms', Controlling the heater and cooler.
- void SS_Display (uint8_t num)

  Function to display number in the seven segment display.

4.19.1 Macro Definition Documentation

4.19.1.1 SS_Conrol_PIN0

```
#define SS_Conrol_PIN0 PIN4
```

4.19.1.2 SS_Conrol_PIN1

```
#define SS_Conrol_PIN1 PIN5
```

4.19.1.3 SS_Control_PORT

```
#define SS_Control_PORT GPIOA
```

4.19.1.4 SS_Data_PORT

```
#define SS_Data_PORT GPIOD
```

4.19.2 Function Documentation

4.19.2.1 SS_Display()

```
void SS_Display (
            uint8_t num )
```

Function to display number in the seven segment display.

Parameters

num	to be printed.

Returns

void.

4.19.2.2 SS_Init()

```
void SS_Init ( )
```

Control Temp task run every 1000 'ms', Controlling the heater and cooler.

Heater-ON/Cooler-OFF if : AVG_TEMP <= SET_TEMP - 5. Heater-OFF/Cooler-ON if : AVG_TEMP >= SET_T↩EMP + 5.

Parameters

void.	

Returns

void.

4.20 SS_prog.c File Reference

```
#include <xc.h>
#include <stdint.h>
#include "DIO_int.h"
#include "SS_int.h"
```

Macros

- #define _XTAL_FREQ 8000000

Functions

- void SS_Init ()

  Control Temp task run every 1000 'ms', Controlling the heater and cooler.
- void SS_Display (uint8_t num)

  Function to display number in the seven segment display.

Variables

- uint8_t segments_code [10] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x67}

4.20.1 Macro Definition Documentation

4.20.1.1 _XTAL_FREQ

```
#define _XTAL_FREQ 8000000
```

4.20.2 Function Documentation

4.20.2.1 SS_Display()

```
void SS_Display (
            uint8_t num )
```

Function to display number in the seven segment display.

Parameters

num	to be printed.

Returns

void.

4.20.2.2 SS_Init()

```
void SS_Init ( )
```

Control Temp task run every 1000 'ms', Controlling the heater and cooler.

Heater-ON/Cooler-OFF if : AVG_TEMP <= SET_TEMP - 5. Heater-OFF/Cooler-ON if : AVG_TEMP >= SET_T↩EMP + 5.

Parameters

void.	

Returns

void.

4.20.3 Variable Documentation

4.20.3.1 segments_code

```
uint8_t segments_code[10] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x67}
```