Reflow Oven firmware

1.0

Generated by Doxygen 1.9.1

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 BS_DATA_OBJ Union Reference

Data Fields

- struct {
  uint8_t heatProfileID:5
  uint8_t startHeating:1
  uint8_t stopHeating:1
  uint8_t setHeatProfile:1
  } properties

- uint8_t raw

3.1.1 Detailed Description

Definition at line 239 of file baseSW.c.

3.1.2 Field Documentation

3.1.2.1 heatProfileID

```
uint8_t heatProfileID
```

Definition at line 241 of file baseSW.c.

3.1.2.2

```
struct { ... } properties
```

3.1.2.3 raw

```
uint8_t raw
```

Definition at line 246 of file baseSW.c.

3.1.2.4 setHeatProfile

```
uint8_t setHeatProfile
```

Definition at line 244 of file baseSW.c.

3.1.2.5 startHeating

```
uint8_t startHeating
```

Definition at line 242 of file baseSW.c.

3.1.2.6 stopHeating

```
uint8_t stopHeating
```

Definition at line 243 of file baseSW.c.

The documentation for this union was generated from the following file:

- baseSW.c

3.2 IC_MAX31855_DATA Union Reference

Data Fields

- struct {
  uint16_t thermocouple_temperature_data:14
  uint8_t reserved1:1
  uint8_t fault:1
  uint16_t internal_temperature_data:12
  uint8_t reserved2:1
  uint8_t scv:1
  uint8_t scg:1
  uint8_t oc:1
  } s

- uint32_t rawData

3.2.1 Detailed Description

Definition at line 170 of file baseSW.c.

3.2.2 Field Documentation

3.2.2.1 fault

```
uint8_t fault
```

Definition at line 176 of file baseSW.c.

3.2.2.2 internal_temperature_data

```
uint16_t internal_temperature_data
```

Definition at line 177 of file baseSW.c.

3.2.2.3 oc

```
uint8_t oc
```

Definition at line 181 of file baseSW.c.

3.2.2.4 rawData

```
uint32_t rawData
```

Definition at line 183 of file baseSW.c.

3.2.2.5 reserved1

```
uint8_t reserved1
```

Definition at line 175 of file baseSW.c.

3.2.2.6 reserved2

```
uint8_t reserved2
```

Definition at line 178 of file baseSW.c.

3.2.2.7

```
struct { ... } s
```

3.2.2.8 scg

```
uint8_t scg
```

Definition at line 180 of file baseSW.c.

3.2.2.9 scv

```
uint8_t scv
```

Definition at line 179 of file baseSW.c.

3.2.2.10 thermocouple_temperature_data

```
uint16_t thermocouple_temperature_data
```

Definition at line 174 of file baseSW.c.

The documentation for this union was generated from the following file:

- baseSW.c

3.3 SENSOR_DATA Struct Reference

Data Fields

- IC_MAX31855_DATA * dataArrayQue
- struct {
  uint8_t isUploaded:1
  uint8_t currentData:7
  } dataArrayStatus

3.3.1 Detailed Description

Definition at line 186 of file baseSW.c.

3.3.2 Field Documentation

3.3.2.1 currentData

```
uint8_t currentData
```

Definition at line 190 of file baseSW.c.

3.3.2.2 dataArrayQue

```
IC_MAX31855_DATA* dataArrayQue
```

Definition at line 187 of file baseSW.c.

3.3.2.3

```
struct { ... } dataArrayStatus
```

3.3.2.4 isUploaded

```
uint8_t isUploaded
```

Definition at line 189 of file baseSW.c.

The documentation for this struct was generated from the following file:

- baseSW.c

3.4 stateTaskList_s Struct Reference

```
#include <stateTaskHandler.h>
```

Data Fields

- void(∗ fun_ptr)(void)
- struct stateTaskList_s ∗ next
- struct stateTaskList_s ∗ prev
- int simaint

3.4.1 Detailed Description

Definition at line 22 of file stateTaskHandler.h.

3.4.2 Field Documentation

3.4.2.1 fun_ptr

```
void(* fun_ptr) (void)
```

Definition at line 23 of file stateTaskHandler.h.

3.4.2.2 next

```
struct stateTaskList_s* next
```

Definition at line 24 of file stateTaskHandler.h.

3.4.2.3 prev

```
struct stateTaskList_s* prev
```

Definition at line 25 of file stateTaskHandler.h.

3.4.2.4 simaint

```
int simaint
```

Definition at line 26 of file stateTaskHandler.h.

The documentation for this struct was generated from the following file:

- stateTaskHandler.h

3.5 TEMPERATURE_BUFFER Struct Reference

Reference temperature buffer.

Data Fields

- uint8_t * data
- uint16_t offset

3.5.1 Detailed Description

Reference temperature buffer.

This temperature buffer structure conatins a 8 bit unsigned integer pointer, and an offset value. The offset used during memory operations or denotes the actual temperature reference (based on time) in the array depending on the context.

Definition at line 120 of file baseSW.c.

3.5.2 Field Documentation

3.5.2.1 data

```
uint8_t* data
```

Buffer array pointer

Definition at line 122 of file baseSW.c.

3.5.2.2 offset

```
uint16_t offset
```

Buffer offset value

Definition at line 123 of file baseSW.c.

The documentation for this struct was generated from the following file:

- baseSW.c

3.6 TEMPERATURE_PROFILE Struct Reference

Reference temperature profile.

Data Fields

- TEMPERATURE_BUFFER currentProfile
- TEMPERATURE_BUFFER bufferProfile
- TEMPERATURE_BUFFER defaultProfile
- TEMPERATURE_BUFFER_STATUS profileStatus
- uint16_t addressBuffer

3.6.1 Detailed Description

Reference temperature profile.

Reference temperature profile three TEMPERATURE_BUFFER, a TEMPERATURE_BUFFER_STATUS and an addressBuffer. The addressBuffer is used to store the EEPROM address. Only bufferProfile can be used for memory operations, and currentProfile only can be used for heating processes.

Definition at line 147 of file baseSW.c.

3.6.2 Field Documentation

3.6.2.1 addressBuffer

```
uint16_t addressBuffer
```

This buffers stores the EEPROM address

Definition at line 154 of file baseSW.c.

3.6.2.2 bufferProfile

```
TEMPERATURE_BUFFER bufferProfile
```

Buffer profile array pointer

Definition at line 151 of file baseSW.c.

3.6.2.3 currentProfile

```
TEMPERATURE_BUFFER currentProfile
```

Current profile array pointer

Definition at line 150 of file baseSW.c.

3.6.2.4 defaultProfile

```
TEMPERATURE_BUFFER defaultProfile
```

Default profile buffer array pointer

Definition at line 152 of file baseSW.c.

3.6.2.5 profileStatus

```
TEMPERATURE_BUFFER_STATUS profileStatus
```

Temperature profile status

Definition at line 153 of file baseSW.c.

The documentation for this struct was generated from the following file:

- baseSW.c

3.7 TRANSCIEVE_OBJ Struct Reference

Data Fields

- TRANSCIEVE_STATUS status
- bool NextionTransmissionInProgress
- bool FTDITransmissionInProgress

3.7.1 Detailed Description

Definition at line 263 of file baseSW.c.

3.7.2 Field Documentation

3.7.2.1 FTDITransmissionInProgress

```
bool FTDITransmissionInProgress
```

Definition at line 266 of file baseSW.c.

3.7.2.2 NextionTransmissionInProgress

```
bool NextionTransmissionInProgress
```

Definition at line 265 of file baseSW.c.

3.7.2.3 status

```
TRANSCIEVE_STATUS status
```

Definition at line 264 of file baseSW.c.

The documentation for this struct was generated from the following file:

- baseSW.c

Chapter 4

File Documentation

4.1 baseSW.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "baseSW.h"
#include "stateTaskHandler.h"
#include "mcc_generated_files/tmr2.h"
#include "mcc_generated_files/uart1.h"
#include "mcc_generated_files/uart2.h"
#include "mcc_generated_files/spi1.h"
#include "mcc_generated_files/i2c1.h"
#include "EEPROM_driver.h"
#include "mcc_generated_files/pin_manager.h"
```

Data Structures

- struct TEMPERATURE_BUFFER

  Reference temperature buffer.
- struct TEMPERATURE_PROFILE

  Reference temperature profile.
- union IC_MAX31855_DATA
- struct SENSOR_DATA
- union BS_DATA_OBJ
- struct TRANSCIEVE_OBJ

Macros

- #define HEAT_PROFILE_SIZE 1024

  Heat profile buffer size.
- #define SENSOR_DATA_STORE_LENGTH 10

Enumerations

- enum TEMPERATURE_BUFFER_STATUS { TB_RECEIVE_FROM_PC , TB_SEND_TO_EEPROM , TB_RECEIVE_FROM_EEPROM , IDLE }

  Reference temperature buffer status.
- enum FTDI_STATUS { NORMAL_OPERATION , RECEIVE_HEAT_PROFILE , SEND_HEAT_PROFILE_TO_EEPROM , RECEIVE_PROFILE_FROM_ , WRITE_EEPROM_COMMAND_HIGH_BYTE , WRITE_EEPROM_COMMAND_LOW_BYTE }
- enum NEXTION_STATUS { NEXTION_NORMAL_OPERATION , NEXTION_WRITE_EEPROM_COMMAND_HIGH_BYTE , NEXTION_WRITE_EEPROM_COMMAND_LOW_BYTE , NEXTION_RECEIVE_PROFILE_FROM_EEPROM }
- enum TRANSCIEVE_STATUS { TRANSCIEVE_FULL_HEAT_PROFILE , TRANSCIEVE_CURRENT_DATA , TRANSCIEVE_IDLE }

Functions

- void enableHeat ()
- void disableHeat ()
- void IdleState_callback ()
- void loadBuffer ()
- bool checkStartConditions ()
- void ReadTemperatureData_callback ()
- void ReceiveNextionData_callback ()
- void ReceiveFTDI_callback ()
- void genericTranciverFunction ()
- void TranscieveNextionDATA_callback ()
- void TransciveFTDI_callback ()
- void ReadEEPROM_callback ()
- void WriteEEPROM_callback ()
- void baseSW_TMR2_ISR (void)
- void baseSW_UART1_tx_ISR (void)
- void baseSW_UART2_tx_ISR (void)
- void baseSW_UART1_rx_ISR (void)
- void baseSW_UART2_rx_ISR (void)
- stateTaskList * baseSW_Initialize (void)

Variables

- const uint16_t EEPROM_ADDRESS = 0x00
- const uint16_t INTERNAL_MAX_TEMPERATURE = 0x0000
- const uint16_t THERMOCOUPLE_MAX_TEMPERATURE = 0x0000
- stateTaskList * IdleState = NULL

  Idle state.
- stateTaskList * ReadTemperatureData = NULL

  Read data from MAX31855KASA+T state.
- stateTaskList * ReceiveNextionData = NULL

  Receive data from Nextion HMI state.
- stateTaskList * ReceiveFTDI = NULL

  Receive data from FTDI state.
- stateTaskList * TranscieveNextionDATA = NULL

  Broadcast temperature data to the Nextion HMI.

- stateTaskList * TransciveFTDI = NULL

  Broadcast temperature data to the PC.
- stateTaskList * ReadEEPROM = NULL

  Read heat profile from EEPROM.
- stateTaskList * WriteEEPROM = NULL

  Write heat profile into EEPROM.
- uint8_t heatProfileCurrent [HEAT_PROFILE_SIZE]
- uint8_t heatProfileBuffer [HEAT_PROFILE_SIZE]
- uint8_t heatProfileDefult [HEAT_PROFILE_SIZE]
- static TEMPERATURE_PROFILE temperatureHeatProfile
- TEMPERATURE_BUFFER temperatureBufferArray [3]
- static SENSOR_DATA SENSOR_DATA_HANDLER
- static IC_MAX31855_DATA SENSOR_DATA_ARRAYS [SENSOR_DATA_STORE_LENGTH]
- static FTDI_STATUS ftdiStatus = NORMAL_OPERATION
- static NEXTION_STATUS nextionStatus = NEXTION_NORMAL_OPERATION
- static uint8_t HEAT_IN_PROGRESS = false
- static TRANSCIEVE_OBJ transciveObj

4.1.1 Macro Definition Documentation

4.1.1.1 HEAT_PROFILE_SIZE

```
#define HEAT_PROFILE_SIZE 1024
```

Heat profile buffer size.

Each byte in the buffer represents a reference temeprature at a given time. The maximum reference temperature is 25+255=280 celsis degree, and the minimum reference temperature is 25 celsius degree.

Definition at line 38 of file baseSW.c.

4.1.1.2 SENSOR_DATA_STORE_LENGTH

```
#define SENSOR_DATA_STORE_LENGTH 10
```

Definition at line 168 of file baseSW.c.

4.1.2 Enumeration Type Documentation

4.1.2.1 FTDI_STATUS

```
enum FTDI_STATUS
```

Enumerator

NORMAL_OPERATION	
RECEIVE_HEAT_PROFILE	
SEND_HEAT_PROFILE_TO_EEPROM	
RECEIVE_PROFILE_FROM_EEPROM	
WRITE_EEPROM_COMMAND_HIGH_BYTE	
WRITE_EEPROM_COMMAND_LOW_BYTE	

Definition at line 204 of file baseSW.c.

4.1.2.2 NEXTION_STATUS

`enum NEXTION_STATUS`

Enumerator

NEXTION_NORMAL_OPERATION	
NEXTION_WRITE_EEPROM_COMMAND_HIGH_BYTE	
NEXTION_WRITE_EEPROM_COMMAND_LOW_BYTE	
NEXTION_RECEIVE_PROFILE_FROM_EEPROM	

Definition at line 222 of file baseSW.c.

4.1.2.3 TEMPERATURE_BUFFER_STATUS

`enum TEMPERATURE_BUFFER_STATUS`

Reference temperature buffer status.

Actual status of the refernce temperature buffer is stored in this enum. The status.

Enumerator

TB_RECEIVE_FROM_PC	Temperature buffer is receiving from PC is in progress
TB_SEND_TO_EEPROM	Temperature buffer is tranciving to EEPROM is in progress
TB_RECEIVE_FROM_EEPROM	Temperature buffer is receiving from EEPROM is in progress
IDLE	Temperature buffer is in idle state and ready to be used

Definition at line 132 of file baseSW.c.

4.1.2.4 TRANSCIEVE_STATUS

```
enum TRANSCIEVE_STATUS
```

Enumerator

TRANSCIEVE_FULL_HEAT_PROFILE	
TRANSCIEVE_CURRENT_DATA	
TRANSCIEVE_IDLE	

Definition at line 257 of file baseSW.c.

4.1.3 Function Documentation

4.1.3.1 baseSW_Initialize()

```
stateTaskList* baseSW_Initialize (
            void  )
```

Definition at line 559 of file baseSW.c.

4.1.3.2 baseSW_TMR2_ISR()

```
void baseSW_TMR2_ISR (
            void  )
```

Definition at line 532 of file baseSW.c.

4.1.3.3 baseSW_UART1_rx_ISR()

```
void baseSW_UART1_rx_ISR (
            void  )
```

Definition at line 551 of file baseSW.c.

4.1.3.4 baseSW_UART1_tx_ISR()

```
void baseSW_UART1_tx_ISR (
            void  )
```

Definition at line 543 of file baseSW.c.

4.1.3.5 baseSW_UART2_rx_ISR()

```
void baseSW_UART2_rx_ISR (
             void  )
```

Definition at line 555 of file baseSW.c.

4.1.3.6 baseSW_UART2_tx_ISR()

```
void baseSW_UART2_tx_ISR (
             void  )
```

Definition at line 547 of file baseSW.c.

4.1.3.7 checkStartConditions()

```
bool checkStartConditions ( )
```

Definition at line 317 of file baseSW.c.

4.1.3.8 disableHeat()

```
void disableHeat ( )
```

Definition at line 282 of file baseSW.c.

4.1.3.9 enableHeat()

```
void enableHeat ( )
```

Definition at line 278 of file baseSW.c.

4.1.3.10 genericTranciverFunction()

```
void genericTranciverFunction ( )
```

Definition at line 457 of file baseSW.c.

4.1.3.11 IdleState_callback()

```
void IdleState_callback ( )
```

Definition at line 287 of file baseSW.c.

4.1.3.12 loadBuffer()

```
void loadBuffer ( )
```

Definition at line 312 of file baseSW.c.

4.1.3.13 ReadEEPROM_callback()

```
void ReadEEPROM_callback ( )
```

Definition at line 493 of file baseSW.c.

4.1.3.14 ReadTemperatureData_callback()

```
void ReadTemperatureData_callback ( )
```

Definition at line 342 of file baseSW.c.

4.1.3.15 ReceiveFTDI_callback()

```
void ReceiveFTDI_callback ( )
```

Definition at line 395 of file baseSW.c.

4.1.3.16 ReceiveNextionData_callback()

```
void ReceiveNextionData_callback ( )
```

Definition at line 352 of file baseSW.c.

4.1.3.17 TranscieveNextionDATA_callback()

```
void TranscieveNextionDATA_callback ( )
```

Definition at line 484 of file baseSW.c.

4.1.3.18 TransciveFTDI_callback()

```
void TransciveFTDI_callback ( )
```

Definition at line 488 of file baseSW.c.

4.1.3.19 WriteEEPROM_callback()

```
void WriteEEPROM_callback ( )
```

WriteEEPROM_callback

@Summary Copy data from temperature buffer to EEPROM

@Description This defines the object in the i2c queue. Each entry is a composed of a list of TRBs, the number of the TRBs and the status of the currently processed TRB.

Definition at line 510 of file baseSW.c.

4.1.4 Variable Documentation

4.1.4.1 EEPROM_ADDRESS

```
const uint16_t EEPROM_ADDRESS = 0x00
```

Definition at line 20 of file baseSW.c.

4.1.4.2 ftdiStatus

```
FTDI_STATUS ftdiStatus = NORMAL_OPERATION  [static]
```

Definition at line 214 of file baseSW.c.

4.1.4.3 HEAT_IN_PROGRESS

```
uint8_t HEAT_IN_PROGRESS = false  [static]
```

Definition at line 248 of file baseSW.c.

4.1.4.4 heatProfileBuffer

```
uint8_t heatProfileBuffer[HEAT_PROFILE_SIZE]
```

Heat profile buffer for memory operations. It can't be used directily, it has to be loaded into heatProfileCurrent

Definition at line 109 of file baseSW.c.

4.1.4.5 heatProfileCurrent

```
uint8_t heatProfileCurrent[HEAT_PROFILE_SIZE]
```

Currently selected heat profile.

Definition at line 108 of file baseSW.c.

4.1.4.6 heatProfileDefult

```
uint8_t heatProfileDefult[HEAT_PROFILE_SIZE]
```

Default heat profile, cannot be deleted

Definition at line 110 of file baseSW.c.

4.1.4.7 IdleState

```
stateTaskList* IdleState = NULL
```

Idle state.

The task connected to this state cannot be deleted from the task que. This task provides the toggling protection to the SSR, ensuring that the software is properly running. This task is also responsible for disableing the heating process if one of the limits is exceeded.

Definition at line 47 of file baseSW.c.

4.1.4.8 INTERNAL_MAX_TEMPERATURE

```
const uint16_t INTERNAL_MAX_TEMPERATURE = 0x0000
```

Definition at line 21 of file baseSW.c.

4.1.4.9 nextionStatus

```
NEXTION_STATUS nextionStatus = NEXTION_NORMAL_OPERATION  [static]
```

Definition at line 230 of file baseSW.c.

4.1.4.10 ReadEEPROM

```
stateTaskList* ReadEEPROM = NULL
```

Read heat profile from EEPROM.

Heat profile is read from the EEPROM via 400kHz I2C communication interface. This task use sequential read implemented in EEPROM_driver.c for the maximum transmission speed. 24LC64

Definition at line 98 of file baseSW.c.

4.1.4.11 ReadTemperatureData

```
stateTaskList* ReadTemperatureData = NULL
```

Read data from MAX31855KASA+T state.

The task connected to this state reads 4 bytes of data from MAX31855KASA+T. Hot junction temperature is stored in 14 bit format while the cold junction temperature is only 11 bits. Besides the measured temperatures, diagnostic data can also be read from the IC. IC datasheet

Definition at line 57 of file baseSW.c.

4.1.4.12 ReceiveFTDI

```
stateTaskList* ReceiveFTDI = NULL
```

Receive data from FTDI state.

In this task varios control commands are received from the PC via the FTDI UART USB bridge. Heating process can be enabled or disabled, new heat profile can be choosen, and it can be loaded from the EEPROM to the microcontroller. New heat profiles (generated on the PC) can be downloaded into the microcontroller and it can be saved into the EEPROM for further use. FT232R

Definition at line 77 of file baseSW.c.

4.1.4.13 ReceiveNextionData

```
stateTaskList* ReceiveNextionData = NULL
```

Receive data from Nextion HMI state.

In this task varios control commands are received from the Nextion touch screen HMI. Heating process can be enabled or disabled, new heat profile can be choosen, and it can be loaded from the EEPROM to the microcontroller.
NX4832T035

Definition at line 66 of file baseSW.c.

4.1.4.14 SENSOR_DATA_ARRAYS

```
IC_MAX31855_DATA SENSOR_DATA_ARRAYS[SENSOR_DATA_STORE_LENGTH]  [static]
```

Definition at line 195 of file baseSW.c.

4.1.4.15 SENSOR_DATA_HANDLER

```
SENSOR_DATA SENSOR_DATA_HANDLER  [static]
```

Definition at line 194 of file baseSW.c.

4.1.4.16 temperatureBufferArray

```
TEMPERATURE_BUFFER temperatureBufferArray[3]
```

Definition at line 158 of file baseSW.c.

4.1.4.17 temperatureHeatProfile

```
TEMPERATURE_PROFILE temperatureHeatProfile  [static]
```

Global varible, it stores heat profile related data

Definition at line 157 of file baseSW.c.

4.1.4.18 THERMOCOUPLE_MAX_TEMPERATURE

```
const uint16_t THERMOCOUPLE_MAX_TEMPERATURE = 0x0000
```

Definition at line 22 of file baseSW.c.

4.1.4.19 TranscieveNextionDATA

```
stateTaskList* TranscieveNextionDATA = NULL
```

Broadcast temperature data to the Nextion HMI.

Broadcast temperature data to the Nextion HMI. NX4832T035

Definition at line 84 of file baseSW.c.

4.1.4.20 TransciveFTDI

```
stateTaskList* TransciveFTDI = NULL
```

Broadcast temperature data to the PC.

Broadcast temperature data to the PC FT232R

Definition at line 90 of file baseSW.c.

4.1.4.21 transciveObj

```
TRANSCIEVE_OBJ transciveObj [static]
```

Definition at line 269 of file baseSW.c.

4.1.4.22 WriteEEPROM

```
stateTaskList* WriteEEPROM = NULL
```

Write heat profile into EEPROM.

Heat profile is written into the EEPROM via 400kHz I2C communication interface. This task use page write implemented in EEPROM_driver.c for the maximum transmission speed. 24LC64

Definition at line 106 of file baseSW.c.

4.2 baseSW.h File Reference

```
#include "stateTaskHandler.h"
```

Functions

- stateTaskList * baseSW_Initialize (void)

4.2.1 Function Documentation

4.2.1.1 baseSW_Initialize()

```
stateTaskList* baseSW_Initialize (
            void  )
```

Definition at line 559 of file baseSW.c.

4.3 EEPROM_driver.c File Reference

```
#include <xc.h>
#include "EEPROM_driver.h"
#include "mcc_generated_files/i2c1.h"
#include "piclib30_wrapper.h"
```

Functions

- I2C1_MESSAGE_STATUS EEPROM_Read (uint16_t slaveDeviceAddress, uint16_t dataAddress, uint8_t *pData, uint16_t nCount)
- I2C1_MESSAGE_STATUS EEPROM_Write (uint16_t slaveDeviceAddress, uint16_t dataAddress, uint8_t *pData, uint16_t nCount)
- I2C1_MESSAGE_STATUS EEPROM_WritePage (uint16_t slaveDeviceAddress, uint16_t dataAddress, uint8_t *pData)

4.3.1 Function Documentation

4.3.1.1 EEPROM_Read()

```
I2C1_MESSAGE_STATUS EEPROM_Read (
            uint16_t slaveDeviceAddress,
            uint16_t dataAddress,
            uint8_t * pData,
            uint16_t nCount )
```

Definition at line 6 of file EEPROM_driver.c.

4.3.1.2 EEPROM_Write()

```
I2C1_MESSAGE_STATUS EEPROM_Write (
            uint16_t slaveDeviceAddress,
            uint16_t dataAddress,
            uint8_t * pData,
            uint16_t nCount )
```

Definition at line 78 of file EEPROM_driver.c.

4.3.1.3 EEPROM_WritePage()

```
I2C1_MESSAGE_STATUS EEPROM_WritePage (
            uint16_t slaveDeviceAddress,
            uint16_t dataAddress,
            uint8_t * pData )
```

Definition at line 153 of file EEPROM_driver.c.

4.4 EEPROM_driver.h File Reference

```
#include "mcc_generated_files/i2c1.h"
```

Macros

- #define DEVICE_RETRY_MAX 100
- #define DEVICE_ADDRESS 0x50
- #define DEVICE_TIMEOUT 50

Functions

- I2C1_MESSAGE_STATUS EEPROM_Read (uint16_t slaveDeviceAddress, uint16_t dataAddress, uint8_t *pData, uint16_t nCount)
- I2C1_MESSAGE_STATUS EEPROM_Write (uint16_t slaveDeviceAddress, uint16_t dataAddress, uint8_t *pData, uint16_t nCount)
- I2C1_MESSAGE_STATUS EEPROM_WritePage (uint16_t slaveDeviceAddress, uint16_t dataAddress, uint8_t *pData)

4.4.1 Macro Definition Documentation

4.4.1.1 DEVICE_ADDRESS

```
#define DEVICE_ADDRESS 0x50
```

Definition at line 12 of file EEPROM_driver.h.

4.4.1.2 DEVICE_RETRY_MAX

```
#define DEVICE_RETRY_MAX 100
```

Definition at line 11 of file EEPROM_driver.h.

4.4.1.3 DEVICE_TIMEOUT

```
#define DEVICE_TIMEOUT 50
```

Definition at line 13 of file EEPROM_driver.h.

4.4.2 Function Documentation

4.4.2.1 EEPROM_Read()

```
I2C1_MESSAGE_STATUS EEPROM_Read (
            uint16_t slaveDeviceAddress,
            uint16_t dataAddress,
            uint8_t * pData,
            uint16_t nCount )
```

Definition at line 6 of file EEPROM_driver.c.

4.4.2.2 EEPROM_Write()

```
I2C1_MESSAGE_STATUS EEPROM_Write (
            uint16_t slaveDeviceAddress,
            uint16_t dataAddress,
            uint8_t * pData,
            uint16_t nCount )
```

Definition at line 78 of file EEPROM_driver.c.

4.4.2.3 EEPROM_WritePage()

```
I2C1_MESSAGE_STATUS EEPROM_WritePage (
            uint16_t slaveDeviceAddress,
            uint16_t dataAddress,
            uint8_t * pData )
```

Definition at line 153 of file EEPROM_driver.c.

4.5 main.c File Reference

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/pin_manager.h"
#include "mcc_generated_files/i2c1.h"
#include "stateTaskHandler.h"
#include "baseSW.h"
#include <stdlib.h>
```

Functions

- int main (void)

4.5.1 Function Documentation

4.5.1.1 main()

```
int main (
            void  )
```

Generated main.c file from MPLAB Code Configurator

@Company Microchip Technology Inc.

@File Name main.c

@Summary This is the generated main.c using PIC24 / dsPIC33 / PIC32MM MCUs.

@Description This source file provides main entry point for system initialization and application code development. Generation Information : Product Revision : PIC24 / dsPIC33 / PIC32MM MCUs - 1.170.0 Device : PIC24FJ256↩ GA702 The generated drivers are tested against the following: Compiler : XC16 v1.61 MPLAB : MPLAB X v5.45 Section: Included Files

Definition at line 66 of file main.c.

4.6 piclib30_wrapper.h File Reference

```
#include <libpic30.h>
```

Macros

- #define FCY 4000000UL

4.6.1 Macro Definition Documentation

4.6.1.1 FCY

```
#define FCY 4000000UL
```

Definition at line 15 of file piclib30_wrapper.h.

4.7 stateTaskHandler.c File Reference

```
#include "stateTaskHandler.h"
#include <stdlib.h>
```

Functions

- void stateTaskHandler (stateTaskList *task)
- void initilaizeTaskHandler (stateTaskList *idleTask)
- stateTaskList * createTask (void(*function)(void))
- void addTask (stateTaskList *idleTask, stateTaskList *newTask)

4.7.1 Function Documentation

4.7.1.1 addTask()

```
void addTask (
            stateTaskList * idleTask,
            stateTaskList * newTask )
```

stateTaskList

@Summary

@Description

Definition at line 77 of file stateTaskHandler.c.

4.7.1.2 createTask()

```
stateTaskList* createTask (
            void(*)(void) function )
```

stateTaskList

@Summary

@Description

Definition at line 59 of file stateTaskHandler.c.

4.7.1.3 initilaizeTaskHandler()

```
void initilaizeTaskHandler (
            stateTaskList * idleTask )
```

stateTaskList

@Summary

@Description

Definition at line 44 of file stateTaskHandler.c.

4.7.1.4 stateTaskHandler()

```
void stateTaskHandler (
            stateTaskList * task )
```

stateTaskList

@Summary

@Description stateTaskList

@Summary

@Description

Definition at line 25 of file stateTaskHandler.c.

4.8 stateTaskHandler.h File Reference

Data Structures

- struct stateTaskList_s

Typedefs

- typedef struct stateTaskList_s stateTaskList

Functions

- void stateTaskHandler (stateTaskList *task)
- stateTaskList * createTask (void(*function)(void))
- void initilaizeTaskHandler (stateTaskList *idleTask)
- void addTask (stateTaskList *idleTask, stateTaskList *newTask)

4.8.1 Typedef Documentation

4.8.1.1 stateTaskList

```
typedef struct stateTaskList_s stateTaskList
```

Definition at line 1 of file stateTaskHandler.h.

4.8.2 Function Documentation

4.8.2.1 addTask()

```
void addTask (
            stateTaskList * idleTask,
            stateTaskList * newTask )
```

stateTaskList

@Summary

@Description

Definition at line 77 of file stateTaskHandler.c.

4.8.2.2 createTask()

```
stateTaskList* createTask (
            void(*)(void) function )
```

stateTaskList

@Summary

@Description

Definition at line 59 of file stateTaskHandler.c.

4.8.2.3 initilaizeTaskHandler()

```
void initilaizeTaskHandler (
            stateTaskList * idleTask )
```

stateTaskList

@Summary

@Description

Definition at line 44 of file stateTaskHandler.c.

4.8.2.4 stateTaskHandler()

```
void stateTaskHandler (
            stateTaskList * task )
```

stateTaskList

@Summary

@Description stateTaskList

@Summary

@Description

Definition at line 25 of file stateTaskHandler.c.

Index