SHT7x PSoC Component

Generated by Doxygen 1.8.14

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 SHT7x.h File Reference

Header file for interfacing with a SHT7x sensor.

```
#include "cytypes.h"
#include "CyLib.h"
```

Macros

- #define SHT7x_ERR_NO_ACK 1
- #define SHT7x_ERR_CRC 2
- #define SHT7x_ERR_TO 3
- #define SHT7x_MEAS_READY 4
- #define SHT7x_READ_TEMP 0
- #define SHT7x_READ_HUMI 1
- #define SHT7x_BLOCK 1
- #define SHT7x_NON_BLOCK 0
- #define SHT7x_BATTERY_LOW 1
- #define SHT7x_BATTERY_OK 0

Functions

- uint8_t SHT7x_Start (void)

  Start the component.
- uint8_t SHT7x_Measure (float ∗temp, float ∗humi, float ∗dew)

  Complete blocking measurement.
- uint8_t SHT7x_MeasureTemp (float ∗temp)

  Temperature blocking measurement.
- uint8_t SHT7x_MeasureHumi (float ∗humi, float temp)

  Humidity blocking measurement.
- uint8_t SHT7x_StartMeasure (uint8_t cmd)

  Initiate measurement.
- uint8_t SHT7x_StartMeasureTemp ()

Initiate non blocking temperature measurement.

- uint8_t SHT7x_StartMeasureHumi ()

  Initiate non blocking humidity measurement.
- uint8_t SHT7x_Meas (uint8_t cmd, uint16_t *result, int block)

  Initiate measurement.
- uint8_t SHT7x_MeasReady (void)

  Check if non-blocking measurement has completed.
- uint8_t SHT7x_PutByte (char command)

  Write a byte to the sensor and check acknowledgement.
- uint8_t SHT7x_GetByte (int ack)

  Get a byte from the sensor.
- void SHT7x_StartTransmission (void)

  Generate Sensirion-specific transmission start sequence.
- void SHT7x_ResetConnection (void)

  Reset connection with sensor.
- uint8_t SHT7x_Reset (void)

  Public reset function.
- uint8_t SHT7x_GetResult (uint16_t *result)

  Get measurement result from sensor (plus CRC, if enabled)
- uint8_t SHT7x_ReadSR (uint8_t *result)

  Read to status register.
- uint8_t SHT7x_WriteSR (uint8_t value)

  Write to status register.
- uint8_t SHT7x_SetHighResolution (void)

  Set resolution to high.
- uint8_t SHT7x_SetLowResolution (void)

  Set resolution to low.
- uint8_t SHT7x_ActivateOTPReload (void)

  Activate OTP Reload.
- uint8_t SHT7x_DeactivateOTPReload (void)

  Deactivate OTP Reload.
- uint8_t SHT7x_ActivateHeater (void)

  Activate on chip heater.
- uint8_t SHT7x_DeactivateHeater (void)

  Deactivate on chip heater.
- uint8_t SHT7x_CheckEndOfBattery (void)

  Check if vdd value is below 2.47 V.
- float SHT7x_CalcTemp (uint16_t rawData)

  Calcuate temperature in degrees C.
- float SHT7x_CalcHum (uint16_t rawData, float temp)

  Calcuate relative humidity with temperature compensation.
- float SHT7x_CalcDewpoint (float humi, float temp)

  Calcuate dewpoint.

2.1.1 Detailed Description

Header file for interfacing with a SHT7x sensor.

This header contains macros and function prototypes to interface the PSoC with a Sensirion SHT7x temperature and humidity sensor. This work is based on the Sensirion SHT7x Arduino library written by Carl Jackson: https://github.com/spease/Sensirion

Author

Davide Marzorati

2.1.2 Macro Definition Documentation

2.1.2.1 SHT7x_BATTERY_LOW

```
#define SHT7x_BATTERY_LOW 1
```

Value returned when battery is low

2.1.2.2 SHT7x_BATTERY_OK

```
#define SHT7x_BATTERY_OK 0
```

Value returned when battery is ok

2.1.2.3 SHT7x_BLOCK

```
#define SHT7x_BLOCK 1
```

Value to pass to functions to start blocking measurements

2.1.2.4 SHT7x_ERR_CRC

```
#define SHT7x_ERR_CRC 2
```

Error code for CRC failure.

2.1.2.5 SHT7x_ERR_NO_ACK

```
#define SHT7x_ERR_NO_ACK 1
```

Error code for acknowledgment not received.

2.1.2.6 SHT7x_ERR_TO

```
#define SHT7x_ERR_TO 3
```

Error code for measurement timeout.

2.1.2.7 SHT7x_MEAS_READY

```
#define SHT7x_MEAS_READY 4
```

Value returned when measurement is ready

2.1.2.8 SHT7x_NON_BLOCK

```
#define SHT7x_NON_BLOCK 0
```

Value to pass to functions to start non-blocking measurements

2.1.2.9 SHT7x_READ_HUMI

```
#define SHT7x_READ_HUMI 1
```

Value to pass to read humidity.

2.1.2.10 SHT7x_READ_TEMP

```
#define SHT7x_READ_TEMP 0
```

Helper value to use as a parameter to read temperature.

2.1.3 Function Documentation

2.1.3.1 SHT7x_ActivateHeater()

```
uint8_t SHT7x_ActivateHeater (
            void  )
```

Activate on chip heater.

Calling this function activates the on chip heater. The heater may increase the temperature of the sensor by 5-10℃ (9-18℉) beyond ambient temperature. The heater draw 8ma @ 5 V supply voltage. The heater can be helpful for functionality analysis. Humidity and temperature readings before and after applying the heater are compared. Temperature shall increase while relative humidity decreases at the same time. Dew point shall remain the same.

Note: the temperature reading will display the temperature of the heated sensor element and not the ambient temperature. Furthermore, the sensor is not qualified fo continuous application of the heater.

2.1.3.2 SHT7x_ActivateOTPReload()

```
uint8_t SHT7x_ActivateOTPReload (
            void  )
```

Activate OTP Reload.

If this is called, the calibration data are uploaded to the register before each measurement. This increases measurement time by about 10 ms.

2.1.3.3 SHT7x_CalcDewpoint()

```
float SHT7x_CalcDewpoint (
            float humi,
            float temp )
```

Calcuate dewpoint.

Starting from the value of humidity and temperature, it computes the dewpoint value

Parameters

humi	humidity
temp	temperature

Returns

dewpoint value

2.1.3.4 SHT7x_CalcHum()

```
float SHT7x_CalcHum (
            uint16_t rawData,
            float temp )
```

Calcuate relative humidity with temperature compensation.

Starting from the raw sensor data, returns the value of relative humidity with temperature compensation.

Parameters

rawData	raw sensor data
temp	value of temperature

Returns

relative humidity (%)

2.1.3.5 SHT7x_CalcTemp()

```
float SHT7x_CalcTemp (
            uint16_t rawData )
```

Calcuate temperature in degrees C.

Starting from the raw sensor data, returns the temperature in degrees (C).

Parameters

rawData	raw sensor data

Returns

temperature in degrees C

2.1.3.6 SHT7x_DeactivateOTPReload()

```
uint8_t SHT7x_DeactivateOTPReload (
            void  )
```

Deactivate OTP Reload.

If this is called, the calibration data are not uploaded to the register before each measurement. This reduces measurement time by about 10 ms.

2.1.3.7 SHT7x_GetByte()

```
uint8_t SHT7x_GetByte (
            int ack )
```

Get a byte from the sensor.

This function gets a byte from the sensor and allows to choose if doing or skipping the acknowledgement.

Parameters

ack	0 if no ack, 1 if ack

Returns

the byte read

2.1.3.8 SHT7x_GetResult()

```
uint8_t SHT7x_GetResult (
            uint16_t * result )
```

Get measurement result from sensor (plus CRC, if enabled)

Parameters

result	Variable where the result will be stored

Return values

S_Err_CRC	if CRC check failed
0	if everything ok

2.1.3.9 SHT7x_Meas()

```
uint8_t SHT7x_Meas (
            uint8_t cmd,
            uint16_t * result,
            int block )
```

Initiate measurement.

This functions starts a measurement, and if this is blocking it waits until the measurement is complete before returning.

Parameters

cmd	SHT7x_TEMP for temperature, SHT7x_HUMI for humidity
result	Variable where to store the result
block	True if blocking

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
SHT7x_ERR_CRC	If CRC check failed
SHT7x_ERR_TO	If there was a timeout in the measurement

2.1.3.10 SHT7x_MeasReady()

```
uint8_t SHT7x_MeasReady (
            void  )
```

Check if non-blocking measurement has completed.

Non-zero return indicates complete.

Returns

SHT7x_MEAS_READY 0 if measurement is complete

2.1.3.11 SHT7x_Measure()

```
uint8_t SHT7x_Measure (
            float * temp,
            float * humi,
            float * dew )
```

Complete blocking measurement.

This functions performs a blocking measurement for both temperature and humidity values.

Parameters

temp	Variable where the temperature value will be stored
humi	Variable where the humidity value will be stored
dew	Variable where the dewpoint value will be stored

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
SHT7x_ERR_CRC	If CRC check failed
SHT7x_ERR_TO	If there was a timeout in the measurement
0	If everything ok

2.1.3.12 SHT7x_MeasureHumi()

```
uint8_t SHT7x_MeasureHumi (
            float * humi,
            float temp )
```

Humidity blocking measurement.

This functions performs a blocking measurement for humidity.

Parameters

humi	Variable where the humidity value will be stored
temp	Value of temperature

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
SHT7x_ERR_CRC	If CRC check failed
SHT7x_ERR_TO	If there was a timeout in the measurement
0	If everything ok

2.1.3.13 SHT7x_MeasureTemp()

```
uint8_t SHT7x_MeasureTemp (
            float * temp )
```

Temperature blocking measurement.

This functions performs a blocking measurement for temperature.

Parameters

temp	Variable where the temperature value will be stored

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
SHT7x_ERR_CRC	If CRC check failed
SHT7x_ERR_TO	If there was a timeout in the measurement
0	If everything ok

2.1.3.14 SHT7x_PutByte()

```
uint8_t SHT7x_PutByte (
            char d )
```

Write a byte to the sensor and check acknowledgement.

This function sends a byte to the sensor and check if the sensor acknowledged it. The sensor returns a value greater than 0 if no acknowledgement was received.

Parameters

d	the byte to write

Return values

S_Err_NoAck	if no ack
0	if everything ok

2.1.3.15 SHT7x_ReadSR()

```
uint8_t SHT7x_ReadSR (
            uint8_t * result )
```

Read to status register.

Parameters

result	Variable where the SR byte will be stored

Return values

S_Err_NoAck	if no acknowledgement was received

Return values

0	otherwise

2.1.3.16 SHT7x_Reset()

```
uint8_t SHT7x_Reset (
            void  )
```

Public reset function.

Reset communication and reset status register to default

2.1.3.17 SHT7x_ResetConnection()

```
void SHT7x_ResetConnection (
            void  )
```

Reset connection with sensor.

Set up a predefined sequence of signals on clock and data line to restore a connection with the sensor.

2.1.3.18 SHT7x_SetHighResolution()

```
uint8_t SHT7x_SetHighResolution (
            void  )
```

Set resolution to high.

Resolution is set to 12 bits for humidity, and to 14 bits for temperature

2.1.3.19 SHT7x_SetLowResolution()

```
uint8_t SHT7x_SetLowResolution (
            void  )
```

Set resolution to low.

Resolution is set to 8 bits for humidity, and to 12 bits for temperature

2.1.3.20 SHT7x_Start()

```
uint8_t SHT7x_Start (
            void  )
```

Start the component.

This function should always be called before starting any measurements, since it sets up the sensor based on the parameters specified in the top design.

Return values

SHT7x_ERR_NO_ACK	if no acknowledgment was received
0	if everything ok

2.1.3.21 SHT7x_StartMeasure()

```
uint8_t SHT7x_StartMeasure (
            uint8_t cmd )
```

Initiate measurement.

This functions starts a measurement for temperature or humidity based on the passed parameter.

Parameters

cmd	SHT7x_TEMP for temperature, SHT7x_HUMI for humidity

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
0	If everything ok

2.1.3.22 SHT7x_StartMeasureHumi()

```
uint8_t SHT7x_StartMeasureHumi ( )
```

Initiate non blocking humidity measurement.

This functions starts a non blocking measurement for humidity.

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
0	If everything ok

2.1.3.23 SHT7x_StartMeasureTemp()

```
uint8_t SHT7x_StartMeasureTemp ( )
```

Initiate non blocking temperature measurement.

This functions starts a non blocking measurement for temperature.

Return values

SHT7x_ERR_NO_ACK	If the sensor did not acknowledge
0	If everything ok

2.1.3.24 SHT7x_StartTransmission()

```
void SHT7x_StartTransmission (
             void  )
```

Generate Sensirion-specific transmission start sequence.

This is where Sensirion does not conform to the I2C standard.

2.1.3.25 SHT7x_WriteSR()

```
uint8_t SHT7x_WriteSR (
             uint8_t value )
```

Write to status register.

Parameters

value	the byte to write to status register

Return values

S_Err_NoAck	if no acknowledgement was received
0	otherwise