L.A.M.P

Genereret af Doxygen 1.8.11

Indhold

1 Indeks over datastrukturer

1.1 Datastrukturer

Her er datastrukturerne med korte beskrivelser:

2 Fil-indeks

2.1 Filoversigt

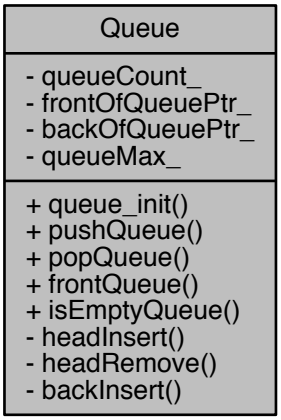
Her er en liste over alle filer med korte beskrivelser:

3 Datastruktur-documentation

3.1 Queue Klasse-reference

```
#include <queue.h>
```

Samarbejdsdiagram for Queue:



Offentlige metoder

- void queue_init (uint8 queueSize)

  Initialize queue.
- void pushQueue (const struct Data data)

  Insert element in FIFO ordre in queue.
- void popQueue ()

  Remove front element of the queue.
- struct Data frontQueue ()

  Return data from the front element in the queue.
- int isEmptyQueue ()

  Return 1 (true) if queue is empty or 0 (false) if not.

Private metoder

- void headInsert (struct Node **headPtr, const struct Data data)

  Insert element in the front of the queue.
- void headRemove (struct Node **headPtr)

  Remove element in front af queue.
- void backInsert (struct Node **backPtr, const struct Data data)

  Insert element in the back of the queue.

Private attributter

- uint8 queueCount_

  Conuter for elements in the queue.

Statiske, private attributter

- static struct Node * frontOfQueuePtr_

  Pointer to front element in queue.
- static struct Node * backOfQueuePtr_

  Pointer to back element in queue.
- static uint8 queueMax_

  Maximum elements in queue.

3.1.1 Detaljeret beskrivelse

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

3.1.2 Dokumentation af medlemsfunktioner

3.1.2.1 void backInsert (struct Node ** *backPtr,* const struct Data *data*) [private]

Insert element in the back of the queue.

Parametre

in	*backPtr*	Pointer to back of queue
in	*data*	Struct of data to be placed in the queue

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 213 i filen queue.c.

Indeholder referencer til Node::data_ og Node::next_.

```
214 {
215   if(*backPtr == NULL)
216   {
217     return;
218   }
219
220   struct Node* next = (*backPtr)->next_;
221   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
222   temp->data_ = data;
223   temp->next_ = next;
224   (*backPtr)->next_ = temp;
225 }
```

3.1.2.2 struct Data frontQueue (void)

Return data from the front element in the queue.

Returnerer

Data

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

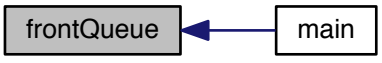
Defineret på linje 141 i filen queue.c.

Indeholder referencer til Node::data_.

Refereret til af main().

```
142 {
143   return frontOfQueuePtr_->data_;
144 }
```

Her er kalder-grafen for denne funktion:



3.1.2.3 void headInsert (struct Node ** *headPtr*, const struct Data *data*) [private]

Insert element in the front of the queue.

Parametre

in	*headPtr*	Pointer to front of queue
in	*data*	Struct of data to be placed in the queue

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 173 i filen queue.c.

Indeholder referencer til Node::data_ og Node::next_.

```
174 {
175   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
176   if(temp == NULL)
177   {
178     return;
179   }
180
181   temp->data_ = data;
182   temp->next_ = NULL;
183
184   *headPtr = temp;
185 }
```

3.1.2.4 void headRemove (struct Node ** *headPtr*) [private]

Remove element in front af queue.

Parametre

in	*headPtr*	Pointer to front of queue

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 194 i filen queue.c.

Indeholder referencer til Node::next_.

```
195 {
196   if(headPtr != NULL)
197   {
198 struct Node* condemned;
199     condemned = *headPtr;
200     *headPtr = (*headPtr)->next_;
201     free(condemned);
202   }
203 }
```

3.1.2.5 int isEmptyQueue (void)

Return 1 (true) if queue is empty or 0 (false) if not.

Returnerer

int

Forfatter

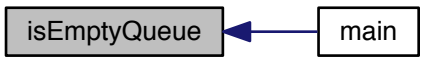
Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 153 i filen queue.c.

Refereret til af main().

```
154 {
155   if(frontOfQueuePtr_ == NULL)
156   {
157     return 1;
158   }
159   else
160   {
161     return 0;
162   }
163 }
```

Her er kalder-grafen for denne funktion:



3.1.2.6 void popQueue (void)

Remove front element of the queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

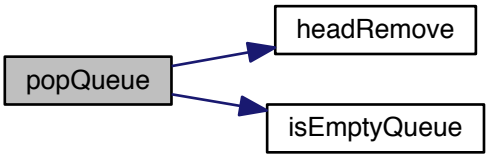
Defineret på linje 124 i filen queue.c.

Indeholder referencer til headRemove(), isEmptyQueue() og queueCount_.

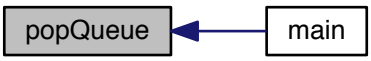
Refereret til af main().

```
125 {
126   headRemove(&frontOfQueuePtr_);
127   queueCount_--;
128   if(isEmptyQueue() == 1)
129   {
130     backOfQueuePtr_ = NULL;
131   }
132 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.1.2.7 void pushQueue (const struct Data *data*)

Insert element in FIFO ordre in queue.

Parametre

in	*data*	Struct of data to be placed in the queue

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 99 i filen queue.c.

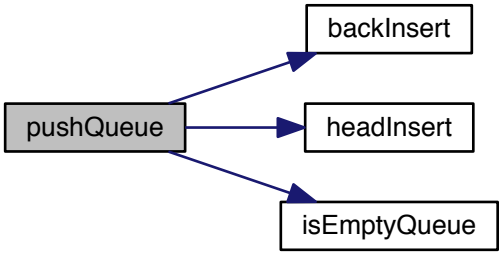
Indeholder referencer til backInsert(), headInsert(), isEmptyQueue(), Node::next_ og queueCount_.

Refereret til af i2c_rx().

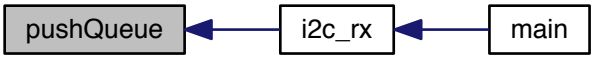
```
100 {
101   if(queueCount_<queueMax_)
102   {
103   if(isEmptyQueue() != 1)
104   {
105     backInsert(&backOfQueuePtr_, data);
106     backOfQueuePtr_ = backOfQueuePtr_->next_;
107     queueCount_++;
108   }
109     else
110     {
```

```
111         headInsert(&frontOfQueuePtr_, data);
112         backOfQueuePtr_ = frontOfQueuePtr_;
113         queueCount_++;
114       }
115     }
116 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.1.2.8 void queue_init (uint8 *queueSize*)

Initialize queue.

Parametre

in	*queueSize*	Size of queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

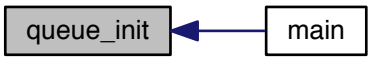
Defineret på linje 82 i filen queue.c.

Indeholder referencer til Node::next_ og queueCount_.

Refereret til af main().

```
83 {
84   frontOfQueuePtr_ = NULL;
85   frontOfQueuePtr_->next_ = NULL;
86   backOfQueuePtr_ = NULL;
87   backOfQueuePtr_->next_ = NULL;
88   queueMax_ = queueSize;
89   queueCount_ = 0;
90 }
```

Her er kalder-grafen for denne funktion:



3.1.3 Felt-dokumentation

3.1.3.1 struct Node* backOfQueuePtr_ [static], [private]

Pointer to back element in queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 44 i filen queue.c.

3.1.3.2 struct Node* frontOfQueuePtr_ [static], [private]

Pointer to front element in queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 36 i filen queue.c.

3.1.3.3 uint8 queueCount_ [private]

Conuter for elements in the queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 52 i filen queue.c.

Refereret til af main().

3.1.3.4 uint8 queueMax_ [static],[private]

Maximum elements in queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 60 i filen queue.c.

Dokumentationen for denne klasse blev genereret ud fra filen:

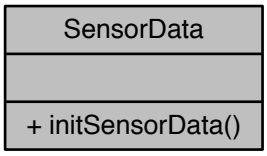
- queue.c

3.2 SensorData Klasse-reference

Container for sensor data.

`#include <SensorData.h>`

Samarbejdsdiagram for SensorData:



Offentlige metoder

- void initSensorData ()

  Initializes the SensorData struct parts that need initial values.

3.2.1 Detaljeret beskrivelse

Container for sensor data.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

3.2.2 Dokumentation af medlemsfunktioner

3.2.2.1 void initSensorData ( )

Initializes the SensorData struct parts that need initial values.

Debug define. Comment out to suppress debug prints

Forfatter

Simon Nejmann (19981127@uni.au.dk)
Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 17 i filen SensorData.c.

Indeholder referencer til sensorDataT::bluePWMPct, sensorDataT::desiredLux, sensorDataT::desiredTimeDistance, sensorDataT::greenPWMPct, initCircularMean(), sensorDataT::ledPower, sensorDataT::LumenMean, sensorDataT::movementAlertOn, sensorDataT::redPWMPct og sensorData.

Refereret til af main().

```
18 {
19     sensorData.desiredLux = 0;
20     sensorData.desiredTimeDistance = 580;
21
22     sensorData.movementAlertOn = 0;
23
24     sensorData.ledPower = 0;
25     sensorData.redPWMPct = 255;
26     sensorData.greenPWMPct = 255;
27     sensorData.bluePWMPct = 255;
28
29 #ifdef DEBUG_ON
30     sensorData.desiredLux = 0;
31     sensorData.movementAlertOn = 1;
32     sensorData.ledPower = 1;
33 #endif
34
35     initCircularMean(&sensorData.LumenMean);
36 }
```

Her er kald-grafen for denne funktion:

Her er kalder-grafen for denne funktion:

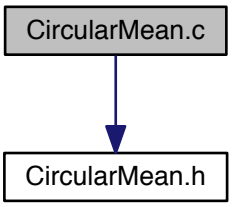
Dokumentationen for denne klasse blev genereret ud fra filerne:

- SensorData.h
- SensorData.c

4 Fil-dokumentation

4.1 CircularMean.c filreference

```
#include "CircularMean.h"
```
Inklusions-afhængighedsgraf for CircularMean.c:



Funktioner

- void initCircularMean (struct CircularMean ∗buf)

  Initialize a CircularMean struct.
- void insertValue (struct CircularMean ∗buf, int val)

  Insert value, possibly overwriting oldest entry.
- int getMeanValue (struct CircularMean ∗buf)

  Get the average value of all the ints in the buffer.

4.1.1 Funktions-dokumentation

4.1.1.1 int getMeanValue (struct CircularMean ∗ *buf*)

Get the average value of all the ints in the buffer.

Parametre

in	*buf*	Pointer to the CircularMean

Returnerer

The average value calculated

Forfatter

Simon Nejmann (19981127@uni.au.dk)

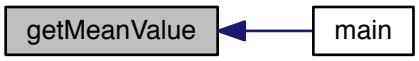
Defineret på linje 53 i filen CircularMean.c.

Indeholder referencer til CircularMean::num og CircularMean::queue.

Refereret til af main().

```
54 {
55     if (buf->num == 0)
56         return 0;
57
58     int i, temp = 0;
59     for (i = 0; i < buf->num; ++i) {
60         temp += buf->queue[i];
61     }
62
63     return temp / buf->num;
64 }
```

Her er kalder-grafen for denne funktion:



4.1.1.2 void initCircularMean (struct CircularMean * *buf*)

Initialize a CircularMean struct.

Parametre

in	*buf*	Pointer to the CircularMean to initialize

Forfatter

Simon Nejmann (19981127@uni.au.dk)

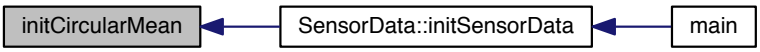
Defineret på linje 19 i filen CircularMean.c.

Indeholder referencer til BUF_SIZE, CircularMean::num, CircularMean::queue og CircularMean::start.

Refereret til af SensorData::initSensorData().

```
20 {
21     int i;
22     for (i = 0; i < BUF_SIZE; ++i) {
23         buf->queue[i] = -1;
24     }
25     buf->start = 0;
26     buf->num = 0;
27 }
```

Her er kalder-grafen for denne funktion:



4.1.1.3 void insertValue (struct CircularMean * *buf*, int *val*)

Insert value, possibly overwriting oldest entry.

Parametre

in	*buf*	Pointer to the CircularMean to insert into
in	*val*	Value to insert

Forfatter

Simon Nejmann (19981127@uni.au.dk)

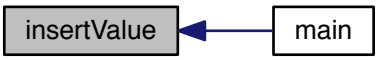
Defineret på linje 36 i filen CircularMean.c.

Indeholder referencer til BUF_SIZE, CircularMean::num, CircularMean::queue og CircularMean::start.

Refereret til af main().

```
37 {
38     buf->queue[buf->start] = val;
39     buf->start = (buf->start +1) % BUF_SIZE;
40     ++(buf->num);
41     if (buf->num > BUF_SIZE) {
42         buf->num = BUF_SIZE;
43     }
44 }
```

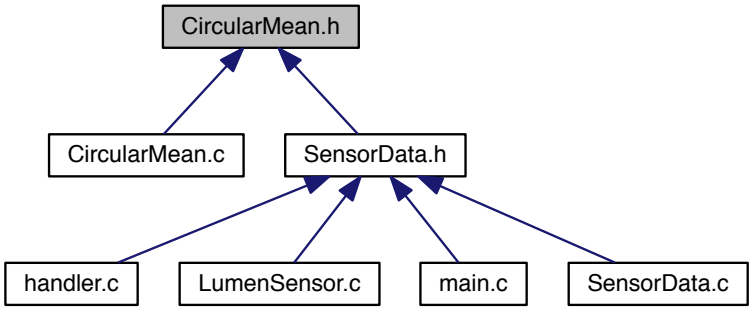
Her er kalder-grafen for denne funktion:



4.2 CircularMean.h filreference

Circular buffer used for calculating the average of a series of values. Queue is circular and can contain BUF_SIZE elements. start points to the next element to overwrite. num is the number of items inserted - it grows to BUF_SIZE and then stays there.

Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



Datastrukturer

- class CircularMean

  Struct for the buffer. Mere...

#Defines

- #define BUF_SIZE 30

Funktioner

- void initCircularMean (struct CircularMean ∗buf)

  Initialize a CircularMean struct.
- void insertValue (struct CircularMean ∗buf, int val)

  Insert value, possibly overwriting oldest entry.
- int getMeanValue (struct CircularMean ∗buf)

  Get the average value of all the ints in the buffer.

4.2.1 Detaljeret beskrivelse

Circular buffer used for calculating the average of a series of values. Queue is circular and can contain BUF_SIZE elements. start points to the next element to overwrite. num is the number of items inserted - it grows to BUF_SIZE and then stays there.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

4.2.2 Datastruktur-documentation

4.2.2.1 class CircularMean

Struct for the buffer.

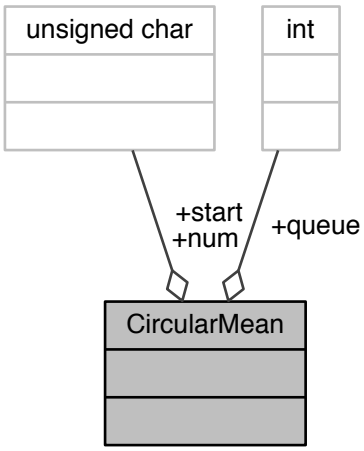
Circular buffer used for calculating the average of a series of values.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 17 i filen CircularMean.h.

Samarbejdsdiagram for CircularMean:



Data-felter

unsigned char	num	Number of values current in the array
int	queue[BUF_SIZE]	Array holding the inserted values
unsigned char	start	Points to the next array-cell to write in

4.2.3 #Define-dokumentation

4.2.3.1 #define BUF_SIZE 30

The size of the buffer. Max 255

Defineret på linje 12 i filen CircularMean.h.

Refereret til af initCircularMean() og insertValue().

4.2.4 Funktions-dokumentation

4.2.4.1 int getMeanValue (struct CircularMean * *buf*)

Get the average value of all the ints in the buffer.

Parametre

in	*buf*	Pointer to the CircularMean

Returnerer

The average value calculated

Forfatter

Simon Nejmann (19981127@uni.au.dk)

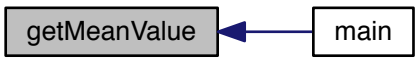
Defineret på linje 53 i filen CircularMean.c.

Indeholder referencer til CircularMean::num og CircularMean::queue.

Refereret til af main().

```
54 {
55     if (buf->num == 0)
56         return 0;
57 
58     int i, temp = 0;
59     for (i = 0; i < buf->num; ++i) {
60         temp += buf->queue[i];
61     }
62 
63     return temp / buf->num;
64 }
```

Her er kalder-grafen for denne funktion:



4.2.4.2 void initCircularMean (struct CircularMean * *buf*)

Initialize a CircularMean struct.

Parametre

in	*buf*	Pointer to the CircularMean to initialize

Forfatter

Simon Nejmann (19981127@uni.au.dk)

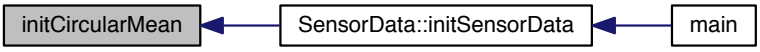
Defineret på linje 19 i filen CircularMean.c.

Indeholder referencer til BUF_SIZE, CircularMean::num, CircularMean::queue og CircularMean::start.

Refereret til af SensorData::initSensorData().

```
20 {
21     int i;
22     for (i = 0; i < BUF_SIZE; ++i) {
23         buf->queue[i] = -1;
24     }
25     buf->start = 0;
26     buf->num = 0;
27 }
```

Her er kalder-grafen for denne funktion:



4.2.4.3 void insertValue (struct CircularMean * *buf*, int *val*)

Insert value, possibly overwriting oldest entry.

Parametre

in	*buf*	Pointer to the CircularMean to insert into
in	*val*	Value to insert

Forfatter

Simon Nejmann (19981127@uni.au.dk)

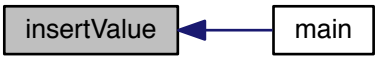
Defineret på linje 36 i filen CircularMean.c.

Indeholder referencer til BUF_SIZE, CircularMean::num, CircularMean::queue og CircularMean::start.

Refereret til af main().

```
37 {
38     buf->queue[buf->start] = val;
39     buf->start = (buf->start +1) % BUF_SIZE;
40     ++(buf->num);
41     if (buf->num > BUF_SIZE) {
42         buf->num = BUF_SIZE;
43     }
44 }
```

Her er kalder-grafen for denne funktion:



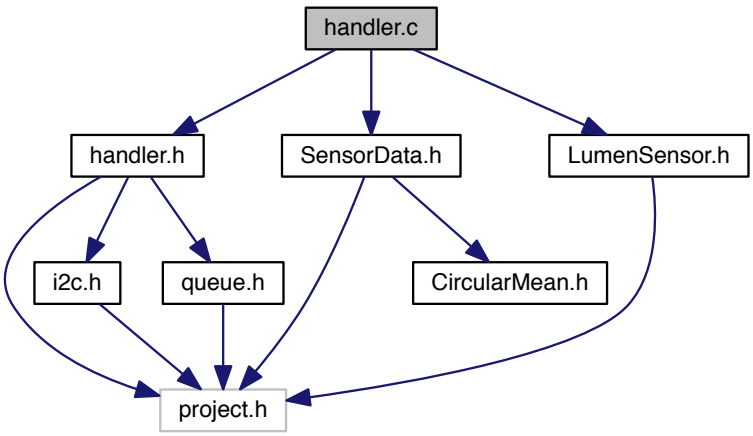
4.3 cyapicallbacks.h filreference

4.4 handler.c filreference

Communication handler for the Sensor-PSoC (to PSoC4Master)

```
#include "handler.h"
#include "SensorData.h"
#include "LumenSensor.h"
```

Inklusions-afhængighedsgraf for handler.c:



#Defines

- #define DEBUG_ON

Funktioner

- void handler (uint8 cmd, uint8 val)

  Communication handler for the Sensor-PSoC.

4.4.1 Detaljeret beskrivelse

Communication handler for the Sensor-PSoC (to PSoC4Master)

Forfatter

Simon Nejmann (19981127@uni.au.dk)

4.4.2 #Define-dokumentation

4.4.2.1 #define DEBUG_ON

Debug define. Comment out to suppress debug prints

Defineret på linje 12 i filen handler.c.

4.4.3 Funktions-dokumentation

4.4.3.1 void handler (uint8 *cmd,* uint8 *val*)

Communication handler for the Sensor-PSoC.

Parametre

in	*cmd*	The command to be executed
in	*val*	Optional command argument

Forfatter

Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 21 i filen handler.c.

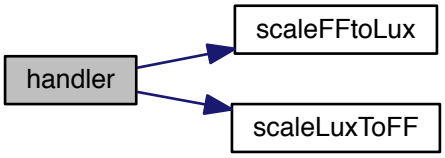
Indeholder referencer til sensorDataT::bluePWMPct, cmdDistanceAlert, cmdGetBlue, cmdGetDistance, cmdGetGreen, cmdGetLumen, cmdGetMovement, cmdGetPower, cmdGetRed, cmdMovementAlert, cmdSetBlue, cmdSetDistance, cmdSetGreen, cmdSetLumen, cmdSetMovement, cmdSetPower, cmdSetRed, sensorDataT::desiredLux, sensorDataT::desiredTimeDistance, sensorDataT::distance, sensorDataT::greenPWMPct, i2cTxBuffer, sensorDataT::ledPower, sensorDataT::lux, sensorDataT::movement, sensorDataT::movementAlertOn, sensorDataT::redPWMPct, scaleFFtoLux(), scaleLuxToFF() og sensorData.

Refereret til af main().

```
22 {
23     switch (cmd) {
24         case cmdGetRed :
25             i2cTxBuffer[1] = cmd;
26             i2cTxBuffer[2] = sensorData.redPWMPct;
27 #ifdef DEBUG_ON
28     DEBUG_PutString("cmdGetRed ");
29     DEBUG_PutHexByte(sensorData.redPWMPct);
30     DEBUG_PutCRLF();
31 #endif
32             break;
33         case cmdSetRed :
34             sensorData.redPWMPct = val;
35             RedPWM_WriteCompare(sensorData.redPWMPct);
36             RedPWM_Start();
37             sensorData.ledPower = 1;
38 #ifdef DEBUG_ON
39     DEBUG_PutString("cmdSetRed ");
40     DEBUG_PutHexByte(sensorData.redPWMPct);
41     DEBUG_PutCRLF();
42 #endif
43             break;
44         case cmdGetGreen :
45             i2cTxBuffer[1] = cmd;
46             i2cTxBuffer[2] = sensorData.greenPWMPct;
47 #ifdef DEBUG_ON
48     DEBUG_PutString("cmdGetGreen ");
49     DEBUG_PutHexByte(sensorData.greenPWMPct);
50     DEBUG_PutCRLF();
51 #endif
52             break;
53         case cmdSetGreen :
54             sensorData.greenPWMPct = val;
55             GreenPWM_WriteCompare(sensorData.greenPWMPct);
56             GreenPWM_Start();
57             sensorData.ledPower = 1;
58 #ifdef DEBUG_ON
59     DEBUG_PutString("cmdSetGreen ");
60     DEBUG_PutHexByte(sensorData.greenPWMPct);
61     DEBUG_PutCRLF();
62 #endif
63             break;
64         case cmdGetBlue :
65             i2cTxBuffer[1] = cmd;
66             i2cTxBuffer[2] = sensorData.bluePWMPct;
67 #ifdef DEBUG_ON
68     DEBUG_PutString("cmdGetBlue ");
69     DEBUG_PutHexByte(sensorData.bluePWMPct);
70     DEBUG_PutCRLF();
71 #endif
72             break;
73         case cmdSetBlue :
74             sensorData.bluePWMPct = val;
75             BluePWM_WriteCompare(sensorData.bluePWMPct);
76             BluePWM_Start();
77             sensorData.ledPower = 1;
78 #ifdef DEBUG_ON
79     DEBUG_PutString("cmdSetBlue ");
80     DEBUG_PutHexByte(sensorData.bluePWMPct);
81     DEBUG_PutCRLF();
82 #endif
83             break;
84
85         case cmdGetPower :
86             i2cTxBuffer[1] = cmd;
87             i2cTxBuffer[2] = sensorData.ledPower;
88 #ifdef DEBUG_ON
89     DEBUG_PutString("cmdGetPower ");
90     DEBUG_PutHexByte(sensorData.ledPower);
91     DEBUG_PutCRLF();
92 #endif
93             break;
94         case cmdSetPower :
95             sensorData.ledPower = val;
96             if (val) {
97                 RedPWM_Start();
98                 GreenPWM_Start();
99                 BluePWM_Start();
100                 RedPWM_WriteCompare(sensorData.redPWMPct);
101                 GreenPWM_WriteCompare(sensorData.greenPWMPct);
102                 BluePWM_WriteCompare(sensorData.bluePWMPct);
103             } else {
104                 RedPWM_Stop();
105                 GreenPWM_Stop();
106                 BluePWM_Stop();
107             }
108 #ifdef DEBUG_ON
```

```
109     DEBUG_PutString("cmdSetPower ");
110     DEBUG_PutHexByte(sensorData.ledPower);
111     DEBUG_PutCRLF();
112 #endif
113             break;
114 
115         case cmdGetLumen :
116             i2cTxBuffer[1] = cmd;
117             i2cTxBuffer[2] = scaleLuxToFF(sensorData.
    lux);
118 #ifdef DEBUG_ON
119     DEBUG_PutString("cmdGetLumen ");
120     DEBUG_PutHexByte(i2cTxBuffer[2]);
121     DEBUG_PutCRLF();
122 #endif
123             break;
124         case cmdSetLumen :
125             sensorData.desiredLux = scaleFFtoLux(val);
126 #ifdef DEBUG_ON
127     DEBUG_PutString("cmdSetLumen ");
128     DEBUG_PutHexByte(sensorData.desiredLux);
129     DEBUG_PutCRLF();
130 #endif
131             break;
132 
133         case cmdGetDistance :
134             i2cTxBuffer[1] = cmd;
135             i2cTxBuffer[2] = (uint8)((sensorData.distance < 255) ?
    sensorData.distance : 255);
136 #ifdef DEBUG_ON
137     DEBUG_PutString("cmdGetDistance ");
138     DEBUG_PutHexByte(i2cTxBuffer[2]);
139     DEBUG_PutCRLF();
140 #endif
141             break;
142         case cmdSetDistance :
143             sensorData.desiredTimeDistance = val * 58;
144 #ifdef DEBUG_ON
145     DEBUG_PutString("cmdSetDistance ");
146     DEBUG_PutHexByte(sensorData.desiredTimeDistance);
147     DEBUG_PutCRLF();
148 #endif
149             break;
150         case cmdDistanceAlert :
151             i2cTxBuffer[1] = cmd;
152             i2cTxBuffer[2] = val;
153 #ifdef DEBUG_ON
154     DEBUG_PutString("cmdDistanceAlert ");
155     DEBUG_PutHexByte(val);
156     DEBUG_PutCRLF();
157 #endif
158             break;
159 
160         case cmdGetMovement :
161             // Return 0xff if there is movement, 0x00 if there is not
162             i2cTxBuffer[1] = cmd;
163             i2cTxBuffer[2] = (sensorData.movement) ? 0xff : 0x00;
164 #ifdef DEBUG_ON
165     DEBUG_PutString("cmdGetMovement ");
166     DEBUG_PutHexByte(sensorData.movement);
167     DEBUG_PutCRLF();
168 #endif
169             break;
170         case cmdSetMovement :
171             sensorData.movementAlertOn = val;
172 #ifdef DEBUG_ON
173     DEBUG_PutString("cmdSetMovement ");
174     DEBUG_PutHexByte(sensorData.movementAlertOn);
175     DEBUG_PutCRLF();
176 #endif
177             break;
178         case cmdMovementAlert :
179             i2cTxBuffer[1] = cmd;
180             i2cTxBuffer[2] = val;
181 #ifdef DEBUG_ON
182     DEBUG_PutString("cmdMovementAlert ");
183     DEBUG_PutHexByte(val);
184     DEBUG_PutCRLF();
185 #endif
186             break;
187 
188         default :
189             break;
190     }
191 }
```

Her er kald-grafen for denne funktion:

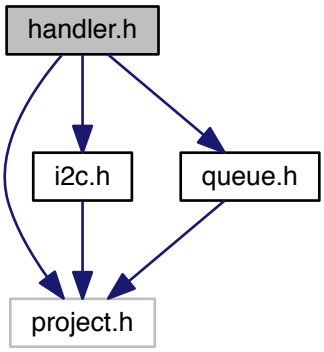


Her er kalder-grafen for denne funktion:

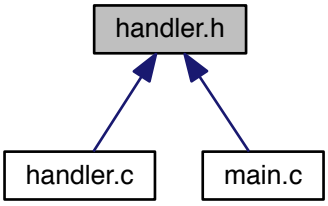
4.5 handler.h filreference

```
#include <project.h>
#include "i2c.h"
#include "queue.h"
```

Inklusions-afhængighedsgraf for handler.h:



Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



#Defines

- #define cmdSetRed (0x30u)
- #define cmdSetGreen (0x31u)
- #define cmdSetBlue (0x32u)
- #define cmdSetLumen (0x33u)
- #define cmdSetPower (0x34u)
- #define cmdGetRed (0x35u)
- #define cmdGetGreen (0x36u)
- #define cmdGetBlue (0x37u)
- #define cmdGetLumen (0x38u)
- #define cmdGetPower (0x39u)
- #define cmdSetDistance (0x40u)
- #define cmdSetMovement (0x41u)
- #define cmdGetDistance (0x42u)
- #define cmdGetMovement (0x43u)
- #define cmdDistanceAlert (0x44u)
- #define cmdMovementAlert (0x45u)

Funktioner

- void handler (uint8 cmd, uint8 val)

  Communication handler for the Sensor-PSoC.

4.5.1 #Define-dokumentation

4.5.1.1 #define cmdDistanceAlert (0x44u)

Defineret på linje 49 i filen handler.h.

Refereret til af handler() og main().

4.5.1.2 #define cmdGetBlue (0x37u)

Defineret på linje 42 i filen handler.h.

Refereret til af handler().

4.5.1.3 #define cmdGetDistance (0x42u)

Defineret på linje 47 i filen handler.h.

Refereret til af handler().

4.5.1.4 #define cmdGetGreen (0x36u)

Defineret på linje 41 i filen handler.h.

Refereret til af handler().

4.5.1.5 #define cmdGetLumen (0x38u)

Defineret på linje 43 i filen handler.h.

Refereret til af handler().

4.5.1.6 #define cmdGetMovement (0x43u)

Defineret på linje 48 i filen handler.h.

Refereret til af handler().

4.5.1.7 #define cmdGetPower (0x39u)

Defineret på linje 44 i filen handler.h.

Refereret til af handler().

4.5.1.8 #define cmdGetRed (0x35u)

Defineret på linje 40 i filen handler.h.

Refereret til af handler().

4.5.1.9 #define cmdMovementAlert (0x45u)

Defineret på linje 50 i filen handler.h.

Refereret til af handler() og main().

4.5.1.10 #define cmdSetBlue (0x32u)

Defineret på linje 37 i filen handler.h.

Refereret til af handler().

4.5.1.11 #define cmdSetDistance (0x40u)

Defineret på linje 45 i filen handler.h.

Refereret til af handler().

4.5.1.12 #define cmdSetGreen (0x31u)

Defineret på linje 36 i filen handler.h.

Refereret til af handler().

4.5.1.13 #define cmdSetLumen (0x33u)

Defineret på linje 38 i filen handler.h.

Refereret til af handler().

4.5.1.14 #define cmdSetMovement (0x41u)

Defineret på linje 46 i filen handler.h.

Refereret til af handler().

4.5.1.15 #define cmdSetPower (0x34u)

Defineret på linje 39 i filen handler.h.

Refereret til af handler().

4.5.1.16 #define cmdSetRed (0x30u)

Defineret på linje 35 i filen handler.h.

Refereret til af handler().

4.5.2 Funktions-dokumentation

4.5.2.1 void handler (uint8 *cmd,* uint8 *val*)

Communication handler for the Sensor-PSoC.

Parametre

in	*cmd*	The command to be executed
in	*val*	Optional command argument

Forfatter

Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 21 i filen handler.c.

Indeholder referencer til sensorDataT::bluePWMPct, cmdDistanceAlert, cmdGetBlue, cmdGetDistance, cmd↩GetGreen, cmdGetLumen, cmdGetMovement, cmdGetPower, cmdGetRed, cmdMovementAlert, cmdSetBlue,

cmdSetDistance, cmdSetGreen, cmdSetLumen, cmdSetMovement, cmdSetPower, cmdSetRed, sensorDataT::desiredLux, sensorDataT::desiredTimeDistance, sensorDataT::distance, sensorDataT::greenPWMPct, i2cTxBuffer, sensorDataT::ledPower, sensorDataT::lux, sensorDataT::movement, sensorDataT::movementAlertOn, sensorDataT::redPWMPct, scaleFFtoLux(), scaleLuxToFF() og sensorData.

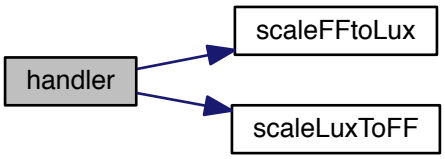
Refereret til af main().

```
22 {
23     switch (cmd) {
24         case cmdGetRed :
25             i2cTxBuffer[1] = cmd;
26             i2cTxBuffer[2] = sensorData.redPWMPct;
27 #ifdef DEBUG_ON
28     DEBUG_PutString("cmdGetRed ");
29     DEBUG_PutHexByte(sensorData.redPWMPct);
30     DEBUG_PutCRLF();
31 #endif
32             break;
33         case cmdSetRed :
34             sensorData.redPWMPct = val;
35             RedPWM_WriteCompare(sensorData.redPWMPct);
36             RedPWM_Start();
37             sensorData.ledPower = 1;
38 #ifdef DEBUG_ON
39     DEBUG_PutString("cmdSetRed ");
40     DEBUG_PutHexByte(sensorData.redPWMPct);
41     DEBUG_PutCRLF();
42 #endif
43             break;
44         case cmdGetGreen :
45             i2cTxBuffer[1] = cmd;
46             i2cTxBuffer[2] = sensorData.greenPWMPct;
47 #ifdef DEBUG_ON
48     DEBUG_PutString("cmdGetGreen ");
49     DEBUG_PutHexByte(sensorData.greenPWMPct);
50     DEBUG_PutCRLF();
51 #endif
52             break;
53         case cmdSetGreen :
54             sensorData.greenPWMPct = val;
55             GreenPWM_WriteCompare(sensorData.greenPWMPct);
56             GreenPWM_Start();
57             sensorData.ledPower = 1;
58 #ifdef DEBUG_ON
59     DEBUG_PutString("cmdSetGreen ");
60     DEBUG_PutHexByte(sensorData.greenPWMPct);
61     DEBUG_PutCRLF();
62 #endif
63             break;
64         case cmdGetBlue :
65             i2cTxBuffer[1] = cmd;
66             i2cTxBuffer[2] = sensorData.bluePWMPct;
67 #ifdef DEBUG_ON
68     DEBUG_PutString("cmdGetBlue ");
69     DEBUG_PutHexByte(sensorData.bluePWMPct);
70     DEBUG_PutCRLF();
71 #endif
72             break;
73         case cmdSetBlue :
74             sensorData.bluePWMPct = val;
75             BluePWM_WriteCompare(sensorData.bluePWMPct);
76             BluePWM_Start();
77             sensorData.ledPower = 1;
78 #ifdef DEBUG_ON
79     DEBUG_PutString("cmdSetBlue ");
80     DEBUG_PutHexByte(sensorData.bluePWMPct);
81     DEBUG_PutCRLF();
82 #endif
83             break;
84
85         case cmdGetPower :
86             i2cTxBuffer[1] = cmd;
87             i2cTxBuffer[2] = sensorData.ledPower;
88 #ifdef DEBUG_ON
89     DEBUG_PutString("cmdGetPower ");
90     DEBUG_PutHexByte(sensorData.ledPower);
91     DEBUG_PutCRLF();
92 #endif
93             break;
94         case cmdSetPower :
95             sensorData.ledPower = val;
96             if (val) {
97                 RedPWM_Start();
```

```
98                  GreenPWM_Start();
99                  BluePWM_Start();
100                  RedPWM_WriteCompare(sensorData.redPWMPct);
101                  GreenPWM_WriteCompare(sensorData.greenPWMPct);
102                  BluePWM_WriteCompare(sensorData.bluePWMPct);
103             } else {
104                 RedPWM_Stop();
105                 GreenPWM_Stop();
106                 BluePWM_Stop();
107             }
108 #ifdef DEBUG_ON
109     DEBUG_PutString("cmdSetPower ");
110     DEBUG_PutHexByte(sensorData.ledPower);
111     DEBUG_PutCRLF();
112 #endif
113             break;
114 
115         case cmdGetLumen :
116             i2cTxBuffer[1] = cmd;
117             i2cTxBuffer[2] = scaleLuxToFF(sensorData.
      lux);
118 #ifdef DEBUG_ON
119     DEBUG_PutString("cmdGetLumen ");
120     DEBUG_PutHexByte(i2cTxBuffer[2]);
121     DEBUG_PutCRLF();
122 #endif
123             break;
124         case cmdSetLumen :
125             sensorData.desiredLux = scaleFFtoLux(val);
126 #ifdef DEBUG_ON
127     DEBUG_PutString("cmdSetLumen ");
128     DEBUG_PutHexByte(sensorData.desiredLux);
129     DEBUG_PutCRLF();
130 #endif
131             break;
132 
133         case cmdGetDistance :
134             i2cTxBuffer[1] = cmd;
135             i2cTxBuffer[2] = (uint8)((sensorData.distance < 255) ?
      sensorData.distance : 255);
136 #ifdef DEBUG_ON
137     DEBUG_PutString("cmdGetDistance ");
138     DEBUG_PutHexByte(i2cTxBuffer[2]);
139     DEBUG_PutCRLF();
140 #endif
141             break;
142         case cmdSetDistance :
143             sensorData.desiredTimeDistance = val * 58;
144 #ifdef DEBUG_ON
145     DEBUG_PutString("cmdSetDistance ");
146     DEBUG_PutHexByte(sensorData.desiredTimeDistance);
147     DEBUG_PutCRLF();
148 #endif
149             break;
150         case cmdDistanceAlert :
151             i2cTxBuffer[1] = cmd;
152             i2cTxBuffer[2] = val;
153 #ifdef DEBUG_ON
154     DEBUG_PutString("cmdDistanceAlert ");
155     DEBUG_PutHexByte(val);
156     DEBUG_PutCRLF();
157 #endif
158             break;
159 
160         case cmdGetMovement :
161             // Return 0xff if there is movement, 0x00 if there is not
162             i2cTxBuffer[1] = cmd;
163             i2cTxBuffer[2] = (sensorData.movement) ? 0xff : 0x00;
164 #ifdef DEBUG_ON
165     DEBUG_PutString("cmdGetMovement ");
166     DEBUG_PutHexByte(sensorData.movement);
167     DEBUG_PutCRLF();
168 #endif
169             break;
170         case cmdSetMovement :
171             sensorData.movementAlertOn = val;
172 #ifdef DEBUG_ON
173     DEBUG_PutString("cmdSetMovement ");
174     DEBUG_PutHexByte(sensorData.movementAlertOn);
175     DEBUG_PutCRLF();
176 #endif
177             break;
178         case cmdMovementAlert :
179             i2cTxBuffer[1] = cmd;
180             i2cTxBuffer[2] = val;
181 #ifdef DEBUG_ON
182     DEBUG_PutString("cmdMovementAlert ");
```

```
183     DEBUG_PutHexByte(val);
184     DEBUG_PutCRLF();
185 #endif
186             break;
187 
188         default :
189             break;
190     }
191 }
```

Her er kald-grafen for denne funktion:

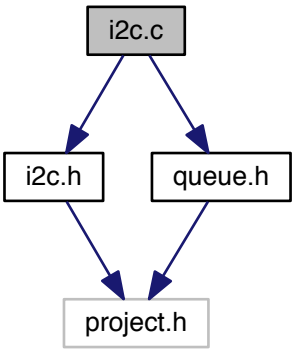


Her er kalder-grafen for denne funktion:

4.6 i2c.c filreference

```
#include "i2c.h"
#include "queue.h"
```

Inklusions-afhængighedsgraf for i2c.c:



Funktioner

- void i2c_init ()
- void i2c_rx ()
- void i2c_tx ()

Variable

- uint8 i2cTxBuffer [I2C_BUFFER_SIZE] = {I2C_PACKET_SOP, I2C_STS_CMD_FAIL, I2C_STS_CMD_FAIL, I2C_PACKET_EOP}
- uint8 i2cRxBuffer [I2C_BUFFER_SIZE]

4.6.1 Funktions-dokumentation

4.6.1.1 void i2c_init (void)

Defineret på linje 25 i filen i2c.c.

Indeholder referencer til I2C_BUFFER_SIZE, i2cRxBuffer og i2cTxBuffer.

Refereret til af main().

```
26 {
27     I2CS_I2CSlaveInitReadBuf(i2cTxBuffer, I2C_BUFFER_SIZE);
28     I2CS_I2CSlaveClearReadBuf();
29     I2CS_I2CSlaveClearReadStatus();
30 
31     I2CS_I2CSlaveInitWriteBuf(i2cRxBuffer, I2C_BUFFER_SIZE);
32     I2CS_I2CSlaveClearWriteBuf();
33     I2CS_I2CSlaveClearWriteStatus();
34 
35     I2CS_Start();
36 }
```

Her er kalder-grafen for denne funktion:

4.6.1.2 void i2c_rx (void)

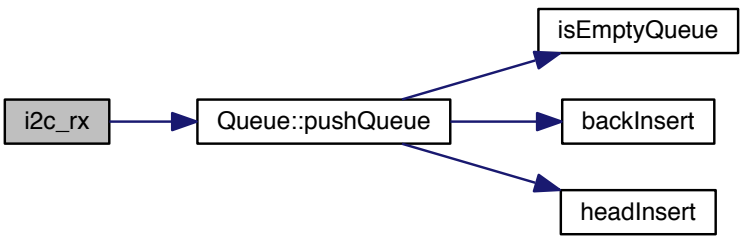
Defineret på linje 38 i filen i2c.c.

Indeholder referencer til Data::cmd_, I2C_BUFFER_SIZE, I2C_PACKET_CMD_POS, I2C_PACKET_EOP, I2C_PACKET_EOP_POS, I2C_PACKET_SOP, I2C_PACKET_SOP_POS, I2C_PACKET_VAL_POS, i2cRxBuffer, Queue::pushQueue() og Data::val_.

Refereret til af main().

```
39 {
40   if(0u != (I2CS_I2CSlaveStatus() & I2CS_I2C_SSTAT_WR_CMPLT))
41   {
42     if(I2C_BUFFER_SIZE == I2CS_I2CSlaveGetWriteBufSize())
43     {
44       if((i2cRxBuffer[I2C_PACKET_SOP_POS] ==
      I2C_PACKET_SOP) && (i2cRxBuffer[I2C_PACKET_EOP_POS] ==
      I2C_PACKET_EOP))
45       {
46         struct Data action;
47
48         action.cmd_ = i2cRxBuffer[I2C_PACKET_CMD_POS];
49         action.val_ = i2cRxBuffer[I2C_PACKET_VAL_POS];
50
51         pushQueue(action);
52       }
53       else
54       {
55       }
56     }
57     I2CS_I2CSlaveClearWriteBuf();
58     (void) I2CS_I2CSlaveClearWriteStatus();
59   }
60 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:

4.6.1.3 void i2c_tx (void)

Defineret på linje 62 i filen i2c.c.

Refereret til af main().

```
63 {
64     if(0u != (I2CS_I2CSlaveStatus() & I2CS_I2C_SSTAT_RD_CMPLT))
65     {
66         I2CS_I2CSlaveClearReadBuf();
67         (void) I2CS_I2CSlaveClearReadStatus();
68     }
69 }
```

Her er kalder-grafen for denne funktion:

4.6.2 Variabel-dokumentation

4.6.2.1 uint8 i2cRxBuffer[I2C_BUFFER_SIZE]

Defineret på linje 23 i filen i2c.c.

Refereret til af i2c_init() og i2c_rx().

4.6.2.2 uint8 i2cTxBuffer[I2C_BUFFER_SIZE] = {I2C_PACKET_SOP, I2C_STS_CMD_FAIL, I2C_STS_CMD_FAIL, I2C_PACKET_EOP}

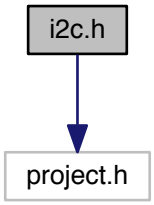
Defineret på linje 22 i filen i2c.c.

Refereret til af handler() og i2c_init().

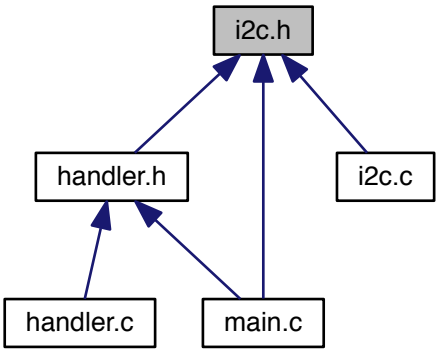
4.7 i2c.h filreference

```
#include <project.h>
```

Inklusions-afhængighedsgraf for i2c.h:



Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



#Defines

- #define I2C_BUFFER_SIZE (4u)
- #define I2C_PACKET_SIZE (I2C_BUFFER_SIZE)
- #define I2C_PACKET_SOP_POS (0u)
- #define I2C_PACKET_CMD_POS (1u)
- #define I2C_PACKET_VAL_POS (2u)
- #define I2C_PACKET_EOP_POS (3u)
- #define I2C_PACKET_SOP (0xBEu)
- #define I2C_PACKET_EOP (0xEFu)
- #define I2C_STS_CMD_DONE (0xAAu)
- #define I2C_STS_CMD_FAIL (0xEEu)

Funktioner

- void i2c_init (void)
- void i2c_rx (void)
- void i2c_tx (void)

Variable

- uint8 i2cTxBuffer [I2C_BUFFER_SIZE]
- uint8 i2cRxBuffer [I2C_BUFFER_SIZE]

4.7.1 #Define-dokumentation

4.7.1.1 #define I2C_BUFFER_SIZE (4u)

Defineret på linje 29 i filen i2c.h.

Refereret til af i2c_init() og i2c_rx().

4.7.1.2 #define I2C_PACKET_CMD_POS (1u)

Defineret på linje 34 i filen i2c.h.

Refereret til af i2c_rx().

4.7.1.3 #define I2C_PACKET_EOP (0xEFu)

Defineret på linje 40 i filen i2c.h.

Refereret til af i2c_rx().

4.7.1.4 #define I2C_PACKET_EOP_POS (3u)

Defineret på linje 36 i filen i2c.h.

Refereret til af i2c_rx().

4.7.1.5 #define I2C_PACKET_SIZE (I2C_BUFFER_SIZE)

Defineret på linje 30 i filen i2c.h.

4.7.1.6 #define I2C_PACKET_SOP (0xBEu)

Defineret på linje 39 i filen i2c.h.

Refereret til af i2c_rx().

4.7.1.7 #define I2C_PACKET_SOP_POS (0u)

Defineret på linje 33 i filen i2c.h.

Refereret til af i2c_rx().

4.7.1.8 #define I2C_PACKET_VAL_POS (2u)

Defineret på linje 35 i filen i2c.h.

Refereret til af i2c_rx().

4.7.1.9 #define I2C_STS_CMD_DONE (0xAAu)

Defineret på linje 43 i filen i2c.h.

4.7.1.10 #define I2C_STS_CMD_FAIL (0xEEu)

Defineret på linje 44 i filen i2c.h.

4.7.2 Funktions-dokumentation

4.7.2.1 void i2c_init (void)

Defineret på linje 25 i filen i2c.c.

Indeholder referencer til I2C_BUFFER_SIZE, i2cRxBuffer og i2cTxBuffer.

Refereret til af main().

```
26 {
27     I2CS_I2CSlaveInitReadBuf(i2cTxBuffer, I2C_BUFFER_SIZE);
28     I2CS_I2CSlaveClearReadBuf();
29     I2CS_I2CSlaveClearReadStatus();
30
31     I2CS_I2CSlaveInitWriteBuf(i2cRxBuffer, I2C_BUFFER_SIZE);
32     I2CS_I2CSlaveClearWriteBuf();
33     I2CS_I2CSlaveClearWriteStatus();
34
35     I2CS_Start();
36 }
```

Her er kalder-grafen for denne funktion:

4.7.2.2 void i2c_rx (void)

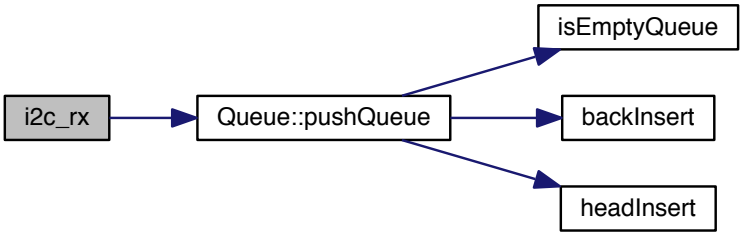
Defineret på linje 38 i filen i2c.c.

Indeholder referencer til Data::cmd_, I2C_BUFFER_SIZE, I2C_PACKET_CMD_POS, I2C_PACKET_EOP, I2C_PACKET_EOP_POS, I2C_PACKET_SOP, I2C_PACKET_SOP_POS, I2C_PACKET_VAL_POS, i2cRxBuffer, Queue::pushQueue() og Data::val_.

Refereret til af main().

```
39 {
40   if(0u != (I2CS_I2CSlaveStatus() & I2CS_I2C_SSTAT_WR_CMPLT))
41   {
42     if(I2C_BUFFER_SIZE == I2CS_I2CSlaveGetWriteBufSize())
43     {
44       if((i2cRxBuffer[I2C_PACKET_SOP_POS] ==
      I2C_PACKET_SOP) && (i2cRxBuffer[I2C_PACKET_EOP_POS] ==
      I2C_PACKET_EOP))
45       {
46         struct Data action;
47
48         action.cmd_ = i2cRxBuffer[I2C_PACKET_CMD_POS];
49         action.val_ = i2cRxBuffer[I2C_PACKET_VAL_POS];
50
51         pushQueue(action);
52       }
53       else
54       {
55       }
56     }
57     I2CS_I2CSlaveClearWriteBuf();
58     (void) I2CS_I2CSlaveClearWriteStatus();
59   }
60 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:

4.7.2.3 void i2c_tx (void)

Defineret på linje 62 i filen i2c.c.

Refereret til af main().

```
63 {
64     if(0u != (I2CS_I2CSlaveStatus() & I2CS_I2C_SSTAT_RD_CMPLT))
65     {
66         I2CS_I2CSlaveClearReadBuf();
67         (void) I2CS_I2CSlaveClearReadStatus();
68     }
69 }
```

Her er kalder-grafen for denne funktion:

4.7.3 Variabel-dokumentation

4.7.3.1 uint8 i2cRxBuffer[I2C_BUFFER_SIZE]

Defineret på linje 47 i filen i2c.h.

4.7.3.2 uint8 i2cTxBuffer[I2C_BUFFER_SIZE]

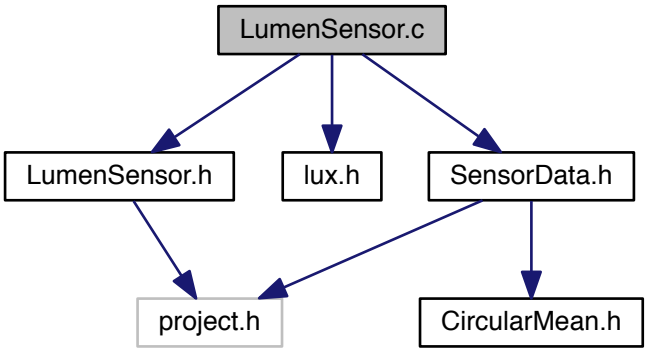
Defineret på linje 46 i filen i2c.h.

4.8 LumenSensor.c filreference

Handles the light sensor. Both initialization and data extraction.

```
#include "LumenSensor.h"
#include "lux.h"
#include "SensorData.h"
```

Inklusions-afhængighedsgraf for LumenSensor.c:



#Defines

- #define LUMEN_ADDR 0b0101001

Funktioner

- void initLumenSensor ()

  Initialize the light sensor.
- unsigned int readLumenSensor ()

  Read sensor data from the light sensor.
- uint8 scaleLuxToFF (unsigned int val)

  Scale a lux value into a 0-255 range value Due to the limited range to scale into, lux over 1530 is scaled to 255. The top 1530 vas chosen because 1530/255 = 6.
- unsigned int scaleFFtoLux (uint8 val)

  Scale a 0-255 range value into lux Due to the limited input range 1530 has been chosen as max value for lux. The top 1530 vas chosen because 1530/255 = 6.

4.8.1 Detaljeret beskrivelse

Handles the light sensor. Both initialization and data extraction.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

4.8.2 #Define-dokumentation

4.8.2.1 #define LUMEN_ADDR 0b0101001

The I2C address of the light sensor

Defineret på linje 11 i filen LumenSensor.c.

Refereret til af initLumenSensor() og readLumenSensor().

4.8.3 Funktions-dokumentation

4.8.3.1 void initLumenSensor ( )

Initialize the light sensor.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

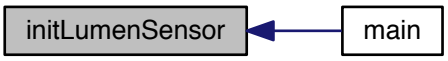
Defineret på linje 18 i filen LumenSensor.c.

Indeholder referencer til LUMEN_ADDR.

Refereret til af main().

```
19 {
20     uint8 writeBuf[2];
21     uint32 err;
22 
23     // Add internal pull-up resistors on the SCL and SDA lines
24     LumenCom_scl_SetDriveMode(LumenCom_scl_DM_RES_UP);
25     LumenCom_sda_SetDriveMode(LumenCom_sda_DM_RES_UP);
26 
27     LumenCom_Start();
28     // Send command to sensor:
29     writeBuf[0] = 0x80; // Highest bit set = command, low bits 0000 = control register
30     writeBuf[1] = 0x03; // Power up sensor
31     err = LumenCom_I2CMasterWriteBuf(LUMEN_ADDR, writeBuf, 2, LumenCom_I2C_MODE_COMPLETE_XFER);
32     // Wait for the I2C command to finish transferring:
33     while(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_INP) {}
34 }
```

Her er kalder-grafen for denne funktion:



4.8.3.2 unsigned int readLumenSensor ( )

Read sensor data from the light sensor.

Returnerer

The lux value of the perceived light

Forfatter

Simon Nejmann (19981127@uni.au.dk)

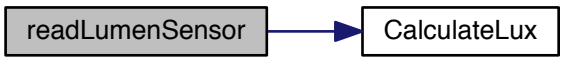
Defineret på linje 42 i filen LumenSensor.c.

Indeholder referencer til CalculateLux() og LUMEN_ADDR.

Refereret til af main().

```
43 {
44     uint8 readBuf[2];
45     uint8 writeBuf[1];
46     unsigned int channel0 = 0;
47     unsigned int channel1 = 0;
48 
49     // Send "read channel 0" command
50     writeBuf[0] = 0xAC;
51     LumenCom_I2CMasterWriteBuf(LUMEN_ADDR, writeBuf, 1, LumenCom_I2C_MODE_NO_STOP);
52 
53     // Wait for the I2C command to finish transferring
54     while(!(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_HALT)) {}
55     // Read two bytes of data
56     LumenCom_I2CMasterReadBuf(LUMEN_ADDR, readBuf, 2,
57         LumenCom_I2C_MODE_REPEAT_START | LumenCom_I2C_MODE_NO_STOP);
58 
59     // Wait for the I2C command to finish transferring
60     while(!(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_HALT)) {}
61 
62     channel0 = readBuf[0] + readBuf[1] * 256;
63 
64     // Send "read channel 1" command
65     writeBuf[0] = 0xAE;
66     LumenCom_I2CMasterWriteBuf(LUMEN_ADDR, writeBuf, 1,
67         LumenCom_I2C_MODE_REPEAT_START | LumenCom_I2C_MODE_NO_STOP);
68 
69     // Wait for the I2C command to finish transferring
70     while(!(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_HALT)) {}
71     // Read two bytes of data
72     LumenCom_I2CMasterReadBuf(LUMEN_ADDR, readBuf, 2, LumenCom_I2C_MODE_REPEAT_START);
73 
74     // Wait for the I2C command to finish transferring
75     while(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_INP) {}
76 
77     channel1 = readBuf[0] + readBuf[1] * 256;
78 
79     // Use Datasheet function to calculate lux value from read values
80     return CalculateLux(0, 2, channel0, channel1, 0);
81 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:

4.8.3.3 unsigned int scaleFFtoLux (uint8 *val*)

Scale a 0-255 range value into lux Due to the limited input range 1530 has been chosen as max value for lux. The top 1530 vas chosen because 1530/255 = 6.

Returnerer

The an upscaled lux value

Parametre

in	*val*	uint8 containing a value to scale up

Forfatter

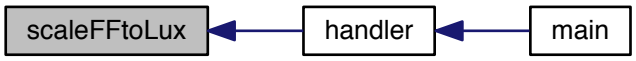
Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 110 i filen LumenSensor.c.

Refereret til af handler().

```
111 {
112 //    return  (val * 1530) / 255;
113     return  val * 6;
114 }
```

Her er kalder-grafen for denne funktion:



4.8.3.4 uint8 scaleLuxToFF (unsigned int *val*)

Scale a lux value into a 0-255 range value Due to the limited range to scale into, lux over 1530 is scaled to 255. The top 1530 vas chosen because 1530/255 = 6.

Returnerer

The lux value scaled down into a single uint8

Parametre

in	*val*	The lux value to scale down

Forfatter

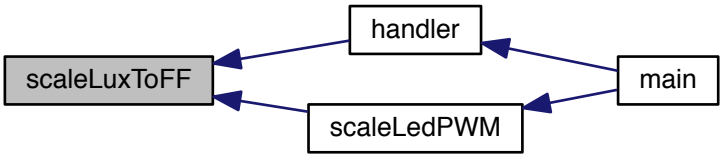
Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 93 i filen LumenSensor.c.

Refereret til af handler() og scaleLedPWM().

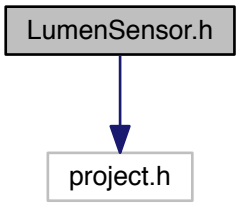
```
94 {
95 //    int tmp = (val * 255) / 1530;
96     int tmp = val / 6;
97     return (uint8)((tmp < 255) ? tmp : 255);
98 }
```

Her er kalder-grafen for denne funktion:

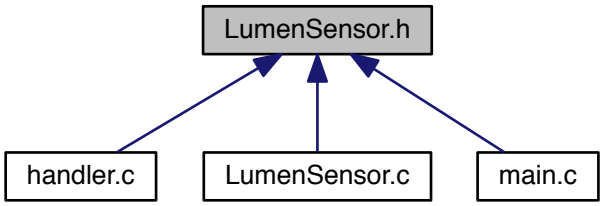


4.9 LumenSensor.h filreference

`#include <project.h>`
Inklusions-afhængighedsgraf for LumenSensor.h:



Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



Funktioner

- void initLumenSensor ()

  Initialize the light sensor.
- unsigned int readLumenSensor ()

  Read sensor data from the light sensor.
- uint8 scaleLuxToFF (unsigned int val)

  Scale a lux value into a 0-255 range value Due to the limited range to scale into, lux over 1530 is scaled to 255. The top 1530 vas chosen because 1530/255 = 6.
- unsigned int scaleFFtoLux (uint8 val)

  Scale a 0-255 range value into lux Due to the limited input range 1530 has been chosen as max value for lux. The top 1530 vas chosen because 1530/255 = 6.

4.9.1 Funktions-dokumentation

4.9.1.1 void initLumenSensor ( )

Initialize the light sensor.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

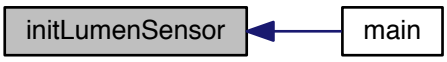
Defineret på linje 18 i filen LumenSensor.c.

Indeholder referencer til LUMEN_ADDR.

Refereret til af main().

```
19 {
20     uint8 writeBuf[2];
21     uint32 err;
22 
23     // Add internal pull-up resistors on the SCL and SDA lines
24     LumenCom_scl_SetDriveMode(LumenCom_scl_DM_RES_UP);
25     LumenCom_sda_SetDriveMode(LumenCom_sda_DM_RES_UP);
26 
27     LumenCom_Start();
28     // Send command to sensor:
29     writeBuf[0] = 0x80; // Highest bit set = command, low bits 0000 = control register
30     writeBuf[1] = 0x03; // Power up sensor
31     err = LumenCom_I2CMasterWriteBuf(LUMEN_ADDR, writeBuf, 2, LumenCom_I2C_MODE_COMPLETE_XFER);
32     // Wait for the I2C command to finish transferring:
33     while(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_INP) {}
34 }
```

Her er kalder-grafen for denne funktion:



4.9.1.2 unsigned int readLumenSensor ( )

Read sensor data from the light sensor.

Returnerer

The lux value of the perceived light

Forfatter

Simon Nejmann (19981127@uni.au.dk)

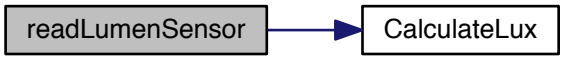
Defineret på linje 42 i filen LumenSensor.c.

Indeholder referencer til CalculateLux() og LUMEN_ADDR.

Refereret til af main().

```
43 {
44     uint8 readBuf[2];
45     uint8 writeBuf[1];
46     unsigned int channel0 = 0;
47     unsigned int channel1 = 0;
48 
49     // Send "read channel 0" command
50     writeBuf[0] = 0xAC;
51     LumenCom_I2CMasterWriteBuf(LUMEN_ADDR, writeBuf, 1, LumenCom_I2C_MODE_NO_STOP);
52 
53     // Wait for the I2C command to finish transferring
54     while(!(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_HALT)) {}
55     // Read two bytes of data
56     LumenCom_I2CMasterReadBuf(LUMEN_ADDR, readBuf, 2,
57         LumenCom_I2C_MODE_REPEAT_START | LumenCom_I2C_MODE_NO_STOP);
58 
59     // Wait for the I2C command to finish transferring
60     while(!(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_HALT)) {}
61 
62     channel0 = readBuf[0] + readBuf[1] * 256;
63 
64     // Send "read channel 1" command
65     writeBuf[0] = 0xAE;
66     LumenCom_I2CMasterWriteBuf(LUMEN_ADDR, writeBuf, 1,
67         LumenCom_I2C_MODE_REPEAT_START | LumenCom_I2C_MODE_NO_STOP);
68 
69     // Wait for the I2C command to finish transferring
70     while(!(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_HALT)) {}
71     // Read two bytes of data
72     LumenCom_I2CMasterReadBuf(LUMEN_ADDR, readBuf, 2, LumenCom_I2C_MODE_REPEAT_START);
73 
74     // Wait for the I2C command to finish transferring
75     while(LumenCom_I2CMasterStatus() & LumenCom_I2C_MSTAT_XFER_INP) {}
76 
77     channel1 = readBuf[0] + readBuf[1] * 256;
78 
79     // Use Datasheet function to calculate lux value from read values
80     return CalculateLux(0, 2, channel0, channel1, 0);
81 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:

4.9.1.3 unsigned int scaleFFtoLux (uint8 *val*)

Scale a 0-255 range value into lux Due to the limited input range 1530 has been chosen as max value for lux. The top 1530 vas chosen because 1530/255 = 6.

Returnerer

The an upscaled lux value

Parametre

in	*val*	uint8 containing a value to scale up

Forfatter

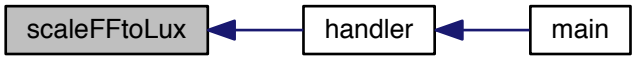
Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 110 i filen LumenSensor.c.

Refereret til af handler().

```
111 {
112 //    return  (val * 1530) / 255;
113     return  val * 6;
114 }
```

Her er kalder-grafen for denne funktion:



4.9.1.4 uint8 scaleLuxToFF (unsigned int *val*)

Scale a lux value into a 0-255 range value Due to the limited range to scale into, lux over 1530 is scaled to 255. The top 1530 vas chosen because 1530/255 = 6.

Returnerer

The lux value scaled down into a single uint8

Parametre

in	*val*	The lux value to scale down

Forfatter

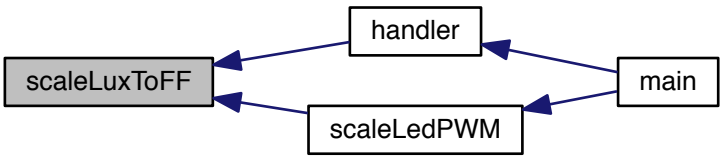
Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 93 i filen LumenSensor.c.

Refereret til af handler() og scaleLedPWM().

```
94 {
95 //    int tmp = (val * 255) / 1530;
96     int tmp = val / 6;
97     return (uint8)((tmp < 255) ? tmp : 255);
98 }
```

Her er kalder-grafen for denne funktion:



4.10 lux.c filreference

Code copied from the "TSL2561 Light-to-digital converter" datasheet.

#Defines

- #define LUX_SCALE 14
- #define RATIO_SCALE 9
- #define CH_SCALE 10
- #define CHSCALE_TINT0 0x7517
- #define CHSCALE_TINT1 0x0fe7
- #define K1T 0x0040
- #define B1T 0x01f2
- #define M1T 0x01be
- #define K2T 0x0080
- #define B2T 0x0214
- #define M2T 0x02d1
- #define K3T 0x00c0
- #define B3T 0x023f
- #define M3T 0x037b
- #define K4T 0x0100
- #define B4T 0x0270
- #define M4T 0x03fe
- #define K5T 0x0138
- #define B5T 0x016f
- #define M5T 0x01fc
- #define K6T 0x019a
- #define B6T 0x00d2
- #define M6T 0x00fb
- #define K7T 0x029a
- #define B7T 0x0018
- #define M7T 0x0012
- #define K8T 0x029a
- #define B8T 0x0000
- #define M8T 0x0000
- #define K1C 0x0043
- #define B1C 0x0204
- #define M1C 0x01ad
- #define K2C 0x0085
- #define B2C 0x0228
- #define M2C 0x02c1
- #define K3C 0x00c8
- #define B3C 0x0253
- #define M3C 0x0363
- #define K4C 0x010a
- #define B4C 0x0282
- #define M4C 0x03df
- #define K5C 0x014d
- #define B5C 0x0177
- #define M5C 0x01dd
- #define K6C 0x019a
- #define B6C 0x0101
- #define M6C 0x0127

- #define K7C 0x029a
- #define B7C 0x0037
- #define M7C 0x002b
- #define K8C 0x029a
- #define B8C 0x0000
- #define M8C 0x0000

Funktioner

- unsigned int CalculateLux (unsigned int iGain, unsigned int tInt, unsigned int ch0, unsigned int ch1, int iType)

  Calculate perceived lux from sensor values The sensor has two detectors, one that detects visible and infrared light while the second only detects infrared light. The amount of visible light is then calculated by substracting a scaling fraction of the second detector value from a scaling fraction of the first.

4.10.1 Detaljeret beskrivelse

Code copied from the "TSL2561 Light-to-digital converter" datasheet.

Forfatter

TAOS, Inc.

4.10.2 #Define-dokumentation

4.10.2.1 #define B1C 0x0204

Defineret på linje 106 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.2 #define B1T 0x01f2

Defineret på linje 59 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.3 #define B2C 0x0228

Defineret på linje 109 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.4 #define B2T 0x0214

Defineret på linje 62 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.5 #define B3C 0x0253

Defineret på linje 112 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.6 #define B3T 0x023f

Defineret på linje 65 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.7 #define B4C 0x0282

Defineret på linje 115 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.8 #define B4T 0x0270

Defineret på linje 68 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.9 #define B5C 0x0177

Defineret på linje 118 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.10 #define B5T 0x016f

Defineret på linje 71 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.11 #define B6C 0x0101

Defineret på linje 121 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.12 #define B6T 0x00d2

Defineret på linje 74 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.13 #define B7C 0x0037

Defineret på linje 124 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.14 #define B7T 0x0018

Defineret på linje 77 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.15 #define B8C 0x0000

Defineret på linje 127 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.16 #define B8T 0x0000

Defineret på linje 80 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.17 #define CH_SCALE 10

Defineret på linje 25 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.18 #define CHSCALE_TINT0 0x7517

Defineret på linje 26 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.19 #define CHSCALE_TINT1 0x0fe7

Defineret på linje 27 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.20 #define K1C 0x0043

Defineret på linje 105 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.21 #define K1T 0x0040

Defineret på linje 58 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.22 #define K2C 0x0085

Defineret på linje 108 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.23 #define K2T 0x0080

Defineret på linje 61 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.24 #define K3C 0x00c8

Defineret på linje 111 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.25 #define K3T 0x00c0

Defineret på linje 64 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.26 #define K4C 0x010a

Defineret på linje 114 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.27 #define K4T 0x0100

Defineret på linje 67 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.28 #define K5C 0x014d

Defineret på linje 117 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.29 #define K5T 0x0138

Defineret på linje 70 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.30 #define K6C 0x019a

Defineret på linje 120 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.31 #define K6T 0x019a

Defineret på linje 73 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.32 #define K7C 0x029a

Defineret på linje 123 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.33 #define K7T 0x029a

Defineret på linje 76 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.34 #define K8C 0x029a

Defineret på linje 126 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.35 #define K8T 0x029a

Defineret på linje 79 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.36 #define LUX_SCALE 14

Defineret på linje 20 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.37 #define M1C 0x01ad

Defineret på linje 107 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.38 #define M1T 0x01be

Defineret på linje 60 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.39 #define M2C 0x02c1

Defineret på linje 110 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.40 #define M2T 0x02d1

Defineret på linje 63 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.41 #define M3C 0x0363

Defineret på linje 113 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.42 #define M3T 0x037b

Defineret på linje 66 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.43 #define M4C 0x03df

Defineret på linje 116 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.44 #define M4T 0x03fe

Defineret på linje 69 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.45 #define M5C 0x01dd

Defineret på linje 119 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.46 #define M5T 0x01fc

Defineret på linje 72 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.47 #define M6C 0x0127

Defineret på linje 122 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.48 #define M6T 0x00fb

Defineret på linje 75 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.49 #define M7C 0x002b

Defineret på linje 125 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.50 #define M7T 0x0012

Defineret på linje 78 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.51 #define M8C 0x0000

Defineret på linje 128 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.52 #define M8T 0x0000

Defineret på linje 81 i filen lux.c.

Refereret til af CalculateLux().

4.10.2.53 #define RATIO_SCALE 9

Defineret på linje 21 i filen lux.c.

Refereret til af CalculateLux().

4.10.3 Funktions-dokumentation

4.10.3.1 unsigned int CalculateLux (unsigned int *iGain,* unsigned int *tInt,* unsigned int *ch0,* unsigned int *ch1,* int *iType*)

Calculate perceived lux from sensor values The sensor has two detectors, one that detects visible and infrared light while the second only detects infrared light. The amount of visible light is then calculated by substracting a scaling fraction of the second detector value from a scaling fraction of the first.

Returnerer

The calculated lux value

Parametre

in	*iGain*	Is internal scaling enabled in the sensor (0 = x1, 1 = x16)
in	*tInt*	Integration time set in the sensor (0 = 13.7ms, 1 = 101ms, 2 = 402ms)
in	*ch0*	Value read from detector 0
in	*ch1*	Value read from detector 1
in	*iType*	Physical sensor package (0 = T, FN or CL, 1 = CS)

Forfatter

TAOS, Inc.

Defineret på linje 147 i filen lux.c.

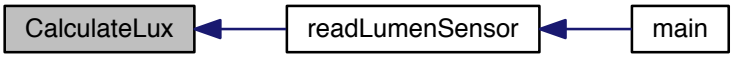
Indeholder referencer til B1C, B1T, B2C, B2T, B3C, B3T, B4C, B4T, B5C, B5T, B6C, B6T, B7C, B7T, B8C, B8T, CH_SCALE, CHSCALE_TINT0, CHSCALE_TINT1, K1C, K1T, K2C, K2T, K3C, K3T, K4C, K4T, K5C, K5T, K6C, K6T, K7C, K7T, K8C, K8T, LUX_SCALE, M1C, M1T, M2C, M2T, M3C, M3T, M4C, M4T, M5C, M5T, M6C, M6T, M7C, M7T, M8C, M8T og RATIO_SCALE.

Refereret til af readLumenSensor().

```
149 {
150   //
151   // first, scale the channel values depending on the gain and integration time
152   // 16X, 402mS is nominal.
153   // scale if integration time is NOT 402 msec
154   unsigned long chScale;
155   unsigned long channel1;
156   unsigned long channel0;
157   switch (tInt)
158   {
159     case 0: // 13.7 msec
160       chScale = CHSCALE_TINT0;
161       break;
162     case 1: // 101 msec
163       chScale = CHSCALE_TINT1;
164       break;
165     default: // assume no scaling
166       chScale = (1 << CH_SCALE);
167       break;
168   }
169   // scale if gain is NOT 16X
170   if (!iGain) chScale = chScale << 4; // scale 1X to 16X
171   // scale the channel values
172   channel0 = (ch0 * chScale) >> CH_SCALE;
173   channel1 = (ch1 * chScale) >> CH_SCALE;
174   //
175   // find the ratio of the channel values (Channel1/Channel0)
176   // protect against divide by zero
177   unsigned long ratio1 = 0;
178   if (channel0 != 0) ratio1 = (channel1 << (RATIO_SCALE+1)) / channel0;
179   // round the ratio value
180   unsigned long ratio = (ratio1 + 1) >> 1;
181   // is ratio <= eachBreak ?
182   unsigned int b = 0, m = 0;
183   switch (iType)
184   {
185     case 0: // T, FN and CL package
186       if ((ratio >= 0) && (ratio <= K1T))
187       {b=B1T; m=M1T;}
188       else if (ratio <= K2T)
189       {b=B2T; m=M2T;}
190       else if (ratio <= K3T)
191       {b=B3T; m=M3T;}
192       else if (ratio <= K4T)
193       {b=B4T; m=M4T;}
194       else if (ratio <= K5T)
195       {b=B5T; m=M5T;}
196       else if (ratio <= K6T)
197       {b=B6T; m=M6T;}
198       else if (ratio <= K7T)
199       {b=B7T; m=M7T;}
200       else if (ratio > K8T)
201       {b=B8T; m=M8T;}
202       break;
203     case 1:// CS package
204       if ((ratio >= 0) && (ratio <= K1C))
205       {b=B1C; m=M1C;}
206       else if (ratio <= K2C)
207       {b=B2C; m=M2C;}
208       else if (ratio <= K3C)
209       {b=B3C; m=M3C;}
210       else if (ratio <= K4C)
211       {b=B4C; m=M4C;}
212       else if (ratio <= K5C)
213       {b=B5C; m=M5C;}
214       else if (ratio <= K6C)
215       {b=B6C; m=M6C;}
216       else if (ratio <= K7C)
217       {b=B7C; m=M7C;}
218       else if (ratio > K8C)
219       {b=B8C; m=M8C;}
220       break;
221   }
222   unsigned long temp;
223   temp = ((channel0 * b) - (channel1 * m));
224   // do not allow negative lux value
```

```
225   if (temp < 0) temp = 0;
226   // round lsb (2^(LUX_SCALE1))
227   temp += (1 << (LUX_SCALE-1));
228   // strip off fractional portion
229   unsigned long lux = temp >> LUX_SCALE;
230   return(lux);
231 }
```

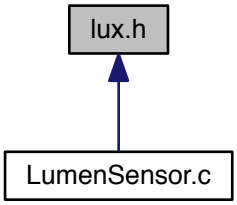
Her er kalder-grafen for denne funktion:



4.11 lux.h filreference

Code copied from the "TSL2561 Light-to-digital converter" datasheet.

Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



Funktioner

- unsigned int CalculateLux (unsigned int iGain, unsigned int tInt, unsigned int ch0, unsigned int ch1, int iType)

  Calculate perceived lux from sensor values The sensor has two detectors, one that detects visible and infrared light while the second only detects infrared light. The amount of visible light is then calculated by substracting a scaling fraction of the second detector value from a scaling fraction of the first.

4.11.1 Detaljeret beskrivelse

Code copied from the "TSL2561 Light-to-digital converter" datasheet.

Forfatter

TAOS, Inc.

4.11.2 Funktions-dokumentation

4.11.2.1 unsigned int CalculateLux (unsigned int *iGain,* unsigned int *tInt,* unsigned int *ch0,* unsigned int *ch1,* int *iType*)

Calculate perceived lux from sensor values The sensor has two detectors, one that detects visible and infrared light while the second only detects infrared light. The amount of visible light is then calculated by substracting a scaling fraction of the second detector value from a scaling fraction of the first.

Returnerer

The calculated lux value

Parametre

in	*iGain*	Is internal scaling enabled in the sensor (0 = x1, 1 = x16)
in	*tInt*	Integration time set in the sensor (0 = 13.7ms, 1 = 101ms, 2 = 402ms)
in	*ch0*	Value read from detector 0
in	*ch1*	Value read from detector 1
in	*iType*	Physical sensor package (0 = T, FN or CL, 1 = CS)

Forfatter

TAOS, Inc.

Defineret på linje 147 i filen lux.c.

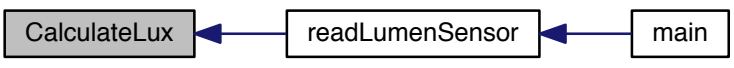
Indeholder referencer til B1C, B1T, B2C, B2T, B3C, B3T, B4C, B4T, B5C, B5T, B6C, B6T, B7C, B7T, B8C, B8T, CH_SCALE, CHSCALE_TINT0, CHSCALE_TINT1, K1C, K1T, K2C, K2T, K3C, K3T, K4C, K4T, K5C, K5T, K6C, K6T, K7C, K7T, K8C, K8T, LUX_SCALE, M1C, M1T, M2C, M2T, M3C, M3T, M4C, M4T, M5C, M5T, M6C, M6T, M7C, M7T, M8C, M8T og RATIO_SCALE.

Refereret til af readLumenSensor().

```
149 {
150   //
151   // first, scale the channel values depending on the gain and integration time
152   // 16X, 402mS is nominal.
153   // scale if integration time is NOT 402 msec
154   unsigned long chScale;
155   unsigned long channel1;
156   unsigned long channel0;
157   switch (tInt)
158   {
159     case 0: // 13.7 msec
160       chScale = CHSCALE_TINT0;
161       break;
162     case 1: // 101 msec
163       chScale = CHSCALE_TINT1;
164       break;
165     default: // assume no scaling
166       chScale = (1 << CH_SCALE);
167       break;
168   }
169   // scale if gain is NOT 16X
170   if (!iGain) chScale = chScale << 4; // scale 1X to 16X
171   // scale the channel values
172   channel0 = (ch0 * chScale) >> CH_SCALE;
173   channel1 = (ch1 * chScale) >> CH_SCALE;
174   //
175   // find the ratio of the channel values (Channel1/Channel0)
176   // protect against divide by zero
177   unsigned long ratio1 = 0;
178   if (channel0 != 0) ratio1 = (channel1 << (RATIO_SCALE+1)) / channel0;
```

```
179   // round the ratio value
180   unsigned long ratio = (ratio1 + 1) >> 1;
181   // is ratio <= eachBreak ?
182   unsigned int b = 0, m = 0;
183   switch (iType)
184   {
185     case 0: // T, FN and CL package
186       if ((ratio >= 0) && (ratio <= K1T))
187       {b=B1T; m=M1T;}
188       else if (ratio <= K2T)
189       {b=B2T; m=M2T;}
190       else if (ratio <= K3T)
191       {b=B3T; m=M3T;}
192       else if (ratio <= K4T)
193       {b=B4T; m=M4T;}
194       else if (ratio <= K5T)
195       {b=B5T; m=M5T;}
196       else if (ratio <= K6T)
197       {b=B6T; m=M6T;}
198       else if (ratio <= K7T)
199       {b=B7T; m=M7T;}
200       else if (ratio > K8T)
201       {b=B8T; m=M8T;}
202       break;
203     case 1:// CS package
204       if ((ratio >= 0) && (ratio <= K1C))
205       {b=B1C; m=M1C;}
206       else if (ratio <= K2C)
207       {b=B2C; m=M2C;}
208       else if (ratio <= K3C)
209       {b=B3C; m=M3C;}
210       else if (ratio <= K4C)
211       {b=B4C; m=M4C;}
212       else if (ratio <= K5C)
213       {b=B5C; m=M5C;}
214       else if (ratio <= K6C)
215       {b=B6C; m=M6C;}
216       else if (ratio <= K7C)
217       {b=B7C; m=M7C;}
218       else if (ratio > K8C)
219       {b=B8C; m=M8C;}
220       break;
221   }
222   unsigned long temp;
223   temp = ((channel0 * b) - (channel1 * m));
224   // do not allow negative lux value
225   if (temp < 0) temp = 0;
226   // round lsb (2^(LUX_SCALE1))
227   temp += (1 << (LUX_SCALE-1));
228   // strip off fractional portion
229   unsigned long lux = temp >> LUX_SCALE;
230   return(lux);
231 }
```

Her er kalder-grafen for denne funktion:

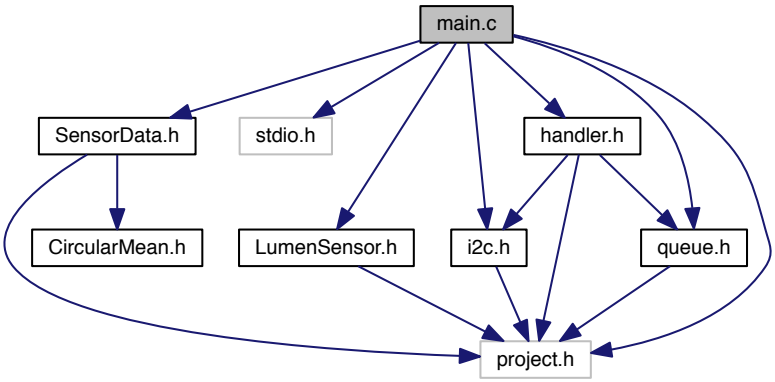


4.12 main.c filreference

Main file for Sensor-PSoC.

```
#include <project.h>
#include <stdio.h>
#include "SensorData.h"
#include "LumenSensor.h"
#include "handler.h"
#include "i2c.h"
#include "queue.h"
```

Inklusions-afhængighedsgraf for main.c:



#Defines

- #define DEBUG_ON

Enumerationer

- enum sensor {
  DIST, LUMEN, PIR, DIST_ALERT,
  MOVE_ALERT }
- enum ctrl { COUNT, RATE, FLAG }

Funktioner

- void scaleLedPWM ()

  Should control LED output according to measured and desired lux Does not currently work: LEDs cannot produce enough light to affect the measured lux in a meaningful way. Should have implemented a primitive PID controller (so far only P-part has been attempted implemented).
- void initCtrlFlags ()
- void incrCtrlFlag (enum sensor se)
- CY_ISR (Metronome_Interrupt)
- CY_ISR (DistTimer_Interrupt)
- int main ()

  Main function.

Variable

- char bla [150]
- char controlFlags [5][3]
- uint8 distAlertTriggered = 0

4.12.1 Detaljeret beskrivelse

Main file for Sensor-PSoC.

Contains the main loop and control structure for the Sensor-PSoC. The main control is the Metronome timer which generates an interrupt every half second. This interrupt then increases a set of counters which, in turn, set flags when the counters overflow. This allows us to define periodic events that trigger on integer multiple of half seconds.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

4.12.2 #Define-dokumentation

4.12.2.1 #define DEBUG_ON

Debug define. Comment out to suppress debug prints

Defineret på linje 22 i filen main.c.

4.12.3 Dokumentation af enumerations-typer

4.12.3.1 enum ctrl

Flag names

Enumerationsværdier

COUNT

RATE

FLAG

Defineret på linje 31 i filen main.c.

```
31 {COUNT, RATE, FLAG};
```

4.12.3.2 enum sensor

List of event names

Enumerationsværdier

DIST

LUMEN

PIR

DIST_ALERT

MOVE_ALERT

Defineret på linje 29 i filen main.c.

```
29 {DIST, LUMEN, PIR, DIST_ALERT, MOVE_ALERT};
```

4.12.4 Funktions-dokumentation

4.12.4.1 CY_ISR (Metronome_Interrupt)

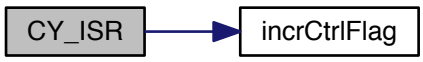
Metronome timer interrupt handler

Defineret på linje 58 i filen main.c.

Indeholder referencer til DIST, incrCtrlFlag(), LUMEN, MOVE_ALERT og PIR.

```
59 {
60     // Clear interrupt
61     MetronomeTimer_ReadStatusRegister();
62 
63     incrCtrlFlag(DIST);
64     incrCtrlFlag(LUMEN);
65     incrCtrlFlag(PIR);
66     // Not DIST_ALERT
67     incrCtrlFlag(MOVE_ALERT);
68 
69 #ifdef DEBUG_ON
70 //    DEBUG_PutString("Beat\r\n");
71 #endif
72 }
```

Her er kald-grafen for denne funktion:



4.12.4.2 CY_ISR (DistTimer_Interrupt)

Timer interrupt for ultrasonic distance sensor

Defineret på linje 78 i filen main.c.

Indeholder referencer til controlFlags, COUNT, sensorDataT::desiredTimeDistance, DIST_ALERT, distAlertTriggered, sensorDataT::distance, FLAG og sensorData.

```
79 {
80     //  Calculate distance from delay. Basic formula: cm = micro-seconds / 58
81     //      Timer counts down from 2^16, so start at 2^16 and subtract timer.
82     int echoDelay = (1<<16) - DistTimer_ReadCapture();
83 
84     //  Integer calculations means everything gets rounded down, so add
85     //      half a cm (58/2) to make 3.5 round up instead.
86     sensorData.distance = (echoDelay + 58/2) / 58;
87 
88     // Check if we are too close - check done in time domain due to greater resolution
89     if (echoDelay < sensorData.desiredTimeDistance) {
90         controlFlags[DIST_ALERT][FLAG] = 1;
91     } else {
92         distAlertTriggered = 0;
93         DistInterruptPin_Write(0);
94         controlFlags[DIST_ALERT][COUNT] = 0;
95     }
96 
97     // Reset the timer
98     DistReset_Write(1);
99 
100 #ifdef DEBUG_ON
101 //    sprintf(bla, "Dist: %i\n\r", sensorData.distance);
102 //    DEBUG_PutString(bla);
103 #endif
104 }
```

4.12.4.3 void incrCtrlFlag (enum sensor *se*)

Helper function: Increase count, check for overflow, and raise flag if needed

Defineret på linje 50 i filen main.c.

Indeholder referencer til controlFlags, COUNT, FLAG og RATE.

Refereret til af CY_ISR().

```
51 {
52     controlFlags[se][COUNT] = (controlFlags[se][
      COUNT] + 1) % controlFlags[se][RATE];
53     if (controlFlags[se][COUNT] == 0) {
54         controlFlags[se][FLAG] = 1;
55     }
56 }
```

Her er kalder-grafen for denne funktion:

4.12.4.4 void initCtrlFlags ( )

Initializer for control structure

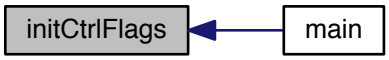
Defineret på linje 41 i filen main.c.

Indeholder referencer til controlFlags, DIST, LUMEN, MOVE_ALERT, PIR og RATE.

Refereret til af main().

```
42 {
43     controlFlags[DIST][RATE] = 3;
44     controlFlags[LUMEN][RATE] = 2;
45     controlFlags[PIR][RATE] = 1;
46 
47     controlFlags[MOVE_ALERT][RATE] = 1;
48 }
```

Her er kalder-grafen for denne funktion:



4.12.4.5 int main ( )

Main function.

Forfatter

Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 111 i filen main.c.

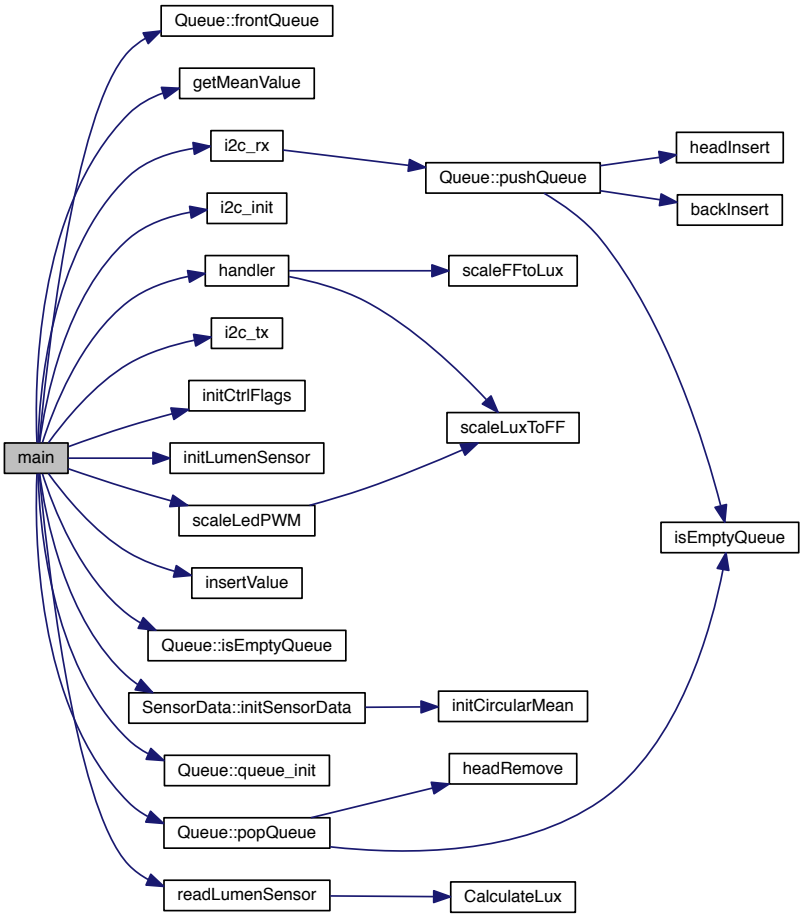
Indeholder referencer til bla, sensorDataT::bluePWMPct, cmdDistanceAlert, cmdMovementAlert, controlFlags, C↩
OUNT, sensorDataT::desiredLux, DIST, DIST_ALERT, distAlertTriggered, FLAG, Queue::frontQueue(), getMean↩
Value(), sensorDataT::greenPWMPct, handler(), i2c_init(), i2c_rx(), i2c_tx(), initCtrlFlags(), initLumenSensor(), SensorData::initSensorData(), insertValue(), Queue::isEmptyQueue(), sensorDataT::ledPower, LUMEN, sensor↩
DataT::LumenMean, sensorDataT::lux, MOVE_ALERT, sensorDataT::movement, sensorDataT::movementAlertOn, PIR, Queue::popQueue(), Queue::queue_init(), Queue::queueCount_, readLumenSensor(), sensorDataT::redP↩
WMPct, scaleLedPWM() og sensorData.

```
112 {
113     CyGlobalIntEnable; /* Enable global interrupts. */
114 
115     // Data collection
116     initSensorData();
117 
118     // Afstandssensor
119     DistTimer_Start();
120     DistTimerInt_StartEx(DistTimer_Interrupt);
121 
122     // Lumen sensor
123     initLumenSensor();
```

```
124
125     // LED PWM
126     GreenPWM_Start();
127     RedPWM_Start();
128     BluePWM_Start();
129
130     // Main command loop
131     initCtrlFlags();
132     MetronomeTimer_Start();
133     MetronomeISR_StartEx(Metronome_Interrupt);
134     queue_init(6u);
135     i2c_init();
136
137     for(;;)
138     {
139         // Handle communication with PSoC4Master
140         i2c_rx();
141         if(isEmptyQueue() != 1)
142         {   char queue[25];
143             struct Data action;
144             action = frontQueue();
145             handler(action.cmd_, action.val_);
146             sprintf(queue, "Queue: %i Cmd: %i Cal: %i\n\r", queueCount_,action.cmd_, action.val_
    );
147             DEBUG_PutString(queue);
148             popQueue();
149         }
150         i2c_tx();
151
152         // Afstandssensor
153         if (controlFlags[DIST][FLAG]) {
154             controlFlags[DIST][FLAG] = 0;
155             // Start the distance sensor by holding the trigger pin high for 10us
156             DistReset_Write(0);
157             DistTrigger_Write(1);
158             CyDelayUs(10);
159             DistTrigger_Write(0);
160         }
161
162         if (controlFlags[DIST_ALERT][FLAG]) {
163             controlFlags[DIST_ALERT][FLAG] = 0;
164
165             // Have to get two "too close" readings in a row to trigger alert
166             ++(controlFlags[DIST_ALERT][COUNT]);
167             if (!distAlertTriggered && controlFlags[
    DIST_ALERT][COUNT] >= 1) {
168                 // Set counter to fixed value - keeps it high but prevents overflow
169                 controlFlags[DIST_ALERT][COUNT] = 10;
170                 distAlertTriggered = 1;
171                 DistInterruptPin_Write(1);
172                 handler(cmdDistanceAlert, 0xff);
173             }
174         }
175
176         // PIR sensor
177         if (controlFlags[PIR][FLAG]) {
178             controlFlags[PIR][FLAG] = 0;
179
180             if (sensorData.movementAlertOn) {
181                 // Read the PIR sensor output
182                 int tmp = PIR_Trig_Read();
183                 // If there is movement now but not before
184                 if (tmp) {
185                     // Movement detected - set movement high
186                     sensorData.movement = 10;
187
188                     if (!sensorData.movement) {
189                         // Did it just happen?
190                         handler(cmdMovementAlert, 0xff);
191                     }
192                 } else {
193                     // No movement
194                     if (sensorData.movement) {
195                         // But there were movement before - so slowly decay the indicator
196                         --(sensorData.movement);
197                     }
198                 }
199             } else {
200                 // Movement alert is off
201                 sensorData.movement = 0;
202             }
203         }
204
205         if (controlFlags[MOVE_ALERT][FLAG]) {
206             controlFlags[MOVE_ALERT][FLAG] = 0;
207
208             if (sensorData.movementAlertOn) {
```

```
209                 if (sensorData.movement) {
210                     sensorData.ledPower = 1;
211                     RedPWM_Start();
212                     GreenPWM_Start();
213                     BluePWM_Start();
214                     RedPWM_WriteCompare(sensorData.redPWMPct);
215                     GreenPWM_WriteCompare(sensorData.greenPWMPct);
216                     BluePWM_WriteCompare(sensorData.bluePWMPct);
217                 } else {
218                     sensorData.ledPower = 0;
219                     RedPWM_Stop();
220                     GreenPWM_Stop();
221                     BluePWM_Stop();
222                 }
223             }
224         }
225 
226         // Lumen sensor
227         if (controlFlags[LUMEN][FLAG]) {
228             controlFlags[LUMEN][FLAG] = 0;
229             // Read the lux value from the sensor
230             unsigned int luxValue = readLumenSensor();
231             // Add lux value to history
232             insertValue(&sensorData.LumenMean, luxValue);
233             sensorData.lux = getMeanValue(&sensorData.
    LumenMean);
234 
235             if (sensorData.ledPower && sensorData.
    desiredLux != 0) {
236                 scaleLedPWM();
237             }
238 
239 #ifdef DEBUG_ON
240             sprintf(bla, "Lux: %u, mean: %i\n\r", luxValue, sensorData.
    lux);
241             DEBUG_PutString(bla);
242 #endif
243         }
244     }
245 }
```

Her er kald-grafen for denne funktion:



4.12.4.6 void scaleLedPWM ( )

Should control LED output according to measured and desired lux Does not currently work: LEDs cannot produce enough light to affect the measured lux in a meaningful way. Should have implemented a primitive PID controller (so far only P-part has been attempted implemented).

Forfatter

Simon Nejmann (19981127@uni.au.dk)

Defineret på linje 256 i filen main.c.

Indeholder referencer til bla, sensorDataT::bluePWMPct, sensorDataT::desiredLux, sensorDataT::greenPWMPct, sensorDataT::lux, sensorDataT::redPWMPct, scaleLuxToFF() og sensorData.

Refereret til af main().

```
257 {
258     // Work in scaled units since the LedPWMs also use the 0-255 range
259     uint8 luxFF = scaleLuxToFF(sensorData.lux);
260     uint8 luxDesFF = scaleLuxToFF(sensorData.desiredLux);
261     int16 delta = luxDesFF - luxFF;
262 
263     // Keep delta in check - don't want light-levels jumping too much
264     delta = (delta > 10) ? 10 : delta;
265     delta = (delta < -10) ? -10 : delta;
266 
267 #ifdef DEBUG_ON
268     sprintf(bla, "Delta: %i, RGB: %i, %i, %i\n\r", delta,
269         sensorData.redPWMPct, sensorData.
      greenPWMPct, sensorData.bluePWMPct);
270     DEBUG_PutString(bla);
271 #endif
272 
273 // Scale LEDs by delta
274     sensorData.redPWMPct += delta;
275     sensorData.greenPWMPct += delta;
276     sensorData.bluePWMPct += delta;
277 
278     RedPWM_WriteCompare(sensorData.redPWMPct);
279     GreenPWM_WriteCompare(sensorData.greenPWMPct);
280     BluePWM_WriteCompare(sensorData.bluePWMPct);
281 }
```

Her er kald-grafen for denne funktion:

Her er kalder-grafen for denne funktion:

4.12.5 Variabel-dokumentation

4.12.5.1 char bla[150]

Defineret på linje 23 i filen main.c.

Refereret til af main() og scaleLedPWM().

4.12.5.2 char controlFlags[5][3]

Startværdi:

```
= {
    {1,-1, 0},
    {1,-1, 0},
    {1,-1, 0},
    {1,-1, 0},
    {1,-1, 0}
}
```

Control matrix: flags x event_names

Defineret på linje 33 i filen main.c.

Refereret til af CY_ISR(), incrCtrlFlag(), initCtrlFlags() og main().

4.12.5.3 uint8 distAlertTriggered = 0

Flag: Have we already triggered a "too close" alert?

Defineret på linje 75 i filen main.c.

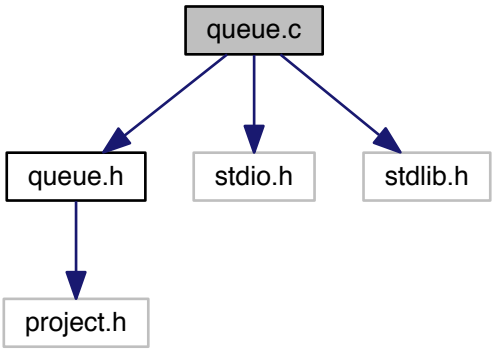
Refereret til af CY_ISR() og main().

4.13 queue.c filreference

A queue for incoming commands.

```
#include "queue.h"
#include <stdio.h>
#include <stdlib.h>
```

Inklusions-afhængighedsgraf for queue.c:



Datastrukturer

- struct Node

  Struct to contain a element in the queue. Mere...

Funktioner

- static void headInsert (struct Node **headPtr, const struct Data data)
- static void headRemove (struct Node **headPtr)
- static void backInsert (struct Node **backPtr, const struct Data data)

4.13.1 Detaljeret beskrivelse

A queue for incoming commands.

4.13.2 Datastruktur-documentation

4.13.2.1 struct Node

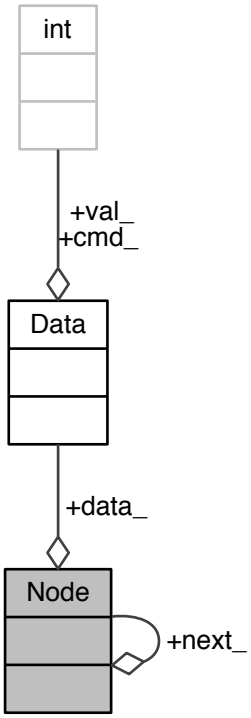
Struct to contain a element in the queue.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 20 i filen queue.c.

Samarbejdsdiagram for Node:



Data-felter

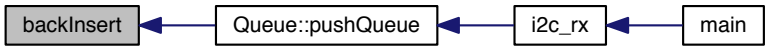
struct Data	data_	Data stored in queue
struct Node *	next_	Next node in queue

4.13.3 Funktions-dokumentation

4.13.3.1 static void backInsert (struct Node ** *backPtr,* const struct Data *data*) [static]

Refereret til af Queue::pushQueue().

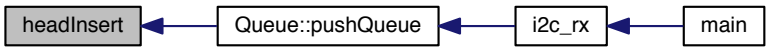
Her er kalder-grafen for denne funktion:



4.13.3.2 static void headInsert (struct Node ** *headPtr,* const struct Data *data*) [static]

Refereret til af Queue::pushQueue().

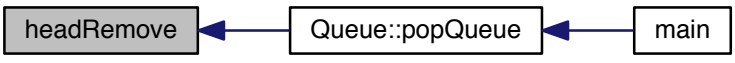
Her er kalder-grafen for denne funktion:



4.13.3.3 static void headRemove (struct Node ** *headPtr*) [static]

Refereret til af Queue::popQueue().

Her er kalder-grafen for denne funktion:

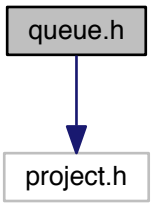


4.14 queue.h filreference

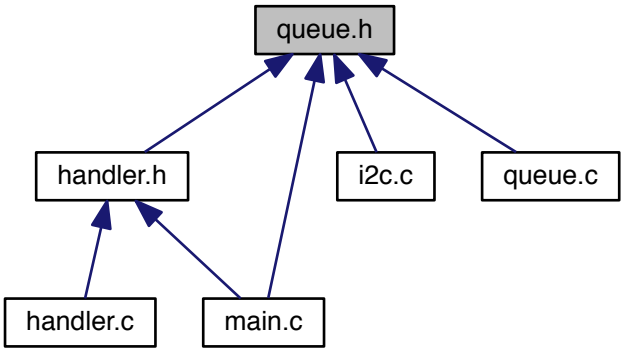
A queue for incoming commands.

```
#include <project.h>
```

Inklusions-afhængighedsgraf for queue.h:



Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



Datastrukturer

- struct Data

  Struct to contain a command and value. Mere...

Funktioner

- void queue_init (uint8 queueSize)
- void pushQueue (const struct Data data)
- void popQueue (void)
- struct Data frontQueue (void)
- int isEmptyQueue (void)

Variable

- uint8 queueCount_

4.14.1 Detaljeret beskrivelse

A queue for incoming commands.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

4.14.2 Datastruktur-documentation

4.14.2.1 struct Data

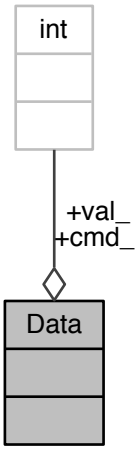
Struct to contain a command and value.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 24 i filen queue.h.

Samarbejdsdiagram for Data:



Data-felter

int	cmd↩ _	Command stored in queue
int	val↩	Value stored in queue
	_	

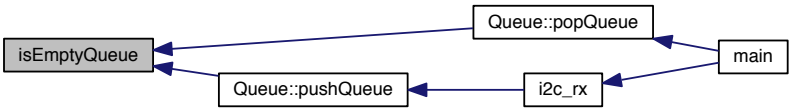
4.14.3 Funktions-dokumentation

4.14.3.1 struct Data frontQueue (void)

4.14.3.2 int isEmptyQueue (void)

Refereret til af Queue::popQueue() og Queue::pushQueue().

Her er kalder-grafen for denne funktion:



4.14.3.3 void popQueue (void)

4.14.3.4 void pushQueue (const struct Data *data*)

4.14.3.5 void queue_init (uint8 *queueSize*)

4.14.4 Variabel-dokumentation

4.14.4.1 uint8 queueCount_

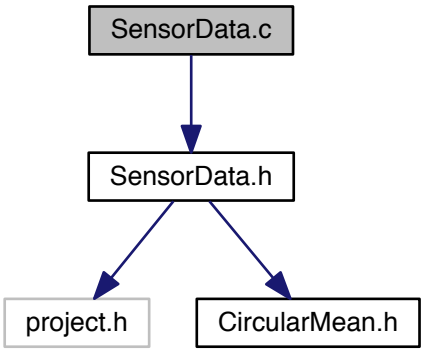
Refereret til af Queue::popQueue(), Queue::pushQueue() og Queue::queue_init().

4.15 SensorData.c filreference

Container for sensor data.

```
#include "SensorData.h"
```

Inklusions-afhængighedsgraf for SensorData.c:



4.15.1 Detaljeret beskrivelse

Container for sensor data.

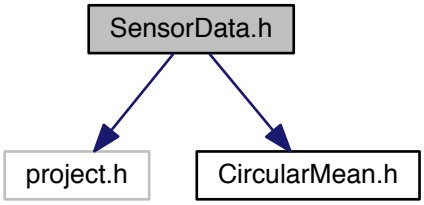
Forfatter

Simon Nejmann (19981127@uni.au.dk)

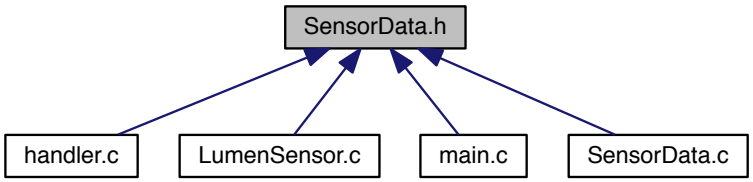
4.16 SensorData.h filreference

```
#include <project.h>
#include "CircularMean.h"
```

Inklusions-afhængighedsgraf for SensorData.h:



Denne graf viser, hvilke filer der direkte eller indirekte inkluderer denne fil:



Datastrukturer

- struct sensorDataT

  Container for sensor data. Mere...

Variable

- struct sensorDataT sensorData

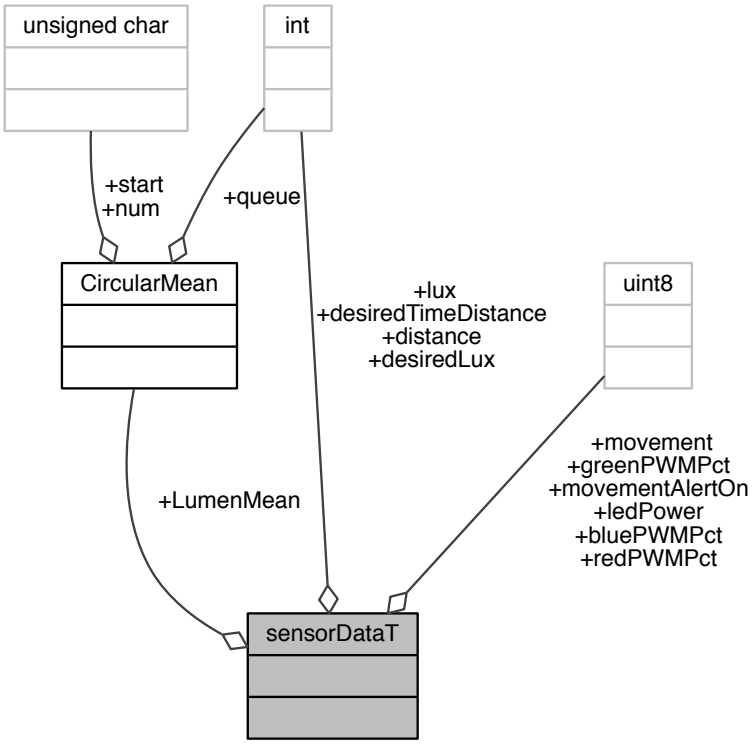
4.16.1 Datastruktur-documentation

4.16.1.1 struct sensorDataT

Container for sensor data.

Defineret på linje 16 i filen SensorData.h.

Samarbejdsdiagram for sensorDataT:



Data-felter

uint8	bluePWMPct	Power level of the blue LED: Scale 0-255.
unsigned int	desiredLux	The lux value the system should try to maintain
int	desiredTimeDistance	The distance the lamp should not be lowered below. Stored in micro-seconds for greater resolution
int	distance	The latest reading from the ultrasonic distance sensor
uint8	greenPWMPct	Power level of the green LED: Scale 0-255.
uint8	ledPower	Are the LEDs currently on or off
struct CircularMean	LumenMean	Circular structure. Can add values and get their average. Used for lux measurements
unsigned int	lux	The average value of lux measured
uint8	movement	The latest reading from the PIR movement sensor
uint8	movementAlertOn	Should the system react to movement or ignore it
uint8	redPWMPct	Power level of the red LED: Scale 0-255.

4.16.2 Variabel-dokumentation

4.16.2.1 struct sensorDataT sensorData

Refereret til af CY_ISR(), handler(), SensorData::initSensorData(), main() og scaleLedPWM().