

L.A.M.P

Genereret af Doxygen 1.8.11

Indhold

1 Indeks over datastrukturer

1.1 Datastrukturer

Her er datastrukturerne med korte beskrivelser:

Data	
Data class	??
Handler	
Handler class	??
I2C	
I2C class	??
LED	
LED class	??
Queue	
Queue class	??
XY	
XY class	??

2 Fil-indeks

2.1 Filoversigt

Her er en liste over alle filer med korte beskrivelser:

cyapicallbacks.h	??
data.c	
Data modul	??
data.h	
Data modul	??
handler.c	
Handler modul	??
handler.h	
Handler modul	??
i2c.c	
I2C modul	??
i2c.h	
I2C modul	??
led.c	
LED modul	??

led.h	
LED modul	??
main.c	
Hovedprogram	??
queue.c	
Queue modul	??
queue.h	
Queue modul	??
xy.c	
XY modul	??
xy.h	
XY modul	??

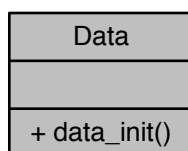
3 Datastruktur-documentation

3.1 Data Klasse-reference

[Data](#) class.

```
#include <data.h>
```

Samarbejdsdiagram for Data:



Offentlige metoder

- void [data_init](#) (void)
Initialiser data modulet.

3.1.1 Detaljeret beskrivelse

[Data](#) class.

Indeholder data vedr. [XY](#) modulet.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

3.1.2 Dokumentation af medlemsfunktioner

3.1.2.1 void data_init (void)

Initialiser data modulet.

Initialiser dataXY structen med start værdier.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 24 i filen data.c.

Indeholder referencer til DataXY::calibratedX, DataXY::calibratedY, dataXY, DataXY::interruptX, DataXY::interruptY, DataXY::isrStopX, DataXY::isrStopY, DataXY::xFlag, DataXY::xMax, DataXY::xPos, DataXY::yFlag, DataXY::yMax og DataXY::yPos.

Refereret til af main().

```
25 {  
26   dataXY.calibratedX = 1;  
27   dataXY.calibratedY = 1;  
28   dataXY.interruptX = 0;  
29   dataXY.interruptY = 0;  
30   dataXY.isrStopX = 0;  
31   dataXY.isrStopY = 0;  
32   dataXY.xFlag = 0;  
33   dataXY.yFlag = 0;  
34   dataXY.xMax = 3460;  
35   dataXY.xPos = 0;  
36   dataXY.yMax = 1475;  
37   dataXY.yPos = 0;  
38 }
```

Her er kalder-grafen for denne funktion:



Dokumentationen for denne klasse blev genereret ud fra filerne:

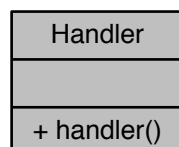
- [data.h](#)
- [data.c](#)

3.2 Handler Klasse-reference

[Handler](#) class.

```
#include <handler.h>
```

Samarbejdsdiagram for Handler:



Offentlige metoder

- void [handler](#) (uint8 cmd, uint8 val)
Håndter kommando med tilhørende værdi.

3.2.1 Detaljeret beskrivelse

[Handler](#) class.

Håndtere indkommende kommandoer med tilhørende værdier.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

3.2.2 Dokumentation af medlemsfunktioner

3.2.2.1 void handler (uint8 cmd, uint8 val)

Håndter kommando med tilhørende værdi.

Fortager en defineret handling ud fra den modtaget kommando med den tilhørende værdi.

Parametre

in	<i>cmd</i>	Er den modtaget kommando.
in	<i>val</i>	Er den tilhørende værdi.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

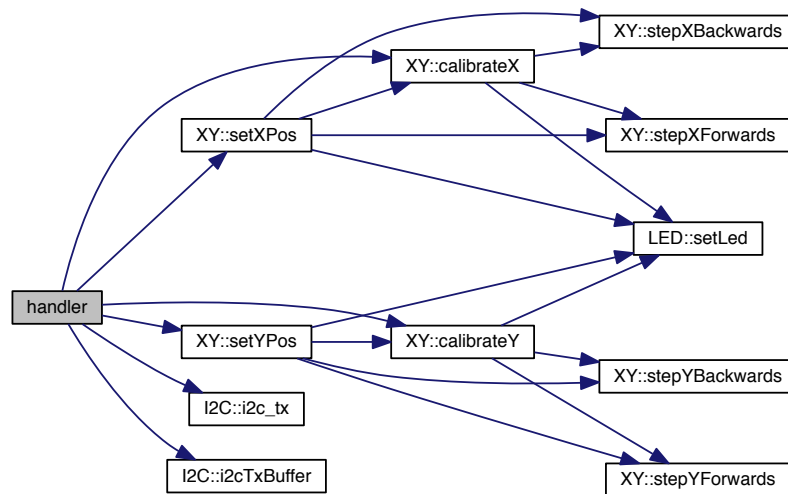
Defineret på linje 29 i filen handler.c.

Indeholder referencer til XY::calibrateX(), XY::calibrateY(), CMD_GET_X_POS, CMD_GET_Y_POS, CMD_SET_X_POS, CMD_SET_Y_POS, CMD_X_CAL, CMD_X_STP, CMD_Y_CAL, CMD_Y_STP, dataXY, I2C_PACKET_CMD_POS, I2C_PACKET_VAL_POS, I2C::i2c_tx(), I2C::i2cTxBuffer(), DataXY::isrStopX, DataXY::isrStopY, resolution, XY::setXPos(), XY::setYPos(), DataXY::xMax, DataXY::xPos, DataXY::yMax og DataXY::yPos.

Refereret til af main().

```
30 {
31     switch (cmd) {
32         case CMD_SET_X_POS :
33             setXPos(val);
34             break;
35         case CMD_SET_Y_POS:
36             setYPos(val);
37             break;
38         case CMD_GET_X_POS :
39             i2cTxBuffer[I2C_PACKET_CMD_POS] = cmd;
40             i2cTxBuffer[I2C_PACKET_VAL_POS] = (uint8)((resolution *
dataXY.xPos) / dataXY.xMax + 1);
41             i2c_tx();
42             break;
43         case CMD_GET_Y_POS :
44             i2cTxBuffer[I2C_PACKET_CMD_POS] = cmd;
45             i2cTxBuffer[I2C_PACKET_VAL_POS] = (uint8)((resolution *
dataXY.yPos) / dataXY.yMax + 1);
46             i2c_tx();
47             break;
48         case CMD_X_STP:
49             dataXY.isrStopX = 1;
50             break;
51         case CMD_Y_STP:
52             dataXY.isrStopY = 1;
53             break;
54         case CMD_X_CAL:
55             calibrateX();
56             break;
57         case CMD_Y_CAL:
58             calibrateY();
59             break;
60         default :
61             break;
62     }
63 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



Dokumentationen for denne klasse blev genereret ud fra filerne:

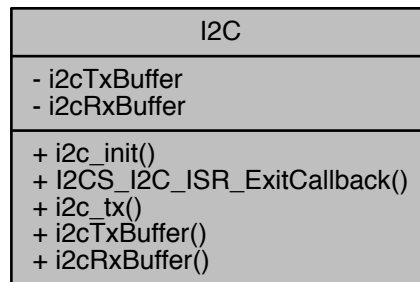
- [handler.h](#)
- [handler.c](#)

3.3 I2C Klasse-reference

I2C class.

```
#include <i2c.h>
```

Samarbejdsdiagram for I2C:



Offentlige metoder

- void `i2c_init()`
Initialiser I2C modulet.
- void `I2CS_I2C_ISR_ExitCallback()`
Modtager "Exit Callback" fra I2C.
- void `i2c_tx()`
Ryder om efter I2C.
- uint8 `i2cTxBuffer[I2C_BUFFER_SIZE]`
Buffer til afsendelse af data.
- uint8 `i2cRxBuffer[I2C_BUFFER_SIZE]`
Buffer til modtagelse af data.

Private attributter

- uint8 `i2cTxBuffer[I2C_BUFFER_SIZE]` = {`I2C_PACKET_SOP`, `I2C_STS_CMD_FAIL`, `I2C_STS_CMD_FAIL`, `I2C_PACKET_EOP`}
Buffer til afsendelse af data.
- uint8 `i2cRxBuffer[I2C_BUFFER_SIZE]`
Buffer til modtagelse af data.

3.3.1 Detaljeret beskrivelse

`I2C` class.

Håndter kommunikation via I2C-busset.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

3.3.2 Dokumentation af medlemsfunktioner

3.3.2.1 void i2c_init (void)

Initialiser I2C modulet.

Initailiser I2C komponent på PSoC'en.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

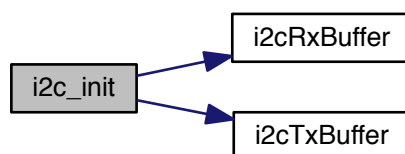
Defineret på linje 48 i filen i2c.c.

Indeholder referencer til I2C_BUFFER_SIZE, i2cRxBuffer() og i2cTxBuffer().

Refereret til af main().

```
49 {  
50     I2CS_I2CSlaveInitReadBuf(i2cTxBuffer, I2C_BUFFER_SIZE);  
51     I2CS_I2CSlaveClearReadBuf();  
52     I2CS_I2CSlaveClearReadStatus();  
53  
54     I2CS_I2CSlaveInitWriteBuf(i2cRxBuffer, I2C_BUFFER_SIZE);  
55     I2CS_I2CSlaveClearWriteBuf();  
56     I2CS_I2CSlaveClearWriteStatus();  
57  
58     I2CS_Start();  
59 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.3.2.2 void i2c_tx (void)

Ryder om efter I2C.

Efter fuldført afsendelse af pakke til I2C-master, bliver status nulstillet.

Forfatter

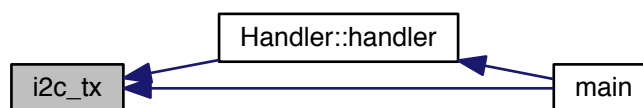
Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 120 i filen i2c.c.

Refereret til af Handler::handler() og main().

```
121 {  
122     if(0u != (I2CS_I2CSlaveStatus() & I2CS_I2C_SSTAT_RD_CMPLT))  
123     {  
124         I2CS_I2CSlaveClearReadBuf();  
125         (void) I2CS_I2CSlaveClearReadStatus();  
126     }  
127 }
```

Her er kalder-grafen for denne funktion:



3.3.2.3 uint8 i2cRxBuffer ()

Buffer til modtagelse af data.

En buffer der indeholder de data pakker der skal modtagelse over I2C-busset.

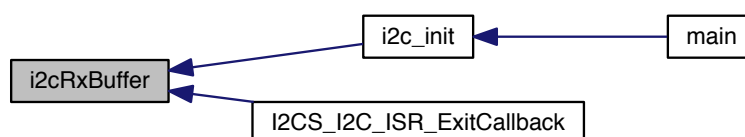
Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 74 i filen i2c.h.

Refereret til af `i2c_init()` og `I2CS_I2C_ISR_ExitCallback()`.

Her er kalder-grafen for denne funktion:



3.3.2.4 void I2CS_I2C_ISR_ExitCallback (void)

Motager "Exit Callback" fra [I2C](#).

En "Interrupt Service Routine(ISR)" der aktiveres ved færdig modtagelse af kald via I2C-busset, det modtaget data behandles og håndteres.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 68 i filen i2c.c.

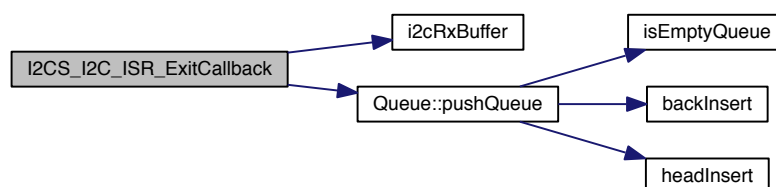
Indeholder referencer til Action::cmd, CMD_SET_X_POS, CMD_SET_Y_POS, dataXY, I2C_BUFFER_SIZE, I2C_PACKET_CMD_POS, I2C_PACKET_VAL_POS, i2cRxBuffer(), DataXY::isrStopX, DataXY::isrStopY, Queue::pushQueue() og Action::val.

```

69 {
70     if(I2CS_I2CSlaveGetWriteBufSize() == I2C_BUFFER_SIZE)
71     {
72         DEBUG_PutCRLF();
73         DEBUG_PutString("*** isr exit callback ***");
74         DEBUG_PutCRLF();
75         DEBUG_PutString("I> i2cRxBuffer[0]: ");
76         DEBUG_PutHexByte(i2cRxBuffer[0]);
77         DEBUG_PutString(" [1]: ");
78         DEBUG_PutHexByte(i2cRxBuffer[1]);
79         DEBUG_PutString(" [2]: ");
80         DEBUG_PutHexByte(i2cRxBuffer[2]);
81         DEBUG_PutString(" [3]: ");
82         DEBUG_PutHexByte(i2cRxBuffer[3]);
83         DEBUG_PutString(" buffer size: ");
84         DEBUG_PutHexByte(I2CS_I2CSlaveGetWriteBufSize());
85         DEBUG_PutCRLF();
86
87         struct Action action;
88         action.cmd = i2cRxBuffer[I2C_PACKET_CMD_POS];
89         action.val = i2cRxBuffer[I2C_PACKET_VAL_POS];
90
91         switch(i2cRxBuffer[I2C_PACKET_CMD_POS]) {
92             case CMD_SET_X_POS :
93                 dataXY.isrStopX = 1;
94                 DEBUG_PutString(" isrStopX = 1");
95                 DEBUG_PutCRLF();
96                 pushQueue(action);
97                 break;
98             case CMD_SET_Y_POS :
99                 dataXY.isrStopY = 1;
100                 DEBUG_PutString(" isrStopY = 1");
101                 DEBUG_PutCRLF();
102                 pushQueue(action);
103                 break;
104             default :
105                 pushQueue(action);
106                 break;
107         }
108         I2CS_I2CSlaveClearWriteBuf();
109         (void) I2CS_I2CSlaveClearWriteStatus();
110     }
111 }

```

Her er kald-grafen for denne funktion:



3.3.2.5 uint8 i2cTxBuffer ()

Buffer til afsendelse af data.

En buffer der indeholder de data pakker der skal sende over I2C-busset.

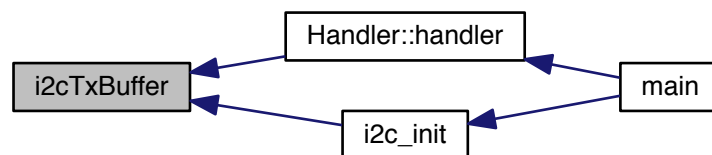
Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 65 i filen i2c.h.

Refereret til af Handler::handler() og i2c_init().

Her er kalder-grafen for denne funktion:



3.3.3 Felt-dokumentation

3.3.3.1 uint8 i2cRxBuffer[I2C_BUFFER_SIZE] [private]

Buffer til modtagelse af data.

En buffer der indeholder de data pakker der skal modtagelse over I2C-busset.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 34 i filen i2c.c.

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

Buffer til afsendelse af data.

En buffer der indeholder de data pakker der skal sende over I2C-busset.

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 25 i filen i2c.c.

Dokumentationen for denne klasse blev genereret ud fra filerne:

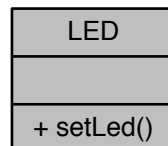
- [i2c.h](#)
- [i2c.c](#)

3.4 LED Klasse-reference

LED class.

```
#include <led.h>
```

Samarbejdsdiagram for LED:



Offentlige metoder

- void [setLed](#) (uint8 red, uint8 green, uint8 blue, uint8 delay)
Sætter den defineret farve og angivet delay.

3.4.1 Detaljeret beskrivelse

LED class.

Håndtere PSoC'ens røde, grønne og blå led

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
 Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
 Jeppe Stærk Antonsen (201271201@uni.au.dk)

3.4.2 Dokumentation af medlemsfunktioner

3.4.2.1 void setLed (uint8 red, uint8 green, uint8 blue, uint8 delay)

Sætter den defineret farve og angivet delay.

Metoden sætter den/de valgte farver og venter i det angivet delay.

Parametre

in	<i>red</i>	Tænder/slukker den røde led.
in	<i>green</i>	Tænder/slukker den grønne led.
in	<i>blue</i>	Tænder/slukker den blå led.
in	<i>delay</i>	Tid i microsekunder til delay.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk (201271201@uni.au.dk)

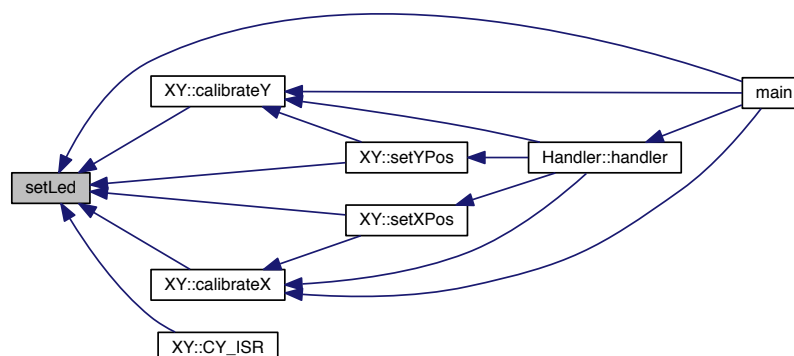
Defineret på linje 28 i filen led.c.

Indeholder referencer til LED_OFF og LED_ON.

Refereret til af XY::calibrateX(), XY::calibrateY(), XY::CY_ISR(), main(), XY::setXPos() og XY::setYPos().

```
29 {  
30     red ? LED_RED_Write(LED_ON) : LED_RED_Write(LED_OFF);  
31     green ? LED_GREEN_Write(LED_ON) : LED_GREEN_Write(LED_OFF);  
32     blue ? LED_BLUE_Write(LED_ON) : LED_BLUE_Write(LED_OFF);  
33  
34     CyDelay(delay);  
35 }
```

Her er kalder-grafen for denne funktion:



Dokumentationen for denne klasse blev genereret ud fra filerne:

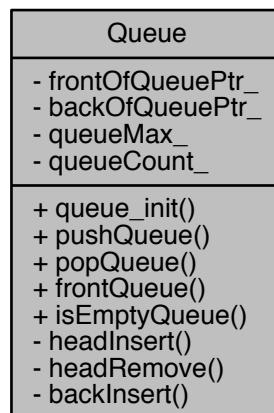
- [led.h](#)
- [led.c](#)

3.5 Queue Klasse-reference

Queue class.

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

Samarbejdsdiagram for Queue:



Offentlige metoder

- void `queue_init` (uint8 queueMaxSize)
Initialiser Queue modulet.
- void `pushQueue` (const struct `Action` data)
Indsætter et element i køen.
- void `popQueue` ()
Fjerner et element i køen.
- struct `Action` `frontQueue` ()
Viser et element fra køen.
- uint8 `isEmptyQueue` ()
Retuner status af køen.

Private metoder

- void `headInsert` (struct `Node` **headPtr, const struct `Action` data)
Indsætter forreste i listen.
- void `headRemove` (struct `Node` **headPtr)
Fjerner fra listen.
- void `backInsert` (struct `Node` **backPtr, const struct `Action` data)
Indsætter bagerst i listen.

Statiske, private attributter

- static struct `Node` * `frontOfQueuePtr_`
Pointer til foreste element i køen.
- static struct `Node` * `backOfQueuePtr_`
Pointer til bagerste element i køen.
- static uint8 `queueMax_`
Køens max.
- static uint8 `queueCount_`
Kø element tæller.

3.5.1 Detaljeret beskrivelse

[Queue](#) class.

En FIFO kø der er opbygget af en single linked liste.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

3.5.2 Dokumentation af medlemsfunktioner

3.5.2.1 void backInsert (struct Node ** backPtr, const struct Action data) [private]

Indsætter bagerst i listen.

Indsætter det angivet element bagerst i den underlægende linked liste.

Parametre

in	<i>backPtr</i>	Pointer til det bagerste element i listen.
in	<i>data</i>	Data der skal indsættes i listen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 246 i filen queue.c.

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

```
247 {  
248     if(*backPtr == NULL)  
249     {  
250         return;  
251     }  
252  
253     struct Node* next = (*backPtr)->next_;  
254     struct Node* temp = (struct Node*)malloc(sizeof(struct Node));  
255     temp->data_ = data;  
256     temp->next_ = next;  
257     (*backPtr)->next_ = temp;  
258 }
```

3.5.2.2 struct Action frontQueue (void)

Viser et element fra køen.

Viser det foreste element i FIFO køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 168 i filen queue.c.

Indeholder referencer til Node::data_.

Refereret til af main().

```
169 {
170     DEBUG_PutString("Q=: count: ");
171     DEBUG_PutHexByte(queueCount_);
172     DEBUG_PutCRLF();
173     return frontOfQueuePtr->data_;
174 }
```

Her er kalder-grafen for denne funktion:



3.5.2.3 void headInsert (struct Node ** headPtr, const struct Action data) [private]

Indsætter forreste i listen.

Indsætter det angivet element forreste i den underlægende linked liste.

Parametre

in	headPtr	Pointer til det foreste element i listen.
in	data	Data der skal indsættes i listen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 204 i filen queue.c.

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

```
205 {
206     struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
207     if(temp == NULL)
208     {
209         return;
210     }
211     temp->data_ = data;
212     temp->next_ = NULL;
213     *headPtr = temp;
214 }
215 }
```

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

Fjerner fra listen.

Fjerner det forreste element i den underlæggende linked liste

Parametre

in	headPtr	Pointer til det forreste element i listen.
----	---------	--------------------------------------------

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 226 i filen queue.c.

Indeholder referencer til Node::next_.

```

227 {
228     if(headPtr != NULL)
229     {
230         struct Node* condemned;
231         condemned = *headPtr;
232         *headPtr = (*headPtr)->next_;
233         free(condemned);
234     }
235 }
```

3.5.2.5 uint8 isEmptyQueue (void)

Retuner status af køen.

Kontrollere om køen er tom.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 183 i filen queue.c.

Refereret til af main().

```

184 {
185     if(frontOfQueuePtr_ == NULL)
186     {
187         return 1;
188     }
189     else
190     {
191         return 0;
192     }
193 }
```

Her er kalder-grafen for denne funktion:



3.5.2.6 void popQueue (void)

Fjerner et element i køen.

Fjerner det foreste element i FIFO køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

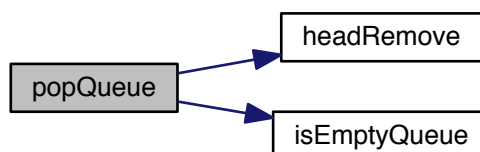
Defineret på linje 147 i filen queue.c.

Indeholder referencer til headRemove() og isEmptyQueue().

Refereret til af main().

```
148 {  
149     headRemove (&frontOfQueuePtr_);  
150     queueCount_--;  
151     if (isEmptyQueue () == 1)  
152     {  
153         backOfQueuePtr_ = NULL;  
154     }  
155     DEBUG_PutString ("-Q: count: ");  
156     DEBUG_PutHexByte (queueCount_);  
157     DEBUG_PutCRLF ();  
158     DEBUG_PutCRLF ();  
159 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.5.2.7 void pushQueue (const struct Action data)

Indsætter et element i køen.

Indsætter det angivet element bagerst i FIFO køen.

Parametre

in	data	Data der skal indsættes i køen.
----	------	---------------------------------

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

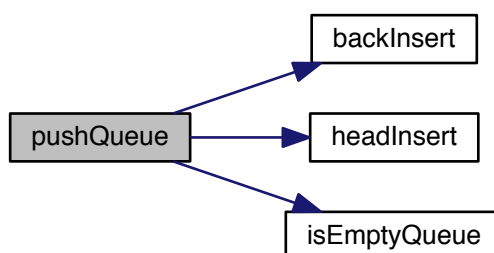
Defineret på linje 105 i filen queue.c.

Indeholder referencer til backInsert(), Action::cmd, headInsert(), isEmptyQueue(), Node::next_ og Action::val.

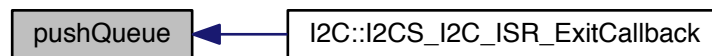
Refereret til af I2C::I2CS_I2C_ISR_ExitCallback().

```
106 {
107     if(queueCount_<queueMax_)
108     {
109         if(isEmptyQueue() != 1)
110         {
111             backInsert(&backOfQueuePtr_, data);
112             backOfQueuePtr_ = backOfQueuePtr_>next_;
113             queueCount_++;
114         }
115         else
116         {
117             headInsert(&frontOfQueuePtr_, data);
118             backOfQueuePtr_ = frontOfQueuePtr_;
119             queueCount_++;
120         }
121         DEBUG_PutString("Q+: count: ");
122         DEBUG_PutHexByte(queueCount_);
123         DEBUG_PutString(" cmd: ");
124         DEBUG_PutHexByte(data.cmd);
125         DEBUG_PutString(" val: ");
126         DEBUG_PutHexByte(data.val);
127         DEBUG_PutCRLF();
128         DEBUG_PutCRLF();
129     }
130     else
131     {
132         DEBUG_PutString("Q~: ERROR! Queue FULL!!! count: ");
133         DEBUG_PutHexByte(queueCount_);
134         DEBUG_PutCRLF();
135         DEBUG_PutCRLF();
136     }
137 }
138 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.5.2.8 void queue_init (uint8 queueMaxSize)

Initialiser [Queue](#) modulet.

Initailiser køen med den ønsket max størrelse.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 87 i filen queue.c.

Indeholder referencer til `Node::next_`.

Refereret til af `main()`.

```

88 {
89     frontOfQueuePtr_ = NULL;
90     frontOfQueuePtr_->next_ = NULL;
91     backOfQueuePtr_ = NULL;
92     backOfQueuePtr_->next_ = NULL;
93     queueMax_ = queueMaxSize;
94     queueCount_ = 0;
95 }
  
```

Her er kalder-grafen for denne funktion:



3.5.3 Felt-dokumentation

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

Pointer til bagerste element i køen.

En [Node](#) pointer der indeholder adressen på det bagerste elementet i køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 47 i filen queue.c.

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

Pointer til foreste element i køen.

En [Node](#) pointer der indeholder adressen på det foreste elementet i køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 38 i filen queue.c.

3.5.3.3 uint8 queueCount_ [static],[private]

Kø element tæller.

Bruges til at tælle hvor mange elementer der er i køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 65 i filen queue.c.

3.5.3.4 uint8 queueMax_ [static],[private]

Køens max.

Laver ved initialisering der ønsket antal for max elementer i køen

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 56 i filen queue.c.

Dokumentationen for denne klasse blev genereret ud fra filerne:

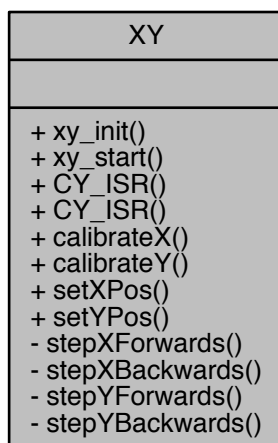
- [queue.h](#)
- [queue.c](#)

3.6 XY Klasse-reference

XY class.

```
#include <xy.h>
```

Samarbejdsdiagram for XY:



Offentlige metoder

- void `xy_init` ()
Initialiser XY modulet.
- void `xy_start` ()
Starter XY modulet.
- `CY_ISR` (isr_X)
Afvikler "Interrupt" fra X.
- `CY_ISR` (isr_Y)
Afvikler "Interrupt" fra Y.
- void `calibrateX` ()
Kalibrere X.
- void `calibrateY` ()
Kalibrere Y.
- void `setXPos` (uint8 xVal)
Sætter ny X position.
- void `setYPos` (uint8 yVal)
Sætter ny Y position.

Private metoder

- void `stepXForwards()`
Køre X motor et step frem.
- void `stepXBackwards()`
Køre X motor et step tilbage.
- void `stepYForwards()`
Køre Y motor et step frem.
- void `stepYBackwards()`
Køre Y motor et step tilbage.

3.6.1 Detaljeret beskrivelse

XY class.

Styre XY modulets funktioner.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

3.6.2 Dokumentation af medlemsfunktioner

3.6.2.1 void `calibrateX()`

Kalibrere X.

Metoden kalibrerer X og sætter en ny max værdi for X.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 164 i filen xy.c.

Indeholder referencer til `DataXY::calibratedX`, `dataXY`, `interruptSteps`, `DataXY::interruptX`, `LED::setLed()`, `stepXBackwards()`, `stepXForwards()`, `DataXY::xFlag`, `DataXY::xMax` og `DataXY::xPos`.

Refereret til af `Handler::handler()`, `main()` og `setXPos()`.

```
165 {  
166     DEBUG_PutString("X calibrate pre-xMax: ");  
167     DEBUG_PutHexInt(dataXY.xMax);  
168     DEBUG_PutCRLF();  
169  
170     dataXY.calibratedX = 0;  
171     dataXY.xFlag = 1;  
172     dataXY.xMax = 0;  
173  
174     DEBUG_PutString("Going forwards to max");  
175     while(dataXY.interruptX == 0 && dataXY.xFlag == 1)  
176     {  
177         DEBUG_PutString(".");  
178         setLed(1, 0, 0, 0);  
179         stepXForwards();  
180     }
```

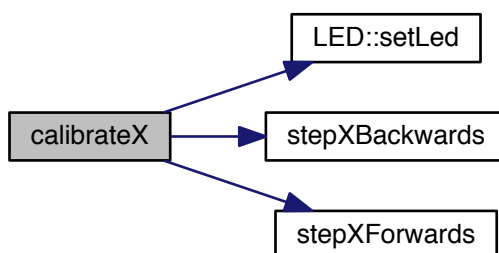


```

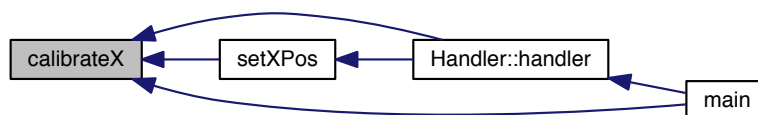
180 }
181 dataXY.interruptX = 0;
182
183 DEBUG_PutString("done");
184 DEBUG_PutCRLF();
185 DEBUG_PutString("Going backwards to zero");
186 while(dataXY.interruptX == 0 && dataXY.xFlag == 0)
187 {
188     DEBUG_PutString(".");
189     setLed(1,0,0,0);
190     stepXBackwards();
191     dataXY.xMax++;
192 }
193 DEBUG_PutString("done");
194
195 setLed(0,0,0,0);
196
197 dataXY.xPos = 0;
198 dataXY.xMax = dataXY.xMax - interruptSteps;
199
200 DEBUG_PutString(" post-xMax: ");
201 DEBUG_PutHexInt(dataXY.xMax);
202 DEBUG_PutString(" new xPos: ");
203 DEBUG_PutHexInt(dataXY.xPos);
204 DEBUG_PutCRLF();
205 DEBUG_PutCRLF();
206
207 dataXY.calibratedX = 1;
208 dataXY.interruptX = 0;
209 }

```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.6.2.2 void calibrateY (void)

Kalibrere Y.

Metoden kalibrerer Y og sætter en ny max værdi for Y.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

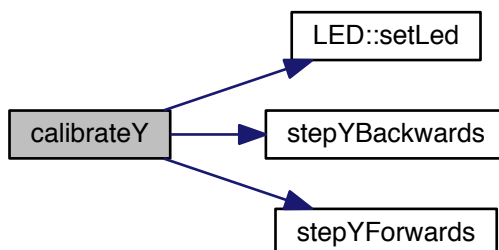
Defineret på linje 219 i filen xy.c.

Indeholder referencer til DataXY::calibratedX, DataXY::calibratedY, dataXY, interruptSteps, DataXY::interruptY, LED::setLed(), stepYBackwards(), stepYForwards(), DataXY::yFlag, DataXY::yMax og DataXY::yPos.

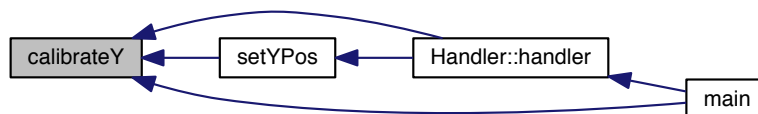
Refereret til af Handler::handler(), main() og setYPos().

```
220 {
221     DEBUG_PutString("Y calibrate pre yMax: ");
222     DEBUG_PutHexInt(dataXY.yMax);
223     DEBUG_PutCRLF();
224
225     dataXY.calibratedX = 0;
226     dataXY.yFlag = 1;
227     dataXY.yMax = 0;
228
229     DEBUG_PutString("Going forwards to max");
230     while(dataXY.interruptY == 0 && dataXY.yFlag == 1)
231     {
232         DEBUG_PutString(".");
233         setLed(1,0,0,0);
234         stepYForwards();
235     }
236     dataXY.interruptY = 0;
237
238     DEBUG_PutString("done");
239     DEBUG_PutCRLF();
240     DEBUG_PutString("Going backwards to zero");
241     while(dataXY.interruptY == 0 && dataXY.yFlag == 0)
242     {
243         DEBUG_PutString(".");
244         setLed(1,0,0,0);
245         stepYBackwards();
246         dataXY.yMax++;
247     }
248     DEBUG_PutString("done");
249
250     setLed(0,0,0,0);
251
252     dataXY.yPos = 0;
253     dataXY.yMax = dataXY.yMax - interruptSteps;
254
255     DEBUG_PutString(" post-yMax: ");
256     DEBUG_PutHexInt(dataXY.yMax);
257     DEBUG_PutString(" new yPos: ");
258     DEBUG_PutHexInt(dataXY.yPos);
259     DEBUG_PutCRLF();
260     DEBUG_PutCRLF();
261
262     dataXY.calibratedY = 1;
263     dataXY.interruptY = 0;
264 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.6.2.3 CY_ISR(isr_X)

Afvikler "Interrupt" fra X.

En "Interrupt Service Routine(ISR)" for X der aktiveres ved interrupt fra X modulet.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

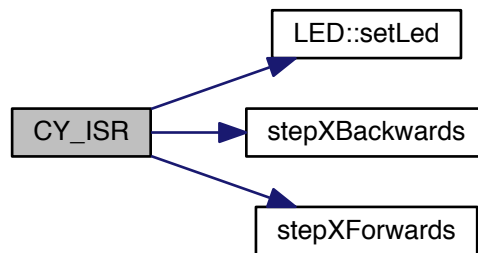
Defineret på linje 84 i filen xy.c.

Indeholder referencer til dataXY, interruptSteps, DataXY::interruptX, LED::setLed(), stepXBackwards(), stepXForwards() og DataXY::xFlag.

```

85 {
86   interrupt_X_Disable();
87
88   uint32 i;
89
90   dataXY.interruptX = 1;
91
92   if(dataXY.xFlag == 0)
93   {
94     setLed(0,0,1,0);
95     for(i = 0; i < interruptSteps; i++)
96     {
97       stepXForwards();
98     }
99     dataXY.xFlag = 1;
100   }
101   else if(dataXY.xFlag == 1)
102   {
103     setLed(0,0,1,0);
104     for(i = 0; i < interruptSteps; i++)
105     {
106       stepXBackwards();
107     }
108     dataXY.xFlag = 0;
109   }
110   setLed(0,0,0,0);
111   interrupt_X_ClearPending();
112   interrupt_X_Enable();
113 }
114 }
```

Her er kald-grafen for denne funktion:



3.6.2.4 CY_ISR (isr_Y)

Afvikler "Interrupt" fra Y.

En "Interrupt Service Routine(ISR)" for Y der aktiveres ved interrupt fra Y modulet.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

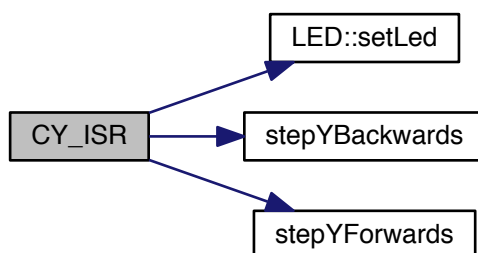
Defineret på linje 124 i filen xy.c.

Indeholder referencer til dataXY, interruptSteps, DataXY::interruptY, LED::setLed(), stepYBackwards(), stepY↔Forwards() og DataXY::yFlag.

```

125 {
126   interrupt_Y_Disable();
127
128   uint32 i;
129
130   dataXY.interruptY = 1;
131
132   if(dataXY.yFlag == 0)
133   {
134     setLed(0,0,1,0);
135     for(i = 0; i < interruptSteps; i++)
136     {
137       stepYForwards();
138     }
139     dataXY.yFlag = 1;
140   }
141   else if(dataXY.yFlag == 1)
142   {
143     setLed(0,0,1,0);
144     for(i = 0; i < interruptSteps; i++)
145     {
146       stepYBackwards();
147     }
148     dataXY.yFlag = 0;
149   }
150   setLed(0,0,0,0);
151
152   interrupt_Y_ClearPending();
153   interrupt_Y_Enable();
154 }
  
```

Her er kald-grafen for denne funktion:



3.6.2.5 void setXPos (uint8 xVal)

Sætter ny X position.

Ud fra den modtagne værdi udregnes antal step og vej til den ønsket destination.

Parametre

in	xVal	Værdi for position.
----	------	---------------------

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 275 i filen xy.c.

Indeholder referencer til DataXY::calibratedX, calibrateX(), dataXY, DataXY::interruptX, DataXY::isrStopX, resolution, LED::setLed(), stepXBackwards(), stepXForwards(), DataXY::xFlag, DataXY::xMax og DataXY::xPos.

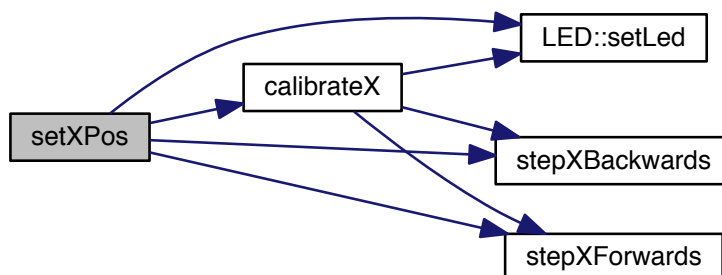
Refereret til af Handler::handler().

```

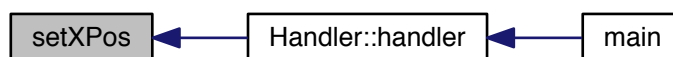
276 {
277     uint32 i;
278     uint32 xDes = 0;
279     uint32 xSteps = 0;
280
281     dataXY.isrStopX = 0;
282
283     if(dataXY.calibratedX == 1)
284     {
285         xDes = xVal * dataXY.xMax / resolution;
286
287         DEBUG_PutString("X set value: ");
288         DEBUG_PutHexInt(xVal);
289         DEBUG_PutString(" pre-xPos: ");
290         DEBUG_PutHexInt(dataXY.xPos);
291         DEBUG_PutString(" xDes: ");
292         DEBUG_PutHexInt(xDes);
293
294         if(xDes > dataXY.xPos)
295         {
  
```

```
296     setLed(0,1,0,0);
297     dataXY.interruptX = 0;
298     dataXY.xFlag = 1;
299     xSteps = xDes - dataXY.xPos;
300
301     DEBUG_PutString(" going forwards steps: ");
302     DEBUG_PutHexInt(xSteps);
303     for(i = 0; i < xSteps && dataXY.isrStopX == 0 && dataXY.
interruptX == 0 && dataXY.xFlag == 1; i++)
304     {
305         DEBUG_PutString(".");
306         stepXForwards();
307         dataXY.xPos++;
308     }
309     DEBUG_PutString("done");
310
311     if(dataXY.interruptX == 1u)
312     {
313         dataXY.xPos = dataXY.xMax;
314     }
315     DEBUG_PutString(" new-xPos: ");
316     DEBUG_PutHexInt(dataXY.xPos);
317     DEBUG_PutCRLF();
318     DEBUG_PutCRLF();
319
320     setLed(0,0,0,0);
321 }
322 else if(xDes < dataXY.xPos)
323 {
324     setLed(0,1,0,0);
325
326     dataXY.interruptX = 0;
327     dataXY.xFlag = 0;
328     xSteps = dataXY.xPos - xDes;
329
330     DEBUG_PutString(" going backwards steps: ");
331     DEBUG_PutHexInt(xSteps);
332     for(i = 0; i < xSteps && dataXY.isrStopX == 0 && dataXY.
interruptX == 0 && dataXY.xFlag == 0; i++)
333     {
334         DEBUG_PutString(".");
335         stepXBackwards();
336         dataXY.xPos--;
337     }
338     DEBUG_PutString("done");
339     if(dataXY.interruptX == 1u)
340     {
341         dataXY.xPos = 0;
342     }
343     DEBUG_PutString(" new-xPos: ");
344     DEBUG_PutHexInt(dataXY.xPos);
345     DEBUG_PutCRLF();
346     DEBUG_PutCRLF();
347
348     setLed(0,0,0,0);
349 }
350 }
351 else
352 {
353     calibrateX();
354     setXPos(xVal);
355 }
356 dataXY.interruptX = 0;
357 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.6.2.6 void setYPos (uint8 yVal)

Sætter ny Y position.

Ud fra den modtaget værdi udregnes antal step og vej til den ønsket destination.

Parametre

in	yVal	Værdi for position.
----	------	---------------------

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

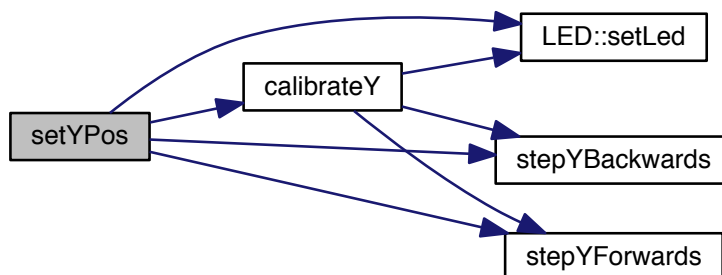
Defineret på linje 368 i filen xy.c.

Indeholder referencer til `DataXY::calibratedY`, `calibrateY()`, `dataXY`, `DataXY::interruptY`, `DataXY::isrStopY`, `resolution`, `LED::setLed()`, `stepYBackwards()`, `stepYForwards()`, `DataXY::yFlag`, `DataXY::yMax` og `DataXY::yPos`.

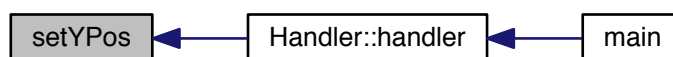
Refereret til af `Handler::handler()`.

```
369 {
370     uint32 i;
371     uint32 yDes = 0;
372     uint32 ySteps = 0;
373
374     dataXY.isrStopY = 0;
375
376     if(dataXY.calibratedY == 1)
377     {
378         yDes = yVal * dataXY.yMax / resolution;
379
380         DEBUG_PutString("Y set value: ");
381         DEBUG_PutHexInt(yVal);
382         DEBUG_PutString(" pre-yPos: ");
383         DEBUG_PutHexInt(dataXY.yPos);
384         DEBUG_PutString(" yDes: ");
385         DEBUG_PutHexInt(yDes);
386
387         if(yDes > dataXY.yPos)
388         {
389             setLed(0,1,0,0);
390
391             dataXY.interruptY = 0;
392             dataXY.yFlag = 1;
393             ySteps = yDes - dataXY.yPos;
394
395             DEBUG_PutString(" going forwards steps: ");
396             DEBUG_PutHexInt(ySteps);
397             for(i = 0; i < ySteps && dataXY.isrStopY == 0 && dataXY.
interruptY == 0 && dataXY.yFlag == 1; i++)
398             {
399                 DEBUG_PutString(".");
400                 stepYForwards();
401                 dataXY.yPos++;
402             }
403             DEBUG_PutString("done");
404             if(dataXY.interruptY == 1u)
405             {
406                 dataXY.yPos = dataXY.yMax;
407             }
408             DEBUG_PutString(" new-yPos: ");
409             DEBUG_PutHexInt(dataXY.yPos);
410             DEBUG_PutCRLF();
411             DEBUG_PutCRLF();
412
413             setLed(0,0,0,0);
414         }
415         else if(yDes < dataXY.yPos)
416         {
417             setLed(0,1,0,0);
418
419             dataXY.interruptY = 0;
420             dataXY.yFlag = 0;
421             ySteps = dataXY.yPos - yDes;
422
423             DEBUG_PutString(" going backwards steps: ");
424             DEBUG_PutHexInt(ySteps);
425             for(i = 0; i < ySteps && dataXY.isrStopY == 0u && dataXY.
interruptY == 0 && dataXY.yFlag == 0; i++)
426             {
427                 DEBUG_PutString(".");
428                 stepYBackwards();
429                 dataXY.yPos--;
430             }
431             DEBUG_PutString("done");
432             if(dataXY.interruptY == 1u)
433             {
434                 dataXY.yPos = 0;
435             }
436             DEBUG_PutString(" new-yPos: ");
437             DEBUG_PutHexInt(dataXY.yPos);
438             DEBUG_PutCRLF();
439             DEBUG_PutCRLF();
440
441             setLed(0,0,0,0);
442         }
443     }
444     else
445     {
446         calibrateY();
447         setYPos(yVal);
448     }
449     dataXY.interruptY = 0;
450 }
```


Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.6.2.7 void stepXBackwards (void) [private]

Køre X motor et step tilbage.

Køre X motoren et step tilbage.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 500 i filen `xy.c`.

Indeholder referencer til `stepDelay`.

Refereret til af `calibrateX()`, `CY_ISR()`, `setXPos()` og `xy_start()`.

```

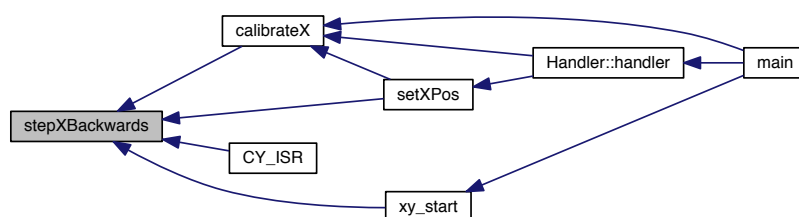
501 {
502   Pin_1a_X_Write(0);
503   Pin_2a_X_Write(0);
504   Pin_1b_X_Write(0);
505   Pin_2b_X_Write(1);
506   CyDelay(stepDelay);
507
508   Pin_1a_X_Write(0);
509   Pin_2a_X_Write(0);
  
```

```

510  Pin_1b_X_Write(1);
511  Pin_2b_X_Write(0);
512  CyDelay(stepDelay);
513
514  Pin_1a_X_Write(0);
515  Pin_2a_X_Write(1);
516  Pin_1b_X_Write(0);
517  Pin_2b_X_Write(0);
518  CyDelay(stepDelay);
519
520  Pin_1a_X_Write(1);
521  Pin_2a_X_Write(0);
522  Pin_1b_X_Write(0);
523  Pin_2b_X_Write(0);
524  CyDelay(stepDelay);
525 }

```

Her er kalder-grafen for denne funktion:



3.6.2.8 void stepXForwards (void) [private]

Køre X motor et step frem.

Køre X motoren et step fremad.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 465 i filen xy.c.

Indeholder referencer til stepDelay.

Refereret til af calibrateX(), CY_ISR() og setXPos().

```

466 {
467  Pin_1a_X_Write(1);
468  Pin_2a_X_Write(0);
469  Pin_1b_X_Write(0);
470  Pin_2b_X_Write(0);
471  CyDelay(stepDelay);
472
473  Pin_1a_X_Write(0);
474  Pin_2a_X_Write(1);
475  Pin_1b_X_Write(0);
476  Pin_2b_X_Write(0);
477  CyDelay(stepDelay);
478
479  Pin_1a_X_Write(0);
480  Pin_2a_X_Write(0);
481  Pin_1b_X_Write(1);

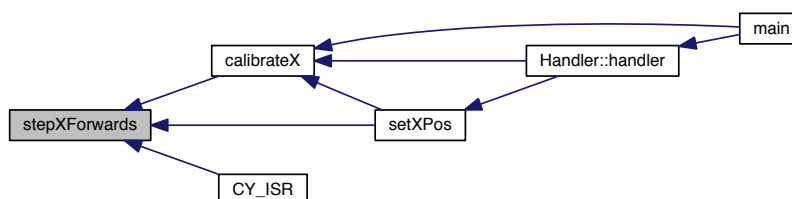
```

```

482  Pin_2b_X_Write(0);
483  CyDelay(stepDelay);
484
485  Pin_1a_X_Write(0);
486  Pin_2a_X_Write(0);
487  Pin_1b_X_Write(0);
488  Pin_2b_X_Write(1);
489  CyDelay(stepDelay);
490 }

```

Her er kalder-grafen for denne funktion:



3.6.2.9 void stepYBackwards (void) [private]

Køre Y motor et step tilbage.

Køre Y motoren et step tilbage.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 570 i filen xy.c.

Indeholder referencer til stepDelay.

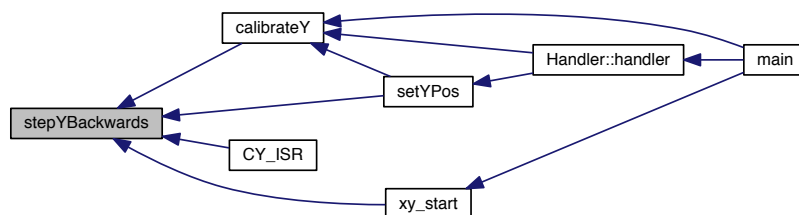
Refereret til af calibrateY(), CY_ISR(), setYPos() og xy_start().

```

571 {
572  Pin_1a_Y_Write(0);
573  Pin_2a_Y_Write(0);
574  Pin_1b_Y_Write(0);
575  Pin_2b_Y_Write(1);
576  CyDelay(stepDelay);
577
578  Pin_1a_Y_Write(0);
579  Pin_2a_Y_Write(0);
580  Pin_1b_Y_Write(1);
581  Pin_2b_Y_Write(0);
582  CyDelay(stepDelay);
583
584  Pin_1a_Y_Write(0);
585  Pin_2a_Y_Write(1);
586  Pin_1b_Y_Write(0);
587  Pin_2b_Y_Write(0);
588  CyDelay(stepDelay);
589
590  Pin_1a_Y_Write(1);
591  Pin_2a_Y_Write(0);
592  Pin_1b_Y_Write(0);
593  Pin_2b_Y_Write(0);
594  CyDelay(stepDelay);
595 }

```

Her er kalder-grafen for denne funktion:



3.6.2.10 void stepYForwards (void) [private]

Køre Y motor et step frem.

Køre Y motoren et step fremad.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 535 i filen xy.c.

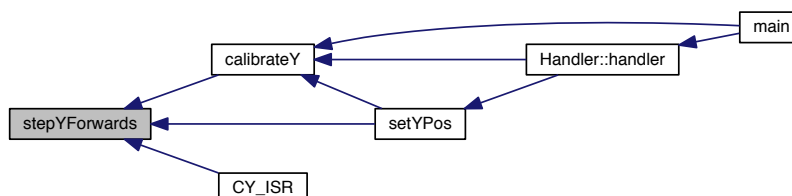
Indeholder referencer til stepDelay.

Refereret til af calibrateY(), CY_ISR() og setYPos().

```

536 {
537     Pin_1a_Y_Write(1);
538     Pin_2a_Y_Write(0);
539     Pin_1b_Y_Write(0);
540     Pin_2b_Y_Write(0);
541     CyDelay(stepDelay);
542
543     Pin_1a_Y_Write(0);
544     Pin_2a_Y_Write(1);
545     Pin_1b_Y_Write(0);
546     Pin_2b_Y_Write(0);
547     CyDelay(stepDelay);
548
549     Pin_1a_Y_Write(0);
550     Pin_2a_Y_Write(0);
551     Pin_1b_Y_Write(1);
552     Pin_2b_Y_Write(0);
553     CyDelay(stepDelay);
554
555     Pin_1a_Y_Write(0);
556     Pin_2a_Y_Write(0);
557     Pin_1b_Y_Write(0);
558     Pin_2b_Y_Write(1);
559     CyDelay(stepDelay);
560 }
  
```

Her er kalder-grafen for denne funktion:



3.6.2.11 void xy_init (void)

Initialiser XY modulet.

Initialiser XY modulets interrupt.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Defineret på linje 35 i filen xy.c.

Refereret til af main().

```

36 {
37     interrupt_X_StartEx(isr_X);
38     interrupt_Y_StartEx(isr_Y);
39 }
  
```

Her er kalder-grafen for denne funktion:



3.6.2.12 void xy_start (void)

Starter XY modulet.

Starter XY modulet, og køre til position 0,0

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

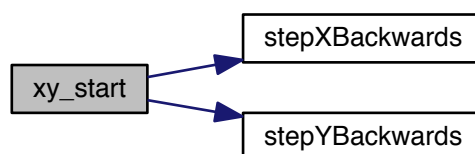
Defineret på linje 49 i filen xy.c.

Indeholder referencer til dataXY, DataXY::interruptX, DataXY::interruptY, stepXBackwards(), stepYBackwards(), DataXY::xFlag og DataXY::yFlag.

Refereret til af main().

```
50 {  
51     DEBUG_PutString("X initializing going to zero");  
52  
53     while(dataXY.interruptX == 0 && dataXY.xFlag == 0)  
54     {  
55         DEBUG_PutString(".");  
56         stepXBackwards();  
57     }  
58     DEBUG_PutString("done");  
59     DEBUG_PutCRLF();  
60  
61     dataXY.interruptX = 0;  
62  
63     DEBUG_PutString("Y initializing going to zero");  
64  
65     while(dataXY.interruptY == 0 && dataXY.yFlag == 0)  
66     {  
67         DEBUG_PutString(".");  
68         stepYBackwards();  
69     }  
70     DEBUG_PutString("done");  
71     DEBUG_PutCRLF();  
72  
73     dataXY.interruptY = 0;  
74 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



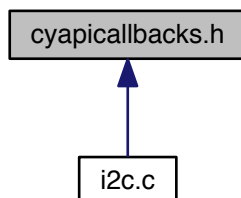
Dokumentationen for denne klasse blev genereret ud fra filerne:

- [xy.h](#)
- [xy.c](#)

4 Fil-dokumentation

4.1 cyapicallbacks.h filreference

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



#Defines

- `#define I2CS_I2C_ISR_EXIT_CALLBACK`

Funktioner

- `void I2CS_I2C_ISR_ExitCallback (void)`

4.1.1 #Define-dokumentation

4.1.1.1 #define I2CS_I2C_ISR_EXIT_CALLBACK

Defineret på linje 15 i filen `cyapicallbacks.h`.

4.1.2 Funktions-dokumentation

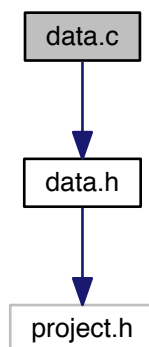
4.1.2.1 void I2CS_I2C_ISR_ExitCallback (void)

4.2 data.c filreference

Data modul.

```
#include "data.h"
```

Inklusions-afhængighedsgraf for data.c:



4.2.1 Detaljeret beskrivelse

Data modul.

Indeholder data vedr. XY modulet.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

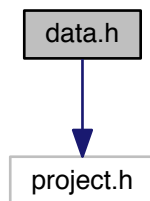
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.3 data.h filreference

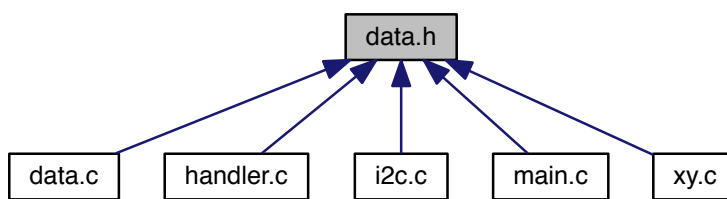
Data modul.


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

Inklusions-afhængighedsgraf for data.h:



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



Datastrukturer

- struct [DataXY](#)

Funktioner

- void [data_init](#) (void)

Variable

- struct [DataXY](#) [dataXY](#)

4.3.1 Detaljeret beskrivelse

[Data](#) modul.

Indeholder data vedr. [XY](#) modulet.

Forfatter

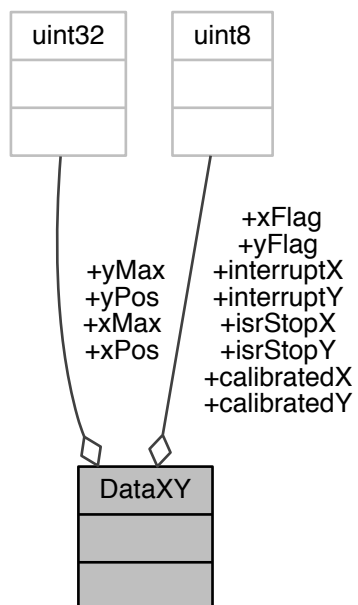
Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.3.2 Datastruktur-documentation

4.3.2.1 struct DataXY

Defineret på linje 32 i filen data.h.

Samarbejdsdiagram for DataXY:



Data-felter

uint8	calibratedX	
uint8	calibratedY	
uint8	interruptX	
uint8	interruptY	
uint8	isrStopX	
uint8	isrStopY	
uint8	xFlag	
uint32	xMax	
uint32	xPos	
uint8	yFlag	
uint32	yMax	
uint32	yPos	

4.3.3 Funktions-dokumentation

4.3.3.1 void data_init (void)

4.3.4 Variabel-dokumentation

4.3.4.1 struct DataXY dataXY

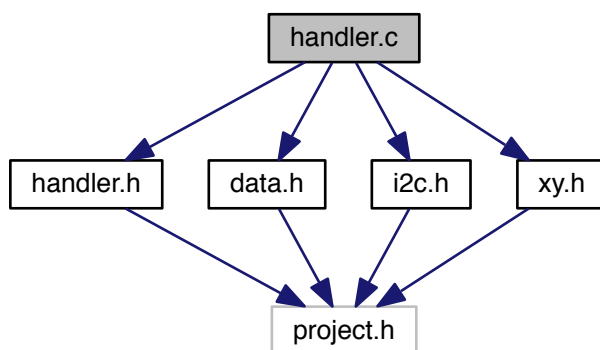
Refereret til af XY::calibrateX(), XY::calibrateY(), XY::CY_ISR(), Data::data_init(), Handler::handler(), I2C::I2CS_↔ I2C_ISR_ExitCallback(), XY::setXPos(), XY::setYPos() og XY::xy_start().

4.4 handler.c filreference

Handler modul.

```
#include "handler.h"
#include "data.h"
#include "i2c.h"
#include "xy.h"
```

Inklusions-afhængighedsgraf for handler.c:



4.4.1 Detaljeret beskrivelse

Handler modul.

Håndtere indkommende kommandoer med tilhørende værdier.

Forfatter

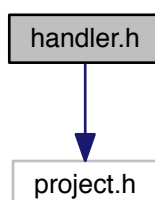
Casper Dieu Le (201370338@uni.au.dk)
 Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
 Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.5 handler.h filreference

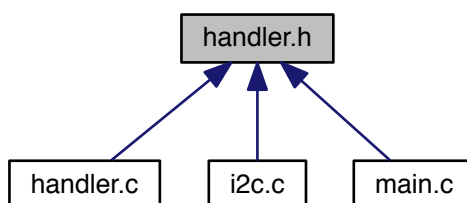
Handler modul.

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

Inklusions-afhængighedsgraf for handler.h:



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



#Defines

- #define `CMD_SET_X_POS` (0x10u)
- #define `CMD_SET_Y_POS` (0x11u)
- #define `CMD_GET_X_POS` (0x12u)
- #define `CMD_GET_Y_POS` (0x13u)
- #define `CMD_X_STP` (0x16u)
- #define `CMD_Y_STP` (0x17u)
- #define `CMD_X_CAL` (0x18u)
- #define `CMD_Y_CAL` (0x19u)

Funktioner

- void `handler` (uint8 cmd, uint8 val)

4.5.1 Detaljeret beskrivelse

[Handler](#) modul.

Håndtere indkommende kommandoer med tilhørende værdier.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.5.2 #Define-dokumentation

4.5.2.1 #define CMD_GET_X_POS (0x12u)

Defineret på linje 41 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.2 #define CMD_GET_Y_POS (0x13u)

Defineret på linje 42 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.3 #define CMD_SET_X_POS (0x10u)

Defineret på linje 39 i filen handler.h.

Refereret til af Handler::handler() og I2C::I2CS_I2C_ISR_ExitCallback().

4.5.2.4 #define CMD_SET_Y_POS (0x11u)

Defineret på linje 40 i filen handler.h.

Refereret til af Handler::handler() og I2C::I2CS_I2C_ISR_ExitCallback().

4.5.2.5 #define CMD_X_CAL (0x18u)

Defineret på linje 45 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.6 #define CMD_X_STP (0x16u)

Defineret på linje 43 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.7 #define CMD_Y_CAL (0x19u)

Defineret på linje 46 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.8 #define CMD_Y_STP (0x17u)

Defineret på linje 44 i filen handler.h.

Refereret til af Handler::handler().

4.5.3 Funktions-dokumentation

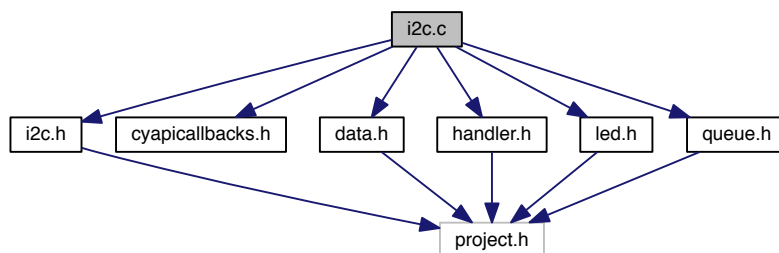
4.5.3.1 void handler (uint8 cmd, uint8 val)

4.6 i2c.c filreference

I2C modul.

```
#include "i2c.h"  
#include "cyapicallbacks.h"  
#include "data.h"  
#include "handler.h"  
#include "led.h"  
#include "queue.h"
```

Inklusions-afhængighedsgraf for i2c.c:



4.6.1 Detaljeret beskrivelse

I2C modul.

Håndter kommunikation via I2C-busset

Forfatter

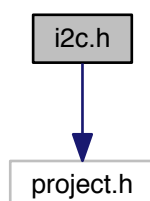
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.7 i2c.h filreference

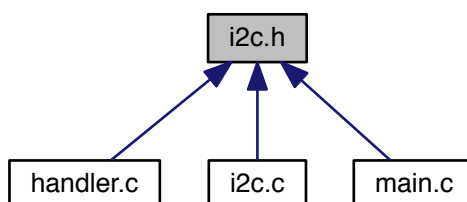
I2C modul.

```
#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 (4u)
- #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_tx (void)

4.7.1 Detaljeret beskrivelse

I2C modul.

Håndter kommunikation via I2C-busset.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.7.2 #Define-dokumentation

4.7.2.1 #define I2C_BUFFER_SIZE (4u)

Defineret på linje 36 i filen i2c.h.

Refereret til af I2C::i2c_init() og I2C::I2CS_I2C_ISR_ExitCallback().

4.7.2.2 #define I2C_PACKET_CMD_POS (1u)

Defineret på linje 41 i filen i2c.h.

Refereret til af Handler::handler() og I2C::I2CS_I2C_ISR_ExitCallback().

4.7.2.3 #define I2C_PACKET_EOP (0xEFu)

Defineret på linje 47 i filen i2c.h.

4.7.2.4 #define I2C_PACKET_EOP_POS (3u)

Defineret på linje 43 i filen i2c.h.

4.7.2.5 #define I2C_PACKET_SIZE (4u)

Defineret på linje 37 i filen i2c.h.

4.7.2.6 #define I2C_PACKET_SOP (0xBEu)

Defineret på linje 46 i filen i2c.h.

4.7.2.7 #define I2C_PACKET_SOP_POS (0u)

Defineret på linje 40 i filen i2c.h.

4.7.2.8 #define I2C_PACKET_VAL_POS (2u)

Defineret på linje 42 i filen i2c.h.

Refereret til af Handler::handler() og I2C::I2CS_I2C_ISR_ExitCallback().

4.7.2.9 #define I2C_STS_CMD_DONE (0xAAu)

Defineret på linje 50 i filen i2c.h.

4.7.2.10 #define I2C_STS_CMD_FAIL (0xEEu)

Defineret på linje 51 i filen i2c.h.

4.7.3 Funktions-dokumentation

4.7.3.1 void i2c_init (void)

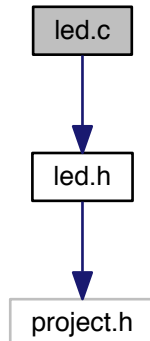
4.7.3.2 void i2c_tx (void)

4.8 led.c filreference

LED modul.

```
#include "led.h"
```

Inklusions-afhængighedsgraf for led.c:



4.8.1 Detaljeret beskrivelse

LED modul.

Håndtere PSoC'ens røde, grønne og blå led.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

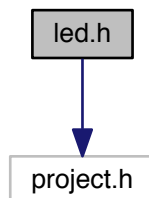
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.9 led.h filreference

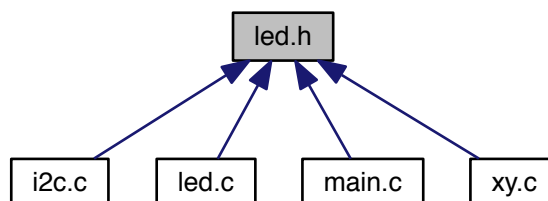
LED modul.

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

Inklusions-afhængighedsgraf for led.h:



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



#Defines

- #define LED_ON (0u)
- #define LED_OFF (1u)

Funktioner

- void setLed (uint8 red, uint8 green, uint8 blue, uint8 delay)

4.9.1 Detaljeret beskrivelse

LED modul.

Håndtere PSoC'ens røde, grønne og blå led.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.9.2 #Define-dokumentation

4.9.2.1 #define LED_OFF (1u)

Defineret på linje 41 i filen led.h.

Refereret til af LED::setLed().

4.9.2.2 #define LED_ON (0u)

Defineret på linje 40 i filen led.h.

Refereret til af LED::setLed().

4.9.3 Funktions-dokumentation

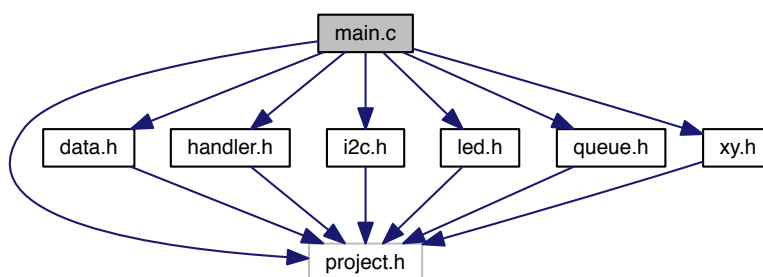
4.9.3.1 void setLed (uint8 *red*, uint8 *green*, uint8 *blue*, uint8 *delay*)

4.10 main.c filreference

Hovedprogram.

```
#include <project.h>
#include "data.h"
#include "handler.h"
#include "i2c.h"
#include "led.h"
#include "queue.h"
#include "xy.h"
```

Inklusions-afhængighedsgraf for main.c:



Funktioner

- int [main](#) ()

4.10.1 Detaljeret beskrivelse

Hovedprogram.

Intilize modulerne og køre derefter i loop hvor der bliver kontrollet om der er nogle actions i køen der skal håndteres af handleren.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk (201271201@uni.au.dk)

4.10.2 Funktions-dokumentation

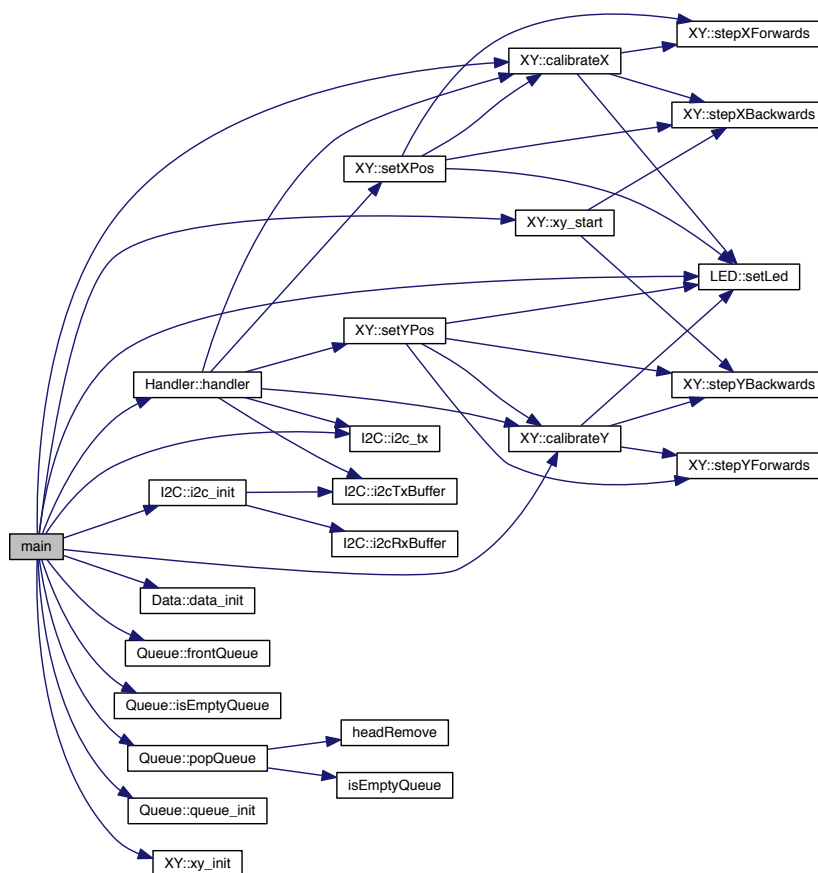
4.10.2.1 int main ()

Defineret på linje 17 i filen main.c.

Indeholder referencer til XY::calibrateX(), XY::calibrateY(), Data::data_init(), Queue::frontQueue(), Handler↵::handler(), I2C::i2c_init(), I2C::i2c_tx(), Queue::isEmptyQueue(), Queue::popQueue(), Queue::queue_init(), LED↵::setLed(), XY::xy_init() og XY::xy_start().

```
18 {
19     CyGlobalIntEnable;
20
21     data_init();
22     queue_init(6u);
23     xy_init();
24     i2c_init();
25
26     DEBUG_PutCRLF();
27     DEBUG_PutString("==== Initializing PSoC XY =====");
28     DEBUG_PutCRLF();
29
30     setLed(0,1,0,0);
31     CyDelay(100);
32     setLed(0,0,0,0);
33
34     xy_start();
35
36     for(;;)
37     {
38         if(SW2_Read() == 0u)
39         {
40             CyDelay(5u);
41             if(SW2_Read() == 0u)
42             {
43                 calibrateX();
44                 calibrateY();
45             }
46             while(SW2_Read() == 0u)
47             {
48                 /* Wait till button released */
49             }
50         }
51
52         while(!isEmptyQueue())
53         {
54             struct Action action;
55             action = frontQueue();
56             handler(action.cmd, action.val);
57             popQueue();
58         }
59         i2c_tx();
60     }
61 }
```

Her er kald-grafen for denne funktion:



4.11 queue.c filreference

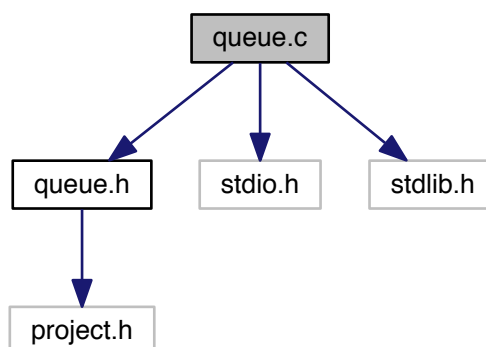
Queue modul.

```

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

```

Inklusions-afhængighedsgraf for queue.c:



Datastrukturer

- struct [Node](#)
Node struct. Mere...

Funktioner

- static void [headInsert](#) (struct [Node](#) **headPtr, const struct [Action](#) data)
- static void [headRemove](#) (struct [Node](#) **headPtr)
- static void [backInsert](#) (struct [Node](#) **backPtr, const struct [Action](#) data)

4.11.1 Detaljeret beskrivelse

[Queue](#) modul.

En FIFO kø der er opbygget af en single linket liste.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.11.2 Datastruktur-documentation

4.11.2.1 struct Node

[Node](#) struct.

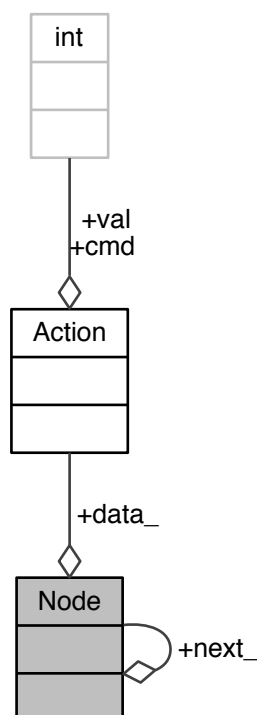
En struct til at oprette et element der kan indsættes i køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 20 i filen queue.c.

Samarbejdsdiagram for Node:



Data-felter

struct Action	data↔	Data til køen
struct Node *	next↔	Pointer til næste node i køen

4.11.3 Funktions-dokumentation

4.11.3.1 static void backInsert (struct **Node** ** *backPtr*, const struct **Action** *data*) [static]

Refereret til af Queue::pushQueue().

Her er kalder-grafen for denne funktion:



4.11.3.2 `static void headInsert (struct Node ** headPtr, const struct Action data) [static]`

Refereret til af `Queue::pushQueue()`.

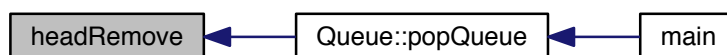
Her er kalder-grafen for denne funktion:



4.11.3.3 `static void headRemove (struct Node ** headPtr) [static]`

Refereret til af `Queue::popQueue()`.

Her er kalder-grafen for denne funktion:

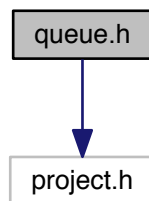


4.12 queue.h filreference

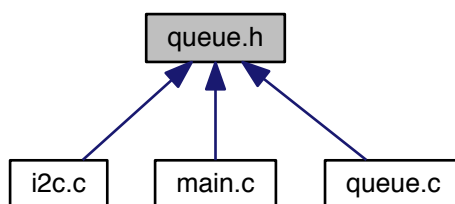
[Queue](#) modul.


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

Inklusions-afhængighedsgraf for queue.h:



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



Datastrukturer

- struct [Action](#)
Action struct. Mere...

Funktioner

- void [queue_init](#) (uint8 queueMaxSize)
- void [pushQueue](#) (const struct [Action](#) data)
- void [popQueue](#) (void)
- struct [Action](#) [frontQueue](#) (void)
- uint8 [isEmptyQueue](#) (void)

4.12.1 Detaljeret beskrivelse

[Queue](#) modul.

En FIFO kø der er opbygget af en single linket liste.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.12.2 Datastruktur-documentation

4.12.2.1 struct Action

Action struct.

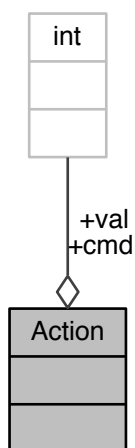
Structen kan indeholde en kommando og tilhørende værdi, som kan indsættes i FIFO køen.

Forfatter

Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 33 i filen queue.h.

Samarbejdsdiagram for Action:



Data-felter

int	cmd	Kommando
int	val	Værdi

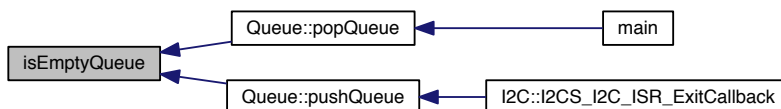
4.12.3 Funktions-dokumentation

4.12.3.1 struct Action frontQueue (void)

4.12.3.2 uint8 isEmptyQueue (void)

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

Her er kalder-grafen for denne funktion:



4.12.3.3 void popQueue (void)

4.12.3.4 void pushQueue (const struct Action data)

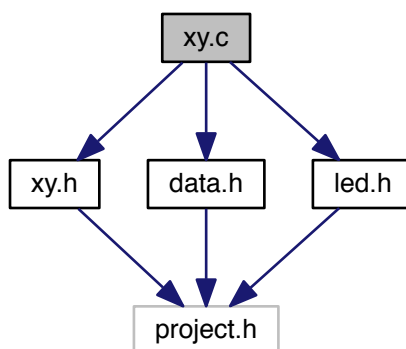
4.12.3.5 void queue_init (uint8 queueMaxSize)

4.13 xy.c filreference

XY modul.

```
#include "xy.h"
#include "data.h"
#include "led.h"
```

Inklusions-afhængighedsgraf for xy.c:



Funktioner

- static void [stepXForwards](#) (void)
- static void [stepXBackwards](#) (void)
- static void [stepYForwards](#) (void)
- static void [stepYBackwards](#) (void)

4.13.1 Detaljeret beskrivelse

XY modul.

Styre XY modulets funktioner.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)

Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

4.13.2 Funktions-dokumentation

4.13.2.1 `static void stepXBackwards (void) [static]`

4.13.2.2 `static void stepXForwards (void) [static]`

4.13.2.3 `static void stepYBackwards (void) [static]`

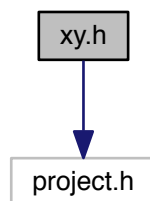
4.13.2.4 `static void stepYForwards (void) [static]`

4.14 xy.h filreference

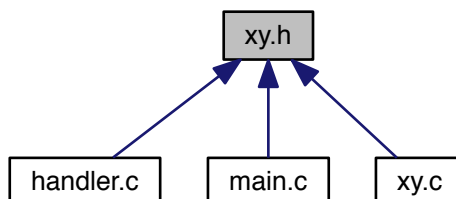
XY modul.

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

Inklusions-afhængighedsgraf for xy.h:



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



#Defines

- #define `stepDelay` (3u)
- #define `interruptSteps` (50u)
- #define `resolution` (255u)

Funktioner

- void `xy_init` (void)
- void `xy_start` (void)
- `CY_ISR_PROTO` (isr_X)
- `CY_ISR_PROTO` (isr_Y)
- void `calibrateX` (void)
- void `calibrateY` (void)
- void `setXPos` (uint8 xVal)
- void `setYPos` (uint8 yVal)

4.14.1 Detaljeret beskrivelse

`XY` modul.

Styre `XY` modulets funktioner.

Forfatter

Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)

4.14.2 #Define-dokumentation

4.14.2.1 #define interruptSteps (50u)

Defineret på linje 47 i filen `xy.h`.

Refereret til af `XY::calibrateX()`, `XY::calibrateY()` og `XY::CY_ISR()`.

4.14.2.2 #define resolution (255u)

Defineret på linje 48 i filen `xy.h`.

Refereret til af `Handler::handler()`, `XY::setXPos()` og `XY::setYPos()`.

4.14.2.3 #define stepDelay (3u)

Defineret på linje 46 i filen `xy.h`.

Refereret til af `XY::stepXBackwards()`, `XY::stepXForwards()`, `XY::stepYBackwards()` og `XY::stepYForwards()`.

4.14.3 Funktions-dokumentation

4.14.3.1 void calibrateX (void)

4.14.3.2 void calibrateY (void)

4.14.3.3 CY_ISR_PROTO (isr_X)

4.14.3.4 CY_ISR_PROTO (isr_Y)

4.14.3.5 void setXPos (uint8 xVal)

4.14.3.6 void setYPos (uint8 yVal)

4.14.3.7 void xy_init (void)

4.14.3.8 void xy_start (void)

