L.A.M.P - PSoC Z

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:

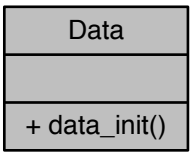
3 Datastruktur-documentation

3.1 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

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 dataZ structen med start værdier.

Forfatter

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

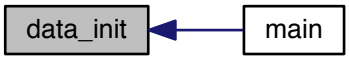
Defineret på linje 21 i filen data.c.

Indeholder referencer til DataZ::calibratedZ, dataZ, DataZ::interruptZ, DataZ::isrStopZ, DataZ::zFlag, DataZ::zMax og DataZ::zPos.

Refereret til af main().

```
22 {
23   dataZ.calibratedZ = 1;
24   dataZ.interruptZ = 0;
25   dataZ.isrStopZ = 0;
26   dataZ.zFlag = 0;
27   dataZ.zMax = 3000;
28   dataZ.zPos = 0;
29 }
```

Her er kalder-grafen for denne funktion:



Dokumentationen for denne klasse blev genereret ud fra filerne:

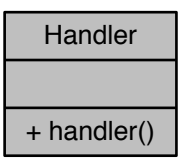
- data.h
- data.c

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

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	*cmd*	Er den modtaget kommando.
in	*val*	Er den tilhørende værdi.

Forfatter

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

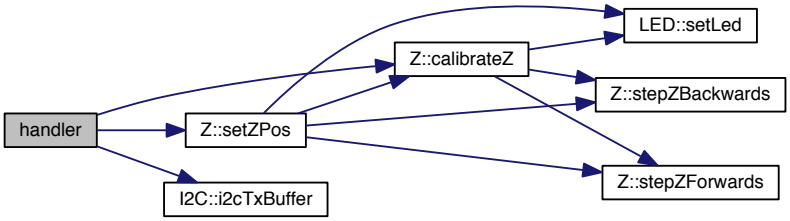
Defineret på linje 26 i filen handler.c.

Indeholder referencer til Z::calibrateZ(), CMD_GET_Z_POS, CMD_SET_Z_POS, CMD_Z_CAL, CMD_Z_STP, dataZ, I2C_PACKET_CMD_POS, I2C_PACKET_VAL_POS, I2C::i2cTxBuffer(), DataZ::isrStopZ, resolution, Z::setZPos(), DataZ::zMax og DataZ::zPos.

Refereret til af main().

```
27 {
28   switch (cmd) {
29     case CMD_SET_Z_POS :
30       setZPos(val);
31       break;
32     case CMD_GET_Z_POS :
33       i2cTxBuffer[I2C_PACKET_CMD_POS] = cmd;
34       i2cTxBuffer[I2C_PACKET_VAL_POS] = (uint8)((resolution *
     dataZ.zPos) / dataZ.zMax + 1);;
35       break;
36     case CMD_Z_STP :
37       dataZ.isrStopZ = 1;
38       break;
39     case CMD_Z_CAL :
40       calibrateZ();
41       break;
42     default :
43       break;
44   }
45 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:

Dokumentationen for denne klasse blev genereret ud fra filerne:

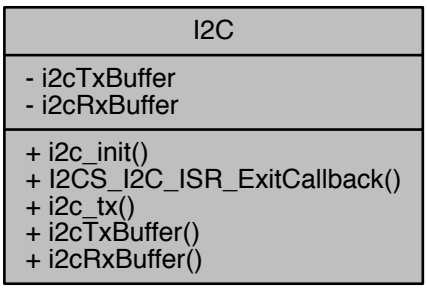
- handler.h
- handler.c

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 ()

  Motager "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)

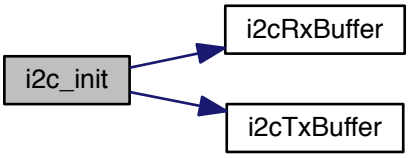
Defineret på linje 49 i filen i2c.c.

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

Refereret til af main().

```
50 {
51   I2CS_I2CSlaveInitReadBuf(i2cTxBuffer, I2C_BUFFER_SIZE);
52   I2CS_I2CSlaveClearReadBuf();
53   I2CS_I2CSlaveClearReadStatus();
54
55   I2CS_I2CSlaveInitWriteBuf(i2cRxBuffer, I2C_BUFFER_SIZE);
56   I2CS_I2CSlaveClearWriteBuf();
57   I2CS_I2CSlaveClearWriteStatus();
58
59   I2CS_Start();
60 }
```

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 115 i filen i2c.c.

Refereret til af main().

```
116 {
117   if(0u != (I2CS_I2CSlaveStatus() & I2CS_I2C_SSTAT_RD_CMPLT))
118   {
119     I2CS_I2CSlaveClearReadBuf();
120     (void) I2CS_I2CSlaveClearReadStatus();
121   }
122 }
```

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.

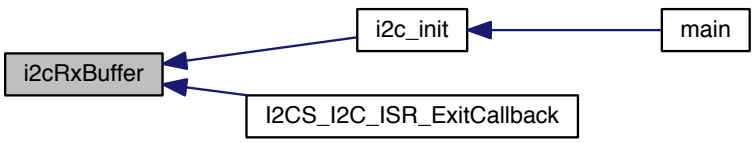
Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 76 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

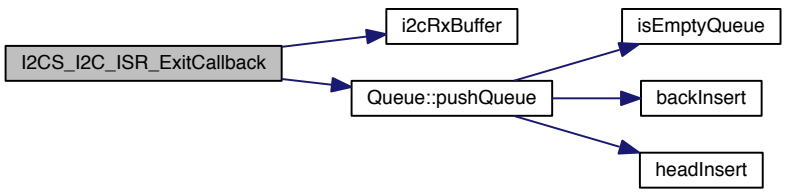
Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 69 i filen i2c.c.

Indeholder referencer til Action::cmd, CMD_SET_Z_POS, dataZ, I2C_BUFFER_SIZE, I2C_PACKET_CMD_POS, I2C_PACKET_VAL_POS, i2cRxBuffer(), DataZ::isrStopZ, Queue::pushQueue() og Action::val.

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

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.

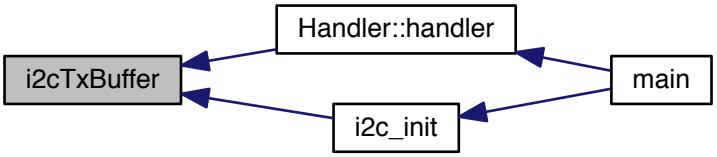
Forfatter

Jeppe Stærk (201271201@uni.au.dk)

Defineret på linje 67 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 35 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 26 i filen i2c.c.

Dokumentationen for denne klasse blev genereret ud fra filerne:

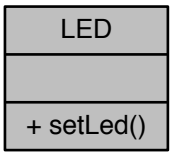
- i2c.h
- i2c.c

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

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

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

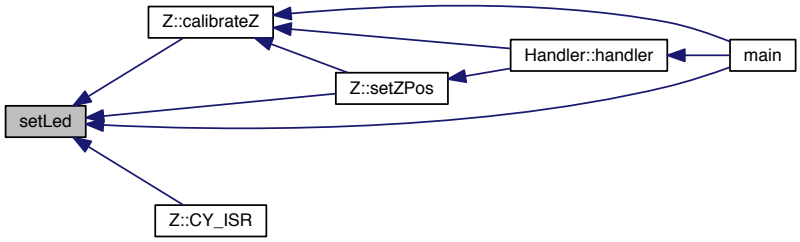
Defineret på linje 25 i filen led.c.

Indeholder referencer til LED_OFF og LED_ON.

Refereret til af Z::calibrateZ(), Z::CY_ISR(), main() og Z::setZPos().

```
26 {
27   red ? LED_RED_Write(LED_ON) : LED_RED_Write(LED_OFF);
28   green ? LED_GREEN_Write(LED_ON) : LED_GREEN_Write(LED_OFF);
29   blue ? LED_BLUE_Write(LED_ON) : LED_BLUE_Write(LED_OFF);
30
31   CyDelay(delay);
32 }
```

Her er kalder-grafen for denne funktion:



Dokumentationen for denne klasse blev genereret ud fra filerne:

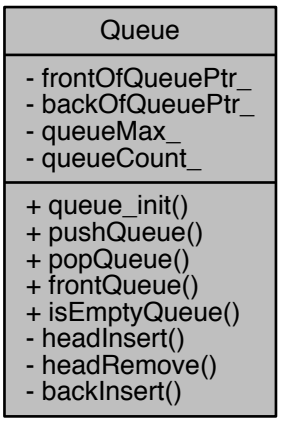
- led.h
- led.c

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 linket 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	*backPtr*	Pointer til det bagerste element i listen.
in	*data*	Data der skal indsættes i listen.

Forfatter

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

Defineret på linje 248 i filen queue.c.

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

```
249 {
250   if(*backPtr == NULL)
251   {
252     return;
253   }
254
255   struct Node* next = (*backPtr)->next_;
256   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
257   temp->data_ = data;
258   temp->next_ = next;
259   (*backPtr)->next_ = temp;
260 }
```

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)

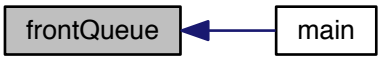
Defineret på linje 170 i filen queue.c.

Indeholder referencer til Node::data_.

Refereret til af main().

```
171 {
172   DEBUG_PutString("Q=: count: ");
173   DEBUG_PutHexByte(queueCount_);
174   DEBUG_PutCRLF();
175   return frontOfQueuePtr_->data_;
176 }
```

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 206 i filen queue.c.

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

```
207 {
208   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
209   if(temp == NULL)
210   {
211     return;
212   }
213
214   temp->data_ = data;
215   temp->next_ = NULL;
216
217   *headPtr = temp;
218 }
```

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 228 i filen queue.c.

Indeholder referencer til Node::next_.

```
229 {
230   if(headPtr != NULL)
231   {
232     struct Node* condemned;
233     condemned = *headPtr;
234     *headPtr = (*headPtr)->next_;
235     free(condemned);
236   }
237 }
```

3.5.2.5 uint8 isEmptyQueue (void)

Retuner status af køen.

Kontrollere om køen er tom.

Forfatter

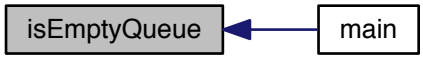
Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 185 i filen queue.c.

Refereret til af main().

```
186 {
187   if(frontOfQueuePtr_ == NULL)
188   {
189     return 1;
190   }
191   else
192   {
193     return 0;
194   }
195 }
```

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)

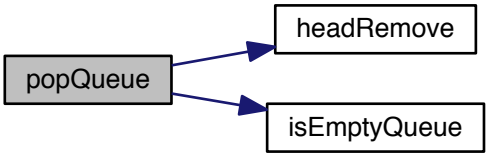
Defineret på linje 149 i filen queue.c.

Indeholder referencer til headRemove() og isEmptyQueue().

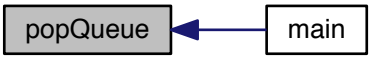
Refereret til af main().

```
150 {
151   headRemove(&frontOfQueuePtr_);
152   queueCount_--;
153   if(isEmptyQueue() == 1)
154   {
155     backOfQueuePtr_ = NULL;
156   }
157   DEBUG_PutString("-Q: count: ");
158   DEBUG_PutHexByte(queueCount_);
159   DEBUG_PutCRLF();
160   DEBUG_PutCRLF();
161 }
```

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)

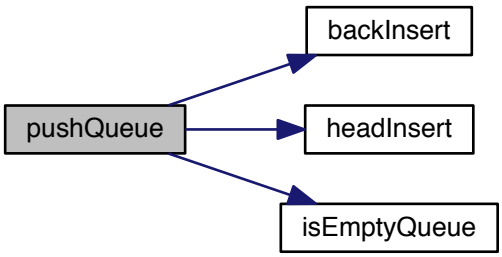
Defineret på linje 107 i filen queue.c.

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

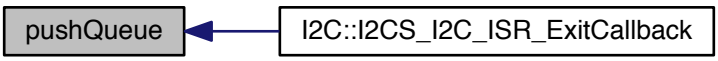
Refereret til af I2C::I2CS_I2C_ISR_ExitCallback().

```
108 {
109   if(queueCount_<queueMax_)
110   {
111     if(isEmptyQueue() != 1)
112     {
113       backInsert(&backOfQueuePtr_, data);
114       backOfQueuePtr_ = backOfQueuePtr_->next_;
115       queueCount_++;
116     }
117     else
118     {
119       headInsert(&frontOfQueuePtr_, data);
120       backOfQueuePtr_ = frontOfQueuePtr_;
121       queueCount_++;
122     }
123     DEBUG_PutString("Q+: count: ");
124     DEBUG_PutHexByte(queueCount_);
125     DEBUG_PutString(" cmd: ");
126     DEBUG_PutHexByte(data.cmd);
127     DEBUG_PutString(" val: ");
128     DEBUG_PutHexByte(data.val);
129     DEBUG_PutCRLF();
130     DEBUG_PutCRLF();
131   }
132   else
133   {
134     DEBUG_PutString("Q~: ERROR! Queue FULL!!! count: ");
135     DEBUG_PutHexByte(queueCount_);
136     DEBUG_PutCRLF();
137     DEBUG_PutCRLF();
138   }
139
140 }
```

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 størelse.

Forfatter

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

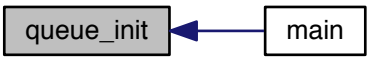
Defineret på linje 89 i filen queue.c.

Indeholder referencer til Node::next_.

Refereret til af main().

```
90 {
91   frontOfQueuePtr_ = NULL;
92   frontOfQueuePtr_->next_ = NULL;
93   backOfQueuePtr_ = NULL;
94   backOfQueuePtr_->next_ = NULL;
95   queueMax_ = queueMaxSize;
96   queueCount_ = 0;
97 }
```

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 48 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 39 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 66 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 57 i filen queue.c.

Dokumentationen for denne klasse blev genereret ud fra filerne:

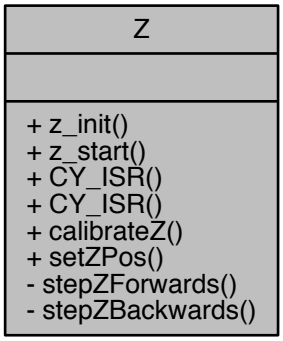
- queue.h
- queue.c

3.6 Z Klasse-reference

Z class.

```
#include <z.h>
```

Samarbejdsdiagram for Z:



Offentlige metoder

- void z_init ()

  Initialiser Z modulet.
- void z_start ()

  Starter Z modulet.
- CY_ISR (isr_Z)

  Afvikler "Interrupt" fra Z.
- CY_ISR (isr_S)

  Afvikler "Interrupt" fra Z.
- void calibrateZ ()

  Kalibrere Z.
- void setZPos (uint8 zVal)

  Sætter ny Z position.

Private metoder

- void stepZForwards ()

  Køre Z motor et step frem.
- void stepZBackwards ()

  Køre Z motor et step tilbage.

3.6.1 Detaljeret beskrivelse

Z class.

Styre Z modulets funktioner.

Forfatter

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

3.6.2 Dokumentation af medlemsfunktioner

3.6.2.1 void calibrateZ (void)

Kalibrere Z.

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

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 137 i filen z.c.

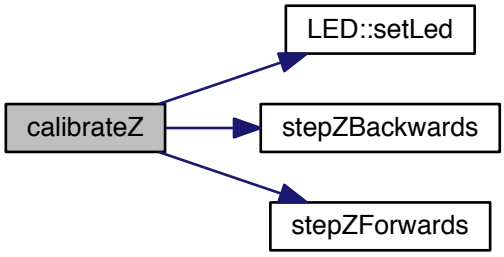
Indeholder referencer til DataZ::calibratedZ, dataZ, interruptSteps, DataZ::interruptZ, LED::setLed(), stepZ↩Backwards(), stepZForwards(), DataZ::zFlag, DataZ::zMax og DataZ::zPos.

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

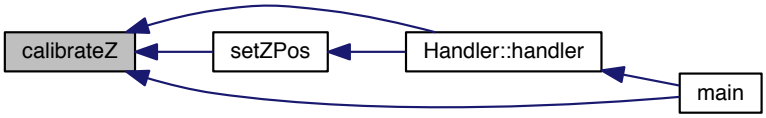
```
138 {
139   DEBUG_PutString("Z calibrate pre-zMax: ");
140   DEBUG_PutHexInt(dataZ.zMax);
141   DEBUG_PutCRLF();
142 
143   dataZ.calibratedZ = 0;
144   dataZ.zFlag = 1;
145   dataZ.zMax = 0;
146 
147   DEBUG_PutString("Going forwards to max");
148   while(dataZ.interruptZ == 0 && dataZ.zFlag == 1)
149   {
150     DEBUG_PutString(".");
151     setLed(1,0,0,0);
152     stepZForwards();
153   }
154   dataZ.interruptZ = 0;
155 
156   DEBUG_PutString("done");
157   DEBUG_PutCRLF();
158 
159   DEBUG_PutString("Going backwards to zero");
160   while(dataZ.interruptZ == 0 && dataZ.zFlag == 0)
161   {
162     DEBUG_PutString(".");
163     setLed(1,0,0,0);
164     stepZBackwards();
165     dataZ.zMax++;
166   }
167   DEBUG_PutString("done");
168 
169   setLed(0,0,0,0);
```

```
170
171   dataZ.zPos = 0;
172   dataZ.zMax = dataZ.zMax - interruptSteps;
173
174   DEBUG_PutString(" post-zMax: ");
175   DEBUG_PutHexInt(dataZ.zMax);
176   DEBUG_PutString(" new xPos: ");
177   DEBUG_PutHexInt(dataZ.zPos);
178   DEBUG_PutCRLF();
179   DEBUG_PutCRLF();
180
181   dataZ.calibratedZ = 1;
182   dataZ.interruptZ = 0;
183 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.6.2.2 CY_ISR (isr_Z)

Afvikler "Interrupt" fra Z.

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

Forfatter

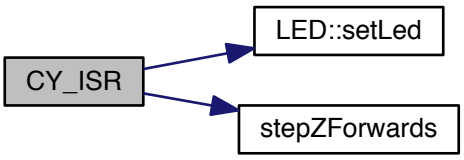
Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 75 i filen z.c.

Indeholder referencer til dataZ, interruptSteps, DataZ::interruptZ, LED::setLed(), stepZForwards() og DataZ::zFlag.

```
76 {
77   DEBUG_PutString("Interrupt Z");
78   DEBUG_PutCRLF();
79   interrupt_Z_Disable();
80 
81   uint32 i;
82 
83   dataZ.interruptZ = 1;
84 
85   setLed(0,0,1,0);
86   for(i = 0; i < interruptSteps; i++)
87   {
88     stepZForwards();
89   }
90   dataZ.zFlag = 1;
91   setLed(0,0,0,0);
92 
93   interrupt_Z_ClearPending();
94   interrupt_Z_Enable();
95 }
```

Her er kald-grafen for denne funktion:



3.6.2.3 CY_ISR (isr_S)

Afvikler "Interrupt" fra Z.

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

Forfatter

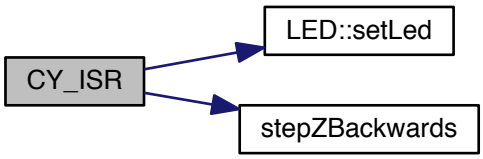
Casper Dieu Le (201370338@uni.au.dk)
Kasper Hinkler Uldbjerg (201370281@uni.au.dk)
Jeppe Stærk Antonsen (201271201@uni.au.dk)

Defineret på linje 106 i filen z.c.

Indeholder referencer til dataZ, interruptSteps, DataZ::interruptZ, LED::setLed(), stepZBackwards() og DataZ::z↩
Flag.

```
107 {
108   DEBUG_PutString("Interrupt S");
109   DEBUG_PutCRLF();
110   interrupt_S_Disable();
111
112   uint32 i;
113
114   dataZ.interruptZ = 1;
115
116   setLed(0,0,1,0);
117   for(i = 0; i < interruptSteps; i++)
118   {
119     stepZBackwards();
120   }
121   dataZ.zFlag = 0;
122   setLed(0,0,0,0);
123
124   interrupt_S_ClearPending();
125   interrupt_S_Enable();
126 }
```

Her er kald-grafen for denne funktion:



3.6.2.4 void setZPos (uint8 *zVal*)

Sætter ny Z position.

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

Parametre

in	*zVal*	Værdi for position.

Forfatter

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

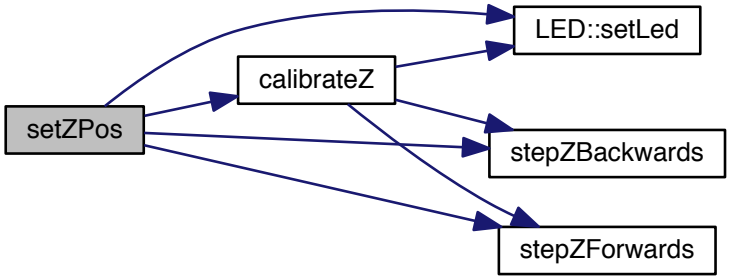
Defineret på linje 195 i filen z.c.

Indeholder referencer til DataZ::calibratedZ, calibrateZ(), dataZ, DataZ::interruptZ, DataZ::isrStopZ, resolution, L↩ ED::setLed(), stepZBackwards(), stepZForwards(), DataZ::zFlag, DataZ::zMax og DataZ::zPos.

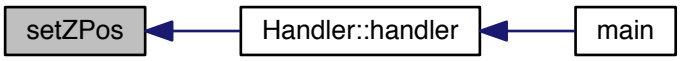
Refereret til af Handler::handler().

```
196 {
197   uint32 i;
198   uint32 zDes = 0;
199   uint32 zSteps = 0;
200 
201   dataZ.isrStopZ = 0;
202 
203   if(dataZ.calibratedZ == 1)
204   {
205     zDes = zVal * dataZ.zMax / resolution;
206 
207     DEBUG_PutString("Z set value: ");
208     DEBUG_PutHexInt(zVal);
209     DEBUG_PutString(" pre-zPos: ");
210     DEBUG_PutHexInt(dataZ.zPos);
211     DEBUG_PutString(" zDes: ");
212     DEBUG_PutHexInt(zDes);
213 
214     if(zDes > dataZ.zPos)
215     {
216       setLed(0,1,0,0);
217       dataZ.interruptZ = 0;
218       dataZ.zFlag = 1;
219       zSteps = zDes - dataZ.zPos;
220 
221       DEBUG_PutString(" going forwards steps: ");
222       DEBUG_PutHexInt(zSteps);
223       for(i = 0; i < zSteps  && dataZ.isrStopZ == 0 && dataZ.
    interruptZ == 0 && dataZ.zFlag == 1; i++)
224       {
225         DEBUG_PutString(".");
226         stepZForwards();
227         dataZ.zPos++;
228       }
229       DEBUG_PutString("done");
230 
231       if(dataZ.interruptZ == 1)
232       {
233         dataZ.zPos = dataZ.zMax;
234       }
235       DEBUG_PutString(" new-zPos: ");
236       DEBUG_PutHexInt(dataZ.zPos);
237       DEBUG_PutCRLF();
238       DEBUG_PutCRLF();
239 
240       setLed(0,0,0,0);
241     }
242     else if(zDes < dataZ.zPos)
243     {
244       setLed(0,1,0,0);
245 
246       dataZ.interruptZ = 0;
247       dataZ.zFlag = 0;
248       zSteps = dataZ.zPos - zDes;
249 
250       DEBUG_PutString(" going backwards steps: ");
251       DEBUG_PutHexInt(zSteps);
252       for(i = 0; i < zSteps && dataZ.isrStopZ == 0 && dataZ.
    interruptZ == 0 && dataZ.zFlag == 0; i++)
253       {
254         DEBUG_PutString(".");
255         stepZBackwards();
256         dataZ.zPos--;
257       }
258       DEBUG_PutString("done");
259       if(dataZ.interruptZ == 1)
260       {
261         dataZ.zPos = 0;
262       }
263       DEBUG_PutString(" new-zPos: ");
264       DEBUG_PutHexInt(dataZ.zPos);
265       DEBUG_PutCRLF();
266       DEBUG_PutCRLF();
267 
268       setLed(0,0,0,0);
269     }
270   }
271   else
272   {
273     calibrateZ();
274     setZPos(zVal);
275   }
276   dataZ.interruptZ = 0;
277 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:



3.6.2.5 void stepZBackwards (void) [private]

Køre Z motor et step tilbage.

Køre Z motoren et step tilbage.

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 329 i filen z.c.

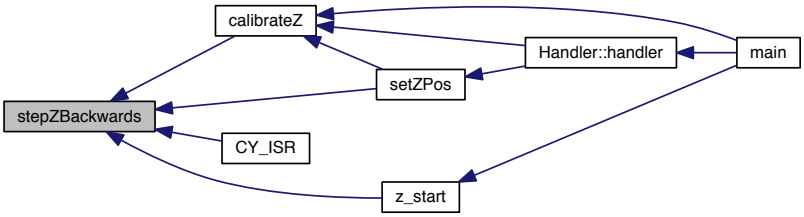
Indeholder referencer til stepDelay.

Refereret til af calibrateZ(), CY_ISR(), setZPos() og z_start().

```
330 {
331   Pin_1a_Z_Write(0);
332   Pin_2a_Z_Write(0);
333   Pin_1b_Z_Write(0);
334   Pin_2b_Z_Write(1);
335   CyDelay(stepDelay);
336 
337   Pin_1a_Z_Write(0);
```

```
338   Pin_2a_Z_Write(0);
339   Pin_1b_Z_Write(1);
340   Pin_2b_Z_Write(0);
341   CyDelay(stepDelay);
342
343   Pin_1a_Z_Write(0);
344   Pin_2a_Z_Write(1);
345   Pin_1b_Z_Write(0);
346   Pin_2b_Z_Write(0);
347   CyDelay(stepDelay);
348
349   Pin_1a_Z_Write(1);
350   Pin_2a_Z_Write(0);
351   Pin_1b_Z_Write(0);
352   Pin_2b_Z_Write(0);
353   CyDelay(stepDelay);
354 }
```

Her er kalder-grafen for denne funktion:



3.6.2.6 void stepZForwards (void) [private]

Køre Z motor et step frem.

Køre Z motoren et step fremad.

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 293 i filen z.c.

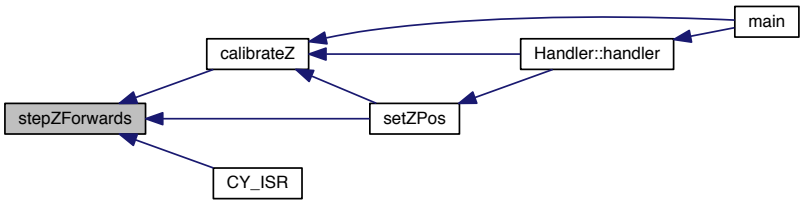
Indeholder referencer til stepDelay.

Refereret til af calibrateZ(), CY_ISR() og setZPos().

```
294 {
295   Pin_1a_Z_Write(1);
296   Pin_2a_Z_Write(0);
297   Pin_1b_Z_Write(0);
298   Pin_2b_Z_Write(0);
299   CyDelay(stepDelay);
300
301   Pin_1a_Z_Write(0);
302   Pin_2a_Z_Write(1);
303   Pin_1b_Z_Write(0);
304   Pin_2b_Z_Write(0);
305   CyDelay(stepDelay);
306
307   Pin_1a_Z_Write(0);
```

```
308   Pin_2a_Z_Write(0);
309   Pin_1b_Z_Write(1);
310   Pin_2b_Z_Write(0);
311   CyDelay(stepDelay);
312
313   Pin_1a_Z_Write(0);
314   Pin_2a_Z_Write(0);
315   Pin_1b_Z_Write(0);
316   Pin_2b_Z_Write(1);
317   CyDelay(stepDelay);
318 }
```

Her er kalder-grafen for denne funktion:



3.6.2.7 void z_init (void)

Initialiser Z modulet.

Initialiser Z modulets interrupt.

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 35 i filen z.c.

Refereret til af main().

```
36 {
37   interrupt_Z_StartEx(isr_Z);
38   interrupt_S_StartEx(isr_S);
39 }
```

Her er kalder-grafen for denne funktion:

3.6.2.8 void z_start (void)

Starter Z modulet.

Starter Z modulet, og køre til position 0

Forfatter

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

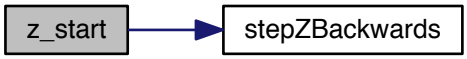
Defineret på linje 50 i filen z.c.

Indeholder referencer til dataZ, DataZ::interruptZ, stepZBackwards() og DataZ::zFlag.

Refereret til af main().

```
51 {
52   dataZ.interruptZ = 0;
53   DEBUG_PutString("Z initializing going to zero");
54
55   while(dataZ.interruptZ == 0 && dataZ.zFlag == 0)
56   {
57     DEBUG_PutString(".");
58     stepZBackwards();
59   }
60   DEBUG_PutString("done");
61   DEBUG_PutCRLF();
62
63   dataZ.interruptZ = 0;
64 }
```

Her er kald-grafen for denne funktion:



Her er kalder-grafen for denne funktion:

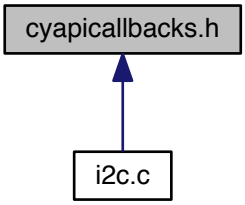
Dokumentationen for denne klasse blev genereret ud fra filerne:

- z.h
- z.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.

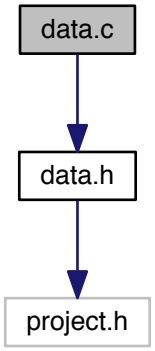
4.1.2 Funktions-dokumentation

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. Z modulet.

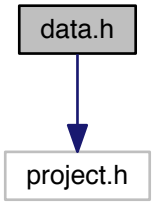
Forfatter

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

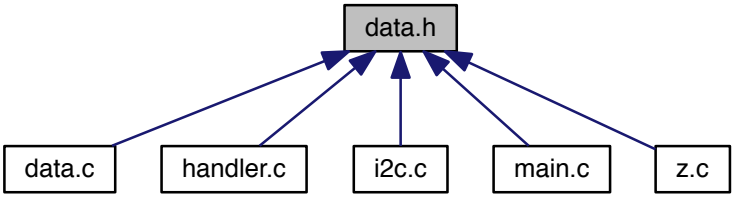
4.3 data.h filreference

Data modul.

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



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



Datastrukturer

- struct DataZ

Funktioner

- void data_init (void)

Variable

- struct DataZ dataZ

4.3.1 Detaljeret beskrivelse

Data modul.

Indeholder data vedr. Z modulet.

Forfatter

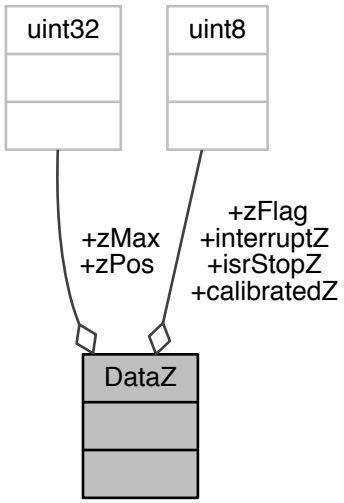
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.3.2 Datastruktur-documentation

4.3.2.1 struct DataZ

Defineret på linje 29 i filen data.h.

Samarbejdsdiagram for DataZ:



Data-felter

uint8	calibratedZ	
uint8	interruptZ	
uint8	isrStopZ	
uint8	zFlag	
uint32	zMax	
uint32	zPos	

4.3.3 Funktions-dokumentation

4.3.3.1 void data_init (void)

4.3.4 Variabel-dokumentation

4.3.4.1 struct DataZ dataZ

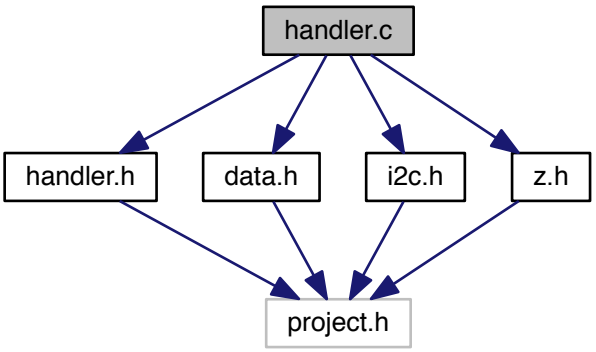
Refereret til af Z::calibrateZ(), Z::CY_ISR(), Data::data_init(), Handler::handler(), I2C::I2CS_I2C_ISR_Exit↩ Callback(), Z::setZPos() og Z::z_start().

4.4 handler.c filreference

Handler modul.

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

Inklusions-afhængighedsgraf for handler.c:



4.4.1 Detaljeret beskrivelse

Handler modul.

Håndtere indkommende kommandoer med tilhørende værdier.

Forfatter

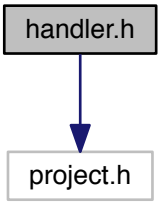
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.5 handler.h filreference

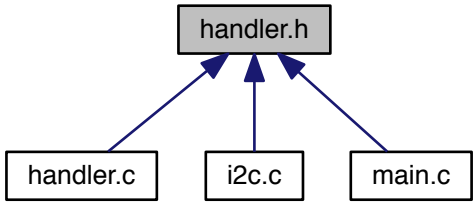
Handler modul.

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

Inklusions-afhængighedsgraf for handler.h:



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



#Defines

- #define CMD_SET_Z_POS (0x20u)
- #define CMD_GET_Z_POS (0x21u)
- #define CMD_Z_STP (0x23u)
- #define CMD_Z_CAL (0x24u)

Funktioner

- void handler (uint8 cmd, uint8 val)

4.5.1 Detaljeret beskrivelse

Handler modul.

Håndtere indkommende kommandoer med tilhørende værdier.

Forfatter

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

4.5.2 #Define-dokumentation

4.5.2.1 #define CMD_GET_Z_POS (0x21u)

Defineret på linje 38 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.2 #define CMD_SET_Z_POS (0x20u)

Defineret på linje 37 i filen handler.h.

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

4.5.2.3 #define CMD_Z_CAL (0x24u)

Defineret på linje 40 i filen handler.h.

Refereret til af Handler::handler().

4.5.2.4 #define CMD_Z_STP (0x23u)

Defineret på linje 39 i filen handler.h.

Refereret til af Handler::handler().

4.5.3 Funktions-dokumentation

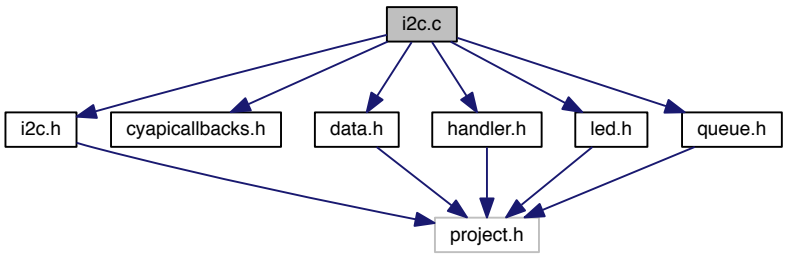
4.5.3.1 void handler (uint8 *cmd,* uint8 *val*)

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

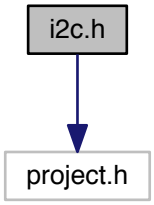
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.7 i2c.h filreference

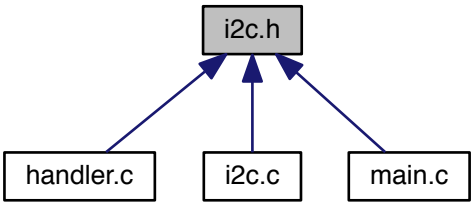
I2C modul.

```
#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 38 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 43 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 49 i filen i2c.h.

4.7.2.4 #define I2C_PACKET_EOP_POS (3u)

Defineret på linje 45 i filen i2c.h.

4.7.2.5 #define I2C_PACKET_SIZE (4u)

Defineret på linje 39 i filen i2c.h.

4.7.2.6 #define I2C_PACKET_SOP (0xBEu)

Defineret på linje 48 i filen i2c.h.

4.7.2.7 #define I2C_PACKET_SOP_POS (0u)

Defineret på linje 42 i filen i2c.h.

4.7.2.8 #define I2C_PACKET_VAL_POS (2u)

Defineret på linje 44 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 52 i filen i2c.h.

4.7.2.10 #define I2C_STS_CMD_FAIL (0xEEu)

Defineret på linje 53 i filen i2c.h.

4.7.3 Funktions-dokumentation

4.7.3.1 void i2c_init (void)

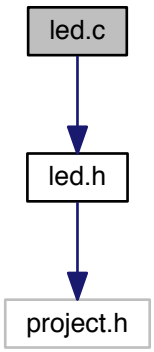
4.7.3.2 void i2c_tx (void)

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

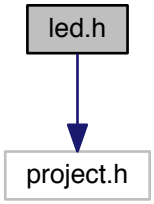
Jeppe Stærk Antonsen (201271201@uni.au.dk)

4.9 led.h filreference

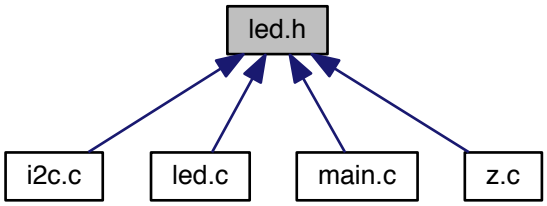
LED modul.

```
#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

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

4.9.2 #Define-dokumentation

4.9.2.1 #define LED_OFF (1u)

Defineret på linje 38 i filen led.h.

Refereret til af LED::setLed().

4.9.2.2 #define LED_ON (0u)

Defineret på linje 37 i filen led.h.

Refereret til af LED::setLed().

4.9.3 Funktions-dokumentation

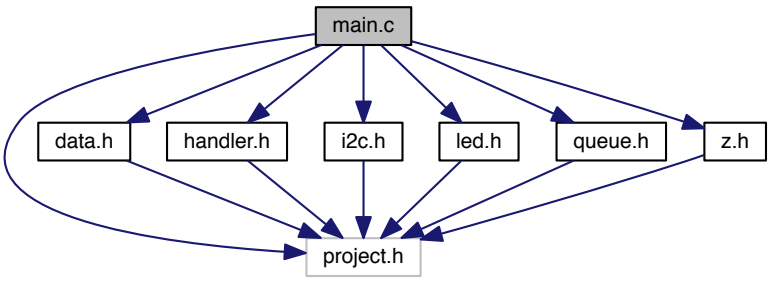
4.9.3.1 void setLed (uint8 *red,* uint8 *green,* uint8 *blue,* uint8 *delay*)

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 "z.h"
```

Inklusions-afhængighedsgraf for main.c:



Funktioner

- int main ()

4.10.1 Detaljeret beskrivelse

Hovedprogram.

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

Forfatter

Jeppe Stærk (201271201@uni.au.dk)

4.10.2 Funktions-dokumentation

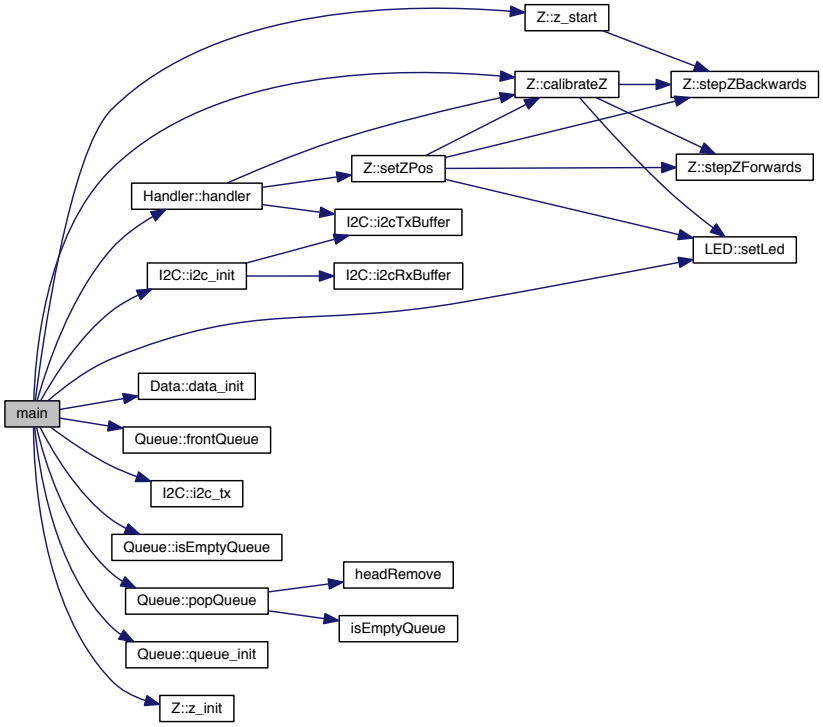
4.10.2.1 int main ( )

Defineret på linje 15 i filen main.c.

Indeholder referencer til Z::calibrateZ(), Data::data_init(), Queue::frontQueue(), Handler::handler(), I2C::i2c_init(), I2C::i2c_tx(), Queue::isEmptyQueue(), Queue::popQueue(), Queue::queue_init(), LED::setLed(), Z::z_init() og Z↩ ::z_start().

```
16 {
17   CyGlobalIntEnable;
18
19   data_init();
20   queue_init(6u);
21   z_init();
22   i2c_init();
23
24   DEBUG_PutCRLF();
25   DEBUG_PutString("===== Initializing PSoC Z =====");
26   DEBUG_PutCRLF();
27
28   setLed(0,1,0,0);
29   CyDelay(100);
30   setLed(0,0,0,0);
31
32   z_start();
33
34   for(;;)
35   {
36     if(SW2_Read() == 0u)
37     {
38       CyDelay(5u);
39       if(SW2_Read() == 0u)
40       {
41         calibrateZ();
42       }
43       while(SW2_Read() == 0u)
44       {
45         ; /* Wait till button released */
46       }
47     }
48
49     while(isEmptyQueue() != 1)
50     {
51       struct Action action;
52       action = frontQueue();
53       handler(action.cmd, action.val);
54       popQueue();
55     }
56     i2c_tx();
57   }
58 }
```

Her er kald-grafen for denne funktion:

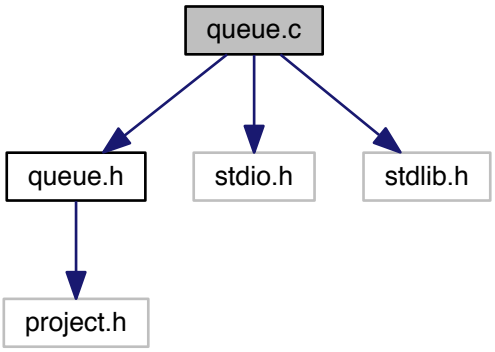


4.11 queue.c filreference

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.

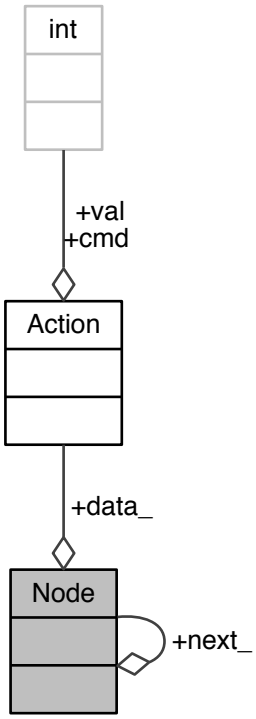
En struct til at oprette et element der kan indsættes i køen.

Forfatter

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

Defineret på linje 21 i filen queue.c.

Samarbejdsdiagram for Node:



Data-felter

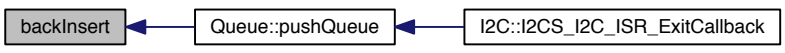
struct Action	data↩ _	Data til køen
struct Node ∗	next↩ _	Pointer til næste node i køen

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]

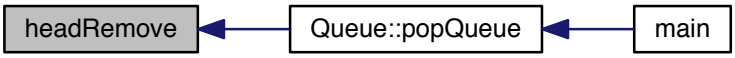
Refereret til af Queue::pushQueue().

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:

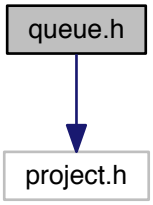


4.12 queue.h filreference

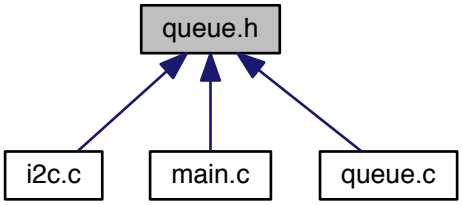
Queue modul.

```
#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.

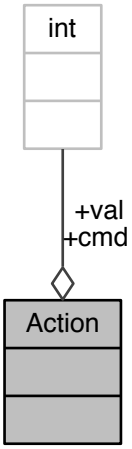
Structen kan indeholde en kommando og tilhørende værdi, som kan indsættes i FIFO køen.

Forfatter

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

Defineret på linje 34 i filen queue.h.

Samarbejdsdiagram for Action:



Data-felter

int	cmd	Kommando
int	val	Værdi

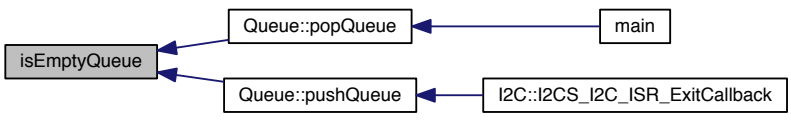
4.12.3 Funktions-dokumentation

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*)

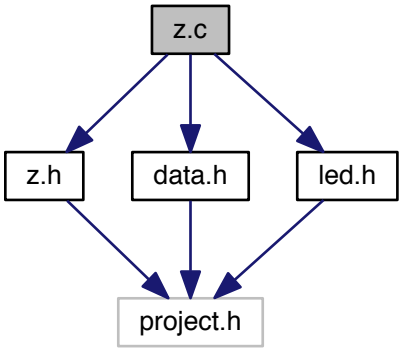
4.12.3.5 void queue_init (uint8 *queueMaxSize*)

4.13 z.c filreference

Z modul.

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

Inklusions-afhængighedsgraf for z.c:



Funktioner

- static void stepZForwards (void)
- static void stepZBackwards (void)

4.13.1 Detaljeret beskrivelse

Z modul.

Styre Z modulets funktioner.

Forfatter

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

4.13.2 Funktions-dokumentation

4.13.2.1 static void stepZBackwards (void) [static]

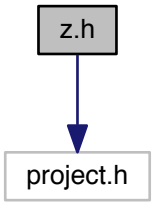
4.13.2.2 static void stepZForwards (void) [static]

4.14 z.h filreference

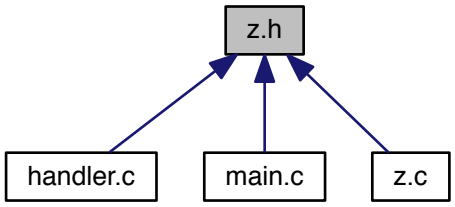
Z modul.

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

Inklusions-afhængighedsgraf for z.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 z_init (void)
- void z_start (void)
- CY_ISR_PROTO (isr_Z)
- CY_ISR_PROTO (isr_S)
- void calibrateZ (void)
- void setZPos (uint8 zVal)

4.14.1 Detaljeret beskrivelse

Z modul.

Styre Z modulets funktioner.

Forfatter

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

4.14.2 #Define-dokumentation

4.14.2.1 #define interruptSteps (50u)

Defineret på linje 47 i filen z.h.

Refereret til af Z::calibrateZ() og Z::CY_ISR().

4.14.2.2 #define resolution (255u)

Defineret på linje 48 i filen z.h.

Refereret til af Handler::handler() og Z::setZPos().

4.14.2.3 #define stepDelay (3u)

Defineret på linje 46 i filen z.h.

Refereret til af Z::stepZBackwards() og Z::stepZForwards().

4.14.3 Funktions-dokumentation

4.14.3.1 void calibrateZ (void)

4.14.3.2 CY_ISR_PROTO (isr_Z)

4.14.3.3 CY_ISR_PROTO (isr_S)

4.14.3.4 void setZPos (uint8 *zVal*)

4.14.3.5 void z_init (void)

4.14.3.6 void z_start (void)