

**nohandcom Nachschlagewerk**  
**0.0.1**

**Erzeugt von Doxygen 1.2.13.1**

**Thu Jun 26 11:51:25 2003**

## Inhaltsverzeichnis

1	nohandcom Datenstruktur-Verzeichnis	1
2	nohandcom Klassen-Dokumentation	1

## 1 nohandcom Datenstruktur-Verzeichnis

### 1.1 nohandcom Übersicht

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

ContentObject	1
DeviceListener	3
Frame	3
Gui	5

## 2 nohandcom Klassen-Dokumentation

### 2.1 ContentObject Klassenreferenz

```
#include <contentobject.h>
```

Oentliche Datenelemente

```
ContentObject ()  
ContentObject (bool isText, bool isGraphic, char text, char graphic,  
char x1, char y1, char x2, char y2)  
ContentObject ()  
bool getTextMode ()  
bool getGraphicMode ()  
char getText ()  
char getGraphic ()  
char getX1 ()  
char getY1 ()  
char getX2 ()  
char getY2 ()  
void setTextMode (bool mode)  
void setGraphicMode (bool mode)  
void setText (char text)  
void setGraphic (char graphic)
```

```
void setX1 (char x1)
void setY1 (char y1)
void setX2 (char x2)
void setY2 (char y2)
```

### Private Attribute

```
int m_actionNumber
    in actionNumber wird die mit dem Objekt assoziierte Aktion gespeichert.

bool m_isText
bool m_isGraphic
char  m_text
char  m_graphic
char m_x1
    in den Attributen x1, y1, x2, y2 wird die Position des Objekts gespeichert.

char m_y1
char m_x2
char m_y2
```

### Statische private Attribute

```
int m_actionCount
```

#### 2.1.1 Ausführliche Beschreibung

Ein ContentObject-Objekt trägt alle Informationen wie Form und Position eines anzeigbaren und mit einer Aktion verknüpften Inhalts. Es wird keine Methode benötigt

#### 2.1.2 Dokumentation der Datenelemente

##### 2.1.2.1 int ContentObject::m\_actionCount **[static, private]**

Die Klassenvariable m\_actionCount zählt die Anzahl der Objekte, die von dieser Klasse instanziiert werden. Beim Hinzufügen einer Aktion kann dieser Wert mit der entsprechenden Aktion verknüpft und in eine Tabelle eingetragen werden

##### 2.1.2.2 char ContentObject::m\_graphic **[private]**

graphic enthält das Bild, das innerhalb des Frames angezeigt werden soll. Es ist auch ein String, da das Format für die Übertragung gleich ist wie für Text. Ob es sich um eine Graphik handelt, wird im Display mittels eines Befehls eingestellt

**2.1.2.3 bool ContentObject::m\_isGraphic [private]**

isgraphic zeigt an, ob das Objekt ein Graphisches ist. Sind sowohl isText wie auch isGraphic gesetzt, bedeutet dies eine OR-Verknüpfung des Text- und Graphikmodus auf dem Display

**2.1.2.4 bool ContentObject::m\_isText [private]**

in isText wird angezeigt, ob es sich bei diesem Objekt um ein Text-basiertes Objekt handelt

**2.1.2.5 char ContentObject::m\_text [private]**

text halt einen String, der an der angegebenen Position im Frame (S. 3) angezeigt wird

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

```
contentobject.h
contentobject.cc
```

**2.2 DeviceListener Klassenreferenz**

```
#include <devicelistener.h>
```

**2.2.1 Ausführliche Beschreibung**

DeviceListener ist die Schnittstelle zur Aktion des users. Ein Objekt dieser Klasse holt den parallelen Port nach einer Aktion am GPIO ab und liest allfallige kurze Spikes an den Pins aus, indem der Port zweimal ausgelesen wird (Im Falle eines massiven Anstiegs der Anzahl Prozesse auf dem System musste dieses Feature gestrichen werden)

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

```
devicelistener.h
```

**2.3 Frame Klassenreferenz**

```
#include <frame.h>
```

**O entliche Datenelemente**

```
Frame ()
Frame (ContentObject objects, int objectCount)
Frame ()
int addObject (bool isText, bool isGraphic, char text, char graphic,
int x1, int y1, int x2, int y2)
```

```
int getObjectCount ()
ContentObject getActiveObject ()
ContentObject getObjects ()
void setObjectCount (int number)
void setActiveObject (int number)
```

#### Private Attribute

```
int m_frameCount
ContentObject m_object
int m_objectCount
```

*Dieser Zahler halt die Anzahl der Objekte im Frame.*

```
ContentObject m_activeObject
```

*Dieser Zeiger zeigt auf das aktuelle und hervorgehobene ContentObjekt.*

#### 2.3.1 Ausführliche Beschreibung

Die Frame Klasse enthält den Inhalt in Form von ContentObject (S.1)-Objekten. Ein Objekt dieser Klasse halt den Inhalt eines gesamten Fensters. Die verschiedenen Frames werden von einem Object der Gui (S.5)-Klasse instanziiert und verwaltet

#### 2.3.2 Dokumentation der Elementfunktionen

**2.3.2.1** int Frame::addObject (bool *isText*, bool *isGraphic*, char *text*, char *graphic*, int *x1*, int *y2*, int *x2*, int *y2*)

Die Methode addObject() (S. 4) fugt dem Frame ein neues Objekt hinzu. Wenn das Objekt in die Liste aufgenommen werden konnte, gibt die Methode true zuruck.

#### 2.3.3 Dokumentation der Datenelemente

**2.3.3.1** int Frame::m\_frameCount **[private]**

Dieses Attribut zahlt die insgesamt vorhandenen Frames.

**2.3.3.2** ContentObject Frame::m\_object **[private]**

Dieses Attribut ist ein Zeiger auf ein Array von ContentObject (S.1)-Objekten, die den Inhalt des Frames bilden. Die ContentObjekte werden zur Laufzeit dynamisch alloziert

Die Dokumentation fur diese Klasse wurde erzeugt aufgrund der Dateien:

```
frame.h
frame.cc
```

## 2.4 Gui Klassenreferenz

**#include <gui.h>**

Oentliche Datenelemente

```
Gui (int frames)
  Gui ()
  bool writeObject (ContentObject content, int size)
  bool init ()
  void run ()
  const ContentObject & scan (Frame frame)
  bool writeScan (char x1, char y1, char x2, char y2)
  void simulate (int timeout)
  void writeBase ()
  void clear (void)
    clear () (S. 6) loscht das Display.

  void lightOn (void)
    lightOn () (S. 5) schaltet die Hintergrundbeleuchtung an.

  void lightO (void)
    lightO schaltet die Hintergrundbeleuchtung aus.

  void displayOn (void)
    displayOn schaltet das Display an.

  void displayO (void)
    displayO schaltet das Display aus.

  void invert (void)
    invert () (S. 5) invertiert die Screen.

  void invertPart (char x1, char y1, char x2, char y2)
  void showLogo (char text, int size)
    showLogo () (S. 5) schreibt ein Text als Logo in die Mitte Oben.

  void writeSquare (char x1, char y1, char x2, char y2)
  void setCursor (char spalte, char zeile)
    setCursor () (S. 5) setzt den Cursor auf die Koordinate (spalte, zeile).

  void setFont (char size)
  void drawLine (char x1, char y1, char x2, char y2)
  void setZoom (char f, char n1, char n2)
```

### Private Attribute

Frame m\_frame  
*mit m\_frames werden die verschiedenen Fenster eingebunden.*

DeviceListener m\_listener  
*Der m\_listener holt den Switch am parallelen Port ab.*

ContentObject m\_contentObject  
char m\_text  
*GsmModule m\_gsm;.*

int m\_frameCount  
Frame m\_activeFrame  
ContentObject m\_activeObject  
bool m\_portFree  
int m\_counter

#### 2.4.1 Ausführliche Beschreibung

Diese Klasse ist zentral für die Verwaltung des Displays und verwaltet die verschiedenen Objekte der anderen Klassen. In einer ersten Implementation verwaltet sie auch die Schnittstelle zu den Ports

#### 2.4.2 Beschreibung der Konstruktoren und Destruktoren

##### 2.4.2.1 Gui::Gui (int *frames*)

Dies ist der Standardkonstruktor des Managerobjekts. Ihm werden als Parameter die Anzahl zu verwaltenden Fenster übergeben.

Parameter:

*frames* Die Anzahl der zu kreierenden Fenster

##### 2.4.2.2 Gui:: Gui ()

Dies ist der Destruktor des Managerobjekts. er deletet die entsprechenden Datenstrukturen

#### 2.4.3 Dokumentation der Elementfunktionen

##### 2.4.3.1 void Gui::clear (void)

clear() (S. 6) löscht das Display.

Die folgenden Methoden sind nötig, um das Display zu steuern

#### 2.4.3.2 void Gui::drawLine (char *x1*, char *y1*, char *x2*, char *y2*)

drawLine zieht eine Linie von *x1*, *y1* nach *x2*, *y2*

#### 2.4.3.3 bool Gui::init ()

Die Methode `init()` (S. 7) übernimmt die Initialisierung des Displays und instanziert die erforderlichen Objekte. Bei erfolgreicher Initialisierung gibt sie `true` zurück

#### 2.4.3.4 void Gui::invertPart (char *x1*, char *y1*, char *x2*, char *y2*)

`invertPart()` (S. 7) invertiert einen Bereich des Bildschirms

Parameter:

*x1* mit diesen Parametern wird die Position übergeben

#### 2.4.3.5 void Gui::run ()

Die Methode `run()` (S. 7) ist der zentrale Loop der Anwendung. Sie initialisiert zuerst die Ports und holt diese darauf nach Aktivität ab

#### 2.4.3.6 const ContentObject & Gui::scan (Frame *frame*)

Diese Methode scannt durch den Inhalt des jeweils aktiven Frames und gibt einen Pointer auf die Struktur zurück, die das Inhaltsobjekt beschreibt

Parameter:

*frame* Mit diesem Parameter wird das aktive Frame (S. 3) und damit sein Content übergeben

#### 2.4.3.7 void Gui::setFont (char *size*)

Die Methode `setFont()` (S. 7) setzt den Font der Textausgabe

Parameter:

*size* Setzt die Größe der Schrift

#### 2.4.3.8 void Gui::setZoom (char *f*, char *n1*, char *n2*)

`setZoom()` (S. 7) zoomt den Font *f* um den Faktor *n1* (Breite) und *n2* (Höhe)

#### 2.4.3.9 void Gui::simulate (int *timeout*)

Die Methode `simulate` aktiviert diverse Aktivitäten im Display. Sie ist v.a. als Testmethode gedacht

Parameter:

*timeout* halt die Länge der Simulationszeit in Loops

#### 2.4.3.10 void Gui::writeBase ()

Die Methode writeBase() (S. 8) zeichnet soviele Reiter an die Basis, wie es Frames hat

#### 2.4.3.11 bool Gui::writeObject (ContentObject *content*, int *size*)

Die Methode writeObject() (S. 8) schreibt einen Frame (S. 3) auf das Display und gibt bei erfolgreichem Transfer true zuruck. Diese Methode hat neben writeScan() (S. 8) den Zugri auf den seriellen Port. um ihn zu synchronisieren wird das Flag portFree von beiden Methoden genutzt.

Parameter:

*content* Im Contentobjekt sind alle zu schreibenden Informationen enthalten.

#### 2.4.3.12 bool Gui::writeScan (char *x1*, char *y1*, char *x2*, char *y2*)

Die Methode scan() (S. 7) invertiert den entsprechenden Bereich des Displays. Sie hat Zugri auf den seriellen Port

Parameter:

*x1*, *y1*, *x2*, *y2* Die Koordinaten des zu invertierenden Bereichs Sie gibt bei Erfolg true zuruck

#### 2.4.3.13 void Gui::writeSquare (char *x1*, char *y1*, char *x2*, char *y2*)

writeSquare() (S. 8) schreibt einen Rahmen in das Display

Parameter:

*x1* mit diesen Parametern wird die Position uebergeben

### 2.4.4 Dokumentation der Datenelemente

#### 2.4.4.1 Frame Gui::m\_activeFrame **[private]**

m\_activeFrame ist ein Zeiger auf den Frame (S. 3), der gerade aktiv und im Display sichtbar ist

#### 2.4.4.2 ContentObject Gui::m\_activeObject **[private]**

m\_activeObject zeigt auf das aktive gerade gescannte Objekt

#### 2.4.4.3 ContentObject Gui::m\_contentObject **[private]**

In m\_contentObject wird die ins Display zu schreibende Information kopiert, bevor sie ausgegeben wird

**2.4.4.4 int Gui::m\_counter [private]**

mit m\_counter hat gui ein Zahler fur die Scannerdurchgange

**2.4.4.5 int Gui::m\_frameCount [private]**

m\_frameCount halt sie Anzahl der instanziierten Frames

**2.4.4.6 bool Gui::m\_portFree [private]**

portFree regelt den Zugri auf den Seriellen Port, da er von zwei Methoden writeScan() (S. 8) und writeFrame() benutzt wird

**2.4.4.7 char Gui::m\_text [private]**

GsmModule m.gsm;

in m\_text konnen beliebige Stringobjekte aus den Applikationen (SMS etc) kopiert werden

Die Dokumentation fur diese Klasse wurde erzeugt aufgrund der Dateien:

gui.h  
gui.cc

---

## Index

ContentObject  
  ContentObject, 1  
Frame  
  Frame, 3  
Gui  
  Gui, 6  
addObject  
  Frame, 4  
clear  
  Gui, 6  
ContentObject  
  ContentObject, 1  
  ContentObject, 1  
  getGraphic, 1  
  getGraphicMode, 1  
  getText, 1  
  getTextMode, 1  
  getX1, 1  
  getX2, 1  
  getY1, 1  
  getY2, 1  
  m.actionNumber, 2  
  m.x1, 2  
  m.x2, 2  
  m.y1, 2  
  m.y2, 2  
  setGraphic, 1  
  setGraphicMode, 1  
  setText, 1  
  setTextMode, 1  
  setX1, 1  
  setX2, 1  
  setY1, 1  
  setY2, 1  
ContentObject, 1  
  m.actionCount, 2  
  m.graphic, 2  
  m.isGraphic, 2  
  m.isText, 2  
  m.text, 3  
DeviceListener, 3  
displayO  
  Gui, 5  
displayOn  
  Gui, 5  
  Gui, 6  
drawLine  
  Gui, 6  
Frame, 3  
  Frame, 3  
  addObject, 4  
  Frame, 3  
  getActiveObject, 3  
  getObjectCount, 3  
  getObjects, 3  
  m.activeObject, 4  
  m.frameCount, 4  
  m.object, 4  
  m.objectCount, 4  
  setActiveObject, 3  
  setObjectCount, 3  
getActiveObject  
  Frame, 3  
getGraphic  
  ContentObject, 1  
getGraphicMode  
  ContentObject, 1  
getObjectCount  
  Frame, 3  
getObjects  
  Frame, 3  
getText  
  ContentObject, 1  
getTextMode  
  ContentObject, 1  
getX1  
  ContentObject, 1  
getX2  
  ContentObject, 1  
getY1  
  ContentObject, 1  
getY2  
  ContentObject, 1  
Gui, 4  
  Gui, 6  
  clear, 6  
  displayO , 5  
  displayOn, 5  
  drawLine, 6  
  Gui, 6

---

- init, 7
- invert, 5
- invertPart, 7
- lightO , 5
- lightOn, 5
- m\_activeFrame, 8
- m\_activeObject, 8
- m\_contentObject, 8
- m\_counter, 8
- m\_frame, 6
- m\_frameCount, 9
- m\_listener, 6
- m\_portFree, 9
- m\_text, 9
- run, 7
- scan, 7
- setCursor, 5
- setFont, 7
- setZoom, 7
- showLogo, 5
- simulate, 7
- writeBase, 7
- writeObject, 8
- writeScan, 8
- writeSquare, 8
  
- init
  - Gui, 7
- invert
  - Gui, 5
- invertPart
  - Gui, 7
  
- lightO
  - Gui, 5
- lightOn
  - Gui, 5
  
- m\_actionCount
  - ContentObject, 2
- m\_actionNumber
  - ContentObject, 2
- m\_activeFrame
  - Gui, 8
- m\_activeObject
  - Frame, 4
  - Gui, 8
- m\_contentObject
  - Gui, 8
- m\_counter
  
- Gui, 8
- m\_frame
  - Gui, 6
- m\_frameCount
  - Frame, 4
  - Gui, 9
- m\_graphic
  - ContentObject, 2
- m\_isGraphic
  - ContentObject, 2
- m\_isText
  - ContentObject, 2
- m\_listener
  - Gui, 6
- m\_object
  - Frame, 4
- m\_objectCount
  - Frame, 4
- m\_portFree
  - Gui, 9
- m\_text
  - ContentObject, 3
  - Gui, 9
- m\_x1
  - ContentObject, 2
- m\_x2
  - ContentObject, 2
- m\_y1
  - ContentObject, 2
- m\_y2
  - ContentObject, 2
  
- run
  - Gui, 7
  
- scan
  - Gui, 7
- setActiveObject
  - Frame, 3
- setCursor
  - Gui, 5
- setFont
  - Gui, 7
- setGraphic
  - ContentObject, 1
- setGraphicMode
  - ContentObject, 1
- setObjectCount
  - Frame, 3
- setText

---

- ContentObject, 1
- setTextMode
  - ContentObject, 1
- setX1
  - ContentObject, 1
- setX2
  - ContentObject, 1
- setY1
  - ContentObject, 1
- setY2
  - ContentObject, 1
- setZoom
  - Gui, 7
- showLogo
  - Gui, 5
- simulate
  - Gui, 7
- writeBase
  - Gui, 7
- writeObject
  - Gui, 8
- writeScan
  - Gui, 8
- writeSquare
  - Gui, 8