Funktionen der libluise.so
(**L**CD-**U**SB-**I**nterface **s**econd **e**dition)

Inhalt:

Allgemeines:

Die libluise.so stellt eine sehr einfache Schnittstelle zur Programmierung des USB-LCDs bereit. Dem Programmierer stehen drei unabhängige Screens zur Verfügung, ein Text-Screen und zwei Grafik-Screens. Die Screens umfassen jeweils 320 x 240 Pixel. Durch die zwei unabhängigen Grafik-Screens lassen sich sehr einfach übereinander liegende statische und dynamische Daten darstellen. Dies ist zum Beispiel bei Einsatz von Hintergrundbildern hilfreich, da die Umgebung der dynamischen Daten/Grafiken nicht ständig neu berechnet werden muss. Alle drei Screens erscheinen auf dem LCD „übereinander“ und werden durch vorher festgelegte Parameter (LUI_LCDmode()) entsprechend miteinander bitweise verknüpft. Der Koordinatenursprung (x=0, y=0) eines Screens befindet sich immer in der linken oberen Ecke. Weiterhin werden die Invertierung und das Drehen der Screens unterstützt (90° Schritte), wodurch dem Programmierer die Koordinatenumrechnungen erspart bleiben.

Hinweise:

Das LCD muss zu Beginn des Programms geöffnet (LUI_OpenDevice()) und am Ende des Programms geschlossen (LUI_CloseDevice()) werden.

Diese Beta-Version 0.1.1 für Linux unterstützt derzeit nur ein LCD, d.h. DevNum muss immer mit 0 angegeben werden! Da zukünftig auch unter Linux mehrere LCDs gleichzeitig unterstützt werden sollen, wurden die Angaben in den Funktionsbeschreibungen so belassen.

Bis auf GetDllVersion() geben alle Funktionen einen Fehlercode („ErrorCode“) zurück.

Typ: unsigned char

Bedeutung der Fehlercodes:

0x00 = OK
0x01 = can't open USB-device
0x02 = USB write error
0x03 = USB read error
0x04 = file doesn't exist
0x05 = faulty file
0x06 = out of range

GetDllVersion

Beschreibung:

Gibt die Versionsnummer der libluise.so zurück.

Syntax:

int GetDllVersion ()

Rückgabewert:

Dreistellige Versionsnummer der libluise.so (100...999), dabei entspricht z.B. 100 = 1.0.0

Bemerkung:

Diese Funktion dient zur Vermeidung von Versionskonflikten mit zukünftigen lib-Versionen.

Beispiel:

```
int DllVersion;
DllVersion = GetDllVersion ();
```

LUI_OpenDevice

Beschreibung:

Öffnet das Gerät (LCD) mit dem im Parameter angegebenen Index.

Syntax:

ErrorCode LUI_OpenDevice (int DevNum)

Parameter:

DevNum Index des zu öffnenden Gerätes (0…3)

Bemerkung:

Diese Funktion öffnet ein Gerät für weitere Operationen. Wenn nur ein Gerät vorhanden ist, hat es immer den Index 0. Zur weiteren Identifikation des LCDs dient die Funktion LUI_GetSerial.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;

ErrorCode = LUI_OpenDevice (DevNum);
if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung, (betreffendes Gerät kann z.B. schon offen sein)
}
else
{
        //das Gerät mit Index 0 wurde geöffnet und ist für weitere Operationen bereit
}
```

LUI_CloseDevice

Beschreibung:

Schließt das Gerät mit dem im Parameter angegebenen Index.

Syntax:

ErrorCode LUI_CloseDevice (int DevNum)

Parameter:

DevNum Index des zu schließenden Gerätes (0…3)

Bemerkung:

Diese Funktion schließt ein Gerät und gibt es wieder frei. Bei Programmende müssen alle geöffneten Geräte wieder geschlossen werden.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;

ErrorCode = LUI_CloseDevice (DevNum);
if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //das Gerät mit Index 0 wurde geschlossen
}
```

LUI_LCDmode

Beschreibung:

Mit dieser Funktion wird festgelegt wie die zwei Grafik-Screens verknüpft werden und ob eine invertierte Ausgabe erfolgen soll. Weiterhin wird die Orientierung des LCDs und die Refreshzeit der IO-Ports festgelegt.

Syntax:

ErrorCode LUI_LCDmode (int DevNum, unsigned char gfxmode, unsigned char gfxinvert, unsigned char ScreenRotation, unsigned char IOrefresh)

Parameter:

DevNum	Index des Gerätes (0…3)
gfxmode	Verknüpfungsfunktion der zwei Grafik-Screens (0=ODER ; 1=UND ; 2=XOR); default=0
gfxinvert	0= normal; 1= invertiert (gesamtes LCD); default=0
ScreenRotation	Rechtsdrehung der Screens in 90° Schritten (0...3); 0 = 0° (Screenformat 320x240, CCFL rechts); 1 = 90° (Screenformat 240x320, CCFL unten); 2 = 180° (Screenformat 320x240, CCFL links); 3 = 270° (Screenformat 240x320, CCFL oben)
IOrefresh	Refreshzeit der IO-Ports in 25ms -Schritten (0=25ms...255=256*25ms)

Bemerkung:

Der Wert für IOrefresh beeinflusst direkt die CPU-Last und sollte, wenn die Verwendung der IO-Ports im Programm nicht vorgesehen ist, möglichst hoch gewählt werden. Normalwert liegt bei ca. 2.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;


ErrorCode = LUI_LCDmode (DevNum, 2, 1, 0, 2);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // Die zwei Grafik-Screens werden XOR-verknüpft und dann invertiert auf dem LCD ausgegeben.
        // Die Screens werden nicht gedreht und die IO-Refreshzeit liegt bei 75ms.
}
```

LUI_Text

Beschreibung:

Textausgabe auf Text-Screen unter Verwendung von Textfeldern und eigenen Schriftarten.

Syntax:

ErrorCode LUI_Text (int DevNum, int ScreenPosX, int ScreenPosY, int TextFieldWidth, int TextFieldHight, unsigned char transparency, char *FontFileName, unsigned char FontWidthSpace, unsigned char FontHightSpace, char *textdata)

Parameter:

DevNum	Index des Gerätes (0…3)
ScreenPosX	X-Koordinate der linken oberen Ecke des Textfeldes
ScreenPosY	Y-Koordinate der linken oberen Ecke des Textfeldes
TextFieldWidth	Breite des Textfeldes in Pixel
TextFieldHight	Höhe des Textfeldes in Pixel
transparency	0 = alle vorherigen Zeichen unter dem Textfeld werden gelöscht 1 = die neuen Zeichen werden über die vorherigen Zeichen gelegt
FontFileName	Zeiger auf Zeichenkette mit Dateinamen der Font-Datei (ASCII, Nullterminierter String); wenn FontFileName=0 wird der interne Zeichensatz verwendet
FontWidthSpace	Freiraum zwischen den Zeichen nebeneinander in Pixel
FontHightSpace	Freiraum zwischen den Zeichen untereinander in Pixel
textdata	Zeiger auf Zeichenkette (ASCII, Nullterminierter String)

Bemerkung:

In der Zeichenkette vorhandene NewLine-Kommandos (0x0A) werden interpretiert und dadurch neue Zeilen innerhalb des Textfeldes begonnen.
Der Parameter 'transparency' wird wohl in den meisten Fällen =0 sein. Das Überlagern der Zeichen wird nur für einige Spezialeffekte benötigt.
Das Format der Font-Datei wird in einer separaten Spezifikation beschrieben.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
char  *textdata= "hello world!\nnew line";
char  *FontFileName= "MyOwnFont.fnt";

ErrorCode = LUI_Text (DevNum, 10, 15, 50, 25, FontFileName, 1, 1, textdata);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //Ein Textfeld (50 x 25 Pixel) wurde an Position x=10 y=15 erzeugt. In der ersten Zeile wurde
        //"hello world!" und durch „\n“ (=0x0A) in der zweiten Zeile „new line“ ausgegeben. Als Schriftart
        //wurde MyOwnFont.fnt verwendet.
}
```

LUI_Bitmap

Beschreibung:

Lädt die übergebenen Bild-Daten in den entsprechenden Grafik-Screen.

Syntax:

ErrorCode LUI_Bitmap (int DevNum, unsigned char ScreenNr, int ScreenPosX, int ScreenPosY, int BMPoffsetX, int BMPoffsetY, int BMPWidth, int BMPHight, int BMPdataWidth, int BMPdataHight, unsigned char *Bitmap)

Parameter:

DevNum	Index des Gerätes (0…3)
ScreenNr	Grafik-Screennummer (0,1)
ScreenPosX	X-Koordinate der linken oberen Ecke des auszugebenden Bitmaps auf dem LCD
ScreenPosY	Y-Koordinate der linken oberen Ecke des auszugebenden Bitmaps auf dem LCD
BMPoffsetX	X-Offset der linken oberen Ecke des auszugebenden Bitmaps innerhalb der Bitmap-Daten (z.B. um nur einen Teilbereich eines Bitmaps auf dem LCD auszugeben)
BMPoffsetY	Y-Offset der linken oberen Ecke des auszugebenden Bitmaps innerhalb der Bitmap-Daten (z.B. um nur einen Teilbereich eines Bitmaps auf dem LCD auszugeben)
BMPWidth	Breite des auszugebenden Bitmaps
BMPHight	Höhe des auszugebenden Bitmaps
BMPdataWidth	Breite der gesamten Bitmap-Daten in Pixel
BMPdataHight	Höhe der gesamten Bitmap-Daten in Pixel
Bitmap	Zeiger auf Array mit Bitmap-Daten

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
unsigned char BMPData[9600];

//BMPData[9600] mit Bitmapdaten laden (schwarz/weiß), z.B. Hintergrundbild 320x240
//…

ErrorCode = LUI_Bitmap (DevNum, 0, 0, 0, 0, 0, 320, 240, 320, 240, BMPData);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //Bitmap (in diesem Beispiel full screen 320x240=9600Byte) wurde an Position x=0 und y=0, mit
        einer Breite von 320 Pixel und eine Höhe von 240 Pixel in Grafik-Screen Nummer Null geladen.
        Da das ganze Bitmap geladen werden sollte, sind BMPoffsetX und BMPoffsetY gleich Null.
}


//BMPData[9600] mit Bitmapdaten laden (schwarz/weiß), z.B. ein 320x240 Pixel Bitmap mit kleineren
//Grafiken für Animationen
//…

ErrorCode = LUI_Bitmap (DevNum, 1, 50, 30, 10, 15, 25, 25, 320, 240, BMPData);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // Ein kleines Bitmap wurde an Position x=50 und y=40, mit einer Breite von 25 Pixel und eine
        Höhe von 25 Pixel in Grafik-Screen Nummer 1 geladen. Das kleine Bitmap hat innerhalb des
        gesamten Bitmaps die Position x=10 und y=15 (BMPoffsetX=10 und BMPoffsetY=15). Das kleine
        25x25 Pixel Bitmap wurde quasi aus dem großen 320x240 Pixel Bitmap (BMPData -Array)  an der
        Position x=10, y=15 “ausgeschnitten” und auf dem LCD an der Position x=50, y=30 eingefügt.
}
```

LUI_BMPfile

Beschreibung:

Lädt ebenfalls wie die Funktion LUI_Bitmap die übergebenen Bild-Daten in den entsprechenden Grafik-Screen, nur dass hier die Bitmap-Daten direkt aus einer Bitmap-Datei ausgelesen werden.

Syntax:

ErrorCode LUI_BMPfile (int DevNum, unsigned char ScreenNr, int ScreenPosX, int ScreenPosY, int BMPoffsetX, int BMPoffsetY, int BMPWidth, int BMPHight, char *file)

Parameter:

DevNum	Index des Gerätes (0…3)
ScreenNr	Grafik-Screennummer (0,1)
ScreenPosX	X-Koordinate der linken oberen Ecke des auszugebenden Bitmaps auf dem LCD
ScreenPosY	Y-Koordinate der linken oberen Ecke des auszugebenden Bitmaps auf dem LCD
BMPoffsetX	X-Offset der linken oberen Ecke des auszugebenden Bitmaps innerhalb der Bitmap-Datei (z.B. um nur einen Teilbereich eines Bitmaps auf dem LCD auszugeben)
BMPoffsetY	Y-Offset der linken oberen Ecke des auszugebenden Bitmaps innerhalb der Bitmap-Datei (z.B. um nur einen Teilbereich eines Bitmaps auf dem LCD auszugeben)
BMPWidth	Breite des auszugebenden Bitmaps
BMPHight	Höhe des auszugebenden Bitmaps
file	Zeiger auf Nullterminierten String, welcher den Dateinamen (und ggf. den Pfad) der Bitmap-Datei enthält.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
char *BMPfile = "pic.bmp";

ErrorCode = LUI_BMPfile (DevNum, 0, 0, 0, 0, 0, 320, 240, BMPfile);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //Das Bitmap (in diesem Beispiel full screen 320x240=9600Byte) wurde aus der Datei pic.bmp
        ausgelesen und an Position x=0 und y=0, mit einer Breite von 320 Pixel und eine Höhe von 240
        Pixel in den Grafik-Screen Nummer Null geladen. Da das ganze Bitmap geladen werden sollte,
        sind BMPoffsetX und BMPoffsetY gleich Null.
}


char *BMPfile2 = "pic2.bmp";

ErrorCode = LUI_BMPfile (DevNum, 1, 50, 30, 10, 15, 25, 25, BMPfile2);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // Ein kleines Bitmap wurde aus der Datei pic2.bmp ab der Position x=10, y=15 ausgelesen und
        an Position x=50 und y=40, mit einer Breite von 25 Pixel und eine Höhe von 25 Pixel in Grafik-
        Screen Nummer 1 geladen. Das kleine Bitmap hat innerhalb der Bitmap-Datei die Position x=10
        und y=15 (BMPoffsetX=10 und BMPoffsetY=15). Das kleine 25x25 Pixel Bitmap wurde quasi aus
        dem großen 320x240 Pixel Bitmap (Bitmap-Datei pic2.bmp)  an der Position x=10, y=15
        “ausgeschnitten” und auf dem LCD an der Position x=50, y=30 eingefügt.
}
```

LUI_SetPixel

Beschreibung:

Setzt ein Pixel an der angegebenen Position auf 0 oder 1.

Syntax:

ErrorCode LUI_SetPixel (int DevNum, unsigned char ScreenNr, int ScreenPosX, int ScreenPosY, char PixelValue)

Parameter:

DevNum	Index des Gerätes (0…3)
ScreenNr	Grafik-Screennummer (0,1)
ScreenPosX	X-Koordinate des Pixel auf dem LCD
ScreenPosY	Y-Koordinate des Pixel auf dem LCD
PixelValue	Wert des Pixel (0,1)

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;

ErrorCode = LUI_SetPixel (DevNum, 0, 100, 200, 1);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //Pixel im Grafik-Screen Nummer 0 mit den Koordinaten x=100, y=200 wurde auf 1 gesetzt
}
```

LUI_GetPixel

Beschreibung:

Ermittelt den Wert eines Pixels an der angegebenen Position.

Syntax:

ErrorCode LUI_GetPixel (int DevNum, unsigned char ScreenNr, int ScreenPosX, int ScreenPosY, char *PixelValue)

Parameter:

DevNum	Index des Gerätes (0…3)
ScreenNr	Grafik-Screennummer (0,1)
ScreenPosX	X-Koordinate des Pixel auf dem LCD
ScreenPosY	Y-Koordinate des Pixel auf dem LCD
PixelValue	Zeiger auf die Variable, in welche der Wert des Pixels gespeichert werden soll.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
char PixelValue;

ErrorCode = LUI_GetPixel (DevNum, 0, 100, 200, &PixelValue);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //PixelValue enthält jetzt den aktuellen Wert des Pixels im Grafik-Screen Nummer 0 mit den
        Koordinaten x=100, y=200 (0 oder 1)
}
```

LUI_DispOnOff

Beschreibung:

Schaltet das LCD ein oder aus.

Syntax:

ErrorCode LUI_DispOnOff (int DevNum, char onoff)

Parameter:

DevNum	Index des Gerätes (0…3)
onoff	0=aus, 1=an

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;

ErrorCode = LUI_DispOnOff (DevNum, 0);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //LCD aus
}
```

LUI_CCFL

Beschreibung:

Schaltet die Hintergrundbeleuchtung ein oder aus.

Syntax:

ErrorCode LUI_CCFL(int DevNum, char onoff)

Parameter:

DevNum	Index des Gerätes (0…3)
onoff	0=aus, 1=an

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;

ErrorCode = LUI_CCFL (DevNum, 0);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //Hintergrundbeleuchtung aus
}
```

LUI_GetContrast

Beschreibung:

Gibt den aktuellen Kontrastwert des LCDs zurück.

Syntax:

ErrorCode LUI_GetContrast (int DevNum, unsigned char *cvalue)

Parameter:

DevNum	Index des Gerätes (0…3)
cvalue	Zeiger auf die Variable, in welche der aktuelle Kontrastwert des LCDs gespeichert werden soll. (0...255) 0=hell, 255=dunkel

Bemerkung:

Diese Funktion kann dazu genutzt werden, um bei Start der Anwendersoftware den Kontrastregler auf die richtige Position voreinzustellen.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
unsigned char cvalue;

ErrorCode = LUI_GetContrast (DevNum, &cvalue);

if (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // cvalue enthält jetzt den aktuellen Kontrastwert des LCDs
}
```

LUI_SetContrast

Beschreibung:

Setzt den Kontrastwert des LCDs.

Syntax:

ErrorCode LUI_SetContrast (int DevNum, unsigned char cvalue)

Parameter:

DevNum	Index des Gerätes (0…3)
cvalue	Kontrastwert (0...255) 0=hell, 255=dunkel

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;


ErrorCode = LUI_SetContrast (DevNum, 100);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // der Kontrastwert des LCDs wurde auf 100 gesetzt
}
```

LUI_InPort

Beschreibung:

Gibt die aktuellen Werte der Eingänge zurück.

Syntax:

ErrorCode LUI_InPort (int DevNum, unsigned char *PortData)

Parameter:

DevNum	Index des Gerätes (0…3)
PortData	Zeiger auf die Variable, in welche die aktuellen Werte der Eingänge gespeichert werden sollen. Dabei ensprechen die 5 niederwertigsten Bits dieser Variable den 5 Tasterzuständen, d.h. Taster 1 = Bit 0, Taster 2 =Bit 1 usw. . 0=Taster offen, 1=Taster geschlossen (gegen Masse)

Bemerkung:

Die zurückgegebenen Bitwerte entsprechen nicht den logischen Pegeln an den Eingängen! Die Werte wurden zur einfacheren Weiterverarbeitung invertiert, da die Taster im Normalfall gegen Masse betrieben werden (keine weiteren Bauelemente (pull up Widerstände) notwendig). Die Eingänge stehen nur bei der Hardwareversion V1 zur Verfügung. Bei der Hardwareversion V2 steht statt der Eingänge ein Touch Screen als Eingabemedium zur Verfügung.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
unsigned char PortData;

ErrorCode = LUI_InPort (DevNum, &PortData);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // PortData enthält jetzt  die aktuellen Werte der Eingänge
}
```

LUI_OutPort

Beschreibung:

Setzt die Ausgänge auf die entsprechenden Werte.

Syntax:

ErrorCode LUI_OutPort (int DevNum, unsigned char PortData)

Parameter:

DevNum	Index des Gerätes (0…3)
PortData	Wert für die Ausgänge. Dabei ensprechen die 3 niederwertigsten Bits dieses Wertes den jeweiligen Ausgangszuständen, d.h. Ausgang 1 = Bit 0, Ausgang 2 =Bit 1 usw. .

Bemerkung:

Die Bitwerte entsprechen den logischen Pegeln an den Ausgängen, d.h. 0 = Low / 0 Volt, 1 = High / +5 Volt.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;

ErrorCode = LUI_OutPort (DevNum, 3);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        // Die Ausgänge 1 und 2 wurden auf 1 (+5 Volt) gesetzt, Ausgang 3 wurde auf 0 (0 Volt) gesetzt
}
```

LUI_GetSerial

Beschreibung:

Gibt die Seriennummer des LCDs zurück.

Syntax:

ErrorCode LUI_GetSerial (int DevNum, unsigned int *Serial)

Parameter:

DevNum	Index des Gerätes (0…3)
Serial	Zeiger auf die Variable, in welcher die Seriennummer des LCDs gespeichert werden soll

Bemerkung:

Die Seriennummer dient bei Betrieb von mehreren LCDs zur eindeutigen Identifizierung des jeweiligen LCDs.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
unsigned int Serial;

ErrorCode = LUI_GetSerial (DevNum, &Serial);

if  (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}
else
{
        //Die Variable Serial enthält jetzt  die Seriennummer des LCDs (Gerät 0)
}
```

LUI_ScreenRefreshControl

Beschreibung:

Steuert die Ausgabe der einzelnen Screens.

Syntax:

ErrorCode LUI_ScreenRefreshControl (int DevNum, unsigned char GfxScreen0,
unsigned char GfxScreen1, unsigned char TXTScreen)

Parameter:

DevNum	Index des Gerätes (0…3)
GfxScreen0	= 1: alle Änderungen auf dem Grafik-Screen 0 werden sofort auf dem LCD ausgegeben = 0: alle Änderungen auf dem Grafik-Screen 0 werden nicht sofort auf dem LCD ausgegeben sondern zwischengespeichert (und erst bei GfxScreen0 =1 mit einmal auf dem LCD ausgegeben)
GfxScreen1	wie GfxScreen0
TXTScreen	wie GfxScreen0

Bemerkung:

Mit dieser Ausgabesteuerung der einzelnen Screens kann man z.B. bei zeitaufwendigen Grafiken deren noch unvollständige Ausgabe verhindern. Default bei allen Screens =1.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
unsigned int Serial;

ErrorCode = LUI_ScreenRefreshControl(DevNum, 1, 0, 1);

if (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}

// Grafik-Screen 1 kann nun in Ruhe beschrieben werden

ErrorCode = LUI_ScreenRefreshControl(DevNum, 1, 1, 1);

if (ErrorCode)
{
        //Fehlercode auswerten, Fehlerbehandlung
}

// Grafik-Screen 1 wird jetzt wieder mit allen Änderungen auf dem LCD ausgegeben
```

LUI_TouchScreen

Beschreibung:

Gibt die Koordinaten des Berührungspunktes vom Touch Screen zurück.

Syntax:

ErrorCode LUI_TouchScreen(int DevNum, int *TSdata)

Parameter:

DevNum	Index des Gerätes (0…3)
TSdata	Zeiger auf ein Array mit drei Elementen vom Typ integer, in welche die Koordinaten des Berührungspunktes vom Touch Screen gespeichert werden sollen.

TSdata[0] : Pen Down (0 = keine Berührung auf dem Touch Screen,
1 = Berührung auf dem Touch Screen, d.h. die nachfolgenden Koordinaten in TSdata[1] und TSdata[2] entsprechen den Koordinaten des Berührungspunktes)
TSdata[1] : X-Koordinate des Berührungspunktes (0...319)
TSdata[2] : Y-Koordinate des Berührungspunktes (0...239)

Bemerkung:

Der "Pen Down" -Wert kann als Gültigkeitssignal verstanden werden. Er ist immer dann =1, wenn das Touch Screen berührt wird.
Das Touch Screen steht nur bei der Hardwareversion V2 zur Verfügung. Bei der Hardwareversion V1 stehen statt des Touch Screens digitale Eingänge z.B. für Taster zur Verfügung. Diese Funktion wird erst ab der Firmware Version 1.0.4 unterstützt.

Beispiel:

```
unsigned char ErrorCode;
int DevNum=0;
int TSdata[3];

ErrorCode = LUI_TouchScreen(DevNum, TSdata);

if  (ErrorCode) { //Fehlercode auswerten, Fehlerbehandlung }
else
{
        if (TSdata[0])  ErrorCode = LUI_SetPixel (DevNum, 0, TSdata[1], TSdata[2], 1);

        //Ein Pixel im Grafik-Screen Nummer 0 mit den Koordinaten des Berührungspunktes wurde auf 1
        //gesetzt (eine einfache Art eines Zeichenprogramms mit Touch Screen)
}
```