Ansteuerung des I²C-GLCD-DOG-M132 PCB-Version 2.2

Displayauflösung 132 x 32 Pixel

Das Programmfile des Displays I2C_DOGM_UC.hex ist für die einfarbige Version , das File I2C_DOGM_RGB.hex für die RGB-Version. Bestückungspläne beachten !

TWInet-Treiber: Display = Slave / Adresse $10
Der I²C-Bus wird bis nach der Ausführung des Befehls blockiert

Es werden maximal 16 Byte Nutzdaten übertragen. Anzahl übertragener Bytes stehen in Klammern hinter dem jeweiligen Befehl. Nicht benutzte Variablen werden nicht ausgewertet.

Nutzdaten-Byte #:

$00	**$01**	**-**	**$0F**
Befehl	----------	siehe unten	----------

Die gesamte Übertragung ist wie folgt aufgebaut: | Adresse | Datenlänge | Daten | Prüfsumme |

Adresse	$10
Datenlänge	siehe beim jeweiligen Befehl *(Datenlänge: x)* und den Aufbau in Anhang 2
Daten	Inhalte des TWItxBuff[0] - [x] siehe Anhang 2
Prüfsumme	Addition aller Bytes in ein Byte wobei der Überlauf ignoriert wird

Erklärungen: $ = hexadezimale Zahl $5A = 5Ah % = binäre Zahl %01010101 = 01010101b

Befehl (Byte 0):

- $00 = chInitLCD *(Datenlänge: 1)*
  (incl. $01 & $02 / wird beim Start einmalig automatisch durchgeführt)

- $01 = chClrScr(Pattern:Byte) *(Datenlänge: 6)*
  Byte 5: Pattern

- $02 = chDispRefresh *(Datenlänge: 1)*

- $03 = chSetAutoRefresh(OnOff:Byte) *(Datenlänge: 2)*
  Byte 1: 0 = off / 1 = on
  default = off

- $0C = chSetLED (**nur RGB-Version !** - *Datenlänge: 2)*
  Byte 1:
  %s t u v w x y z =
  s = 0 = LED 1 / 1 = LED 2
  t-z = Helligkeit (PWM) 0..127
  default = 0

- $0D = chSetBKBright(Bright:Byte); (**nur einfarbige Version !** - *Datenlänge 2**)***
  Byte 1: Helligkeit 0..127
  default = 80

- $0D = chSetBKColor(R,G,B:Byte) (**nur RGB-Version !** - *Datenlänge 4)*
  Byte 1: rot-Wert (0..127)
  Byte 2: grün-Wert (0..127)
  Byte 3: blau-Wert (0..127)

  wenn R=G=B=0 ist Beleuchtung ausgeschaltet

bei R=G=B=127 Beleuchtung weiß & volle Helligkeit
evtl. eingeschaltetes chSetBKRainbow wird abgeschaltet
default = R=G=B=80

- $0E = chSetBKRainbow(OnOff:Byte) (**nur RGB-Version !** - *Datenlänge 2)*
  Byte 1: 0 = off / 1 = on
  default = off

- $0F = chSetContrast *(Datenlänge: 2)*
  Byte 1: Kontrast $00-$3F
  default = $1F

- $10 = chDrawLine(Xs,Ys,Xe,Ye,Pattern:Byte) *(Datenlänge: 6)*
  Byte 1: X-Koordinate Startpunkt
  Byte 2: Y-Koordinate Startpunkt
  Byte 3: X-Koordinate Endpunkt
  Byte 4: Y-Koordinate Endpunkt
  Byte 5: Pattern

- $11 = chDrawRect(Xs,Ys,Xe,Ye,Pattern:Byte) *(Datenlänge: 6)*
  Byte 1: X-Koordinate oben/links
  Byte 2: Y-Koordinate oben/links
  Byte 3: X-Koordinate unten/rechts
  Byte 4: Y-Koordinate unten/rechts
  Byte 5: Pattern

- $12 = chFillRect(Xs,Ys,Xe,Ye,Pattern:Byte) *(Datenlänge: 6)*
  Byte 1: X-Koordinate oben/links
  Byte 2: Y-Koordinate oben/links
  Byte 3: X-Koordinate unten/rechts
  Byte 4: Y-Koordinate unten/rechts
  Byte 5: Pattern

- $13 = chDrawCircle(Xs,Ys,r,Pattern:Byte) *(Datenlänge: 6)*
  Byte 1: X-Koordinate Mittelpunkt
  Byte 2: Y-Koordinate Mittelpunkt
  Byte 3: Radius
  Byte 5: Pattern

- $14 = chFillCircle(Xs,Ys,r,Pattern:Byte) *(Datenlänge: 6)*
  Byte 1: X-Koordinate Mittelpunkt
  Byte 2: Y-Koordinate Mittelpunkt
  Byte 3: Radius
  Byte 5: Pattern

- $15 = chSetPixel(X,Y:Byte) *(Datenlänge: 3)*
  Byte 1: X-Koordinate
  Byte 2: Y-Koordinate

- $16 = chClearPixel(X,Y:Byte) *(Datenlänge: 3)*
  Byte 1: X-Koordinate
  Byte 2: Y-Koordinate

- $17 = chXorPixel(X,Y:Byte) *(Datenlänge: 3)*
  Byte 1: X-Koordinate
  Byte 2: Y-Koordinate

- $18 = chSetLineMode(S:Byte) *(Datenlänge: 2)*
  Byte 1:
  $00 = wmClrPix
  $01 = wmSetPix
  $02 = wmXorPix

- $20 = chDrawString(X,Y,Size&Rotation:Byte; Text:String[10]) *(Datenlänge: 5+Textlänge)*
  Byte 1: X-Koordinate
  Byte 2: Y-Koordinate
  Byte 3:
  %s t u v w x y z =
  s und t = Schriftgröße X-Richtung (1..3)
  u und v = Schriftgröße Y-Richtung (1..3)
  w und x = nicht verwendet
  y und z = Winkel 0..3 entspricht 0,90,180 und 270°
  Byte 4: Textlänge
  Byte 5-14: Text

- $21 = chDrawStringRel(Size&Rotation:Byte; Text:String[10]) *(Datenlänge: 5+Textlänge)*
  Byte 3:
  %s t u v w x y z =
  s und t = Schriftgröße X-Richtung (1..3)
  u und v = Schriftgröße Y-Richtung (1..3)
  w und x = nicht verwendet
  y und z = Winkel 0..3 entspricht 0,90,180 und 270°
  Byte 4: Textlänge
  Byte 5-14: Text

- $22 = chSetTextJustify(S:Byte) *(Datenlänge: 2)*
  Byte 1:
  $00 = alHorLeft, alVertBottom
  $01 = alHorLeft, alVertCenter
  $02 = alHorLeft, alVertTop
  $03 = alHorCenter, alVertBottom
  $04 = alHorCenter, alVertCenter
  $05 = alHorCenter, alVertTop
  $06 = alHorRight, alVertBottom
  $07 = alHorRight, alVertCenter
  $08 = alHorRight, alVertTop

- $23 = chSetTextMode(Mode:Byte) *(Datenlänge: 2)*
  Byte 1:
  $00 = wmClrPix
  $01 = wmSetPix
  $02 = wmXorPix

- $24 = chSetTextBkGround(Bk:Byte) *(Datenlänge: 2)*
  Byte 1:
  $00 = bkTransp
  $01 = bkNormal
  $02 = bkInvers

  Standard ist alHorCenter, alVertTop und wmSetPix

- $30 = chMoveTo(X,Y:Byte) *(Datenlänge: 3)*

Byte 1: X-Koordinate
Byte 2: Y-Koordinate

- $31 = chMoveToRel(X,Y:Byte) *(Datenlänge: 3)*
  Byte 1: X-Koordinate
  Byte 2: Y-Koordinate

- $40 = chDrawBitMap(X,Y,BMPNr:Byte) *(Datenlänge: 4)*
  Byte 1: X-Koordinate
  Byte 2: Y-Koordinate
  Byte 3: Bitmap-Nummer

  Ausgabe Grafik (muß im Controller vordefiniert) sein ! Ausgabefeld wird bei Bitmap $00-$F7 vorher gelöscht (siehe auch Anhang 1).

- $50 = chShowNorth1(X,LoAngle,HiAngle,Drv:Byte) *(Datenlänge: 5)*
  Ausgabe Fahrtrichtung mit Winkelangabe
  (Breite x Höhe fest auf 31 x 32 Pixel)
  Byte 1: X-Koordinate
  Byte 2&3: Winkel (Integer: Lo- & Highbyte)
  Byte 4: wenn <>0 erfolgt keine Winkelausgabe: „----“ wird angezeigt
  Ausgabe des Richtungswinkels; N=0°, O=90°, S=180°, W=270°
  chSetTextJustify muß auf $05 stehen !

- $51 = chShowNorth2(X,LoAngle,HiAngle,Drv:Byte) *(Datenlänge: 5)*
  Ausgabe Fahrtrichtung mit Richtungsangabe
  (Breite x Höhe fest auf 31 x 32 Pixel)
  Byte 1: X-Koordinate
  Byte 2&3: Winkel (Integer: Lo- & Highbyte)
  Byte 4: wenn <>0 erfolgt keine Richtungsausgabe: „--“ wird angezeigt
  Ausgabe von N, NW, W, ...
  chSetTextJustify muß auf $05 stehen !

- $52 = chShowNorth3(X,LoAngle,HiAngle,Drv:Byte) *(Datenlänge: 5)*
  Ausgabe Fahrtrichtung mit Richtungsangabe
  (Breite x Höhe fest auf 31 x 32 Pixel)
  Byte 1: X-Koordinate
  Byte 2&3: Winkel (Integer: Lo- & Highbyte)
  Byte 4: wenn <>0 erfolgt keine Richtungsausgabe: „*“ wird angezeigt
  Ausgabe von N, NW, W, ...
  Darstellung erfolgt durch 9 BitMaps: nur 8 mögliche Positionen + Stop
  chSetTextJustify muß auf $05 stehen !

- $FF = Ausgabe Copyright *(Datenlänge: 1)*

Anhang 1: BitMaps (chDrawBitMap)

BitMap-Nr.	Beschreibung	Größe [Pixel]
$00	Satellit	8x8
$01	Höhe über NN	8x8
$02	Schaltblitz 1	8x8
$03	Schaltblitz 2	8x8
$04	Ölfühler Ölwanne	8x8
$05	Ölfühler Steigleitung	8x8
$0B	On	8x8
$0C	Off	8x8
$0D	Herz gefüllt	8x8
$0E	Herz leer	8x8
$0F	CH	8x8
$10	Lufttemperatur 1*	8x8
$11	Lufttemperatur 2*	8x8
$12	Uhr 1	

* $10 & $11 für animierte Darstellung Lufttemperatur

$40	Öltemperatur 1	16x8
$41	Öltemperatur 2	16x8
$42	Umgebungstemperatur	16x8
$43	Bordspannung	16x8
$44	Tankinhalt	16x8
$45	Reifentemperatur	16x8
$46	Reifenwarnung/Druck	16x8
$47	Öltemperatur 3**	16x8
$48	Öltemperatur 4**	16x8
$49	Öltemperatur 5**	16x8
$4A	Wassersymbol 1***	16x8
$4B	Wassersymbol 2***	16x8
$4C	Heizgrifftemperatur links	16x8
$4D	Heizgrifftemperatur rechts	16x8
$4E	Heizgriff-Leistung 1	16x8
$4F	Kettenöler 1	16x8
$5F	© CH	16x8

** $47 - $49 für animierte Darstellung Öltemperatur
*** $4A & $4B für animierte Darstellung Wassertemperatur

$60	Motor	16x16
$61	Fernlicht	16x16
$62	Abblendlicht	16x16
$63	Nebelscheinwerfer	16x16
$64	Nebelschlussleuchte	16x16
$65	Tacho	16x16
$66	Kettenöler 2	16x16
$67	Uhr 2	16x16

$80	“N”	16x24
$81	“1”	16x24
$82	“2”	16x24
$83	“3”	16x24
$84	“4”	16x24
$85	“5”	16x24
$86	“6”	16x24
$87	“7”	16x24

$88	“8”	16x24
$89	“9”	16x24
$8A	“0”	16x24
$8B	“?”	16x24
$8C	„S“ (für Seitenständer)	16x24
$A0	Windrose „Halt“	32x32
$A1	Windrose „N“	32x32
$A2	Windrose „NW“	32x32
$A3	Windrose „W“	32x32
$A4	Windrose „SW“	32x32
$A5	Windrose „S“	32x32
$A6	Windrose „SO“	32x32
$A7	Windrose „O“	32x32
$A8	Windrose „NO“	32x32

$00 $01 $02 $03 $04 $05 $0B $0C $0D $0E $0F
$10 $11 $12

$40 $41 $42 $43 $44 $45 $46 $47

$48 $49 $4A $4B $4C $4D $4E $4F

$5F

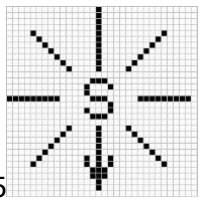
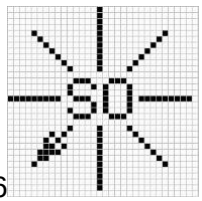
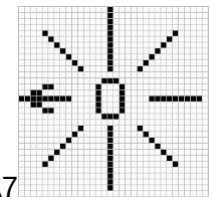
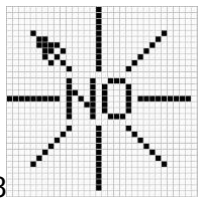
$60 $61 $62 $63 $64 $65 $66

$67

$80 $81 $82 $83 $84 $85 $86 $87

$88 $89 $8A $8B $8C

$A0 $A1 $A2 $A3 $A4

$A5  $A6  $A7  $A8 

Anhang 2: Aufbau der Pascal-Funktionen des Masters

Es müssen natürlich nur die auch verwendeten Funktionen im Master vorhanden sein.

```
Import SysTick, TWInet;

Define
      TWInode       = $01;
      TWIpresc      = TWI_BR100;
      TWIframe      = 16,iData;
      TWInetMode    = Master;

Const
  TWIslv        : Byte = $10;

Function WriteTWIData(Size:Byte):Boolean;
  Var WriteOK  : Boolean;
  Begin
    WriteOK:=false;
    Repeat
      TWIState:=GetTWIslaveStat(TWIslv);
      If [TWIrxEmpty] in TWIState
        then
          WriteOK:=TWItxFrame(TWIslv,Size);
        EndIf;
      Until WriteOK=true;
    Return(WriteOK);
  End WriteTWIData;

Procedure ClrTWIBuffer;
  Var Counter : Byte;
  Begin
    For Counter:=0 to 15 do
      TWItxBuff[Counter]:=0;
      EndFor;
  End ClrTWIBuffer;

Procedure chClrScr(Pattern:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$01;
    TWItxBuff[5]:=Pattern;
    WriteTWIData(6);
  End chClrScr;

Procedure chDispRefresh;
```

```
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$02;
    WriteTWIData(1);
  End chDispRefresh;

Procedure chSetAutoRefresh(OnOff:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$03;
    TWItxBuff[1]:=OnOff and $01;
    WriteTWIData(2);
  End chSetAutoRefresh;

Procedure chSetLED(LED,Bright:Byte); // nur RGB-Version !
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$0C;
    Case LED of
      1:TWItxBuff[1]:=Bright and $7F;|
      2:TWItxBuff[1]:=(Bright and $7F) + $80;|
      EndCase;
    WriteTWIData(2);
  End chSetLED;

Procedure chSetBKBright(Bright:Byte); // nur einfarbige Version !
  Begin
    TWItxBuff[0]:=$0D;
    TWItxBuff[1]:=Bright and $7F;
    WriteTWIData(2);
  End chSetBKBright;

Procedure chSetBKColor(R,G,B:Byte); // nur RGB-Version !
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$0D;
    TWItxBuff[1]:=R and $7F;
    TWItxBuff[2]:=G and $7F;
    TWItxBuff[3]:=B and $7F;
    WriteTWIData(4);
  End chSetBKColor;

Procedure chSetBKRainbow(OnOff:Byte); // nur RGB-Version !
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$0E;
    TWItxBuff[1]:=OnOff and $01;
    WriteTWIData(2);
  End chSetBKRainbow;

Procedure chSetContrast(Contrast:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$0F;
    TWItxBuff[1]:=Contrast;
    WriteTWIData(2);
  End chSetContrast;
```

```
Procedure chDrawLine(Xs,Ys,Xe,Ye,Pattern:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$10;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=Xe;
    TWItxBuff[4]:=Ye;
    TWItxBuff[5]:=Pattern;
    WriteTWIData(6);
  End chDrawLine;

Procedure chDrawRect(Xs,Ys,Xe,Ye,Pattern:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$11;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=Xe;
    TWItxBuff[4]:=Ye;
    TWItxBuff[5]:=Pattern;
    WriteTWIData(6);
  End chDrawRect;

Procedure chFillRect(Xs,Ys,Xe,Ye,Pattern:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$12;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=Xe;
    TWItxBuff[4]:=Ye;
    TWItxBuff[5]:=Pattern;
    WriteTWIData(6);
  End chFillRect;

Procedure chDrawCircle(Xs,Ys,r,Pattern:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$13;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=r;
    TWItxBuff[5]:=Pattern;
    WriteTWIData(6);
  End chDrawCircle;

Procedure chFillCircle(Xs,Ys,r,Pattern:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$14;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=r;
    TWItxBuff[5]:=Pattern;
    WriteTWIData(6);
```

```
  End chFillCircle;

Procedure chSetPixel(Xs,Ys:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$15;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    WriteTWIData(3);
  End chSetPixel;
Procedure chClearPixel(Xs,Ys:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$16;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    WriteTWIData(3);
  End chClearPixel;

Procedure chXorPixel(Xs,Ys:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$17;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    WriteTWIData(3);
  End chXorPixel;

Procedure chSetLineMode(Mode:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$18;
    TWItxBuff[1]:=Mode;
    WriteTWIData(2);
  End chSetLineMode;

Procedure chDrawString(Xs,Ys,Sx,Sy,SRA:Byte;Text:String[10]);
  Var C1,C2 : Byte;
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$20;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=(Sx shl 6) or (Sy shl 4) or SRA;
    TWItxBuff[4]:=Length(Text);
    C2:=TWItxBuff[4];
    For C1:=1 to C2 do
      TWItxBuff[4+C1]:=Ord(Text[C1]);
      EndFor;
    WriteTWIData(5+C2);
  End chDrawString;

Procedure chDrawStringRel(Sx,Sy,SRA:Byte;Text:String[10]);
  Var C1,C2 : Byte;
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$21;
```

```
    TWItxBuff[3]:=(Sx shl 6) or (Sy shl 4) or SRA;
    TWItxBuff[4]:=Length(Text);
    C2:=TWItxBuff[4];
    For C1:=1 to C2 do
      TWItxBuff[4+C1]:=Ord(Text[C1]);
      EndFor;
    WriteTWIData(5+C2);
  End chDrawStringRel;


Procedure chSetTextJustify(StrFormat:Byte);
  Var Counter : Byte;
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$22;
    TWItxBuff[1]:=StrFormat;
    WriteTWIData(2);
  End chSetTextJustify;

Procedure chSetTextMode(StrMode:Byte);
  Var Counter : Byte;
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$23;
    TWItxBuff[1]:=StrMode;
    WriteTWIData(2);
  End chSetTextMode;

Procedure chSetTextBkGround(BkMode:Byte);
  Var Counter : Byte;
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$24;
    TWItxBuff[1]:=BkMode;
    WriteTWIData(2);
  End chSetTextMode;

Procedure chDrawBitmap(Xs,Ys,BNr:Byte);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$40;
    TWItxBuff[1]:=Xs;
    TWItxBuff[2]:=Ys;
    TWItxBuff[3]:=BNr;
    WriteTWIData(4);
  End chDrawBitmap;

Procedure chShowNorth1(X:Byte;Ang:Integer;Drive:Boolean);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$50;
    TWItxBuff[1]:=X;
    If Ang in [0..360]
      then
        TWItxBuff[2]:=Lo(Ang);
        TWItxBuff[3]:=Hi(Ang);
      EndIf;
```

```
    If Drive=false
      then
        TWItxBuff[4]:=$01;
      EndIf;
    WriteTWIData(5);
  End chShowNorth1;

Procedure chShowNorth2(X:Byte;Ang:Integer;Drive:Boolean);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$51;
    TWItxBuff[1]:=X;
    If Ang in [0..360]
      then
        TWItxBuff[2]:=Lo(Ang);
        TWItxBuff[3]:=Hi(Ang);
      EndIf;
    If Drive=false
      then
        TWItxBuff[4]:=$01;
      EndIf;
    WriteTWIData(5);
  End chShowNorth2;

Procedure chShowNorth3(X:Byte;Ang:Integer;Drive:Boolean);
  Begin
    ClrTWIBuffer;
    TWItxBuff[0]:=$52;
    TWItxBuff[1]:=X;
    If Ang in [0..360]
      then
        TWItxBuff[2]:=Lo(Ang);
        TWItxBuff[3]:=Hi(Ang);
      EndIf;
    If Drive=false
      then
        TWItxBuff[4]:=$01;
      EndIf;
    WriteTWIData(5);
  End chShowNorth3;
```