Philips PCD8544 (Nokia 3310) driver

A simple example of interfacing with the 84 x 48 pixel Nokia 3310 LCD.

With just five pins, 3.3V and ground and no other electronics (Some models of LCD may require a 1uf to 10uf capacitor between VOUT and GND pins, as output is distorted), the following sketch will write "Hello World!"

```
#define PIN_SCE   7
#define PIN_RESET 6
#define PIN_DC    5
#define PIN_SDIN  4
#define PIN_SCLK  3

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48

static const byte ASCII[][5] =
{
 {0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 "
,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 23 #
,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 24 $
,{0x23, 0x13, 0x08, 0x64, 0x62} // 25 %
,{0x36, 0x49, 0x55, 0x22, 0x50} // 26 &
,{0x00, 0x05, 0x03, 0x00, 0x00} // 27 '
,{0x00, 0x1c, 0x22, 0x41, 0x00} // 28 (
,{0x00, 0x41, 0x22, 0x1c, 0x00} // 29 )
,{0x14, 0x08, 0x3e, 0x08, 0x14} // 2a *
,{0x08, 0x08, 0x3e, 0x08, 0x08} // 2b +
,{0x00, 0x50, 0x30, 0x00, 0x00} // 2c ,
,{0x08, 0x08, 0x08, 0x08, 0x08} // 2d -
,{0x00, 0x60, 0x60, 0x00, 0x00} // 2e .
,{0x20, 0x10, 0x08, 0x04, 0x02} // 2f /
,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 30 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 31 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 32 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 33 3
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 34 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 35 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 36 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 37 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 38 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 39 9
,{0x00, 0x36, 0x36, 0x00, 0x00} // 3a :
,{0x00, 0x56, 0x36, 0x00, 0x00} // 3b ;
,{0x08, 0x14, 0x22, 0x41, 0x00} // 3c <
,{0x14, 0x14, 0x14, 0x14, 0x14} // 3d =
,{0x00, 0x41, 0x22, 0x14, 0x08} // 3e >
,{0x02, 0x01, 0x51, 0x09, 0x06} // 3f ?
,{0x32, 0x49, 0x79, 0x41, 0x3e} // 40 @
,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 41 A
,{0x7f, 0x49, 0x49, 0x49, 0x36} // 42 B
```

```
,{0x3e, 0x41, 0x41, 0x41, 0x22} // 43 C
,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 44 D
,{0x7f, 0x49, 0x49, 0x49, 0x41} // 45 E
,{0x7f, 0x09, 0x09, 0x09, 0x01} // 46 F
,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 47 G
,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 48 H
,{0x00, 0x41, 0x7f, 0x41, 0x00} // 49 I
,{0x20, 0x40, 0x41, 0x3f, 0x01} // 4a J
,{0x7f, 0x08, 0x14, 0x22, 0x41} // 4b K
,{0x7f, 0x40, 0x40, 0x40, 0x40} // 4c L
,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 4d M
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
,{0x7f, 0x09, 0x09, 0x09, 0x06} // 50 P
,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 51 Q
,{0x7f, 0x09, 0x19, 0x29, 0x46} // 52 R
,{0x46, 0x49, 0x49, 0x49, 0x31} // 53 S
,{0x01, 0x01, 0x7f, 0x01, 0x01} // 54 T
,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 55 U
,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 56 V
,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 57 W
,{0x63, 0x14, 0x08, 0x14, 0x63} // 58 X
,{0x07, 0x08, 0x70, 0x08, 0x07} // 59 Y
,{0x61, 0x51, 0x49, 0x45, 0x43} // 5a Z
,{0x00, 0x7f, 0x41, 0x41, 0x00} // 5b [
,{0x02, 0x04, 0x08, 0x10, 0x20} // 5c ¥
,{0x00, 0x41, 0x41, 0x7f, 0x00} // 5d ]
,{0x04, 0x02, 0x01, 0x02, 0x04} // 5e ^
,{0x40, 0x40, 0x40, 0x40, 0x40} // 5f _
,{0x00, 0x01, 0x02, 0x04, 0x00} // 60 `
,{0x20, 0x54, 0x54, 0x54, 0x78} // 61 a
,{0x7f, 0x48, 0x44, 0x44, 0x38} // 62 b
,{0x38, 0x44, 0x44, 0x44, 0x20} // 63 c
,{0x38, 0x44, 0x44, 0x48, 0x7f} // 64 d
,{0x38, 0x54, 0x54, 0x54, 0x18} // 65 e
,{0x08, 0x7e, 0x09, 0x01, 0x02} // 66 f
,{0x0c, 0x52, 0x52, 0x52, 0x3e} // 67 g
,{0x7f, 0x08, 0x04, 0x04, 0x78} // 68 h
,{0x00, 0x44, 0x7d, 0x40, 0x00} // 69 i
,{0x20, 0x40, 0x44, 0x3d, 0x00} // 6a j
,{0x7f, 0x10, 0x28, 0x44, 0x00} // 6b k
,{0x00, 0x41, 0x7f, 0x40, 0x00} // 6c l
,{0x7c, 0x04, 0x18, 0x04, 0x78} // 6d m
,{0x7c, 0x08, 0x04, 0x04, 0x78} // 6e n
,{0x38, 0x44, 0x44, 0x44, 0x38} // 6f o
,{0x7c, 0x14, 0x14, 0x14, 0x08} // 70 p
,{0x08, 0x14, 0x14, 0x18, 0x7c} // 71 q
,{0x7c, 0x08, 0x04, 0x04, 0x08} // 72 r
,{0x48, 0x54, 0x54, 0x54, 0x20} // 73 s
,{0x04, 0x3f, 0x44, 0x40, 0x20} // 74 t
,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 75 u
,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 76 v
,{0x3c, 0x40, 0x30, 0x40, 0x3c} // 77 w
,{0x44, 0x28, 0x10, 0x28, 0x44} // 78 x
,{0x0c, 0x50, 0x50, 0x50, 0x3c} // 79 y
,{0x44, 0x64, 0x54, 0x4c, 0x44} // 7a z
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ←
,{0x78, 0x46, 0x41, 0x46, 0x78} // 7f →
```

```
};

void LcdCharacter(char character)
{
  LcdWrite(LCD_D, 0x00);
  for (int index = 0; index < 5; index++)
  {
    LcdWrite(LCD_D, ASCII[character - 0x20][index]);
  }
  LcdWrite(LCD_D, 0x00);
}

void LcdClear(void)
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}

void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);
  LcdWrite(LCD_C, 0x21 );  // LCD Extended Commands.
  LcdWrite(LCD_C, 0xB1 );  // Set LCD Vop (Contrast).
  LcdWrite(LCD_C, 0x04 );  // Set Temp coefficent. //0x04
  LcdWrite(LCD_C, 0x14 );  // LCD bias mode 1:48. //0x13
  LcdWrite(LCD_C, 0x20 );  // LCD Basic Commands
  LcdWrite(LCD_C, 0x0C );  // LCD in normal mode.
}

void LcdString(char *characters)
{
  while (*characters)
  {
    LcdCharacter(*characters++);
  }
}

void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

void setup(void)
{
  LcdInitialise();
  LcdClear();
  LcdString("Hello World!");
}

void loop(void)
```

```
{
}
```

A simple modified example of interfacing with the Nokia 3310 LCD that will print characters at an XY position on LCD and also will draw lines on LCD.

```
/*
This Code has extra features
including a XY positioning function on Display
and a Line Draw function on Nokia 3310 LCD
It is modded from the original
http://playground.arduino.cc/Code/PCD8544
*/
// Mods by Jim Park
// jim(^dOt^)buzz(^aT^)gmail(^dOt^)com
// hope it works for you
#define PIN_SCE   7  // LCD CS  .... Pin 3
#define PIN_RESET 6  // LCD RST .... Pin 1
#define PIN_DC    5  // LCD Dat/Com. Pin 5
#define PIN_SDIN  4  // LCD SPIDat . Pin 6
#define PIN_SCLK  3  // LCD SPIClk . Pin 4
                     // LCD Gnd .... Pin 2
                     // LCD Vcc .... Pin 8
                     // LCD Vlcd ... Pin 7

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48
#define LCD_CMD   0

int a = 0;

static const byte ASCII[][5] =
{
 {0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 "
,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 23 #
,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 24 $
,{0x23, 0x13, 0x08, 0x64, 0x62} // 25 %
,{0x36, 0x49, 0x55, 0x22, 0x50} // 26 &
,{0x00, 0x05, 0x03, 0x00, 0x00} // 27 '
,{0x00, 0x1c, 0x22, 0x41, 0x00} // 28 (
,{0x00, 0x41, 0x22, 0x1c, 0x00} // 29 )
,{0x14, 0x08, 0x3e, 0x08, 0x14} // 2a *
,{0x08, 0x08, 0x3e, 0x08, 0x08} // 2b +
,{0x00, 0x50, 0x30, 0x00, 0x00} // 2c ,
,{0x08, 0x08, 0x08, 0x08, 0x08} // 2d -
,{0x00, 0x60, 0x60, 0x00, 0x00} // 2e .
,{0x20, 0x10, 0x08, 0x04, 0x02} // 2f /
,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 30 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 31 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 32 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 33 3
```

```
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 34 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 35 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 36 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 37 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 38 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 39 9
,{0x00, 0x36, 0x36, 0x00, 0x00} // 3a :
,{0x00, 0x56, 0x36, 0x00, 0x00} // 3b ;
,{0x08, 0x14, 0x22, 0x41, 0x00} // 3c <
,{0x14, 0x14, 0x14, 0x14, 0x14} // 3d =
,{0x00, 0x41, 0x22, 0x14, 0x08} // 3e >
,{0x02, 0x01, 0x51, 0x09, 0x06} // 3f ?
,{0x32, 0x49, 0x79, 0x41, 0x3e} // 40 @
,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 41 A
,{0x7f, 0x49, 0x49, 0x49, 0x36} // 42 B
,{0x3e, 0x41, 0x41, 0x41, 0x22} // 43 C
,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 44 D
,{0x7f, 0x49, 0x49, 0x49, 0x41} // 45 E
,{0x7f, 0x09, 0x09, 0x09, 0x01} // 46 F
,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 47 G
,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 48 H
,{0x00, 0x41, 0x7f, 0x41, 0x00} // 49 I
,{0x20, 0x40, 0x41, 0x3f, 0x01} // 4a J
,{0x7f, 0x08, 0x14, 0x22, 0x41} // 4b K
,{0x7f, 0x40, 0x40, 0x40, 0x40} // 4c L
,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 4d M
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
,{0x7f, 0x09, 0x09, 0x09, 0x06} // 50 P
,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 51 Q
,{0x7f, 0x09, 0x19, 0x29, 0x46} // 52 R
,{0x46, 0x49, 0x49, 0x49, 0x31} // 53 S
,{0x01, 0x01, 0x7f, 0x01, 0x01} // 54 T
,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 55 U
,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 56 V
,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 57 W
,{0x63, 0x14, 0x08, 0x14, 0x63} // 58 X
,{0x07, 0x08, 0x70, 0x08, 0x07} // 59 Y
,{0x61, 0x51, 0x49, 0x45, 0x43} // 5a Z
,{0x00, 0x7f, 0x41, 0x41, 0x00} // 5b [
,{0x02, 0x04, 0x08, 0x10, 0x20} // 5c ¥
,{0x00, 0x41, 0x41, 0x7f, 0x00} // 5d ]
,{0x04, 0x02, 0x01, 0x02, 0x04} // 5e ^
,{0x40, 0x40, 0x40, 0x40, 0x40} // 5f _
,{0x00, 0x01, 0x02, 0x04, 0x00} // 60 `
,{0x20, 0x54, 0x54, 0x54, 0x78} // 61 a
,{0x7f, 0x48, 0x44, 0x44, 0x38} // 62 b
,{0x38, 0x44, 0x44, 0x44, 0x20} // 63 c
,{0x38, 0x44, 0x44, 0x48, 0x7f} // 64 d
,{0x38, 0x54, 0x54, 0x54, 0x18} // 65 e
,{0x08, 0x7e, 0x09, 0x01, 0x02} // 66 f
,{0x0c, 0x52, 0x52, 0x52, 0x3e} // 67 g
,{0x7f, 0x08, 0x04, 0x04, 0x78} // 68 h
,{0x00, 0x44, 0x7d, 0x40, 0x00} // 69 i
,{0x20, 0x40, 0x44, 0x3d, 0x00} // 6a j
,{0x7f, 0x10, 0x28, 0x44, 0x00} // 6b k
,{0x00, 0x41, 0x7f, 0x40, 0x00} // 6c l
,{0x7c, 0x04, 0x18, 0x04, 0x78} // 6d m
,{0x7c, 0x08, 0x04, 0x04, 0x78} // 6e n
,{0x38, 0x44, 0x44, 0x44, 0x38} // 6f o
,{0x7c, 0x14, 0x14, 0x14, 0x08} // 70 p
```

```
,{0x08, 0x14, 0x14, 0x18, 0x7c} // 71 q
,{0x7c, 0x08, 0x04, 0x04, 0x08} // 72 r
,{0x48, 0x54, 0x54, 0x54, 0x20} // 73 s
,{0x04, 0x3f, 0x44, 0x40, 0x20} // 74 t
,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 75 u
,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 76 v
,{0x3c, 0x40, 0x30, 0x40, 0x3c} // 77 w
,{0x44, 0x28, 0x10, 0x28, 0x44} // 78 x
,{0x0c, 0x50, 0x50, 0x50, 0x3c} // 79 y
,{0x44, 0x64, 0x54, 0x4c, 0x44} // 7a z
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ←
,{0x00, 0x06, 0x09, 0x09, 0x06} // 7f →
};



void LcdCharacter(char character)
{
  LcdWrite(LCD_D, 0x00);
  for (int index = 0; index < 5; index++)
  {
    LcdWrite(LCD_D, ASCII[character - 0x20][index]);
  }
  LcdWrite(LCD_D, 0x00);
}

void LcdClear(void)
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}

void LcdInitialise(void)
{
  pinMode(PIN_SCE,   OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC,    OUTPUT);
  pinMode(PIN_SDIN,  OUTPUT);
  pinMode(PIN_SCLK,  OUTPUT);

  digitalWrite(PIN_RESET, LOW);
 // delay(1);
  digitalWrite(PIN_RESET, HIGH);

  LcdWrite( LCD_CMD, 0x21 );  // LCD Extended Commands.
  LcdWrite( LCD_CMD, 0xBf );  // Set LCD Vop (Contrast). //B1
  LcdWrite( LCD_CMD, 0x04 );  // Set Temp coefficent. //0x04
  LcdWrite( LCD_CMD, 0x14 );  // LCD bias mode 1:48. //0x13
  LcdWrite( LCD_CMD, 0x0C );  // LCD in normal mode. 0x0d for inverse
  LcdWrite(LCD_C, 0x20);
  LcdWrite(LCD_C, 0x0C);
}

void LcdString(char *characters)
{
  while (*characters)
  {
    LcdCharacter(*characters++);
  }
}

void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

// gotoXY routine to position cursor
// x - range: 0 to 84
// y - range: 0 to 5

void gotoXY(int x, int y)
{
  LcdWrite( 0, 0x80 | x);  // Column.
  LcdWrite( 0, 0x40 | y);  // Row.

}



void drawLine(void)
{
  unsigned char  j;
   for(j=0; j<84; j++) // top
        {
          gotoXY (j,0);
          LcdWrite (1,0x01);
  }
  for(j=0; j<84; j++) //Bottom
        {
          gotoXY (j,5);
          LcdWrite (1,0x80);
  }

  for(j=0; j<6; j++) // Right
        {
          gotoXY (83,j);
          LcdWrite (1,0xff);
  }
        for(j=0; j<6; j++) // Left
        {
          gotoXY (0,j);
          LcdWrite (1,0xff);
  }

}


void setup(void)
{

 LcdInitialise();
  LcdClear();
```

```
}

void loop(void)
{
  // Display some simple character animation
  //
  int a,b;
  char Str[15];
  // Draw a Box
  for(b=1000; b>0; b--){
  drawLine();
  for(a=0; a<=5 ; a++){
  gotoXY(4,1);
  // Put text in Box
  LcdString ("TestDisplay");
  gotoXY(24,3);
  LcdCharacter('H');
  LcdCharacter('E');
  LcdCharacter('L');
  LcdCharacter('L');
  LcdCharacter('O');
  LcdCharacter(' ');
  LcdCharacter('=');
  // Draw + at this position
  gotoXY(10,3);
  LcdCharacter('=');
  delay(500);
  gotoXY(24,3);
  LcdCharacter('h');
  LcdCharacter('e');
  LcdCharacter('l');
  LcdCharacter('l');
  LcdCharacter('o');
  LcdCharacter(' ');
  LcdCharacter('-');
  // Draw - at this position
  gotoXY(10,3);
  LcdCharacter('-');
  delay(500);
  }
  }
}
```

Another example which takes a bitmap via the serial port.

```
#define SER_BAUD  9600

#define PIN_SCE   7
#define PIN_RESET 6
#define PIN_DC    5
#define PIN_SDIN  4
#define PIN_SCLK  3

#define LCD_C     LOW
#define LCD_D     HIGH

void LcdClear(void)
{
  for (int index = 0; index < 84 * 48 / 8; index++)
```

```
    {
      LcdWrite(LCD_D, 0x00);
    }
}

void LcdInitialise(void)
{
  pinMode(PIN_SCE, OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC, OUTPUT);
  pinMode(PIN_SDIN, OUTPUT);
  pinMode(PIN_SCLK, OUTPUT);
  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);
  LcdWrite(LCD_C, 0x22);
  LcdWrite(LCD_C, 0x0C);
  LcdClear();
}

void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

void SerialInitialise(void) {
  Serial.begin(SER_BAUD);
}

void SerialRead(void) {
  if (Serial.available())
  {
    while (Serial.available())
    {
      LcdWrite(LCD_D, Serial.read());
    }
  }
}

void setup(void)
{
  LcdInitialise();
  SerialInitialise();
}

void loop(void)
{
  SerialRead();
}
```

And here's some sample VB.NET code to send bitmaps (loaded from file and generated on the fly) to the Arduino's serial port.

```
Serial_Write(New Bitmap("84x48.bmp"))
Serial_Write(Format(Now(), "HHmm"))
```

```
Private Sub Serial_Write(ByVal theString As String)
    Dim theBitmap As Bitmap = New Bitmap(84, 48)
    Dim theFont As Font = New Font("Courier", "24", FontStyle.Bold,
GraphicsUnit.Pixel)
    Dim theGraphics As Graphics = Graphics.FromImage(theBitmap)
    theGraphics.TextRenderingHint =
Drawing.Text.TextRenderingHint.ClearTypeGridFit
    theGraphics.FillRectangle(Brushes.White, 0, 0, theBitmap.Width,
theBitmap.Height)
    theGraphics.DrawString(theString, theFont, Brushes.Black,
((theBitmap.Width - theGraphics.MeasureString(theString, theFont).Width) /
2), ((theBitmap.Height - theGraphics.MeasureString(theString,
theFont).Height) / 2))
    Serial_Write(theBitmap)
End Sub

Private Sub Serial_Write(ByVal theBitMap As Bitmap)
    Dim theByteArray() As Byte = New Byte() {}
    For theWidth As Integer = 0 To 83
        For theHeight As Integer = 0 To 5
            ReDim Preserve theByteArray(theByteArray.GetUpperBound(0) + 1)
            For theBit As Integer = 0 To 7
                If theBitMap.GetPixel(theWidth, (theHeight * 8) + theBit).R
Then
                        theByteArray(theByteArray.GetUpperBound(0)) =
theByteArray(theByteArray.GetUpperBound(0)) And Not (2 ^ theBit)
                Else
                        theByteArray(theByteArray.GetUpperBound(0)) =
theByteArray(theByteArray.GetUpperBound(0)) Or (2 ^ theBit)
                End If
            Next
        Next
    Next
    SerialPort.Open()
    SerialPort.Write(theByteArray, 0, theByteArray.Length)
    SerialPort.Close()
End Sub
```

Here is a Java version similar to the VB.net code above except that output goes to standard out (allows copy/paste of hex values into your sketch)

```
import java.awt.image.BufferedImage;
import java.io.File;

import javax.imageio.ImageIO;

public class BitmapToLCD {
        public static final int WIDTH = 84;
        public static final int HEIGHT = 48;

        public static void main(String[] args) {
                File f = new File(args[0]);

                try {
                        // Read from a file
                BufferedImage image = ImageIO.read(f);
                // Get all the pixels
                int w = image.getWidth(null);
                int h = image.getHeight(null);
                int[] rgbs = new int[w*h];
                image.getRGB(0, 0, w, h, rgbs, 0, w);

                //iterate through each pixel (and reduce to binary)
                int row = 0;
                int col = 0;
                int bit = 0;
                byte[][] ba = new byte[HEIGHT/8][WIDTH];
                for (int i = 0; i < rgbs.length; i++){
                        byte val = (byte)(rgbs[i] & 0x01);
                        //invert the value
                        val = (byte) (val == 1 ? 0:1);
                                ba[row][col] |= val << bit;

                        //next column
                        col++;

                        //next bit
                        if (col >=WIDTH) {
                                col = 0;
                                bit++;
                        }

                        //next data row
                        if (bit >=8){
                            bit = 0;
                           for (int x= 0; x < WIDTH; x++){
                                   String s =
Integer.toHexString((byte)ba[row][x]);
                                   //Do some formatting
                                   if (s.length() > 2) {
                                           s = s.substring(s.length() - 2);
                                   }
                                   while (s.length() < 2){
                                           s = "0" + s;
                                   }
                               System.out.print( "0x" + s + ",");
                           }
                           System.out.println("");
                           row++;
                        }

                }
                } catch (Exception e) {
                        e.printStackTrace();
                }

        }
}
```