PCD8544 display library usage

The PCD8544 display library is used for interfacing with display based on the Philips PCD8544 controller-based LCDs.

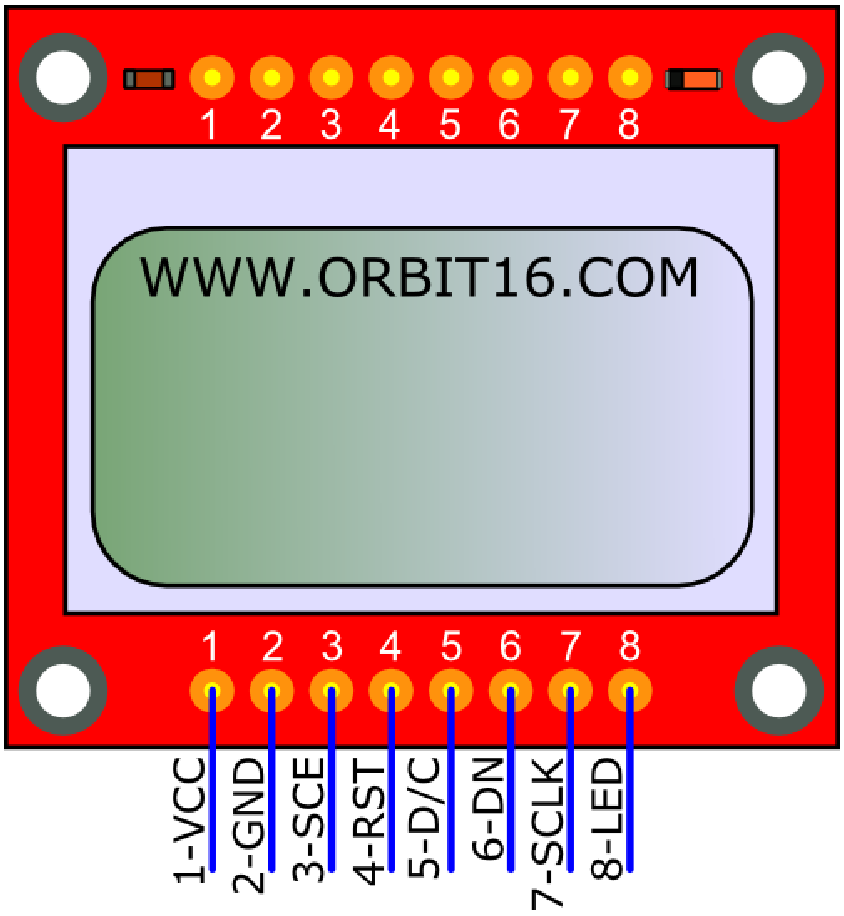
This kind of display is used in some mobile phones such as Nokia 3310 and Nokia 5110. It is a very inexpensive display, operates at 3.3V so it is unnecessary to adapt levels to use this display with the ORbit16™.

Compared to "classical" HD44780 display, the PCD8544 can be considered as a 14 chars x 6 rows text display but have also graphic capabilities.

The library provides a font table since the PCD8544 controller doesn't have one.

Electrical connections

All the displays sold as "Nokia 3310" or "Nokia 5110" display have a connection like the following:



There are 2 rows of pin, connected together (1 with 1, 2 with 2 and so on).

The TOP part of the display is the one with a capacitor and a diode near the pin row (a capacitor, named C1 to the left and a diode, named D1, to the right).

VCC must be connected to a 3.3V source

GND must be connected to ground, or 0V reference

SCE is the Controller Enable line. It is active low, so to enable the display you must tie this pin to GND.

> TIP:
> *If you want to control more than one display, you must drive the SCE pins individually. For only one display is unnecessary drive this, so it can be tied to GND.*

RST is the Reset pin. It is used during initialization routine.

D/C is the Data/Command pin. When this pin is tied to GND, the PCD8544 controller will read the data as command to execute. When this pin is tied to VCC, the controller will read the data as bytes to show on the LCD.

DN is the serial Data In pin. Through this pin we'll transmit bytes to show on the display or commands.

SCLK is the clock signal pin.

LED pin is used for the backlight, you can leave this unconnected or connected to 3.3V (through a small resistor such as 100÷560Ω). You can also drive this pin using a transistor and another I/O of the ORbit16™.

You need **only 4 I/Os** of the ORbit16™ to make this display work (RST, D/C, DN and SCLK), so make care the I/Os you're using are not used as analog inputs.

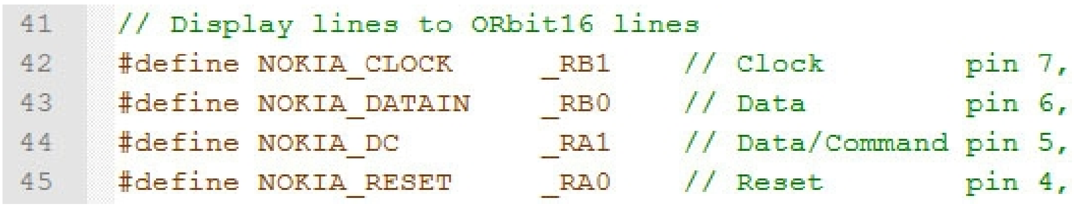
Don't forget to connect SCE pin to GND or the display will not work!

Library settings

Open the file **`nlcd.h`** with a good text editor (such as Notepad++).

STEP 1

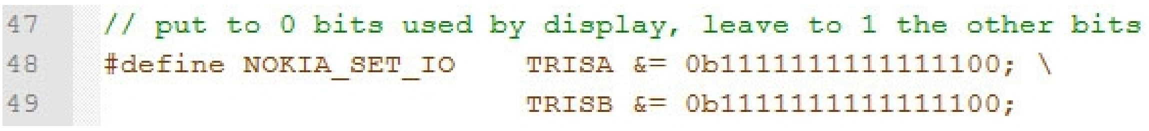
Go to row 42 to set the I/O lines used to control the display:



```
41  // Display lines to ORbit16 lines
42  #define NOKIA_CLOCK      _RB1    // Clock        pin 7,
43  #define NOKIA_DATAIN     _RB0    // Data         pin 6,
44  #define NOKIA_DC         _RA1    // Data/Command pin 5,
45  #define NOKIA_RESET      _RA0    // Reset        pin 4,
```

STEP 2

Go to row 48 to set the pin used to drive display as outputs in the TRIS registers:



```
47  // put to 0 bits used by display, leave to 1 the other bits
48  #define NOKIA_SET_IO     TRISA &= 0b1111111111111100; \
49                           TRISB &= 0b1111111111111100;
```

as remarks says: put a 0 value in the place where the bits are used to drive the display and an 1 value to the other bits.

Library usage

At the beginning of your program, outside the main function, in the "include" section, you must write:

```
#include "nlcd.c"
```

In main() function of your program, outside the infinite loop, you must write:

```
nlcd_init();
```

Now you can use the following high-level functions:

```
void nlcd_clear(void);
void nlcd_goto(unsigned char x, unsigned char y);
void nlcd_putch(unsigned char c);
void nlcd_puts(char *s);
void nlcd_putun(unsigned int c);
void nlcd_putsn(signed int c);
void nlcd_bitmap(const unsigned char *bitmap, unsigned char x,
unsigned char y);
void nlcd_icon(const unsigned char *icon);
```

nlcd_clear
is used to clear the entire lcd area.

usage example:
```
ncld_clear(); // clears display content and locate the cursor to
0,0 position (the upper-left corner of the display)
```

ncld_goto
is used to locate the cursor to x,y position.
x value must be between 0 and 83, y value must be between 0 and 5. the X value represents a pixel along the lcd width, the Y value represents a row index along the lcd height: each row is 8 pixel high.

usage example:
```
nlcd_goto(0,5); // locate the cursor at the beginning of the last
row
```

nlcd_putch
writes a single char on the display. Refer to the ascii table that starts from row 68 of the nlcd.h file and to the custom constant definitions from row 58

usage example:
```
nlcd_putch('A'); // writes: A
nlcd_putch(0x40); // writes: @
ncld_putch(DEG); // writes: ° (the degree symbol)
```

ncld_puts
writes a string on the display.

usage example:
```
nlcd_puts("Hello World!"); // writes: Hello World!
char p[]="Bye bye!";
nlcd_puts(p); // writes: Bye bye!
```

nlcd_putun
writes an unsigned integer on the display (values between 0 and 65535).

usage example:
```
nlcd_putun(1234); // writes: 1234
unsigned int a=14;
nlcd_putun(a); // writes: 14
```

nlcd_putsn
write a signed integer on the display (values between -32768 and 32767).

usage example:
```
nlcd_putsn(-1542); // writes: -1542
signed int b=674;
nlcd_putsn(b); // writes: 674
```

nlcd_bitmap
Paint a bitmap defined as an array stored in the ROM starting from x,y coordinates (0≤x≤83 0≤y≤5). The bitmap width is the first element of the array and the bitmap heigth is the second element of the array. Width and Height are expressed as in pixels. Please refer to document:

Using_LCDAssistant_with_ORbit16_PCD8544_display_library.pdf

usage example:
```
nlcd_bitmap(bitmap,0,0); // paint the bitmap, defined in the
"bitmap" array starting from upper-left corner
```

nlcd_icon
Paint an icon defined as an array stored in the ROM at the actual cursor location. An icon is 8 pixel high. Please refer to document:

Using_LCDAssistant_with_ORbit16_PCD8544_display_library.pdf

usage example:
```
nlcd_icon(icon); // paint the icon, defined in the "icon" array
starting from the current cursor location
```

How can I write float numbers using the library?

The easiest way to do this is to include the stdio library at the beginning of your program and using the **`sprintf()`** function:

```
#include <stdio.h>
```

now you can use the **`sprintf()`** to put a float number into an array defined into the RAM using the %f *format specifier*. Example:

```
double myFloat=12.3456; // float number to print
char bufferToPrint[40]; // buffer, must be long enough to contain
the float number
sprintf(bufferToPrint,"Float is:%f",myFloat); // replaces the %f
with the float value into the buffer
nlcd_puts(bufferToPrint); // writes: Float is:12.3456
```