TerOled

Include files to replace the heavy GFX lib and connect SSD1306 on any pins

Sources on https://github.com/nicoud/TerOled

Oled SSD1306 is most of the time used with Adafruit GFX lib. Many characters to type, long compile and download time, 10k of code min and 1.2k of variables. And it not easy to add functions. Functions you may succeed to add will be large.

We propose a set of include files to fill your needs in an optimal way. But understand it is not a tablet, the SSD1306 is slow and write only. It is great for small applications and as a debugging help you fill find a corner for it.
Note that we do not use a bit map. We can, we have a doc to use the additional files not documented here. There are too many advantages doing direct write to the Oled screen.

OK, you want to display useful data on one Oled. If your application use an Oled on SPI or I2C, you can add an Oled on any 2 pins and use our soft for debugging os showing additional information (no interactions, interrupt compatible).
You decide on which pin to connect the Oled, and you adapt file Bb7654Oled.h file if not I2C pins. As for most libs, you have an associated set-up.
Then you surely need to display text and dots, draw sprites. There is an include file and a clear documentation with examples ready to load.
You need to display numbers and we propose two options; TerOled.h is fully compatible with TerSer, that replaces the heavy Arduino Serial.print facilities.
The other is our 2016 solution on github/nicoud/Oled that have double size character. You are compatible it you replace TerSet.h by TerNum.h.
You many need more graphics, you plan to play ping-pong or another game. Look at the available library files, they are just C programs you can modify and complete. Then, contribute to the Github repo.

Beside the compactness, you get four significant advantages:

1) non-significative zeros are kept, so numbers stay aligned in listings.
2) you can understand the code, modify it, add functions. Need to display in octal? 10 lines of code to add, after looking at the similar hex function.
3) you can easily input data form the terminal keyboard to interact with your program.
4) files are written in portable simple C. Arduino is only used for convenience and only 10 instructions are specific to the AVR328 display port. Easy to port on your preferred micro if you understand its display port and how to change port direction

Here, we propose not to use I2C hardware facilities and widely used libs (see ….) . We bit-bang on any 2 pins, preferably on the same port.

Oled SSD1306

The Oled receive information via two I2C lines. One can only write to the 1306, which is a major weakpoint, but simplify significantly the software. I2C is very simple to implement since you

need only to write bytes: start - address – data-data- ..- stop. Any 2 pins with tree-state control can be used: either you output a zero, or you set the pin as input and the I2C pull-up makes it a one.

Acknowledge value is ignored. But a function can test at power- up if the Oled is connected. We do not see the need to do the I2C transfer by interrupt, but the routines can be interrupted for milliseconds.

The point now is to define correctly on which pins the Oled is connected and use names that will not conflict with other names on your application.

I2C with AtMega328 Twi

Arduino wire lib is used on Adafruit GFX lib. It is a nonsense since one can only write to SSD1306. TwiOled.h lib is only 20 lines of code.

I2C on any pin

I2C is slow, and bit-bang I2C functions are easy to program, of course not using the too slow "digitalWrite". Our definition file supposes the 2 pins are on the same port. The first 5 lines define the wiring, no other change is required in the program.. Ck1 Ck0 Da1 Da0 are the name used further, they cannot be used for other purposes, same as for the function names we have defined. Be carefull, Start and Stop are frequently used name, we have reserved them for I2C.

```
#define Ddr  DDRD
#define Port PORTD
#define Pin  PIND
#define bCk 0  // Oled128x
#define bDa 1  // pin 4
#define Ck1  bitClear (Ddr,bCk)
#define Ck0  bitSet (Ddr,bCk)
#define Da1  bitClear (Ddr,bDa)
#define Da0  bitSet (Ddr,bDa)

void SetupBb() {   // will be called inside Arduino Setup or at the beginning of a pure C prog.
  Ddr  |=  (1<<bCk|1<<bDa) ;
  Port &= ~(1<<bCk|1<<bDa) ;
}
```

Note that the Oled can be powered from processor pins, Example of the required definitions are given in TestBb7654Oled.ino

Naming

TerSer.h include file (https://www.didel.com/TerSer.pdf) replaces the heavy Serial.print() functions with several advantages. TerOled is compatible, but due to the greater complexity handling a bit map, instead of just sending Ascii characters, several include files are proposed. Also, Didel document Oled on many document with include files we had good reason to change to be fully compatible with TerOled, if your objective is to use the SSD1306 to avoid the need of a PC or tablet.

We hence keep the old names OledPix.h and OledMap.h for the 2017 set of fonctions (text, 8 and 16 bit variables, simple graphic), and the communication files OledTwi.h, OledBb'xx'.h.

We have improved the character generator and the display functions on the new files that are like a meccano, you take the functions you need to minimize download time:

TerOled.h Same as TerSer.h, except UART/Oled reference. 2 files must be included before calling TerOled to handle I2C transfers and Oled basic control :

PixOled.h (MapOled.h is not recommended) with the character generator and Car() function.

TwiOled.h or Bb'xx'Oled.h that say how the SSD1306 is wired.

Additional include files are available, see https://www.didel.com/AllOled.zip

NumOled.h is a simplified version of TerOled, with only the 2017 functions.

BigOled.h with the 2017 double size functions.

LineCircleOled.h if you need to draw inclined lines (MapOled only).

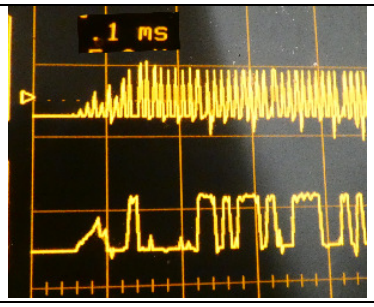
PongOled.h if you want to develop the Pong demo of the Edu-C.

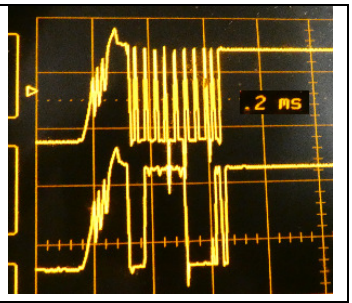
Presence test
On our Oled softwares, SSD1306 is supposed to be connected and operational responding to address 0x78(8bits) = 0x3C(7bits). If not, the processor block on the lack of achnowledge.
Function I2cIsHere (8bit addr); returns a 1 if responding. Its up to you to call that function before Oled setup and decide appropriate action.
For testing if an Oled is good, use TestGencarTwi or TestGencarBb7654 (update pins if others).
If you are not sure if the Oled respond to the address 0x78, or if you want to display on TerSer all the addresses that are recognized on the pins, use TestAvailableOnTwi or TestAvailableOnBb7654.

Note on MapOled
MapOled is not supported any more. You can easily adapt OledMap if you need. See our 2016 documentation https://www.didel.com/OledFunctions.pdf.

Note about powerup

A correct I2C access starts with Sda and Sck high. At power up after initializing the lines as floating, the pull-up takes some time to load the capacitive charge of the Oled, as shown by the oscilloscope.
A delay of 300us is required till power is here. The setup must initialize the direction, the level (Sda Scl high) and wait 300 us.





PixOled.h and NumOled.h – understant the evolution since OledPix
OledPix.h and OledMap.h are documented on https://github.com/nicoud/Oled and were designed as several h files, usually put together. OledPix.h draw directly on the screen, no buffer. You have to take care where texts and variables are displayed if you do not want to clear the screen too frequently, since it takes 50ms. Writing a number takes 5 to 10 ms.
OledPix include several functions that are different to the TerOled ones. Hence OledPix has been split in TerPix and TerNum, the pair being partly compatible with TerOled.
Do not be confused if you are a newcomer. Your world is TerOled plus TwiOled or BBxxOled.

If you have used TerSer or Serial terminal, the difference with an Oled is the text is not scrolling (we do not want it). Texts and variables must to be positioned with the `LiCol (li,col);` function. See https://www.didel.com/OledBbLib.pdf for a quick view.
All our links can be found on https://www.didel.com/prof/Oled.html

Functions available on OledPix and TerNum are

`Licol(li,co);`
`Text("UP and low");`
`Bin8(v);`
`Hex8(v);`
`Hex16(v);`
`Dec8(v16);  unsigned`
`Dec9999(v16)  <9999`
	Double height, lines 1 to 7 only
	`BigText("UP and low");`
	`BigBin8(v);`
	`BigHex8(v);`
	`BigHex16(v);`
	`BigDec8(v);`
	`BigDec9999(v)`
		Pixel (0,0) is on top left
		`Hline (x);`
		`Vline (y);`
		`Dot (x,y);`
		`DDot (x,y);`
		`MySprite (ss);`

Upper case are 5x7 inside 6x8, lower case and digits 4x7 inside 5x8, space after numbers and pointer follows.
Pointer is not updated after texts (may come).
The major change with TerOled is Dec9999(v16) is now Dec16(v16) without limitation.

TerOled.h
Do not get confused if you are new to *Oled*.h documentation. See TestTerOled.ino and play to change what is displayed. You cannot have "big" texts or numbers. It is a debugging tool and must be compact.

TerOled.h functions applies to 8 and 16 bits (32 bits in hexa) and signs are shown if variables are of type int. Appreciate how binary numbers are displayed.
The 4 modes take place in the source code, but the compiler create code for what is used only. Normal(); is the default mode. Zeros() will display non significative zeros.

TerOled functions	Adafruit-GFX equivalent
`Car('s');`	`Display.print('s');`
`Text("xx");`	`Display.print("xx");`
`Bin8(v8);`	`Display.print(v8,BIN);`
`Bin8(v8);`	`Display.print(v8,BIN);`
`Hex8(v8);`	`Display.print(v8,HEX);`
`Dec8(v8);`	`Display.print(v8);`
`Bin16(v16);`	`Display.print(v16,BIN);`
`Hex16(v16);`	`Display.print(v16,HEX);`
`Dec16(v16);`	`Display.print(v16);`

TerOled is compatible with TerSer, the Arduino Serial replacement

See https://git.boxtec.ch/didel/TerSer and https://www.didel.com/TerSer.pdf

TerOled functions list

`Car(cc);` Display a char -- part of PixOled.h (or MapOled.h) file

`Text("abcd");` Display a text and add a space.

`Bin8(v);` Display the 8 low bits of the variable v, converted to integers (all add a space .)

`Bin8(33);` → `00100001.`

`Bin16(v);` Display the 16 bits of the variable v, with a dot in the middle

`Bin8(1035);` → `00000100.00001011.`

`Hex8(v);` Display the 2 nibbles of the variable v, converted to integers

`Hex8(33);` → `21.`

`Hex16(v);` Display the 4 nibbles of the variable v, converted to integers

`Hex16(1035);` → `040B.`

`Hex32(v);` Display the 8 nibbles of the variable v, converted to integers

`Hex32(260);` → `00000104.`

`Dec8(v);` Display the variable v, with its signs if signed and non-significative space or zeros

`Dec8(33);` → `+.33.` or `+033.` or `.+33.`

`Dec16(v);` Display the variable v, with its signs if signed and non-significative space or zeros

`Dec16(1035);` → `+.1035.` or `+01035.` or `.+1035.`

`Dec8u(unsigned);` `Dec8s(signed);` Shorter code

`Dec16u(unsigned);` Dec16s(signed); Shorter code

`ShowS();` `+..nnn.` best for tabular data

`ShowZ();` `+00nnn.` if you prefer

PixOled functions (PixOled.h must be charged before TerOled.h)

`Car(cc);` Send code `cc` to the UART

`Car(' ');` `Car(32);` `Car(0x20);` All three print a space

`Text("abcd");` Send the text and add a space.

`DoubleH(); SimpleH();` For 128x32 and 128x64 SSD1306 screen

`LiCol(li,col);` Set the cursor at line li and column col

`Clear();` Clear all screen

`Dot(x,y);` Set a dot at coordinate (x<128,y<64) (clear other bits of same byte)

`Hline(y); Vline(x); Horizontal and vertical line.`

`Sprite(Name); Table within progmem`

`MySprite(TableInRAM)`

Note: PixOled.h is also the base for the set of modular files compatible with the 2016-2019 Oled software – see **www.didel.com/Oled.html**

I2C transfer variants

TwiOled.h Avr328 I2C pins. Send us what you made work on other microcontrollers

Bb7654Oled.h Power and Ck Da on Arduino pins 7 6 5 4.

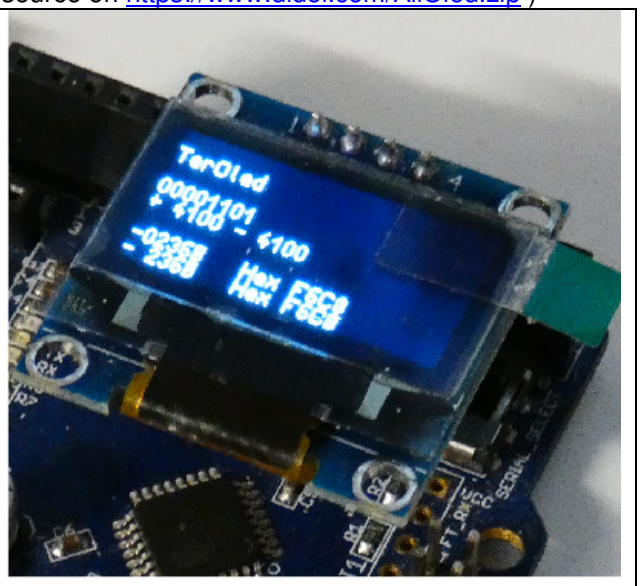
Bb23Oled.h Data on pins A2(16) and A3(17)

5 lines to modify if same port. Possible on different ports.

Bitbanging is not critical and can be used on 1 MHz Tinys. Adapt the delay function.

Example SSD1306 directly inserted on Arduino pins 7654 (source on https://www.didel.com/AllOled.zip)

```
//TestTerOled.ino 190317 2212
#include "Bb7654Oled.h"
#include "PixOled.h"
#include "TerOled.h"
void setup(){
  SetupBb7654Oled();
  SetupPixOled();
//  DoubleH();
}
byte val=13; int v16s=0xEFFC;
void loop() {
  LiCol(1,0); Text("TerOled");
  LiCol(3,0); Bin8(val);
  LiCol(4,0); Dec16(-v16s); Dec16(v16s);
  while (1) {
     LiCol(6,0); ShowZ(); Dec16(v16s);
        Text("  Hex ");Hex16(v16s);
     LiCol(7,0); HideZ(); Dec16(v16s);
         Text("  Hex ");Hex16(v16s);
     v16s++;
  }
}
```



Examples and xxOled.h files can also be found on www.didel.com/TerOled.zip

jdn 190403