# SSD1306-OLED 128x32 module driver for RaspberryPi

kobayashikorio@gmail.com
2017/8/22

Thanks to “http://oleddisplay.squix.ch/#/home” to convert bit-map font for SSD1306

## Prepare font bit-map data of 7 bit height

copy “oleddisplay” produced text then paste as to following settings a0 and b0

```
(* chatacter " shoud be changed to escaped style \" *)
a0 = "0x00, //''
0x49,0x24, //'!'
0xF0, //'\"'
    0x31,0xE5,0x1E,0x50, // '#'
    0x27,0xA7,0xF2,0x00, // '$'
    0xE2,0xC6,0x8D,0x14, // '%'
    0x61,0x0A,0xA6,0x78, // '&'
    0xC0, // '''
    0x6A,0xA0, // '('
    0x95,0x60, // ')'
    0x21,0x08,0x00, // '*'
    0x01,0x3E,0x42,0x00, // '+'
    0x50, // ','
    0xC0, // '-'
    0x40, // '.'
    0x49,0x49,0x00, // '/'
    0x69,0x99,0x60, // '0'
    0x61,0x08,0x47,0x00, // '1'
    0x70,0x88,0x8F,0x00, // '2'
    0xF1,0x61,0xF0, // '3'
    0x32,0x95,0xF1,0x00, // '4'
    0x66,0x11,0x70, // '5'
    0x7E,0x99,0x60, // '6'
    0x71,0x22,0x40, // '7'
    0xF5,0x5E, // '8'
    0xE9,0x71,0xE0, // '9'
    0x41, // ':'
    0x41,0x40, // ';'
    0x00,0xB8,0xC1,0x80, // '<'
    0x79,0xE0, // '='
    0x06,0x0E,0xC8,0x00, // '>'
    0xE5,0x24, // '?'
    0x31,0x3B,0xF3,0xB9,0x03,0x00, // '@'
    0x23,0x14,0xE8,0x80, // 'A'
    0xF9,0xE9,0xF0, // 'B'
    0x74,0x21,0x07,0x00, // 'C'
    0xE9,0x99,0xE0, // 'D'
    0xF8,0xE8,0xF0, // 'E'
    0xE8,0xE8,0x80, // 'F'
    0x7C,0x27,0x17,0x80, // 'G'
    0x99,0xF9,0x90, // 'H'
    0xF8, // 'I'
    0x24,0x92,0xC0, // 'J'
    0x95,0x31,0x49,0x00, // 'K'
    0x88,0x88,0xE0, // 'L'
    0xDE,0xFB,0x58,0x80, // 'M'
    0x9D,0xDB,0x90, // 'N'
    0x74,0x63,0x17,0x00, // 'O'
    0xF7,0x48, // 'P'
    0x74,0x63,0x17,0x08, // 'Q'
    0xEA,0xCA,0xA0, // 'R'
    0xF8,0x61,0xF0, // 'S'
    0xF9,0x08,0x42,0x00, // 'T'
    0x99,0x99,0xF0, // 'U'
```

```
    0x8C,0x94,0xC2,0x00, // 'V'
    0x93,0x69,0xD3,0x66,0xC0, // 'W'
    0x53,0x08,0xA9,0x00, // 'X'
    0x92,0x88,0x42,0x00, // 'Y'
    0xF1,0x10,0x8F,0x00, // 'Z'
    0xEA,0xAC, // '['
    0x92,0x24,0x80, // '\'
    0xD5,0x5C, // ']'
    0x31,0x20, // '^'
    0x78, // '_'
    0x08, // '`'
    0xEE,0xF0, // 'a'
    0x8E,0x99,0xE0, // 'b'
    0x68,0x86, // 'c'
    0x17,0x99,0x70, // 'd'
    0x6F,0x86, // 'e'
    0x6E,0x44,0x40, // 'f'
    0x79,0x97,0x70, // 'g'
    0x9E,0xDA, // 'h'
    0xF8, // 'i'
    0x55,0x70, // 'j'
    0x8A,0xCC,0xA0, // 'k'
    0xF8, // 'l'
    0xFD,0x6B,0x50, // 'm'
    0xF6,0xD0, // 'n'
    0x69,0x96, // 'o'
    0xE9,0x9E,0x80, // 'p'
    0x79,0x97,0x10, // 'q'
    0x1E,0x48, // 'r'
    0x74,0x37, // 's'
    0x46,0x44,0x60, // 't'
    0x16,0xDE, // 'u'
    0x9A,0x66, // 'v'
    0xAA,0xE5,0x92, // 'w'
    0xA6,0x69, // 'x'
    0x96,0x64,0x40, // 'y'
    0xEA,0x70, // 'z'
    0x32,0x24,0x22,0x30, // '{'
    0xFE, // '|'
    0x61,0x08,0x22,0x11,0x80 // '}'
";
b0 = "{{0,1,1,3,0,0}, //''
{1,3,5,4,0,-5}, //'!'
{3,2,2,5,1,-5}, //'\"'
      {     4,   6,   5,   7,    0,   -5 }, // '#'
      {     8,   4,   7,   6,    0,   -5 }, // '$'
      {    12,   6,   5,   9,    1,   -5 }, // '%'
      {    16,   6,   5,   7,    1,   -5 }, // '&'
      {    20,   1,   2,   4,    1,   -5 }, // '''
      {    21,   2,   7,   4,    1,   -5 }, // '('
      {    23,   2,   7,   4,    0,   -5 }, // ')'
      {    25,   5,   4,   6,    0,   -5 }, // '*'
      {    28,   5,   5,   8,    1,   -5 }, // '+'
      {    32,   2,   2,   4,    0,   -1 }, // ','
      {    33,   3,   1,   4,    0,   -2 }, // '-'
      {    34,   2,   1,   4,    0,   -1 }, // '.'
      {    35,   3,   6,   3,    0,   -5 }, // '/'
      {    38,   4,   5,   5,    0,   -5 }, // '0'
      {    41,   5,   5,   5,    0,   -5 }, // '1'
      {    45,   5,   5,   6,    0,   -5 }, // '2'
      {    49,   4,   5,   6,    0,   -5 }, // '3'
      {    52,   5,   5,   6,    0,   -5 }, // '4'
      {    56,   4,   5,   6,    0,   -5 }, // '5'
      {    59,   4,   5,   6,    0,   -5 }, // '6'
      {    62,   4,   5,   6,    0,   -5 }, // '7'
      {    65,   3,   5,   6,    0,   -5 }, // '8'
      {    67,   4,   5,   6,    0,   -5 }, // '9'
      {    70,   2,   4,   3,    0,   -4 }, // ':'
      {    71,   2,   5,   4,    0,   -4 }, // ';'
```

```
        {    73,   5,   5,   7,    1,   -5 }, // '<'
        {    77,   6,   2,   7,    0,   -3 }, // '='
        {    79,   5,   5,   7,    1,   -5 }, // '>'
        {    83,   3,   5,   5,    0,   -5 }, // '?'
        {    85,   6,   7,   9,    1,   -6 }, // '@'
        {    91,   5,   5,   6,    0,   -5 }, // 'A'
        {    95,   4,   5,   5,    0,   -5 }, // 'B'
        {    98,   5,   5,   6,    0,   -5 }, // 'C'
        {   102,   4,   5,   5,    0,   -5 }, // 'D'
        {   105,   4,   5,   5,    0,   -5 }, // 'E'
        {   108,   4,   5,   5,    0,   -5 }, // 'F'
        {   111,   5,   5,   6,    0,   -5 }, // 'G'
        {   115,   4,   5,   5,    0,   -5 }, // 'H'
        {   118,   1,   5,   3,    0,   -5 }, // 'I'
        {   119,   3,   6,   4,   0,   -5 }, // 'J'
        {   122,   5,   5,   6,    0,   -5 }, // 'K'
        {   126,   4,   5,   5,    0,   -5 }, // 'L'
        {   129,   5,   5,   6,    0,   -5 }, // 'M'
        {   133,   4,   5,   5,    0,   -5 }, // 'N'
        {   136,   5,   5,   6,    0,   -5 }, // 'O'
        {   140,   3,   5,   4,    0,   -5 }, // 'P'
        {   142,   5,   6,   6,    0,   -5 }, // 'Q'
        {   146,   4,   5,   5,    0,   -5 }, // 'R'
        {   149,   4,   5,   5,    0,   -5 }, // 'S'
        {   152,   5,   5,   6,    0,   -5 }, // 'T'
        {   156,   4,   5,   5,    0,   -5 }, // 'U'
        {   159,   5,   5,   6,    0,   -5 }, // 'V'
        {   163,   7,   5,   8,    0,   -5 }, // 'W'
        {   168,   5,   5,   6,    0,   -5 }, // 'X'
        {   172,   5,   5,   6,    0,   -5 }, // 'Y'
        {   176,   5,   5,   6,    0,   -5 }, // 'Z'
        {   180,   2,   7,   5,    1,   -5 }, // '['
        {   182,   3,   6,   3,    0,   -5 }, // '\'
        {   185,   2,   7,   4,    0,   -5 }, // ']'
        {   187,   6,   2,   7,    0,   -5 }, // '^'
        {   189,   5,   1,   5,   -1,    1 }, // '_'
        {   190,   3,   2,   5,    0,   -6 }, // '`'
        {   191,   3,   4,   5,    0,   -4 }, // 'a'
        {   193,   4,   5,   5,    0,   -5 }, // 'b'
        {   196,   4,   4,   5,    0,   -4 }, // 'c'
        {   198,   4,   5,   5,    0,   -5 }, // 'd'
        {   201,   4,   4,   5,    0,   -4 }, // 'e'
        {   203,   4,   5,   4,    0,   -5 }, // 'f'
        {   206,   4,   5,   6,   0,   -4 }, // 'g'
        {   209,   3,   5,   4,    0,   -5 }, // 'h'
        {   211,   1,   5,   2,    0,   -5 }, // 'i'
        {   212,   2,   6,   4,    0,   -5 }, // 'j'
        {   214,   4,   5,   5,    0,   -5 }, // 'k'
        {   217,   1,   5,   3,    0,   -5 }, // 'l'
        {   218,   5,   4,   7,    0,   -4 }, // 'm'
        {   221,   3,   4,   4,    0,   -4 }, // 'n'
        {   223,   4,   4,   5,    0,   -4 }, // 'o'
        {   225,   4,   5,   5,    0,   -4 }, // 'p'
        {   228,   4,   5,   5,    0,   -4 }, // 'q'
        {   231,   3,   5,   4,    0,   -5 }, // 'r'
        {   233,   4,   4,   5,    0,   -4 }, // 's'
        {   235,   4,   5,   4,    0,   -5 }, // 't'
        {   238,   3,   5,   5,    0,   -5 }, // 'u'
        {   240,   4,   4,   5,    0,   -4 }, // 'v'
        {   242,   6,   4,   7,    0,   -4 }, // 'w'
        {   245,   4,   4,   5,    0,   -4 }, // 'x'
        {   247,   4,   5,   5,    0,   -4 }, // 'y'
        {   250,   3,   4,   6,    0,   -4 }, // 'z'
        {   252,   4,   7,   6,    0,   -5 }, // '{'
        {   256,   1,   8,   4,    1,   -5 }, // '|'
        {   257,   5,   7,   6,    0,   -5 } // '}'}
";
```

- **Converting to the memory image**

```
(* preparing font bitmap *)
a1 = StringReplace[a0, Whitespace -> ""];
a2 = StringReplace[a1, "//'" → ""];
a3 = StringReplace[a2, "//'" ~~ _ ~~ "'" → ""];
a4 = StringReplace[a3, "'" → ""];
a5 = StringReplace[a4, "0x" → ""];
a = StringSplit[a5, ","];
(* preparing font position parameter *)
b1 = StringReplace[b0, Whitespace -> ""];
b2 = StringReplace[b1, "//'" → ""];
b3 = StringReplace[b2, "//'" ~~ _ ~~ "'" → ""];
b4 = StringReplace[b3, "'" → ""];
b = ToExpression[b4];
(* get bitmap hex data series for each character *)
begin := b[[i, 1]] + 1;
end := b[[i + 1, 1]] /; i < 94;
end := Length[a] /; i == 94;
c = Table[Take[a, {begin, end}], {i, Length[b]}];
(* conbine to long binary before partitioning *)binDigit = Map[Flatten,
   Table[Map[IntegerDigits[FromDigits[#, 16], 2, 8] &, c[[i]]], {i, Length[c]}]];
(* getting font style parameters i=1:"space" to 94:"}" *)
fontTable = ⁝<||>;
  Table[
   width=b[[i,2]];
   height=Min[7,b[[i,3]]];
   xAdvance=b[[i,4]];
   xOffset=b[[i,5]];
   yOffset=Min[b[[i,6]],0];
   (* prepare pads *)
   pad=ConstantArray[0,width];
   bytePad=ConstantArray[0,8];
   (* set padding number *)
   upPadding=6+yOffset;
   downPadding=Max[8-upPadding-height,0];
   (* raw font rectangle *)
   rd=Take[Partition[binDigit[[i]],width],height];
   (* padding rectangle *)
   rt=Join[Table[pad,{upPadding}],rd,Table[pad,{downPadding}]];
   rz=Join[Table[bytePad,{xOffset}],Transpose[rt],Table[bytePad,{xAdvance-width}]];
   (* prepare form for downloading to SSD1306 *)
   hex=IntegerString[Map[FromDigits[#,2]&,rz],16]/."0"→"00";
   strSeries=Map[StringJoin["0x",#]&,hex];
   AssociateTo[fontTable,Rule[FromCharacterCode[31+i],strSeries]],
   {i,94}];
```

* fontTable can save as a file for saving the conversion time and space

# Setup for SSD1306 and output data lines

```
chipAdr = "0x3c";
comStr = Join[{"i2cset", "-y", "1"}, {chipAdr}, {"0x00"}];
rSet[reg_] := RunProcess[Join[comStr, {reg}]];
rSet["0x8d"]; (* charge pump on *)
rSet["0x14"];(* enable charge pump *)
rSet["0xaf"];(* display on in normal mode *)
rSet["0xa1"];(* set segment remap to reverse *)


rSet["0xb6"];(* set page address *)
rSet["0x02"];(* column reset*)
rSet["0x10"];
```

```
(* erase display  *)
blank = ConstantArray["0x00", 32];
dataStr = Join[{"i2cset", "-y", "1", "0x3c", "0x40"}, blank, {"i"}];

Table[RunProcess[dataStr], {4}];



(* output 1-st line *)
str = Map[fontTable, Characters["SSD1306 is a single-chip CMOS"]];
dcom = Map[Join[{"i2cset", "-y", "1", "0x3c", "0x40"}, #, {"i"}] &, str];
Map[RunProcess, dcom];

rSet["0xb5"];(* set page address and column reset*)
rSet["0x02"];
rSet["0x10"];

(* output 2-nd line *)
str = Map[fontTable, Characters["OLED/PLED driver with"]];
dcom = Map[Join[{"i2cset", "-y", "1", "0x3c", "0x40"}, #, {"i"}] &, str];
Map[RunProcess, dcom];

rSet["0xb4"];(* set page address and column reset*)
rSet["0x02"];
rSet["0x10"];

(* output 3-rd line *)
str = Map[fontTable, Characters["controller for organic/polymer"]];
dcom = Map[Join[{"i2cset", "-y", "1", "0x3c", "0x40"}, #, {"i"}] &, str];
Map[RunProcess, dcom];

rSet["0xb3"];(* set page address and column reset*)
rSet["0x02"];
rSet["0x10"];

(* output 4-th line *)
str = Map[fontTable, Characters["light emitting diode dot-matirx"]];
dcom = Map[Join[{"i2cset", "-y", "1", "0x3c", "0x40"}, #, {"i"}] &, str];
Map[RunProcess, dcom];

rSet["0xb2"];(* set page address and column reset*)
rSet["0x02"];
rSet["0x10"];

(* output 5-th line *)
str = Map[fontTable, Characters["graphic display system."]];
dcom = Map[Join[{"i2cset", "-y", "1", "0x3c", "0x40"}, #, {"i"}] &, str];
Map[RunProcess, dcom];
```