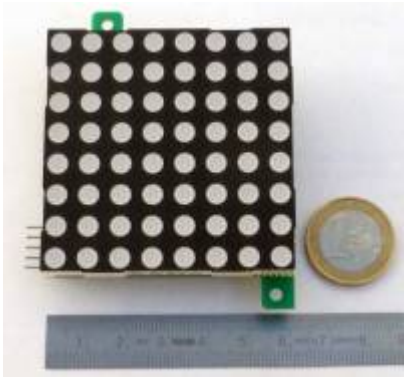
PixBlock



A display, just like you need it

- Bright, colorful dot-matrix display
- 64 bicolor pixels (red, green)
- Modular design: Individual PixBlocks can be daisy-chained and arranged as desired
- Precise LED current sources
- Facilitates borderless installations
- Easy control via SPI interface

The associated Arduino Library enables easy control with the following properties:

- Operation as text and graphics display
- 15 different colors
- Automatic ticker texts
- Multiple character fonts (ASCII, ISO8859, Symbols)

Description

The PixBlock is a two-color (red + green) dot-matrix display with 8 x 8 pixels. Use your favorite microcontroller, e. g. Arduino, PIC or ARM, to control the 64 pixels via an SPI interface. Following a multiplexing scheme you periodically update the 16 leds (8 red + 8 green) of each column. Switching to the next column is done automatically by pulsing the latch signal.

All basic functions from multiplexing to ticker texts are handled by the Arduino Library. The library is written in C ++ for Atmel AVR processors. An adaptation to other processors is easily possible.

Each PixBlock can be optionally equipped with a separate microcontroller (ATtiny84A), which can control up to 10 concatenated PixBlocks. This turns the PixBlock into an intelligent display that works without an external processor. If needed, there is also room to add three push-buttons and an

EEPROM on board.

Assembly

Assembly is straight forward. The SMD components are already mounted on the board. Appropriate skills to build electronic circuits and to safely use the tools are required.

Tools

soldering tools

You need the usual tools for soldering of electronic circuits. If you want to equip the optional microcontroller, you should have thin solder wire (diameter 0.5 mm) and a pair of tweezers.

Components

PixBlock components

The kit contains:

- 1 PixBlock printed circuit board (with SMDs pre-assembled)
- 1 dot-matrix display
- 1 pin header (5 pins, 90 degrees)
- 1 female header (5 pins, 90 degrees)

Depending on the planned use additional parts may be useful. These are not included in the kit.

- Spacer bolts with screws
- Thread locking compound
- Power connector
- Pin header (2 x 5 pins)
- Microcontroller ATtiny84A (SO14)
- Pin header for ISP interface (2 x 3 pins)

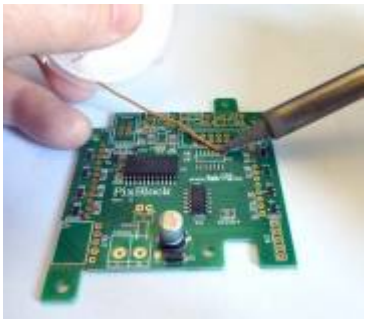
- serielles EEPROM (z. B. 25LC256, SO8-Bauform)
- Push buttons (6 x 6 mm)

Instructions

Important: Soldering of the display is the very last step.
Previously you should carefully consider what additional components you may want assemble. Once the display is soldered, it covers the board and the mounting of other components becomes very difficult or even impossible.

1. If you want to mount the optional microcontroller, this should be the first step. Otherwise, you can skip to the next step.

   First, it is advisable to remove the excess solder from the pads with solder wick. Then cover a single pad of the processor with solder and attach the processor with only one pin to this pad. Starting with only one pin enables you to easily adjust the position of the processor if necessary. If all pins are exactly centered on each pad, solder the remaining pins.



2. Now the 5-pin connectors (male and female) are soldered to the board.


3. If you want to use a 0.1„ pin header or a 5 mm terminal block for the power supply mount the connector and solder it in.

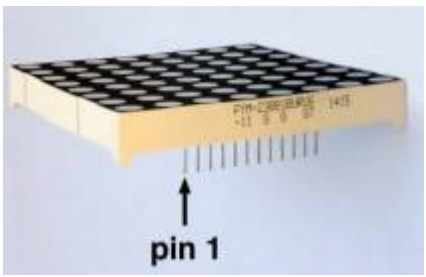


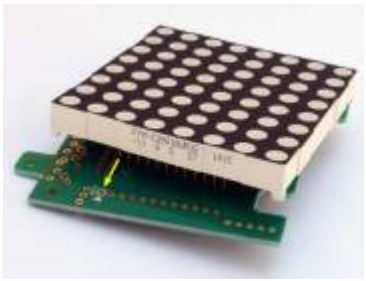
4. Now you have to consider what extra components you want to mount. Here fastening bolts, push buttons or additional pin headers come into question. If using inner fastening bolts fix the screws with some thread locking compound, because they will become inaccessible after the display has been soldered.

   If you use the optional microcontroller it is recommended to assemble a 2 x 3 pin header for in system programming (ISP). Also a reset jumper on J2b might be useful as it allows to permanently disable the controller in case you do not need it.



5. Finally, the dot-matrix display is mounted. Care must be taken to the correct position. If you look at the type print of the display, pin 1 is left front. This pin must be inserted in the square pad which is marked by a triangle. Solder the display so that it rests on three corners on the board. The board has a recess beneath the fourth corner.





This is how an assembled PixBlock looks like. In this example three spacer bolts were mounted and the protruding fixing lugs were snapped off.

Hardware

The PixBlock hardware is described in detail by the schematic and the assembly drawing. Both are available in the download area .

Technical Specification

- Dot-matrix display with 8 x 8 bicolor (red + green) LEDs
- Power supply: 4.0 V - 5.0 V
- Dimensions: approx. 60 mm x 60 mm (excluding fixing lugs)
- Mounting option for ATtiny24/44/84A

Connectors

Serial-IN

< 8em 2em >	
Pin	**Signal**
1	SDI
2	LATCH
3	CLK
4	GND
5	Vin

Serial-OUT

< 8em 2em >	
Pin	**Signal**
1	SDO
2	LATCH
3	CLK
4	GND
5	Vin

ISP connector (optional)

Use the ISP connector (CON4) for in system programming of the optional microcontroller using a programmer.

< 16em 6em 2em 2em 6em >			
Signal	**Pin**	**Pin**	**Signal**
MISO	1	2	VCC
SCK	3	4	MOSI
RESET	5	6	GND

Extension Port (optional)

CON5 provides the I/O pins of the optional microcontroller.

< 22em 6em 2em 2em 12em >			
Signal	**Pin**	**Pin**	**Signal**
GND	1	2	VCC (max. 50 mA)
PA0	3	4	PA1
PA2	5	6	PA3
PA4	7	8	PA5
PA6	9	10	PA7

Power Supply

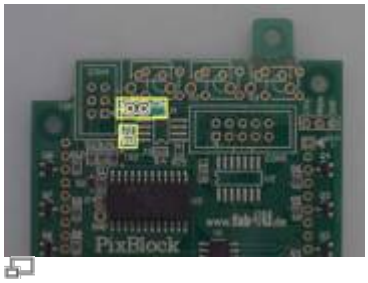
The PixBlock is designed to operate on a stabilized 5 volts DC supply.

Operation down to about 3.3 volts is possible if diodes D1 and D2 are bridged. For this close jumper J1. Then remove diode D2 and replace it by a short piece of wire or a 0 ohm resistance.

Important: After this rework take great care that the polarity of the supply voltage is always correct!

When chaining multiple PixBlocks it is important to note that contact resistance and pcb track resistances account for a small voltage drop from block to block. Through the use of controlled current sources, the LED brightness is relatively immune to such voltage variations. In long chains however it may be useful to feed the supply voltage at several points, e. g. every 5 PixBlocks.

Microcontroller Option



reset jumper

Optionally the PixBlock can be equipped with a processor that controls the display to make it completely autonomous. Suitable processors types are Atmel ATtiny24A, ATtiny44A or ATtiny84A (2k, 4k or 8k Flash memory). The programming requires a programmer and is done via the 6-pin ISP

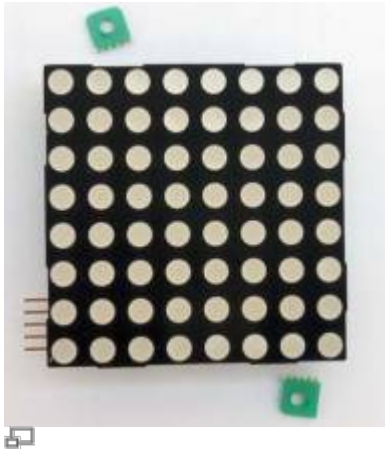
connector.

If the processor is assembled, you can close the reset jumper J2, or J2b, to permanently turn it off. With this setting the PixBlock can be changed back to behave as a purely passive block.

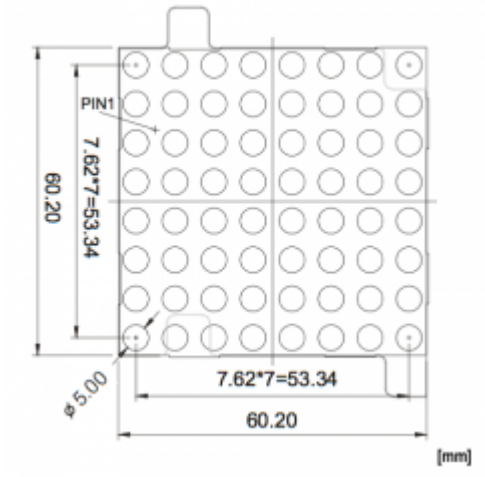
In addition, the board provides space for three push-buttons (straight or angled) and a serial EEPROM (e. g. Microchip 25LC256).

Dimensions

Size of the PixBlock is approximately 60 x 60 mm. Mounting holes have a diameter of 3 mm. For installation in an enclosure please consult the dimensional drawing.



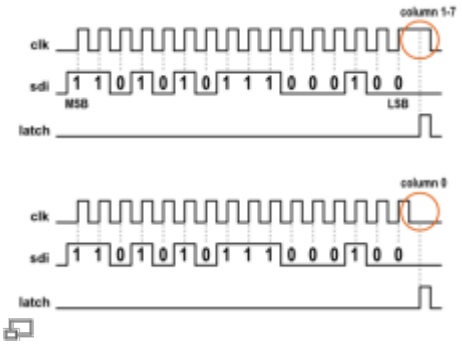
Removable fixing lugs allow a borderless installation



Dimensions of the dot-matrix

When concatenating several blocks one should choose the distance corresponding to the pitch of the light-emitting diodes, i. e. PixBlocks should be arranged in a grid of about 61 mm (8 x 7.62 mm).

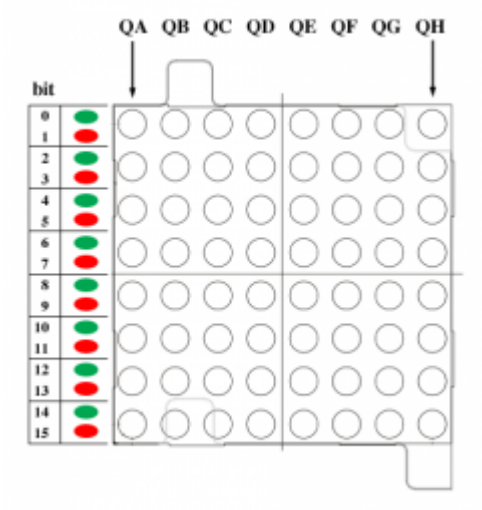
Control



Timing diagramm

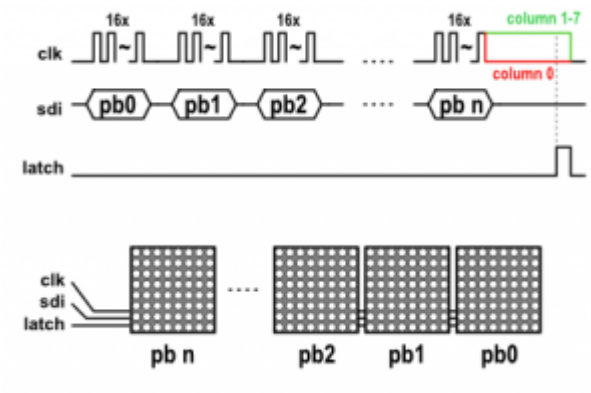
The control is done via a SPI-like interface. With the rising edge of the clock signal (CLK) the state of the data signal is sampled (SDI = serial data in). By a high level on the acquisition line (LATCH), the data is transferred to the PixBlock.

There are 16 bits clocked in that determine the state of the 16 LEDs of a column (1 = on, 0 = off). The most significant bit (MSB) is assigned to the red LED of the bottom pixel. The least significant bit (LSB) determines the state of the green LED of the top pixel.



Assignment of bits and leds

The cycling of the columns is done automatically. With each rising edge of the latch signal, the next column will be selected. Synchronisation is achieved using the state of the clock signal when latch becomes active. When the clock signal is low, column 0 (the leftmost column) will be selected. For all other columns, the clock signal is required to have high level. In this way, the activation of the column is very efficiently implemented.




Concatenating multiple PixBlocks

A chain of several PixBlocks forms a correspondingly larger shift register, e. g. 8 concatenated PixBlocks result in a 128 bit register. For each PixBlock 16 bits of data must be clocked in. The data for the block at the end of the line is sent first.

A simple code example can be found under "software".

Software

The easiest way to control the PixBlock is to use the Arduino library. It does all the tedious work so that you can concentrate on your application.

For a basic understanding you might want to have a look at the following code. It illustrates an example of how the PixBlock is controlled using C as programming language.

```
/*****************************************/
/* Simple example showing how to access   */
/* the PixBlock dot-matrix in principle.  */
/*****************************************/

#include <inttypes.h>
#include <avr/io.h>


/*************
 * constants *
 *************/

// hardware pins for accessing the dot matrix
#define DM_DATA_DDR       DDRB
#define DM_DATA_PORT      PORTB
#define DM_DATA_BIT       0
#define DM_CLK_DDR        DDRB
#define DM_CLK_PORT       PORTB
#define DM_CLK_BIT        1
#define DM_LATCH_DDR      DDRB
#define DM_LATCH_PORT     PORTB
#define DM_LATCH_BIT      2


/********************
 * global variables *
 ********************/

uint16_t pixel_data[8] = {
    0xFFFF, 0xAAAA, 0x5555, 0x0000, 0xC003, 0x2008, 0x0410, 0x03C0
};   // each 16-bit value represents a column of the display
```

```
/*************
 * functions *
 *************/

void dot_matrix_shift_out(const uint16_t* screen)
// with each call shift out one column of pixel data
{
    static uint8_t    column = 0;
    uint16_t    pixel_data;

    pixel_data = screen[column];
    for (uint8_t i = 0; i < 16; i++) {
        if (pixel_data & 0x8000) { DM_DATA_PORT |= (1 << DM_DATA_BIT); }
        else { DM_DATA_PORT &= ~(1 << DM_DATA_BIT); }
        pixel_data <<= 1;

        // clock, rising egde
        DM_CLK_PORT &= ~(1 << DM_CLK_BIT);
        DM_CLK_PORT |=  (1 << DM_CLK_BIT);
    }

    // set final state of clock pin
    if (column == 0) {
        DM_CLK_PORT &= ~(1 << DM_CLK_BIT);
    }
    // pulse latch signal
    DM_LATCH_PORT |=  (1 << DM_LATCH_BIT);
    DM_LATCH_PORT &= ~(1 << DM_LATCH_BIT);

    // next column
    column++;
    column &= 0x07;         // limit column range to 0..7
}


/********
 * main *
 ********/

int main(void)
{
    // initialize i/o pins
    DM_DATA_PORT  &= ~(1 << DM_DATA_BIT);
    DM_CLK_PORT   &= ~(1 << DM_CLK_BIT);
    DM_LATCH_PORT &= ~(1 << DM_LATCH_BIT);
    DM_DATA_DDR   |=  (1 << DM_DATA_BIT);
    DM_CLK_DDR    |=  (1 << DM_CLK_BIT);
    DM_LATCH_DDR  |=  (1 << DM_LATCH_BIT);

    // repeatedly shift out pixel data
```

```
    // (usually one would do this in an interrupt routine)
    while(1)
    {
        dot_matrix_shift_out(pixel_data);
    }
}
```

Download

File	License
Schematic	CC-BY-SA
Assembly drawing	CC-BY-SA
PixBlock Library	see license.md
Instructions (German)	CC-BY-SA

CC-BY-SA = Creative-Commons-Lizenz, attribution, share alike
CC0 = Creative-Commons-Lizenz, no restrictions

PixBlock, electronics, kit, led, dot-matrix, display