

AN1: Using IO-Warrior with a LED matrix

Applicable for IOW24 and IOW40



Code Mercenaries

1. Overview

Since revision V1.0.2.0 both IOW24 and IOW40 offer a Special Mode Function for controlling matrixes of up to $8 \times 32 = 256$ LEDs. Relatively simple external drivers need to be added which can be scaled according to the actual used matrix size and required power.

2. Theory of operation

Driving LEDs in matrixes saves a lot of effort in wiring them and significantly reduces the number of driver circuits needed compared to a discrete direct control. To drive the full 256 LEDs off IO-Warrior only 40 connections to the LEDs and 40 drivers are needed. Compared to 257 connections and 256 drivers for discrete driving. Still a LED matrix allows all possible combinations of lit and dark LEDs just like a discrete arrangement would. A LED matrix is driven in multiplex mode. That means not all LEDs are on at the same time but just one row. This is done by the row drivers feeding current to just one row at a time. At the same time for all LEDs that are on in this row the corresponding column drivers are active so a current flows through that LED. After a short time the next row is driven and the corresponding column drivers are enabled.

This is continuously repeated at a speed high enough to make the human eye see only a continuous light from the active LEDs. The IO-Warrior drives the rows at 1024Hz, resulting in each individual row to be driven active about every $8\text{msec} = 128\text{Hz}$ which is well sufficient to not have any flicker.

2.1 IO-Warrior matrix

The choice of having a 8×32 may seem odd at first glance, 16×16 may sound more natural. Though there are good reasons for this particular layout: Driver requirements, multiplexing ratio, and

scalability.

A column driver is relatively simple to build as it will always only have to care for the current that flows through a single LED. The row drivers on the other hand have to source enough current for all LEDs in a row as in worst case all of them are on at the same time. So it makes sense to keep to relatively few of the powerful drivers and increase the number of the weaker drivers which are simpler to implement.

The multiplexing ratio is directly affecting the maximum brightness that can be obtained from a LED. Since a LED has a maximum current that can be passed through it without damaging it increasing the current is no way to compensate high multiplexing ratios. Keeping to a moderate 1:8 ratio allows high brightness.

And lastly not all applications will require as many as 256 LEDs. Since IO-Warrior is a universal chip it makes sense to keep flexibility high. Cutting off any number of column drivers allows to scale the matrix size from 8×1 to 8×32 .

2.2 LED current

Like with discrete driven LEDs resistors are used to set the proper current through the LED in a matrix. Though the matrix does not need a resistor for every LED but just one for all eight LEDs in a column. Since all LEDs in a column share the same resistor only LEDs of the same type or of identical technical data must be used in a column. Calculating the resistor for a LED in a matrix is a bit more complicated than in a discrete driven arrangement. In any case the data sheet of the LED is helpful, at least two parameters need to be known: The forward voltage V_f and the maximum forward current I_f of the LED.

Forward voltage is the minimum voltage that will be required to drive any current through the LED,

AN1: Using IO-Warrior with a LED matrix

like any diode a LED acts like a kind of barrier that "cuts off" a certain voltage. The forward current is the maximum current you can pass through the diode without causing a degradation of lifetime and/or optical properties. Unfortunately most data sheets specify just a DC forward current and not a pulsed forward current, at best they may specify a maximum single peak forward current which does not help much. This does not apply to the operating conditions in a multiplexed matrix. To keep on the safe side for the lifetime and optical properties of your LEDs the current through the LEDs should stay close to the specified DC I_f .

Once the desired I_f is established the forward voltages of the drivers and the supply voltage for the whole circuit need to be known as well. Then the formula is quite simple:

$$R = (V_{sup} - V_f - V_{fdriver}) / I_f$$

If $V_{sup} - V_f - V_{fdriver}$ should result in a negative value this is an indicator that your supply voltage is too low. Any LED matrix with higher brightness or more than 8x8 size (unless low power LEDs are used) can not be driven from the USB power.

For calculating the power requirements just multiply your I_f with the number of LEDs in a row. Since in worst case all LEDs in a row can be on at the same time the power source must be able to supply this current without any problems.

2.3 Driver design

IO-Warrior controls the LED drivers by a serial data stream designed to connect to serial in-parallel out shift registers. Appropriate shift registers are available in the TTL/HighSpeed CMOS logic

family like the 74HC/HCT595, which require external current drivers. Drivers integrated with the shift registers are available too, i.e. Micrel MIC5891 for the row driver (sourcing driver) and MIC5821 or MIC5822 for the column driver (sinking driver).

Depending on the required current for the LEDs it may be necessary to do a more or less complicated design for the row drivers.

In a simple case of just driving 8x8 LEDs with moderate brightness and 30mA I_f it is possible to supply them from USB power (set IO-Warrior to high power!) and drive the matrix with a MIC5891 and a MIC5821. The total driver circuit amounts to just two chips, three capacitors and nine resistors (see Fig.1).

If the matrix grows bigger more MIC5821 are added up to a total of 4 to drive the full matrix size of 8x32. However the maximum current of the MIC5891 row driver has to be taken into account. If it is approached or exceeded this will not necessarily damage the chip but the forward voltage of the driver increases with the current drawn. This will influence the brightness of the LEDs, it will start to vary with the number of lit LEDs.

The actual design of the drivers depends on a number of parameters. Using the integrated MIC5891/5821 is a simple and compact but costly variant. If PCB space is not the primary concern using 74xx595 in combination with discrete transistors or integrated drivers will result in a more cost efficient solution.

AN1: Using IO-Warrior with a LED matrix

3. Signals from IO-Warrior

IO-Warrior provides four signals to control the external LED registers/drivers. /OE is pulled low by IO-Warrior to enable the drivers. All drivers should be inactive while /OE is high, otherwise random states of the registers may cause LEDs to get a DC current that may be too high (depending on the forward current intended per LED). After enabling the LED function /OE is pulled high. It goes low after the data has been latched into the registers for the first time.

Data to the registers is provided via the Data pin. Each bit is clocked into the external shift registers with the rising edge of Clk. Clk idles in low state between shifting operations.

After shifting out all bits and pulling Clk low again IO-Warrior pulls Strb high for about 1µsec to signal to the registers to store the data.

3.1 Data format

IO-Warrior shifts a block of data every millisecond (to be exact at a rate of 1024Hz). Each block starts with 32 bits for the column drivers, the first bit shifted out corresponds to the highest numbered driver of that row which is set by bit 7 of data3 of the \$15 report. Our programming examples assume this to be the rightmost LED in a row, however actual implementation can be as necessary. Each "1" in that data corresponds to an active driver, i.e. if a column driver gets a 1 it should drive its output to ground.

Following the 32 column bits are the 8 row bits, also highest bit first. The highest bit corresponds to row 7. The programming examples assume row 0 to be the topmost.

Column drivers must also go active if they get a 1, i.e. the driver must then source current into the matrix.

After all 40 bits have been shifted out IO-Warrior issues a Strb pulse and reasserts /OE.

Do not make assumptions going beyond this information, future versions of IO-Warrior may extend this data format to allow driving larger matrixes.

3.2 Example circuits

In Fig.1 you can see a basic driver circuit for 8x8 LEDs. Since this circuit is using a single 5891 for the row drivers it will be limited to a total current draw of about 500mA for a row. For a 8x8 matrix this is well sufficient with about 60mA per LED. Though larger matrixes may either need more powerful drivers or will have to settle for lower brightness of the LEDs.

Fig.2 shows how another group of 8x8 LEDs can be added to the basic circuit. Principally it is possible to extend the matrix to the full 8x32 by adding just three more column drivers. However the maximum available current is set by the row drivers.

There are two approaches to drive larger matrixes with higher current: Either the row drivers can be made more powerful, or the matrix can be segmented and additional row drivers added that each drive just a part of the total matrix.

Fig.3 gives an example how additional 5891s may be added so that each one drives just an 8x8 subset of the matrix. However this adds a lot of effort to the wiring of the matrix.

Fewer wires, but a more complex circuit are the result of using stronger row drivers. Fig.4 shows a higher powered row driver. The transistors and resistors should be chosen according the actual current requirements. The pull down resistors at the '595 outputs are needed since the '595 does disable to tri-state which would leave the transistor bases floating.

An alternative to using the '595 with discrete transistors for the column drivers is to use an ULN2803 as the driver behind the '595.

3.3 Driving 7-Segment Displays

Since the physical arrangement of the LEDs is open it is possible to use the LED matrix function to drive 7-segment displays or other LED displays. Common anode displays are easiest to handle with IO-Warrior as this allows to connect all seven segments and the decimal point to consecutive

AN1: Using IO-Warrior with a LED matrix

column lines, which results in all segments of a single display being controlled by one byte of the data format.

3.4 Component sources for the drivers

The MIC5891 and MIC5821 are probably the most convenient drivers, though they are also the most costly. The parts are made by Micrel Semiconductor:

www.micrel.com

Finding an fully integrated alternative to the MIC5891 sourcing driver seems not to be simple. AllegroMicrosystems had such parts in the program but is discontinuing them:

www.allegromicro.com

The ULN2803 is made by a number of manufacturers, one of them is Motorola:

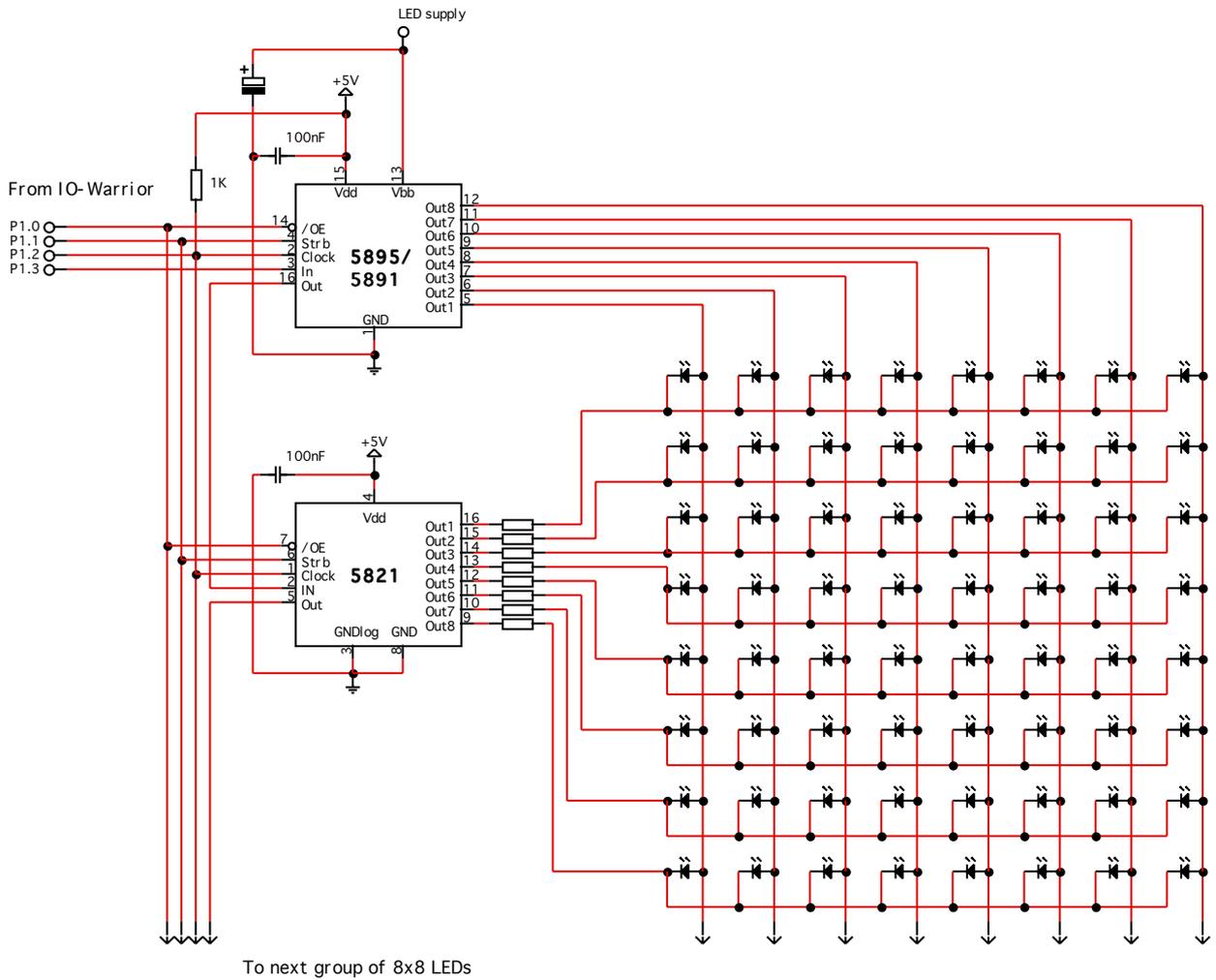
e-www.motorola.com

Another interesting alternative for the column drivers is made by ON Semiconductor. The NLSF595 is basically a 74xx595 with a driver stage, though only for moderate currents of up to 12mA, so it will not be usable for high brightness applications:

www.onsemi.com

AN1: Using IO-Warrior with a LED matrix

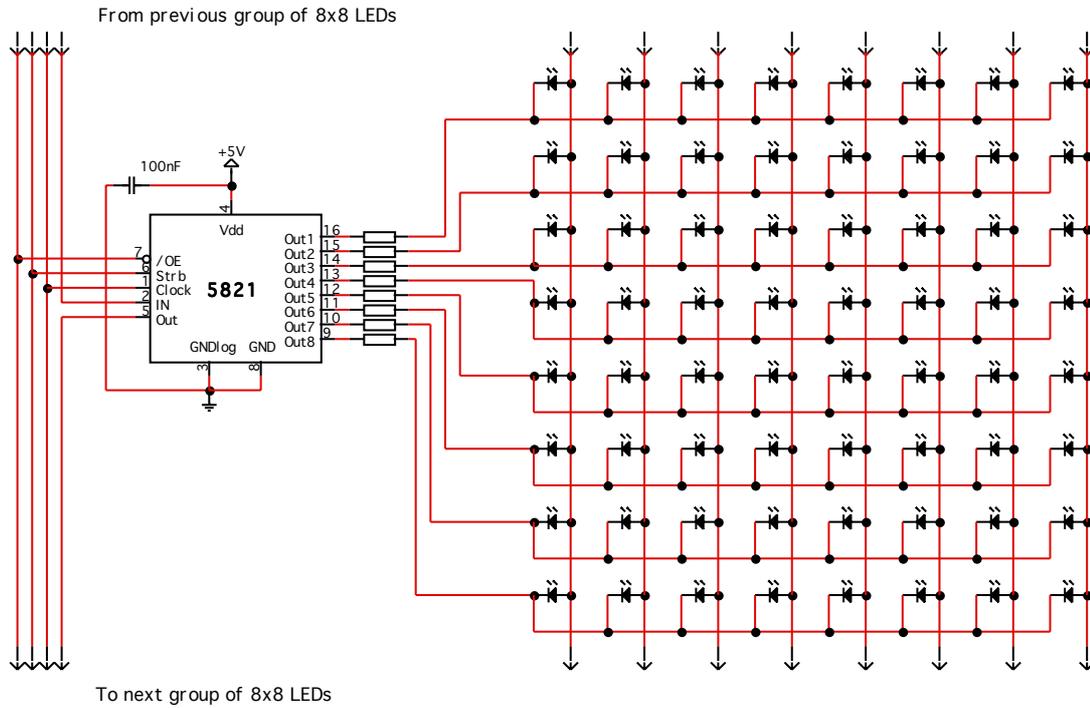
Fig. 1: Basic driver for 8x8 matrix



Circuit: IO-Warrior LED matrix				 Code Mercenaries
Version: 1.0				
Date: 28.1.2004				
Drawn by:				
Function:				
Page:				
Rev.	Date	By	Change	Sign.

AN1: Using IO-Warrior with a LED matrix

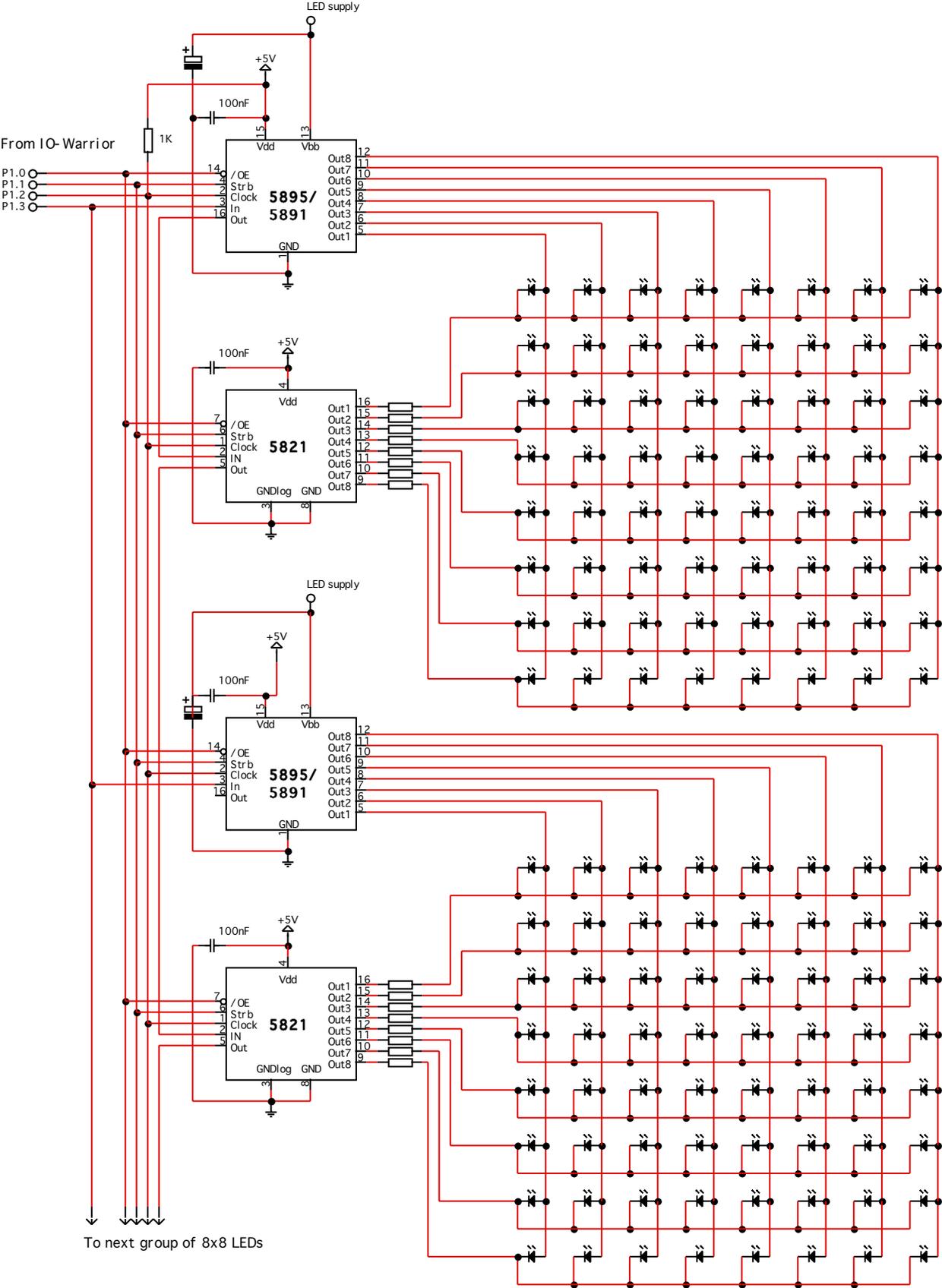
Fig.2: Extending matrix by 8x8 more LEDs



Circuit: IO-Warrior LED matrix					 Code Mercenaries
Version: 1.0					
Date: 28.1.2004					
Drawn by:					
Function:					
Page:					
Rev.	Date	By	Change	Sign.	

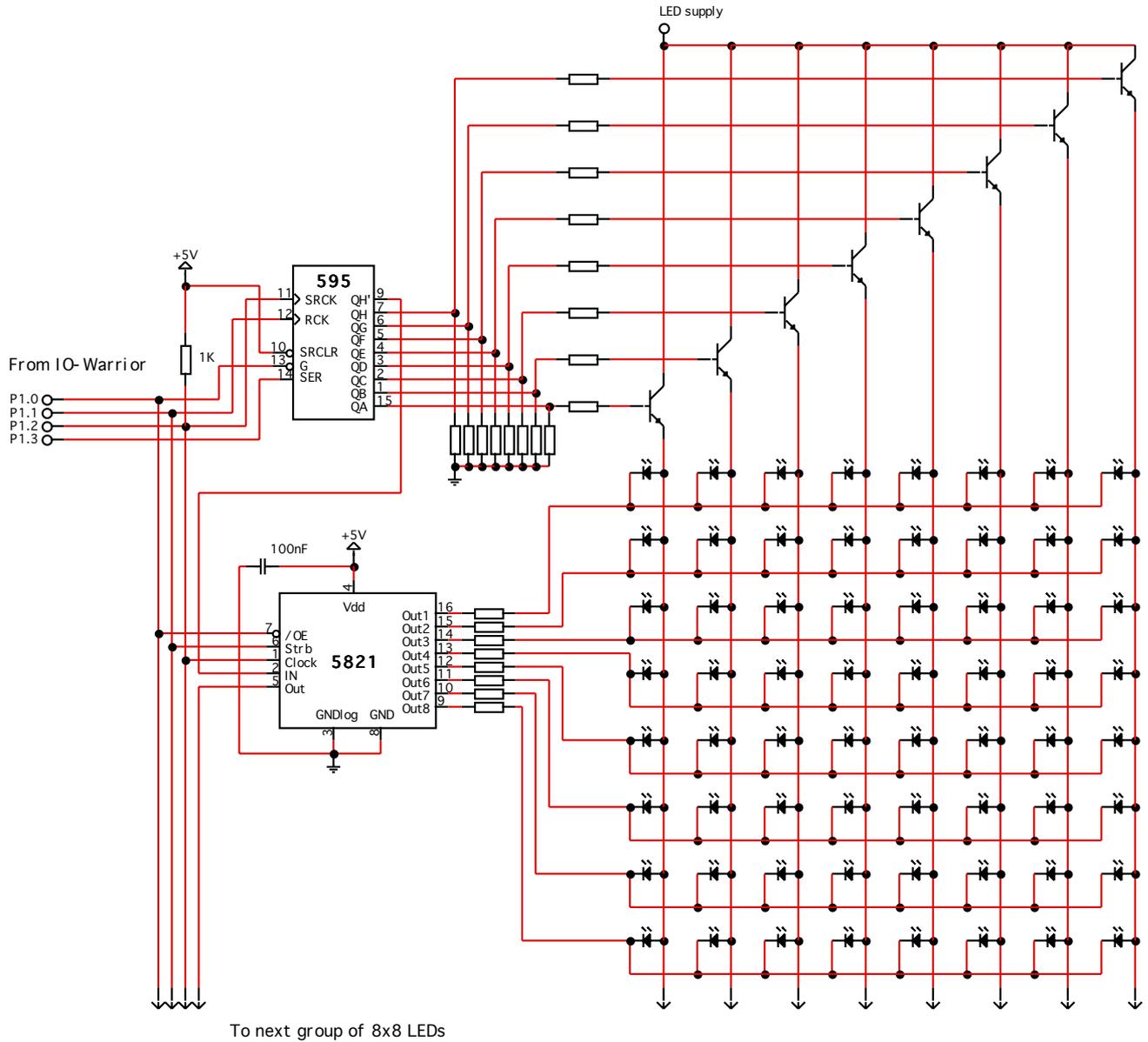
AN1: Using IO-Warrior with a LED matrix

Fig.3: Segmented matrix



AN1: Using IO-Warrior with a LED matrix

Fig.4: Row driver with more power



AN1: Using IO-Warrior with a LED matrix

Legal Stuff

This document is ©2004-2005 by Code Mercenaries.

The information contained herein is subject to change without notice. Code Mercenaries makes no claims as to the completeness or correctness of the information contained in this document.

Code Mercenaries assumes no responsibility for the use of any circuitry other than circuitry embodied in a Code Mercenaries product. Nor does it convey or imply any license under patent or other rights.

Code Mercenaries products may not be used in any medical apparatus or other technical products that are critical for the functioning of lifesaving or supporting systems. We define these systems as such that in the case of failure may lead to the death or injury of a person. Incorporation in such a system requires the explicit written permission of the president of Code Mercenaries.

Trademarks used in this document are properties of their respective owners.

Code Mercenaries
Hard- und Software GmbH
Karl-Marx-Str. 147a
12529 Schönefeld OT Grossziethen
Germany
Tel: x49-3379-20509-20
Fax: x49-3379-20509-30
Mail: support@codemerics.com
Web: www.codemerics.com

HRB 16007 P
Geschäftsführer: Guido Körber, Christian Lucht