Driving an LED matrix using the RL78G13

Introduction

This document explains how to drive an LED matrix by using general purpose I/O ports.

One I/O port is used to drive the common lines of the LED matrix, and up to two I/O ports can be used to generate the LED row activation signals (see Figure 1).

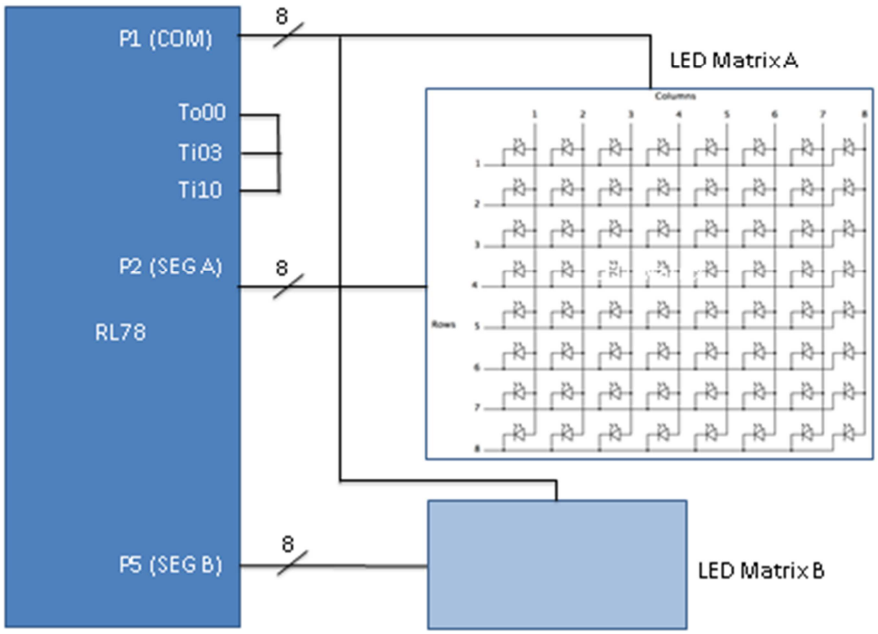


Figure 1 – Principle schematics

Since each I/O port is up to 8 bits wide, a matrix with a total of 8x8 or 8x16 LEDs can be driven with this configuration.

Additionally, each LED is separately dimmable using a total of n dimming steps.

Principle of operation

The shared 8 bit port which generates the common driving signal, P1[0..7] activates each column of the LED matrix. Two dedicated 8 bit segment ports generate the row signals associated with each column, P2[0..7] and P5[0..7].

The table below lists the connections

Port Signal	Led matrix signal
P1[0..7]	COM[1..8]
P2[0..7]	SEG_A[0..7]
P5[0..7]	SEG_B[0..7]

On the COM port there is a repeating pattern being applied which activates each COM line separately in a time division manner. The repeating pattern looks like a walking '1': [10000000], [01000000], [00100000] and repeats periodically.

The timing of the signals is such that each single COM line gets active with a frequency of 100 Hz, which in turn means that the frequency of applying each pattern on the COM port is 800 Hz. This is chosen to have a relatively high refresh rate of 100 Hz on the LED driving lines, to avoid flickering effects.

Each COM activation time is further divided into n 'time slices', seg_t1 to seg_tn (see Figure 2). Notice the trigger points (small triangles) related to Figure 4

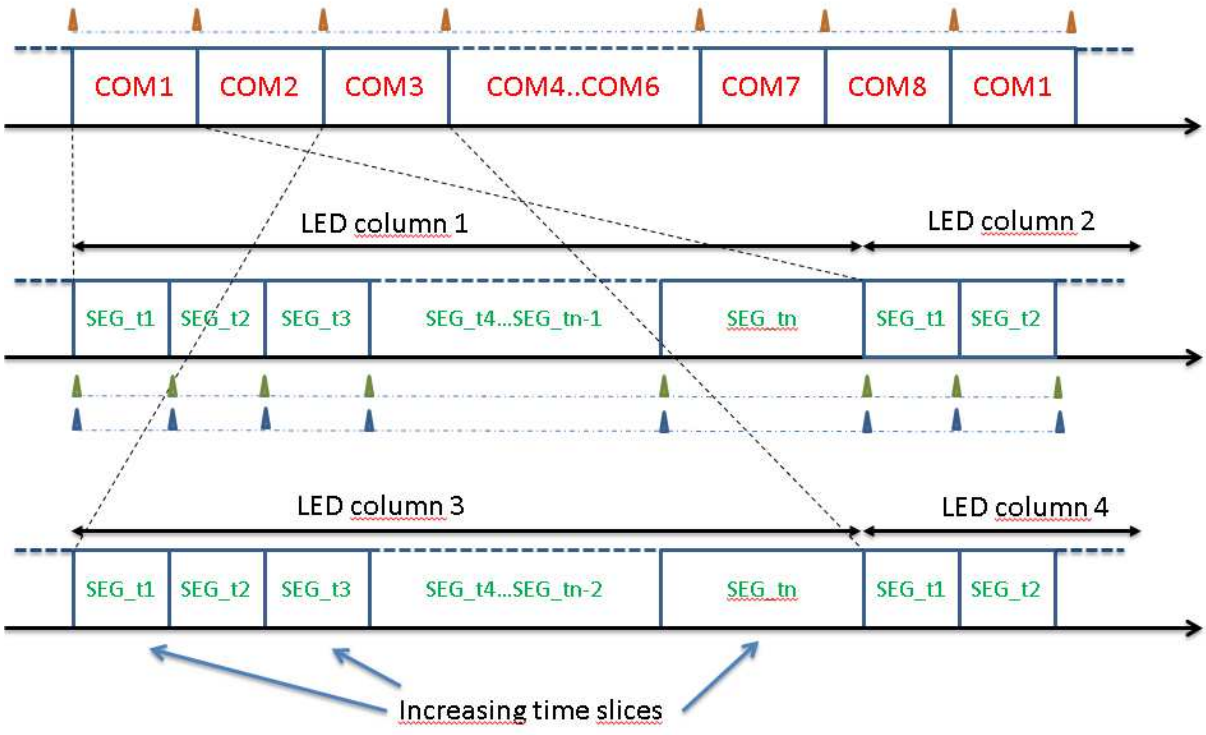


Figure 2 – Time diagram

During the activation time of each column, each I/O port is being written n times and the LED connected to a certain row will be ‘on’ or ‘off’ during a SEG_tn slice depending on the corresponding bit value of the 8 bit port data.

The n ‘time slices’ are determining the brightness of the LED. The more time the LED line is active, the more bright the LED will appear. Each LED can be ‘on’ for one or more consecutive time slices during each COM duration time.

In the simple example of Figure 3, there are 4 time slices defined. During each time slice the 8 bit value is applied to the port, which results to the LED brightness shown in the rightmost column of the table. This scheme repeats during each COM period.

	SEG_t1	SEG_t2	SEG_t3	SEG_t4	
Port	0xFC	0x3C	0x0C	0x0C	
					Brightness
Bit 0	0	0	0	0	OFF
Bit 1	0	0	0	0	OFF
Bit 2	1	1	1	1	Full
Bit 3	1	1	1	1	Full
Bit 4	1	1	0	0	50%
Bit 5	1	1	0	0	50%
Bit 6	1	0	0	0	25%
Bit 7	1	0	0	0	25%

Figure 3 - LED brightness control

A further improvement is made to take into account the sensibility of the human eye to small variations in luminosity at low light intensity. The time duration of each slice is not constant, but starts with a short value which gets incremented exponentially at each step. In this way it is possible to achieve a much finer resolution in the lower brightness range.

The calculation of the period duration in the example software is based on the same formula used for the DALI protocol specification when defining the power steps of a light actuator. With this scheme the proportional increase between successive time slice steps is constant, which gets reflected into a constant proportional increase in LED brightness between each dimming step. The test software uses 64 dim steps

To transfer the data to the COM and SEG ports, and to avoid an excessive load on the CPU, up to 3 DMA channels are used to move the data between the RAM memory and the I/O port buffer.

An additional DMA channel is used for managing the configuration of the time slice durations into the timer.

The system configuration is like in Figure 4:

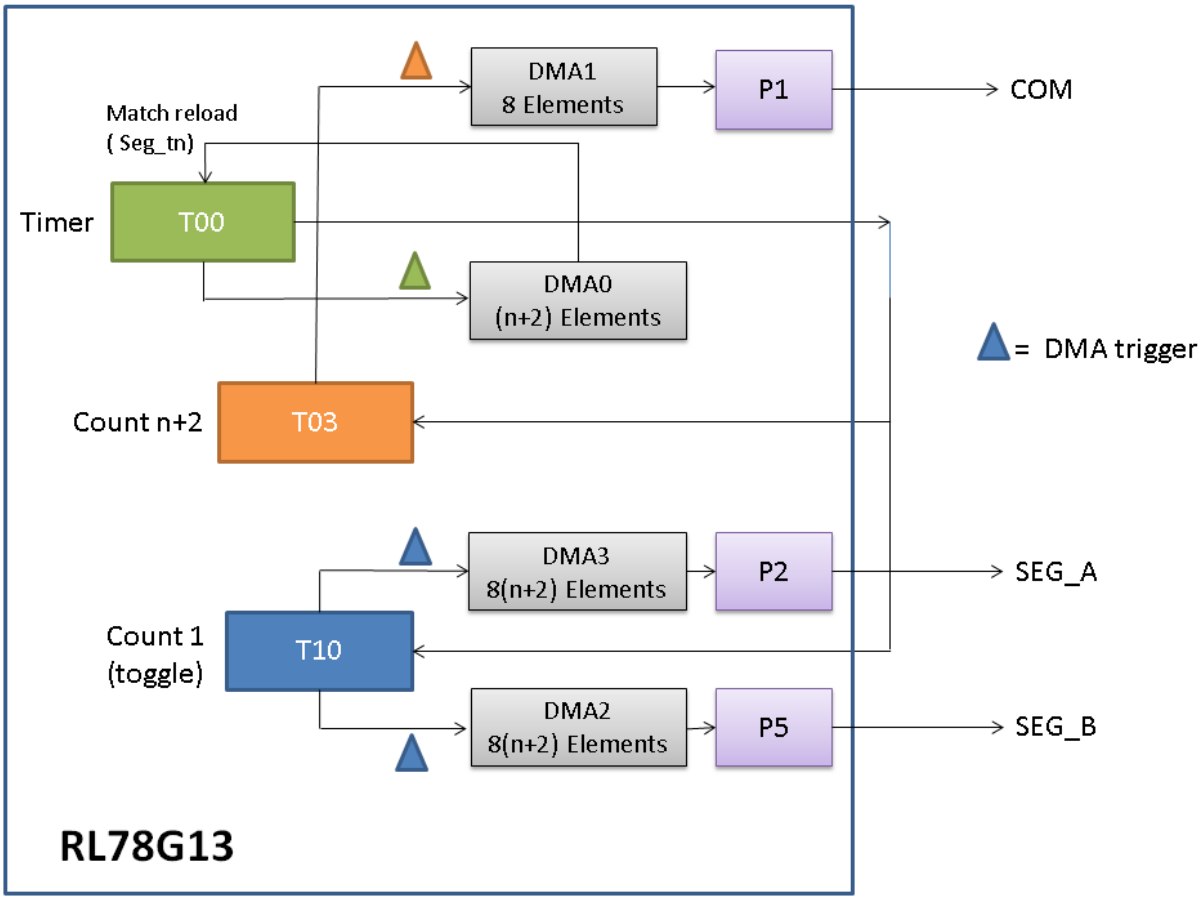


Figure 4 - RL78 G13 configuration

Timer unit 0 generates the time reference for the 'time slices'. At every Timer0 channel 0 (T00) match, a DMA trigger is generated and the DMA0 updates the next match value into the timer match register. These timings are stored into a table of size n+2 (one entry for each active time slice, plus one 'dead time' entry at the beginning and end of each COM period). When n+2 elements have been transferred, one COM period has elapsed and DMA0 has finished its transfer. It issues an interrupt and gets reloaded by the CPU within the interrupt vector routine.

The output signal of Timer0 channel0 is fed back to the input pins of Timer0 channel3 and Timer 1 channel 0. T03 and T10 are used in counter mode. Timer 1 needs to be used because DMA channel 2 and 3 cannot be triggered by Timer0 since this event is not routed to these channels.

T03 counts n+2 values, keeping track of the time slices, and generating a trigger for DMA1 to update the COM port. After DMA1 has transferred 8 elements (one for each COM line), one full refresh cycle has completed. DMA 1 issues an interrupt and gets reloaded by the CPU within the interrupt vector.

T10 counts to one, which means it generates one trigger signal on every T00 output toggle, corresponding to the end of each time slice. This hardware loopback is required to keep T00 and T10 in sync, since these two timers are separate and cannot be started at the exact same time by writing to a common register. DMA channel 2 and 3 are triggered by T10, and output the data used to drive the SEG_A and SEG_B port. DMA 2 and 3 transfers a full segment buffer, and issues an interrupt after 8(n+2) elements have been transferred. These get reloaded within the interrupt routine.

The data holding the brightness information for the LED matrix is stored into a RAM buffer which holds the 8 x n 8-bit values. The COM buffer holds 8 values. An example of the ram segment and com buffers is shown in Figure 5 below

Segment RAM buffer contents

Com active	Time slice	Buffer value
COM1	dummy	0x0
COM1	SEG_t1	...
COM1	SEG_t2	...
COM1	...	...
COM1	SEG_tn	...
COM1	dummy	0x0
COM2	dummy	0x0
COM2	SEG_t1	...
COM2	SEG_t2	...
COM2	...	...
COM2	SEG_tn	...
COM2	dummy	0x0
COM3	dummy	0x0
COM3	SEG_t1	...
COM3	SEG_t2	...
COM3	...	...
COM3	SEG_tn	...
COM3	dummy	0x0
COM4	Dummy	0x0
COM4	SEG_t1	...
...	...	...

COM RAM buffer contents

Com	Port value
COM1	0x01
COM2	0x02
COM3	0x04
COM4	0x08
COM5	0x10
COM6	0x20
COM7	0x40
COM8	0x80

Figure 5 - Segment and com RAM buffers

Note that at the beginning and at the end of each segment block a dummy zero value is inserted, corresponding to a short time slice. This is used to allow a "dead band" time where all the LEDs are off, during the switching time of two consecutive COM lines, to avoid excessive current draw.

Test application

The test application works on the RL78G13 test board (QB-R5F100SL-TB) with E1 emulator on IAR.

It waits for a user to press the button to start the demo.

Then at each successive press the LED brightness is cyclically stepped at each button press from 0 to maximum and then decremented back to 0.

The signals in the waveforms can be observed on the expansion connectors.

For the waveform diagrams, the following were used:

CN2-18 (P10): COM0

CN2-47 (P26): SEG_A[7]

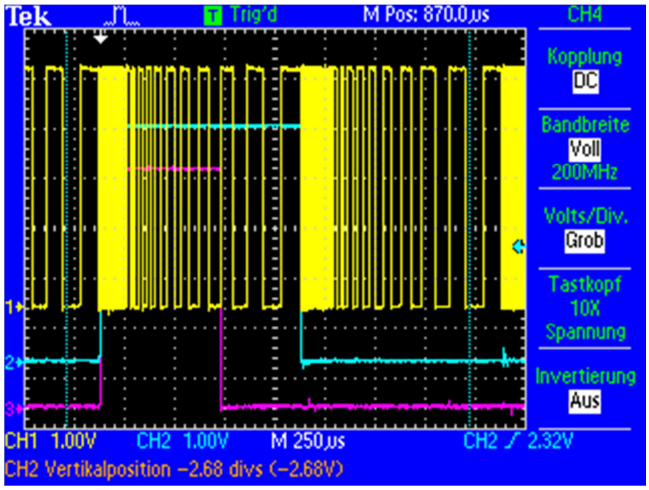
CN2-60 (P01): time slice output from Timer 0 channel 0 (To00)

Note: for the setup, CN2-60 (P01) needs to be externally connected to CN1-41 (P31) and CN1-42 (P64)

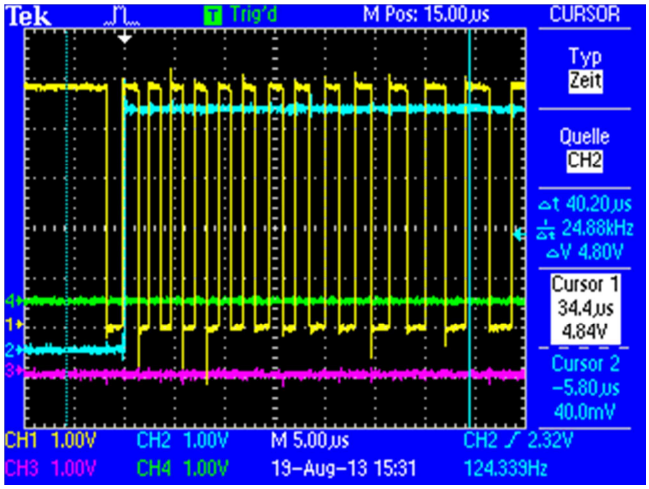
Measured waveforms

The yellow signal is related to the time slices (just for debug), the blue signal is the COM0 activation line, the purple signal is SEG_A[7], showing the different dimming levels, the green one green is the adjacent COM output (COM7 or COM1). The pictures below show several conditions and features of the signals, focusing on the COM0 / ROW7 output.

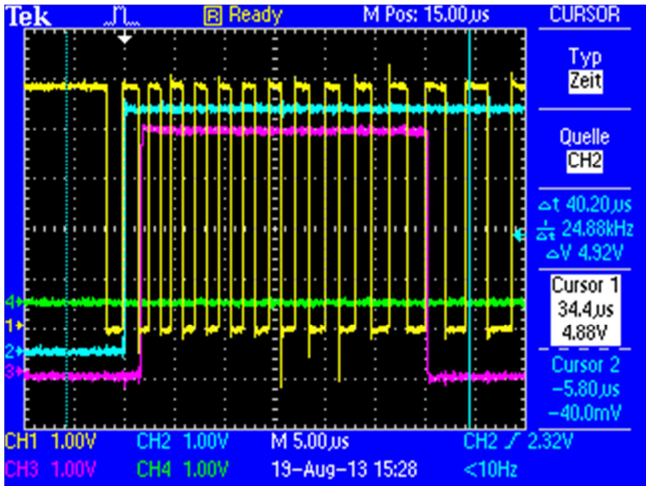
Exponentially incremented time slices



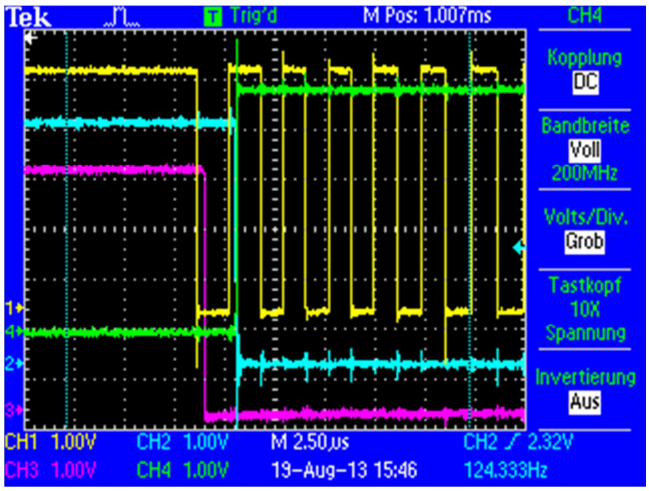
Output off



LED brightness level between 0 and max



Dead band insertion between the LED de-activation signal (purple) and the end of COM0 (blue) / beginning of COM1 (green), when the LED is at maximum brightness.



Dead band insertion between the LED activation signal (purple) and the beginning of COM1 / end of COM7 (green).