

Moving Message Display

Operates
PC-Less

FULL
Character Set!

BY

Trent Jackson

Moving message displays are great for advertising & displaying important information to consumers. You see them everywhere now, clubs train stations, shopping centres, you name it. They are here now and here to stay. . .

PROJECT OVERVIEW

The very good news is that this low cost design presented here boasts many, many numerous features that include the on-board data retention via an EEPROM of up to 4,096 characters for a message. Having said that, it would be obvious to the fact that the unit operates without a PC connected. The PC is only required for programming in your message. This is done via any spare serial port on your PC. Another key feature worth noting is that the Windows based software, it's fast and it's a walk in the park to use. The entire (7 x 35) LED matrix display is made up of 245 individual LEDs. Seven rows and 35 columns. Just the right size. You wouldn't want it any smaller nor would you really want it much bigger unless you have an application specific task intended for it. Going larger would, I feel, price the project out. The use of individual LEDs over block modules means that the colour & brightness can be selected to suit. Yes, LED array modules are far cheaper but also hard to get. They are typically only available in a limited range of colours, most often only red.

Ok so that's all the cause for celebration, what about the gloom & doom?. Well, it's a big circuit, not complicated just big. Mostly step & repeats of sections to drive all the columns via 4017 decade counters that are cascaded as such. There's a fair degree of construction involved, if you like building projects then you'll love this one. Two large PC boards make the project in whole. There's a display board which houses all 245 LEDs and there's a controller board which accommodates all of the required logic circuitry. The display board is stacked on top of the controller board using hex 15mm spacers. Both boards link up electrically through the use of pin headers and rainbow cable. The end result is a neat configuration that reflects a number of hours of tedious work involved. The tedious work is mostly with the 200 odd wire links and 496 solder connections for the LEDs. WHAT?, %&^*\$#@, 200 wire links!. Well, connecting up the LEDs on the display board to form a fully controllable 7 x 35 matrix array on a single sided board, there's really no other way except for the use of many, many small links. They're only very small, about half the size of a staple. This is why most message displays of this size and calibre use double sided through hole plated boards. There's a big cost increase in producing these boards. It took me about 2hrs to leisurely install & solder all the links & LEDs. So, it ain't all that bad.

In fact, this project would ideally suit anyone who's looking to learn how to solder. Or even just to brush up on it. There's a lot of it involved and if you can't solder properly with your eyes closed after building this then soldering just isn't for you. The circuit actually contains a lot of the basic fundamentals of digital electronics. Discrete logic gates and counters in conjunction with a PIC 16f628a micro is used to control the display. In practice, programming in the message involves pressing the program button on the back of the unit so that the PIC micro is ready to receive serial data from the Windows based software. This data is then transferred into a serial 24LC256, 256Kb EEPROM for later recall. After successful programming the display will scroll from right to left the message that you just programmed in. The scroll speed can be "tweaked" via the Windows based software and you can set the number of times the message repeats. In data terms, what's actually stored on the EEPROM is the row data that is broken down from the characters that you program. The characters are all upper case, mostly (7 x 5), which require 5 bytes of row data in storage. Other characters may typically require somewhat less. In normal operation the PIC just scans through this data in the EEPROM and outputs it at the rows and provides a clock source that increments 5 cascaded decade counters that are connected to the columns. It's just a common multiplexing arrangement procedure, so nothing really all that new.

But before we progress any further, lets shed some light on the term "multiplexing". Multiplexing is a rapid switching technique that allows for a matrix array of basically anything, LEDs in this case, to be controlled by a reduced number of of wire connections. This is the actual beauty and reason behind the use of multiplexing, to reduce the number of electrical connections required. Multiplexing a LED matrix array is an optical illusion as such that many, many of the LEDs within the matrix appear to be on at the same time. But, in fact it's really a case of very fast switching

that causes our eyes to perceive this. Typical multiplexing schemes, including ours used here, the columns within the matrix are switched on for a short period of time. During this time of activation the rows are switched simultaneously with appropriate data. This allows for individual LEDs to be controlled at any given time. If the speed of switching is fast enough, no flicker can be noticed. Precise software timing makes the results of multiplexing just as good as using individual lines to control the display. Five cascaded decade counters with gating are used to switch the columns. As each column is switched on via it's corresponding decade counter output, the PIC micro spits out data to the rows which conforms to one broken down part of the text character. As the counters progress in count, more data is shuffled in to the rows. Eventually, after all of the 35 columns have been scanned, the process loops back to the start and resumes again. Each column is switched on 23 times per second for about 1.2ms. This works out to be a 2.8% duty cycle. As a result circuit efficiency is excellent!.

Multiplexing itself isn't a terribly complicated procedure but it does become a little involved when actually "moving" the message and retaining all of the multiplexing at the same time. Especially with 245 LEDs. Another thing that must be taken into account is the duty cycle in which the LEDs are switched at. Even with generous amounts of current being switched, a real low duty cycle will result in insufficient levels of brightness from the LEDs. The duty cycle used in this circuit for switching the columns is extremely low and will result in totally insufficient levels of light output from standard 10mcd rated LEDs. The solution is the use of high-brightness 500mcd LEDs. These LEDs are slightly more expensive but give much more light output for the same amount of current when compared with common ordinary diffused types. Hence they are termed as being much more efficient. From some degree of experience, and reading the words from other authors of similar such projects, I have to agree mostly that the minimum required duty cycle in order to be able to use common diffused LEDs is around 10%.

Multiplexing Techniques

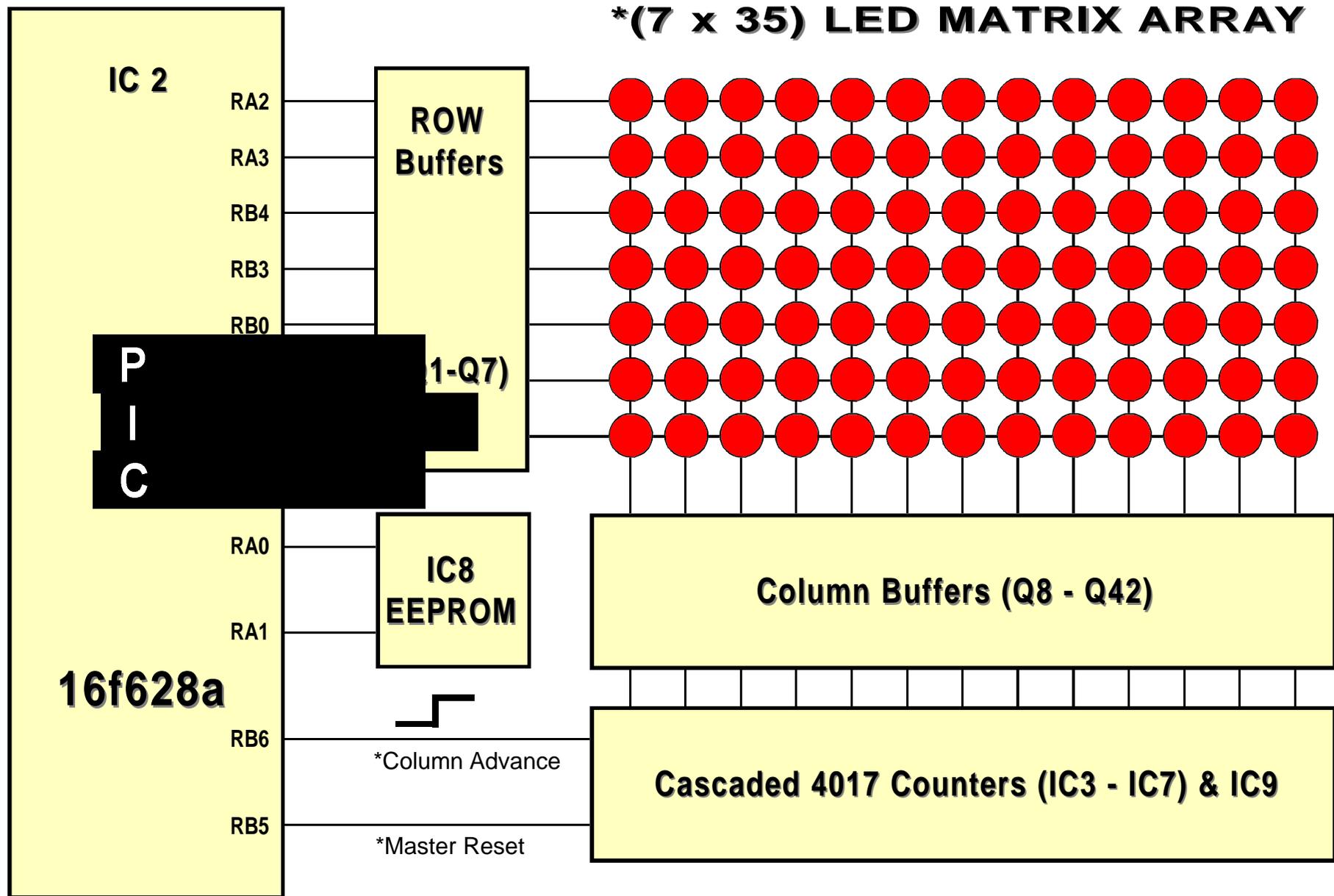
I recall some time ago in the mail bag section of Silicon Chip a reader writing in and asking about the techniques used for moving message displays so that there's no trailing effect on the message. When I first wrote the software for the prototype and powered it up I had this problem actually. The solution isn't very obvious, it took me half an hour to sort it. During the transition of the columns switching you have to switch off all the rows first. And, most importantly before you switch in new data to the rows you must switch on the column. Always take other things into account like propagation delays and any "dead times" that may be within the circuit.

Sequence of events for proper multiplexing & scrolling.

- 1) Switch off all rows
- 2) Switch on desired column
- 3) Output new data to rows

With just a mere couple of options and controls to learn you'll have your first message up on the display in just a few seconds!. It couldn't be any easier than this. Just plug the display into a spare serial port on your PC, run the software & select the port number, press the program button on the display, key

SIMPLIFIED BLOCK DIAGRAM



CIRCUIT EXPLANATION

At first glance the circuit may seem complicated, but it's not. It's merely step and repeats of sections mostly. Step and repeat of sections used to drive the columns of the display. More on this in a moment. Power is derived from any DC source between (9 - 12V) and is feed via reverse polarity protection diode D1 directly to the input of IC2 which is a 5V regulator that provides the entire circuit with an adequate supply. IC2 runs very cool because of the very fast switching of the rows & columns used to drive the LEDs. The heat sink is optional and not really required as such. There's nine integrated circuits & 42 transistors used. All 42 transistors, (Q1 - Q42) are merely current buffers that are used for both the rows & the columns. IC1, PIC 16f628a is the heart of the circuit which calls all of the shots. In normal operation when the display is scrolling a message its task is to read data from the EEPROM, clock out a pulse to the counters which drive the columns, then spit out data at the rows. This is all done very, very fast as you would imagine. A 10MHz crystal and associated 22pf caps provide a clock source for the micro. All of the magic is in the software. The bulk of the hardware is configured as a "work horse" if you like. It's a no brainier. During programming of the message into the display via the Windows based software, the micros main role is to read the serial data and organise it into the EEPROM.

Cascaded 4017 Counters That Drive The LED Columns

In essence a 4017 decade counter receives clock pulses at its clock pin and in turn increments one of its output lines (Q0 - Q9) sequentially. When first powered up Q0 will be set high. As soon as the first clock pulse arrives Q0 will go low and Q1 will go high. At the arrival of the next pulse Q1 will swing low and Q2 will go high. There's only ten outputs on this IC so by itself it can't possible be used to drive the required 35 columns. The solution is what's called a cascaded arrangement. (IC3 - IC7) & IC9 are configured to provide such an arrangement with 35 outputs. Here's how it works. RB6 on the micro provides a clock line for all of the counters. RB6 connects directly to the first counter IC3 while all the other counters receive a gated clock pulse from their associated AND gate output, IC9 which is a quad two input AND gate. These gates ensure that only one counter at a time actually receives a clock pulse. After the first counter IC3 reaches the end of its count Q9 will swing high and enable clocking to the next counter. Q9 is also connected to to the CP1 enable pin on each counter and when Q9 is at a high state the counter will no longer acknowledge any more clock pulses until reset by the proceeding counter. Once any counter has reached the end of its count it will disable itself and enable counting for the next one. With this configuration there's no dead time except for propagation delays. Typically 9nS. RB5 on the micro is connected to an OR gate comprising of 5 diodes (D3 - D7) and provides a master reset for all the counters. RB5 briefly swings high at the start of a message so that the message actually starts scrolling from the immediate right of the display and not some where in between. Without it, the message could start scrolling from any column depending on the position of the counters. All of the counter outputs are buffered via (Q8 - Q42) and are driven quite hard into saturation by their 1K2 base resistors.

The PIC directly controls the rows via emitter follower buffers (Q1 - Q7). These are also driven well and truly into saturation by their respective 1K2 base resistors. 18Ω resistors limit the peak current to the LEDs. RB7 on the micro is the RX line designated to receive the serial data from the PC. Because we are only using the TX line on the serial port to program in the message and due to the excellent I/O architecture on the 16f628a, no RS232 line level converter IC is required nor implemented. Instead we can rely on two 10K resistors to drop the voltage and limit the currents safely. RA0 & RA1 connect to the clock & data lines of IC9, serial EEPROM. This is a 256Kb device which operates serially at up to 400KHz!. The two 47Ω resistors are for noise suppression while the 4K7 pull ups are essential since the data & clock lines are bipolar.

PARTS LISTING

RESISTORS 0.25W 1%

42	1K2 Ω	3	4K7 Ω
7	18 Ω	6	100K Ω
2	47 Ω	2	10K Ω
1	300 Ω		

CAPACITORS

8	100nF MKT
6	10uF Electro
1	1000uF Electro
2	22pF Ceramic Type

SEMICONDUCTORS

2	IN4007 Power Diodes (D1 & D2)
5	IN914 Signal Diodes (D3 - D7)
246	5mm, 500mcd LEDs, Any Col
42	BC548 NPN Transistors (Q1 - Q42)
1	LM7805 5V Regulator (IC2)
1	PIC16f628a Micro Programmed With LEDMSG.hex (IC1)
5	4017 Decade Counters (IC3 - IC7)
1	24LC256 EEPROM (IC8)
1	74HC08 Quad AND (IC9)

1	D9 Female PC Mount
1	DC Socket PC Mount
1	10MHz Crystal (Xtal 1)
2	PC Mount Push Button Switches
4	Mini TO-220 Heat Sink

1	30CM Length Rainbow Cable
1	1M Length Tinned Copper Wire
1	8 Way PC Mount Headers & Connectors
1	4 Way PC Mount Headers & Connectors
1	8 Pin IC Socket
5	16 Pin IC Socket
1	14 Pin IC Socket
1	18 Pin IC Socket

4	25mm M3 Hex Tapped Spacers
4	12mm M3 Hex Untapped Spacers
5	5mm M3 Screws
1	5mm M3 Nut
4	20mm M3 Screws
4	Piece Of Red Perspex 76 x 288 mm
1	12VDC @ 500mA Plug Pack
1	LED Display PCB, 76 x 288 mm
1	Main Controller PCB, 76 x 288 mm