BIG DIGIT LED DOT MATRIX CLOCK

(eeye © 2015 ☺)

Salient features

30x7 Dot Matrix, uses 2.0" high brightness, displays for good visibility even in bright daylight	Extremely low power operation (0.75W) 150mA at max brightness and 20mA at the min brightness.
FAIL SAFE monitoring of microcontroller clock to ensure safe operation of the LED Dot Matrix	20kHz, 10 bit PWM for flicker free brightness control (*auto and manual mode, user preferences, see text*)
70Hz display refresh rate to ensure flicker free visuals and animations	20 levels of manual brightness adjustment
Interlaced scanning to eliminate display artifacts (e.g., skewing of characters)	Date display in fully expanded plain English (*e.g., Thursday 20th November 2014*)
12/24 hour clock operation	Function to adjust deviation of the clock. (*Adjustment up to 2 minutes / day deviation possible see text*)
Adjustable scroll rate of the display from 20ms to 100ms (*user preference*)	User preferences with graphical symbols.
Static and scrolling clock.	Static clock from 8:00 PM until 6:00AM ☺
Hourly Chime and Alarm functions (*user preference*)	Battery backup of the clock and user preferences using a regular 3V coin cell

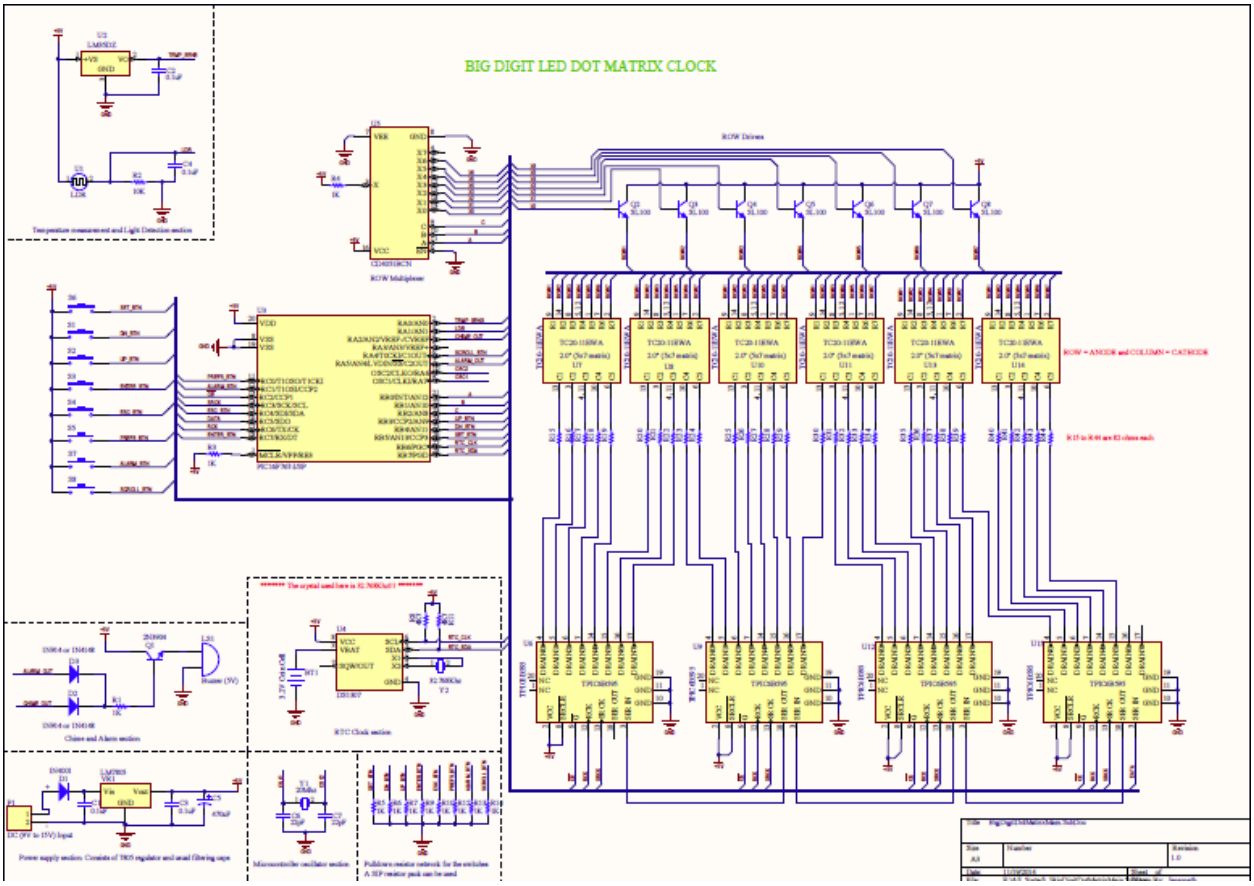

Figure 1: Schematic of the clock

Figure 1, shows the entire clock. The circuit that drives the dot matrix display is fairly simple. It consists of a microcontroller (U3), a bunch of shift registers (U6, U9, U12, U15) and finally the row multiplexer (U5) with the associated row drivers (Q2 to Q8). The remaining blocks are there to acquire the temperature, ambient light and user interactions.

Row Driver

Let's understand the uniqueness of this row driver. If you look at most of the designs presented on the internet, they all invariably use the 74LS138 for multiplexing the row drivers. The problem with the 74LS138 is that firstly it is a negative logic, which means the selected row goes low. Secondly, each and every transistor in the row driver will need a resistor at the base to get the bias current going (*remember transistors are current driven devices [$I_C = \beta I_B$]*). Lastly, the 74LS138 is not CMOS chipset, which in turn means that we will need more power than the microcontroller to keep that chip running. *Honestly, I'm not a great fan of negative logic, unless it is absolutely necessary to do it.* Hence here we have chosen the CD4051 (U5), which is a CMOS chip and has positive logic. The neat feature of this chip is that the voltage applied on pin 3 is the output available on the outputs (*X0 to X7*). Hence, by putting a resistor (R4) between +5V and pin 3 of U5, we have essentially created the bias current needed for those row driver transistors. At any given point in time, only one row transistor is active (*due to multiplexing*) all we need is just one resistor to set the bias current for all those transistors. Cool isn't it.

The row driver transistors have to carry the entire current to the individual dots. Hence the need for selecting the right driver transistors becomes critical. In this case the SL100 (*NPN*) transistor has been selected. Why? Assume that all the LED's on a row are light up. In this case we have 30 LEDs, each one drawing 10mA, which makes it 300mA. The SL100 has an I_C of 500mA, which is well within the limits of our design. *You can use any other NPN transistor that meets this requirement, for example 2N2222A can be used. It is normal for these drivers to become warm during operation and should not be construed as a malfunction.*

Column Driver

The column drivers (U6, U9, U12, and U15) are also unique in the sense that they are shift registers with built in current sinks. The current sinks are not the usual Darlington transistor, but MOSFETs with open drain. This gives the chip the ability to sink 150mA continuously per pin, when all the pins are turned on. We need this capability as each LED at full brightness is driven to 70mA pulsed current and we have 8 LEDs being driven by one chip. It is worth noting that even though we are pushing 70mA continuous current, in reality each LED will see only 10mA as the display is multiplexed. So there is nothing to fear that the LED's will burn out. *Unless, you turn on the matrix without the microcontroller, it is possible to destroy the dot matrix due to the uninitialized behavior of the electronics and the high continuous current in the LEDs.*

The column drivers are cascaded to achieve the 30 dot string (*the serial output from U15 is connected to the serial input of U12 and so on*). To get a fast data transfer rate, the microcontroller communicates with the column drivers over SPI using the microcontroller built in SPI hardware. The SPI protocol needs 3 lines to achieve its objective. In this case the lines DATA, SRCK and RCK are used for the SPI. The data for the shift registers is sent on the DATA line, the serial clock is sent over the SRCK. Once the entire data is clocked out, the RCK is toggled, so that the data is latched into the shift register.

Time Keeper (Real Time Clock / Calendar)

The hard work of time keeping together with the day, date, month and year is done by the RTC chip (U4). The RTC chip takes care of the adjusting for the days in the month, compensates for leap years and it is fully programmed until 2100 (*enough for a lifetime*). It goes without saying that the clock is as accurate as the RTC and the crystal that drives it. So it is very important to ensure that the crystal used to drive the RTC is accurate and is per the specifications of the chip. Fortunately, most of the watch crystals are accurate enough to give a deviation of about 2 seconds/day. Well you may say 2 seconds/day is a lot isn't it? Yes, it definitely is 'cause in 30 days we have a clock that is running either 1 minute fast or slow (*depending on how the crystal is behaving*). Thankfully, this clock has a feature were in you can adjust the deviation due to the crystal. Once set the clock is accurate to the second.

The power consumption of this RTC chip (U4) is extremely low (*about 1.5mA*) with 5V supplied. When powered by the backup battery the chip draws a mere 500nA resulting in a battery backup that will last for 10 years. The RTC is interfaced to the microcontroller with the I2C interface. The microprocessor that does the calculations on the RTC is also used for communicating with the external world (*via I2C*). To ensure that the RTC is able to perform its job accurately, the data from the RTC is requested once every

second. This way we can guarantee that the RTC is not overloaded and is able to perform its job accurately.

Temperature measurement

The ambient temperature is measured with the help of LM35 (U2). The temperature sensor is factory calibrated and is pretty accurate to 0.4^{o}C. The sensor outputs 10mV per degree rise in temperature which is easily measured by the onboard 10bit ADC on the microcontroller. The output from this sensor is connected to pin 2 of the microcontroller (U3), which is configured as an ADC. The ADC samples the sensor for 18ms and provides the averaged value, which is converted into the appropriate humanly understandable centigrade value.

Brightness control (Ambient Light measurement)

The brightness of the dot matrix is adjusted automatically based on the ambient light conditions. The ambient light is measured by the LDR (U1). The resistance of the LDR changes with the amount of light that is impressed on the sensor. This property has been utilized in the traditional way, with a resistive divider network, comprising of the LDR and R2. The change in resistance of the divider network results in variable output voltage. This voltage from the divider network is fed to pin 3 of the microcontroller (U3), which is configured as an ADC. To avoid unwanted display blinking the voltage is sampled 10 times and averaged. The changing voltage influences the PWM which controls the OE (*output enable*) pin of the shift registers (U6, U9, U12 and U15), that in turn sets the brightness for the display. *On a side note, the brightness can also be controlled manually by way of user preference setting.*

The capacitors C2 and C4 serve as a discharge path for the microcontroller's internal sample and hold capacitor. Without these capacitors, the internal sample and hold capacitor does not have a discharge path and due to the ADC being multiplexed, the resulting digital output is not accurate. Additionally, the capacitors server to stabilize the input signals by removing spikes and noise. Using too high/low value for these capacitors will result in over/under damping of the signals. The values shown in the schematic were chosen to critically damp the system.

Alarm & Chime

The alarm and chime functionality is implemented using D2, D3, R1, Q1 and LS1. The D2 and D3 are wired as OR gate and the outputs from Alarm and Chime are fed into them. Q1 acts like a switch to turn on or off the buzzer (LS1). R1 is used to limit the current to Q1.

Hourly chime is implemented as two short beeps of 100ms each. The Alarm function is implemented as implemented as beeps of 500ms. The Alarm keeps sounding until the user presses any key to cancel it.

Microcontroller

The heart of the clock is the microcontroller (U3), which is the PIC16F767. This chip has 8K of flash and 368 bytes of RAM. It contains a 10 bit ADC module, a 10 bit PWM module, the usual input/output ports and most of all it has the "fail safe clock monitoring". Look at Figure 1, and you can see that the

microcontroller is interfaced in the usual way to the various push buttons (S1 to S8). As explained previously the ADC and the shift registers are also connected to the microcontroller. The master reset (MCLR pin) is connected to +5V via a 1K (R3) resistor.

The unique feature of this particular microcontroller is it's "fail safe clock monitoring". The fail safe clock operates at a reduced frequency the moment the main external oscillator fails. This feature has been utilized to safeguard the LED dot matrix in case of microcontroller oscillator failure. (*Remember our earlier discussion that each LED is being pulsed with 70mA current. Should the microcontroller fail, the multiplexing of the dot matrix stops resulting in 70mA of continuous current flowing through each LED killing the dot matrix instantly*). The fail safe feature of the clock operates as follows. During startup the failsafe clock monitors the oscillator startup. If the oscillator does not start up, then a hardware error is set. This error is processed in the software and the microcontroller is switched to the reduced internal clock. This internal clock then provides the multiplexing of the display (*however at a reduced rate, which is obvious as the display can be seen flickering*). The message "Service Clock!!" is displayed continuously.

If the oscillator is okay at startup, then the microcontroller wakes up and the clock is operating. Should the oscillator fail during normal operation, then the software triggers the reduced clock operation. All current operations are cancelled and the message "Service Clock" is displayed.

Construction

The clock can be constructed on a prototype board; however it is advisable to use a PCB. Firstly, the sheer number of wires from the dot matrix display that need to be wired in the correct order is one challenge. Secondly, to ensure accuracy the quartz crystal to the RTC needs some careful consideration about its placement and isolation. Lastly, the I2C and SPI interface need careful consideration to ensure reliable data transmission as they are operating at 100 kHz and 19 kHz respectively.

Software and operation

To keep the article manageable and also to help you build the clock successfully, the software has been released as a precompiled HEX file. The software is a downloadable HEX file, which has all the fuse settings for the microcontroller fully programmed. You'll need a programmer that can program the PIC16F767 including the fuses.

On power up, the software does basic initialization of the hardware, and the display should come to life with the scrolling message "The time is….." If this does not happen, see the troubleshooting section. At first power on the RTC is programmed to the default 24 hour operations and it displays 0:00 hours on 1st Jan 2000.

Setting the Time and Date

As an example let's say we want to set the time to 10:35AM on Thu, 20th Nov 2014. To set the time and date do as follows:

(a) Press SET button.
(b) Display indicates “Set Time & Date?”
(c) Press ENTER button. (*The ENTER button confirms that you are entering the SET TIME AND DATE function*).
(d) Display indicates 24. This is the default operation mode of the RTC (*24 H mode*). Use the UP/DN keys to navigate between AM/PM/24 hours. For the above example use the DN button until the display indicates AM.
(e) Press SET, the display indicates “H->01” (*Hours*)
(f) Press UP key until the display shows “H->10”. If you go past the 10, don’t worry press DN button to come back.
(g) Press SET button, the display indicates “M->00” (*Minutes*)
(h) Press UP key until display shows “M->35”. If you go past 35 don’t worry press DN button until it reaches 35.
(i) Press SET button, the display indicates “S->00” (*Seconds*). (*If you want to adjust seconds, use the UP/DN keys as described above for the hours/minutes setting.*)
(j) We are not adjusting seconds, so press SET key.
(k) The display shows “Yr->14” (*Year*). The current year is 2014, so nothing to do here, press SET button.
(l) The display shows “Mn-> 01” (*Month*). The months start from Jan = 01 until Dec = 12. For the above example press UP until the display indicates “Mn->11” (*which is Nov*).
(m) Press SET button. The display indicates “Dt->01”. Use the UP key until the display shows “Dt->20”. (*The number of days in the month is automatically calculated, so that the month of Feb in normal year will go until 28 and for leap year it will go up to 29 and so on for the other months.*)
(n) Press SET button. The display indicates “Dy->Su”. Press the UP button until the display indicates “Dy->Th”. (*Su = Sunday, Mo = Monday…. and so on.*)
(o) Press SET button. The RTC is updated and the message “Saved” will be displayed for one second.
(p) Now you are done. The clock is programmed to the time and date in the example. The scrolling will be resumed displaying the current time and date programmed

Note:

(1) Pressing the ESC button anytime during the steps above from (d) to (n) will abort the current operation and nothing in the RTC will change. The message “Abort” will be displayed for a second. The scrolling will resume with the old values in the RTC.
(2) Time and Date are set simultaneously to avoid data corruption of the RTC

Alarm Setting

Let’s set an alarm for 6:10AM.

(a) Press the ALARM button.
(b) Display indicates “Set Alarm?”

(c) Press the ENTER key. (*The ENTER button confirms that you are entering the SET ALARM function*).
(d) Use the DN key until the display shows AM. (*The UP/DN keys have no effect if the clock is running in the 24 hour mode. The display will show 24)*
(e) Press the SET key.
(f) The display shows “H->01”.
(g) Use the UP key until the display shows “H->06”
(h) Press the SET key
(i) The display shows “M->01”
(j) Use the UP key until the display shows “M->10”
(k) Press the SET key.
(l) The message “Saved” will be displayed for one second and the alarm is set.

Note:

(1) Pressing the ESC button anytime during the steps above from (d) to (j) will abort the current operation and nothing will change. The message “Abort” will be displayed for a second. The scrolling will resume.
(2) If the operation mode of the clock is changed from AM/PM to 24 hour, then the alarm time needs to be set again. This is because the alarm setting is based on the clock mode.

Checking Alarm Set Time

To check the time for which the alarm is set. Press the ENTER key while the clock is in normal operation (*scrolling / static*). If the Alarm is Enabled in the preferences, then the display will show “Alarm set for x:xx AM/PM”. If the Alarm is Disabled in the preferences, then the display will show “Alarm Disabled”

Scrolling / Static Clock

Press the SCROLL key once to change the mode from scrolling to static and vice versa. In the static mode the Clock and Temperature is alternately displayed, once every 5 seconds. A fade effect is applied to the alternating clock and temperature. The amount of fade applied is based on the ambient lighting or manual brightness. Hence if the clock is in a bright place, the fade is strong, if it is in a dark place the fade is weak. Why? During the night, we want to quickly get to the time instead of waiting for the fade to complete.

In the scroll mode the complete text “The time is x:xx on <Day>, <Date> <Month><Year>. The ambient temperature is <xx.x>^{o}C.” is displayed.

Static mode is automatically activated from 8:00PM until 6:00AM. However should you wish to override the same, press the scroll button to resume scrolling.

Preferences

There are various preferences the user can set. The preferences are described below.

(a) To access the preferences, press the PREFS button.
(b) The display will show “Prefs” and then it will display
(c) “Al->speaker symbol X”, which means Alarm is Disabled. Press UP button to change to enabled. Display will indicate “Al->speaker symbol with waves”. To disable press DN button.
(d) Press SET.
(e) The display will show “Ch: Bell X”, which means Hourly Chime disabled. Press UP button to change to enabled. Display will indicate “Ch: waves Bell waves”. To disable press DN button.
(f) Press SET.
(g) The display will show “Br: Bulb symbol”, which means Automatic Brightness control enabled. Press DN button to change to manual mode. The display will show “Br: up/down arrows”. Press UP again to change to automatic mode. Display should show “Br: Bulb Symbol”.
(h) Press SET.
(i) The display will show “Sr: 30”. (*Sr => Scroll Rate*). The scroll rate can be set from 10 to 50 which correspond to 20ms to 100ms. Therefore the lower the number set, the faster is the scrolling and vice-versa. The default value is 30 which correspond to 60ms. Experiment with the various values to determine the best scrolling rate to your preference.
(j) Press SET.
(k) The display will show “D: 0”. (*D => Deviation in RTC clock*). The value here can be set from -59 to 59. The values set here correspond to the number of seconds the clock is gaining or losing. Negative values indicate that the clock is going fast and positive values indicate clock is going slow. For example if the clock is losing 10 seconds per day. Then you have to set here half the value, which in this case happens to be 5. If the clock is running fast for example if it is gaining 10 seconds per day, then you set here -5. (T*he clock is adjusted twice in the day automatically using this value*. *So after you build the clock observe it for a week. Find out the number of seconds the clock is gaining or losing per day. In this setting, enter half the value of the observed deviation.*)
(l) Press SET. The display will indicate “Saved”. The preferences are saved into the NV RAM.
Note:

(a) The user preferences are restored from the battery backed RTC NV RAM.
(b) Pressing the ESC button anytime during the steps above from (c) to (k) will abort the current operation and nothing will change. The message “Abort” will be displayed for a second. The scrolling/static display will resume.

Special functions

(a) If Manual brightness adjustment is selected in preferences. Then on pressing the UP/DN buttons during the normal clock operation (*scrolling / static*), the brightness can be increased or decreased. *The manually selected brightness is retained and does not change with the ambient light.*

(b) If Alarm is disabled in the preferences, then on pressing the ENTER key during normal clock operation the message “Alarm Disabled” is shown. To enable alarm go back to preferences and enable Alarm.

Assembling and Testing

Now that you have assembled the clock, it is time to test it before it is put to use.

(a) Solder and wire up all the components as per the schematic. Do not insert any IC’s into the sockets.
(b) Do not connect the battery for the RTC backup.
(c) Apply 9V DC to the connector P1. Make the following measurements with respect to GND.
(d) Check the output of VR1 (Voltage Regulator). You should get +5V.
(e) Check the voltage at the VCC pins of the various IC’s. The voltage should be +5V
(f) Check the voltage at the pins of the switches. The voltage should be +5V.
(g) Remove the power, and insert all the chips except the microcontroller. Insert the chips with the proper polarity (i.e., pin 1 should go into pin 1 of the socket).
(h) Turn on the power again. The display should be blank. Check the voltages at the VCC pins of the IC’s. They should be +5V.
(i) Remove the power and insert the programmed microcontroller into its socket.
(j) Apply power again.
(k) If everything is done correctly, the clock will spring to life. Follow the steps previously and you can set up the clock.

Troubleshooting

In the unfortunate event that the clock does not spring to life, follow the steps below.

a. Unplug the clock from the power. Using a multimeter check if all the IC’s have the GND and +5V connection hooked up properly.
b. Check that Pin 1 of the microcontroller is connected to +5V using a 1K resistor (R3). This pin is the reset and leaving it floating will cause the microcontroller to be in reset state.
c. Check if the dot matrix displays are not accidentally connected upside down. Use a multimeter to check that the cathode of the displays is connected to the 82 ohm resistors (R15 to R44).
d. If the problem exists in the above steps and it is fixed the clock will spring to life. If not go to the next step.
e. Power on the clock.
f. Using an oscilloscope or frequency counter measure the frequency on pin 13 and 14 of the microcontroller. The frequency will be in between 19 kHz to 20 kHz (*PWM and SPI clock*).
g. Measure the frequency on pin 21 of the microcontroller. The frequency will be 70Hz (*row refresh clock*).
h. If these frequencies are obtained then the microcontroller is functioning. Otherwise try reprogramming the microcontroller and ensure that the fuses contained in the HEX file

are also programmed. *If the fuses are not programmed correctly, the watchdog monitoring will keep resetting the microcontroller.*

i. If the frequencies specified in steps (f, g) are being obtained and still the clock does not spring to life.
j. It is possible that the communication to the RTC is not happening which is causing the processor to go into endless loop. Hold the ENTER key while power on the clock.
k. Check the output of the RTC pin 7 of U4. If the communication is successful and the crystal (Y2) is oscillating, a 1Hz output will be available on this pin. (*Use an LED with a resistor and connect it to +5V and pin 7. The cathode of the LED should be connected to pin 7. The LED will be blinking at 1 Hz rate*).
l. Fix the problem at the RTC if the 1Hz is not available.
m. Once done the clock should spring to life.

Happy holidays and have fun. I hope you enjoy constructing and using this clock as much as I have enjoyed engineering it.