FINAL PROJECT: CONTROLLING A DOT-MATRIX LED DISPLAY VIA A PIC MICRO-CONTROLLER

AZIZ CONTRACTOR

1. Purpose

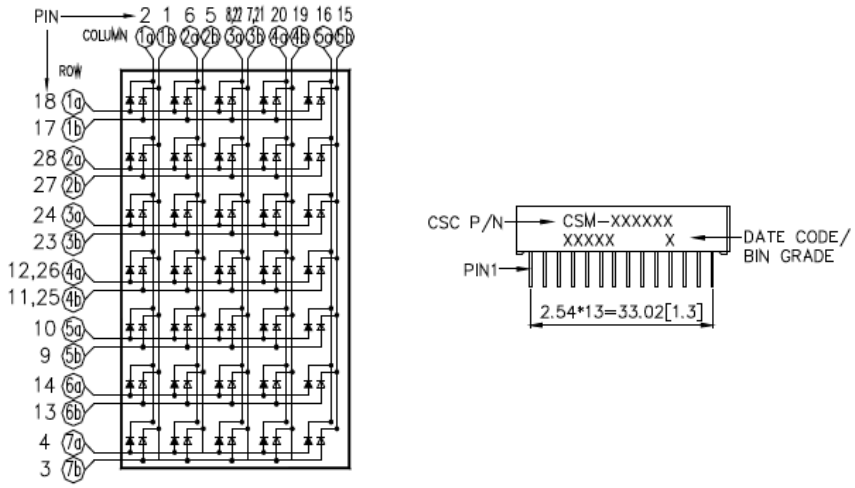
The primary purpose of this project is to help students understand that hardware and software are equivalent. In other words, functions that are normally assigned to hardware components can also be performed by software implementation. The first part of this project includes wiring a 7×5 dot-matrix LED display to a PIC micro-controller, the next, and more important, part consists of implementing a program using a custom FORTH language to loop through all the LEDs of the display one at a time.

2. Introduction

The PIC micro-controller used for this project is a PIC 24FV32KA302, a 28 pin micro-controller part of the PIC24FV32KA304 family. The programming language embedded on this PIC chip is a custom version of the FORTH language called PIC24F FORTH. This system was designed specifically for use with this project and the tutorial is provided on the project website. The major difference between traditional FORTH and PIC24F FORTH is that the PIC24F FORTH does not include some of the predefined words included in traditional FORTH. The purpose of this is to conserve the limited flash memory available

Date: December 14, 2015.

and to help students take a more hands-on approach to the programming aspect of this project. The LED dot-matrix used for this project is the CSM-57281EG, a 2.0 inch 5×7 LED dot-matrix display. This display has 70 total LEDs, 2 different color LEDs for each dot in the matrix. In order to establish a connection between the PIC and the computer, a DB9 USB-to-Serial connector is used along with an RS-232 connector. The circuit diagrams for the dot-matrix display, the PIC and the wiring diagram PIC to the display are shown below.



(A) CSM-57281EG Internal Circuit Diagram

(B) CSM-57281EG Pinout

FIGURE 1. CMS-57281EG dot-matrix display.

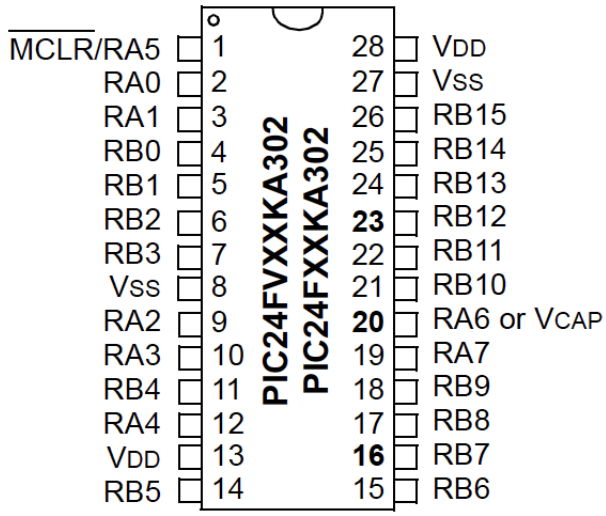


FIGURE 2. PIC24FV32KA302 Pinout

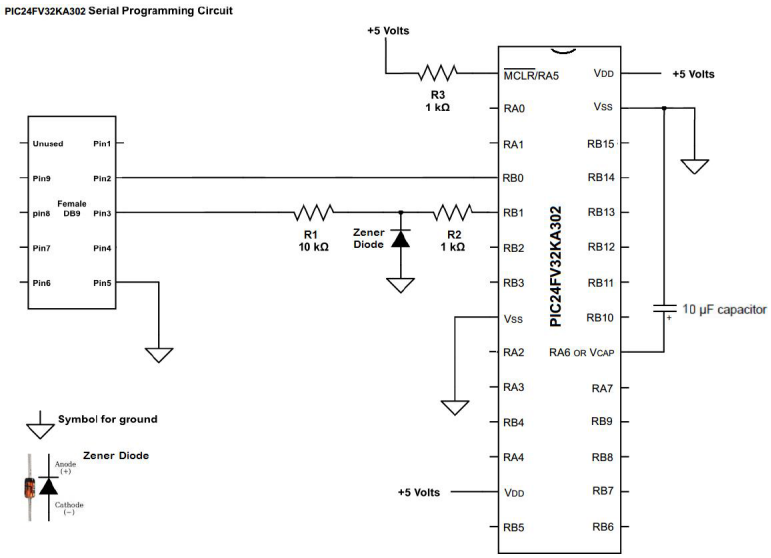


FIGURE 3. PIC24FV32KA302 to RS-232 Circuit diagram

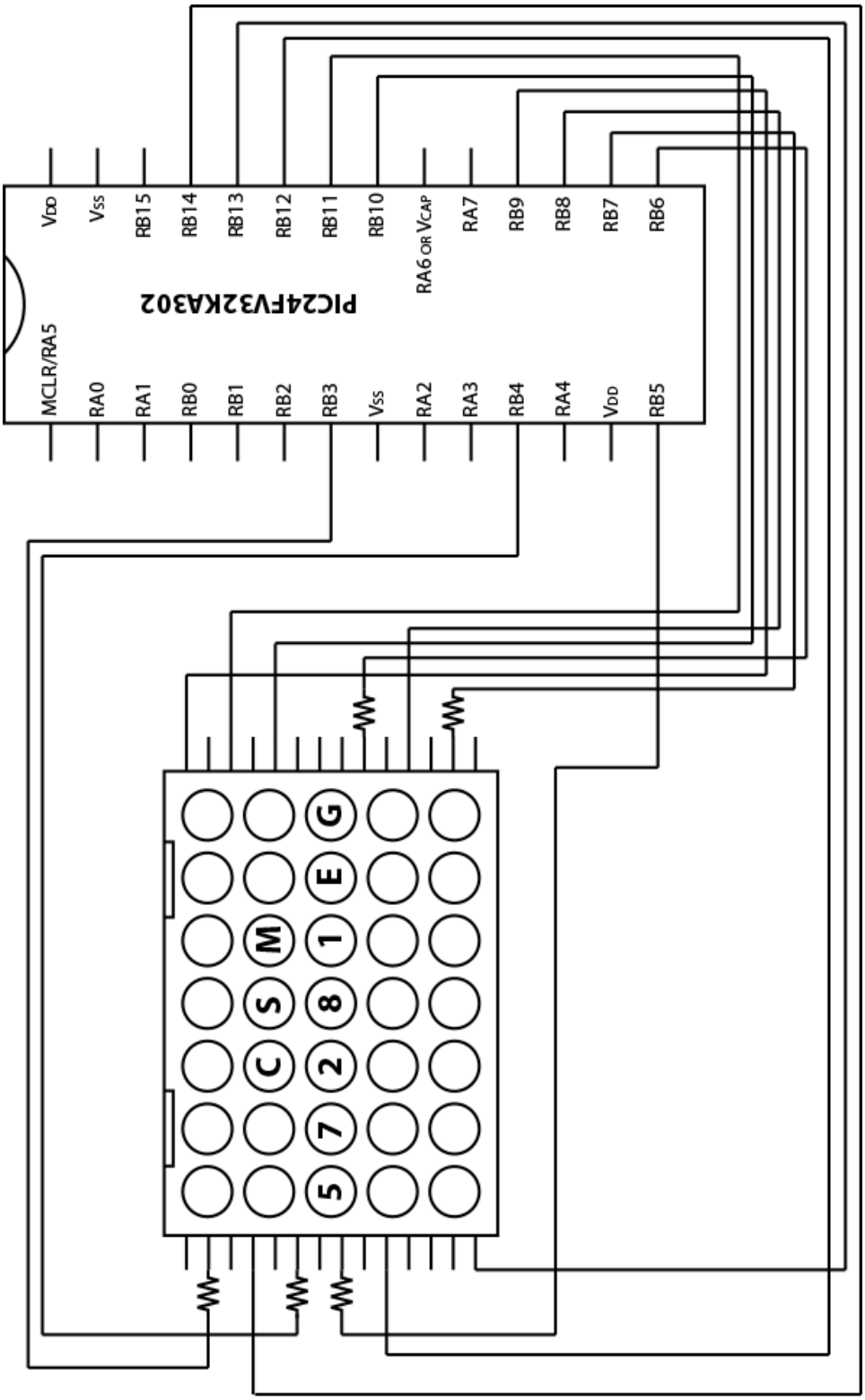


FIGURE 4. Wiring Diagram to connect PIC to Display

3. Materials and Equipment

The materials and equipment used are as follows:

- (1) Breadboard
- (1) PIC24FV32KA302 Chip
- (1) CSM-57281 LED Dot-Matrix Display
- (1) RS-232 Connector
- (1) USB-to-Serial DB9 connector
- (5) 220 Ω Resistor
- (1) 10K Ω Resistor
- (2) 1K Ω Resistor
- (1) 10 μF Capacitor
- (1) Zener Diode
- (3) Red wires for power
- (5) Black wires for ground
- (25) Various color wires for output connections

4. Procedure

4.1. **The Setup.** This section will describe how to setup the RS-232 using the DB9 connector on a Mac. The setup and configurations included here are for Macs only. Windows users should not follow this part of the procedure. The steps are as follows

- Connect the RS-232 to the breadboard.
- Plug the DB9 connector to a USB port of the Mac and connect the other end to the RS-232.
- To test the connection, connect pin 2 and pin 3 of the RS-232.
- Install the drivers for the connector, available here
- Restart the computer.

- Open Terminal.
- Type the following commands in Terminal to check if the connector is installed properly.

```
cd /dev
ls tty.usbserial*
```

- Start the connection to the RS-232 by typing the following command (Note: This command needs to be entered every time you want to access the PIC.)

```
screen /dev/tty.usbserial 38400
```

- Type some characters with the keyboard. If the setup was successful you should see the characters you type on the Terminal screen. If not, then repeat the process or try different equipment.

4.2. **The Wiring.** This section describes the wiring of the PIC chip to the RS-232 and the dot-matrix display. It is essential that the PIC is wired correctly before powering it up. As mentioned before, since only single color LEDs are necessary for the completion of this project, they will be included in the optional wiring section.

4.2.1. *Setting up the PIC.* This section deals with the initial setup and connection of the PIC to the computer. This part should only be done once the connection between the computer and the RS-232 has been established.

- Remove the wire connecting pin 2 and pin 3 of the RS-232 (if applicable).
- Connect the pic to the breadboard.
- Connect pin 1 of the PIC to power using a 1 K Ω resistor.
- Connect pin 4 of the PIC to pin 2 of the RS-232.

- Connect pin 5 of the PIC to the anode of a Zener diode using a 1 K Ω resistor and connect the cathode of the diode to ground
- Use a 10 K Ω resistor to connect the anode of the Zener diode to pin 3 of the RS-232.
- Connect pin 8 of the PIC to ground.
- Connect pin 13 the PIC to power.
- Connect pin 20 of the PIC to the positive end of the capacitor and connect the other end to pin 27 of the PIC.
- Tie pin 27 to ground by connecting a wire behind the capacitor. The connection between the capacitor and pin 27 should not be interrupted
- Connect pin 28 of the PIC to power.
- Connect pin 5 of the RS-232 to ground.
- Lastly, connect the DB9 to the USB port of the computer, turn on the power to the breadboard and access the PIC.

4.2.2. *Wiring up the LED.* Once the PIC is powered up and running, it can be used to control the display. This section describes the wiring of the LED to the PIC. The wiring shown here uses the red LEDs of the display. In order to wire the green color LEDs instead of the red, simply subtract 1 from the display pin numbers listed in the table below. Also, since the port numbers are more important than pin numbers while writing the software, the port numbers will be used to refer to the specific ports of the PIC. The pins that access these ports can be found by using Figure 2. All the cathodes should be connected using a 220Ω resistor.

4.2.3. *Optional Wiring.* This section includes instructions to wire the second color LEDs. For this wiring the anodes of both LEDs will be

Connection	Function
RB3 to pin 2	Connects cathode of column 1a to bit 3 of port B
RB4 to pin 6	Connects cathode of column 2a to bit 4 of port B
RB5 to pin 8	Connects cathode of column 3a to bit 5 of port B
RB6 to pin 20	Connects cathode of column 4a to bit 6 of port B
RB7 to pin 16	Connects cathode of column 5a to bit 7 of port B
RB8 to pin 18	Connects anode of row 1a to bit 8 of port B
RB9 to pin 28	Connects anode of row 2a to bit 9 of port B
RB10 to pin 24	Connects anode of row 3a to bit 10 of port B
RB11 to pin 26	Connects anode of row 4a to bit 11 of port B
RB12 to pin 10	Connects anode of row 5a to bit 12 of port B
RB13 to pin 14	Connects anode of row 6a to bit 13 of port B
RB14 to pin 4	Connects anode of row 7a to bit 14 of port B

TABLE 1. Display to Port Connections and their Purpose

tied together and the cathodes will be connected to separate pins. In order to tie the pins together, take the 220Ω resistor and use it to connect the 2 pins mentioned in the table. Then connect a wire to one of the two pins and use Table 1 to connect the wire to one of the ports of the pic. For example, if you want to tie the anodes of row 1 together, connect pin 18 and pin 17 of the display to each other using the 220Ω resistor. This ties the pins together. Next, connect either pin 18 or pin 17 of the LED to port RB8 of the PIC.

4.3. **The Software.** In this project, the PIC is used to control the LED and code must be written in Terminal and ran on the PIC for this purpose. Every time the PIC is booted up its ports are set to digital input, and must be switched to digital output before programming them.

Connection	Function
RA0 to pin 1	Connects cathode of column 1b to bit 0 of port A
RA1 to pin 5	Connects cathode of column 2b to bit 1 of port A
RA2 to pin 7	Connects cathode of column 3b to bit 2 of port A
RA3 to pin 19	Connects cathode of column 4b to bit 3 of port A
RA4 to pin 15	Connects cathode of column 5b to bit 4 of port A
pin 17 to pin 18	Ties anodes of row 1a and row 1b
pin 27 to pin 28	Ties anodes of row 2a and row 2b
pin 23 to pin 24	Ties anodes of row 3a and row 3b
pin 25 to pin 26	Ties anodes of row 4a and row 4b
pin 9 to pin 10	Ties anodes of row 5a and row 5b
pin 13 to pin 14	Ties anodes of row 6a and row 6b
pin 3 to pin 4	Ties anodes of row 7a and row 7b

TABLE 2. Display to Port Connections and their Purpose

Once this is done, the ports can be set to 1 and 0 using specific commands. A `dowhile` loop can then be written to switch the LEDs off and on using the loop. FORTH allows us to define custom words that can be used to control the ports. These words can be saved to flash memory using the command `CHECKPOINT`. The specific words and their descriptions are listed below. Before using these commands the PIC must be powered on and connected to the computer. The PIC screen should also be loaded using the command `screen /dev/tty.usbserial 38400` in Terminal. Once the PIC is accessible these words can be used to control the pic. For running the loop with both color LEDs please refer to Subsection 4.3.1

```
  :  INIT 0 TRISB 1 + C! 5 [:  DUP 8 SWAP - TRISB SWAP BITCLR
1 - ;] DOWHILE ; // word that sets the needed ports to digital
output.
:  D 15 delay ; // word that sets the delay for the LEDs in
milliseconds.  The number of milliseconds used can be changed.
:  BLINKR PORTB SWAP BITSET D PORTB SWAP BITCLR ; // word that
turns the rows on and off
:  OFFC 5 [:  DUP 8 SWAP - PORTB SWAP BITSET 1 - ;] DOWHILE
; // word that turns all the columns off
:  ROWF 7 [:  DUP 15 SWAP - DUP BLINKR 1 - ;] DOWHILE ; // word
that turn each row off and on one after the other
:  ROWB 7 [:  DUP 7 + DUP BLINKR 1 - ;] DOWHILE ; // word to
do the same as ROWF but backward
:  COLF 5 [:  DUP OFFC 8 SWAP - PORTB SWAP BITCLR ROWF 1 - ;]
DOWHILE ; // word that turns on a single column at a time loops
through all LEDs in that row
:  COLB 5 [:  DUP OFFC 2 + PORTB SWAP BITCLR ROWB 1 - ;] DOWHILE
; // word to do the same as COLF but backward
:  PLAY INIT COLF COLB ; // word that combines all the functionality
into a single command
```

After defining these words and saving them with the `CHECKPOINT` command, simply typing `PLAY` will start the loop for the LEDs.

4.3.1. *Optional Code.* The code below shows how to run the loop with both color LEDs connected. This code loops through the 1a LEDs forward and comes backward with the 1b LEDs, then goes forward with the 1b LEDs and comes back with 1a LEDs. This is to simply demonstrates that all the commands are active and working.

```
: INIT 0 TRISB 1 + C! 0 TRISA ! 5 [: DUP 8 SWAP - TRISB
SWAP BITCLR 1 - ;] DOWHILE ; // word that sets the needed ports
to digital output.
: D 15 delay ; // word that sets the delay for the LEDs in
milliseconds. The number of milliseconds used can be changed.
: BLINKR PORTB SWAP BITSET D PORTB SWAP BITCLR ; // word that
turns the rows on and off
: OFFCA 5 [: DUP 8 SWAP - PORTB SWAP BITSET 1 - ;] DOWHILE
; // word that turns all the 1a columns off
: OFFCB 5 [: DUP 5 SWAP - TRISA SWAP BITSET 1 - ;] DOWHILE
; // word that turns all the 1b columns off
: ROWF 7 [: DUP 15 SWAP - DUP BLINKR 1 - ;] DOWHILE ; // word
that turn each row off and on one after the other
: ROWB 7 [: DUP 7 + DUP BLINKR 1 - ;] DOWHILE ; // word to
do the same as ROWF but backward
: COLFA 5 [: DUP OFFCA OFFCB 8 SWAP - PORTB SWAP BITCLR ROWF
1 - ;] DOWHILE ; // word that turns on a single column of 1a
at a time loops through all 1a LEDs in that row
: COLBA 5 [: DUP OFFCA OFFCB 2 + PORTB SWAP BITCLR ROWB 1
- ;] DOWHILE ; // word to do the same as COLFA but backward
: COLFB 5 [: DUP OFFCA OFFCB 5 SWAP - TRISA SWAP BITCLR ROWF
1 - ;] DOWHILE ; // word that turns on a single column of 1b
at a time loops through all 1b LEDs in that row
: COLBB 5 [: DUP OFFCA OFFCB 1 - TRISA SWAP BITCLR ROWB 1
- ;] DOWHILE //// word to do the same as COLFB but backward
: PLAY INIT COLFA COLBB COLFB COLBA ; // word that combines
all the functionality into a single command
```

5. Results

The results of this project exceeded expectations. The PIC could not only loop through the dot-matrix display, but with a little more manipulation it could also create different patterns using the LEDs. Although the wiring was a bit tedious, it was extremely helpful in understanding the inner workings of hardware component. The software side was fairly straightforward and showed that software can indeed be considered equivalent to hardware. The only drawback was dealing with a beta version of the FORTH language that sometimes caused crashes. Once understood, the setup was fairly simple to manipulate and create new and interesting patterns.

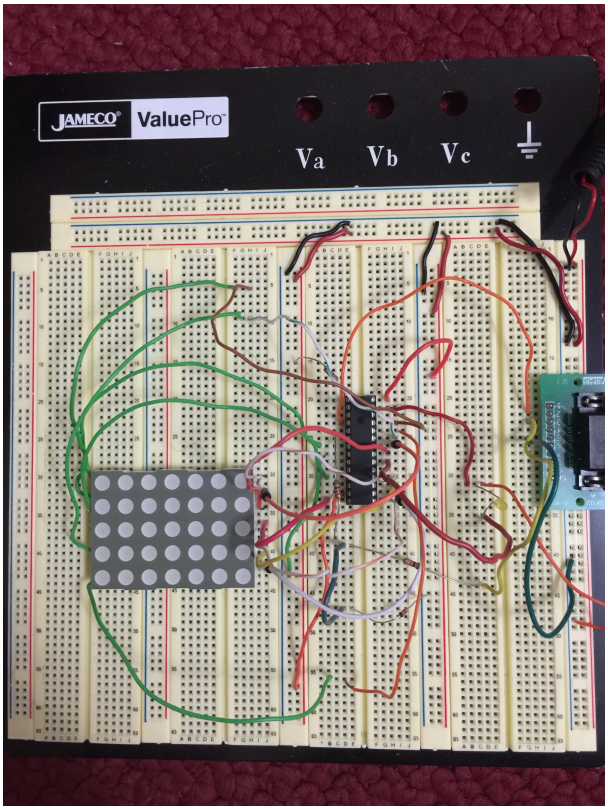


Figure 5. Finished Image of the Breadboard

6. Conclusion

The major part of this project was focused on replacing hardware with software and being able to perform operations using both components. This project provided a thorough understanding of the inner workings of the microarchitecture level of computers in general. It showed how simple commands can be used to perform various functions of the micro-controller. The project also provided an understanding as to how the microarchitecture level can also get bloated quickly if everything was done by software instead of a mixture of software and hardware. Some of the other highlights of this project included understanding memory and stack operations, using and understanding postfix notation and defining new commands that could perform complex functions by using a chain simpler commands. The best part of this project was the flexibility and room to improvise it allowed. Because most of the work was done by software, it allowed for many different ways of accomplishing the same task instead of being limited by hardware. Overall, this project provided a deep understanding of the microarchitecture level design and implementation, as well as micro-controlers and ISA languages in general.