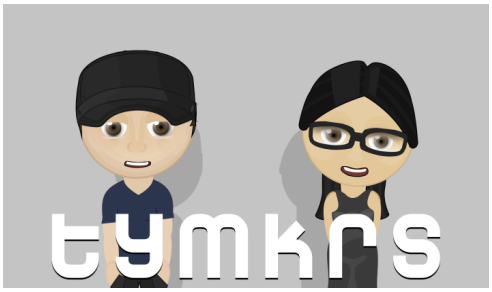
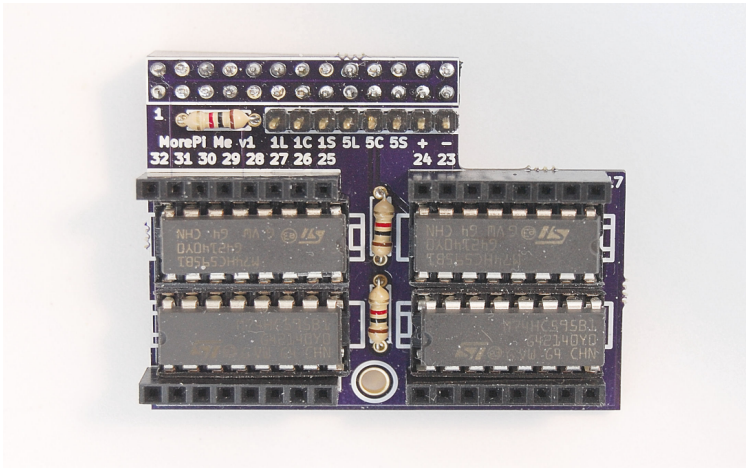


The Toymakers @ tymkrs.com
Questions? Please contact us:
feedback@tymkrs.com

DATASHEET



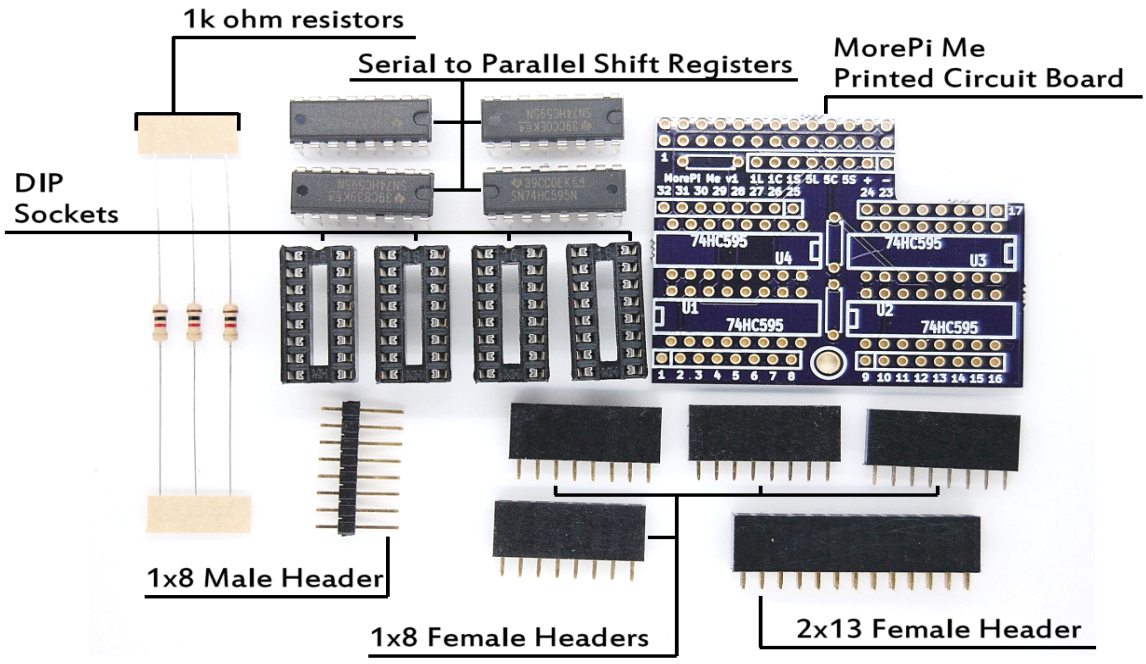
More Pi Me

Raspberry Pi 32-Output Shield

The More Pi Me kit is a shield for the Raspberry Pi that provides 32 5V software-controllable outputs.

- Kit Type: Through-hole soldering
- Assembly instructions: In datasheet
- Function: Raspberry Pi shield providing 32 – 5V software-controllable outputs

KIT CONTENTS



Contents of the MorePi Me Kit:

- MorePi Me printed circuit board (50.90 x 40.74 x 1.60mm)
- 4 1x8 Female Headers
- 1 1x8 Male Header
- 1 2x13 Female Header
- 4 DIP Sockets
- Electrical components

Electrical Components:

Reference	Quantity	Type	Value
Resistor slots	3	Resistor, 1/4W	1k ohm
U1 - U4	4	Serial to Parallel Shift Register	74HC595

Absolute Maximum Ratings

75HC595 Shift Register Maximal Operating Conditions
Datasheet: http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf

Parameter	Maximal Ratings	Unit
Supply Voltage	-0.5 – +7.0	V
Operating Temperature	-40 to +125	ºC
Output Current (Qn)	+/- 35 per pin	mA
Supply Current	70	mA

Note: Absolute maximum ratings are stress ratings only and functional device operation is not implied. The device could be damaged beyond Absolute maximum ratings.

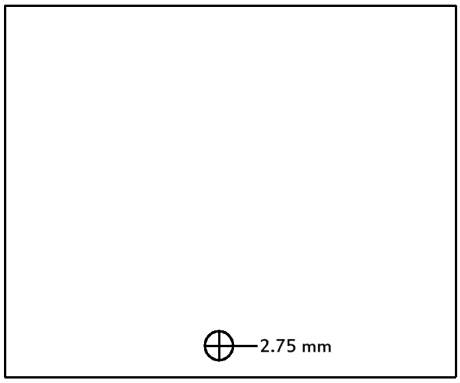
Tools and material required for assembly (not included with the kit):

- Soldering iron
- Solder
- Wire clippers

User provided items required for function:

- Raspberry Pi

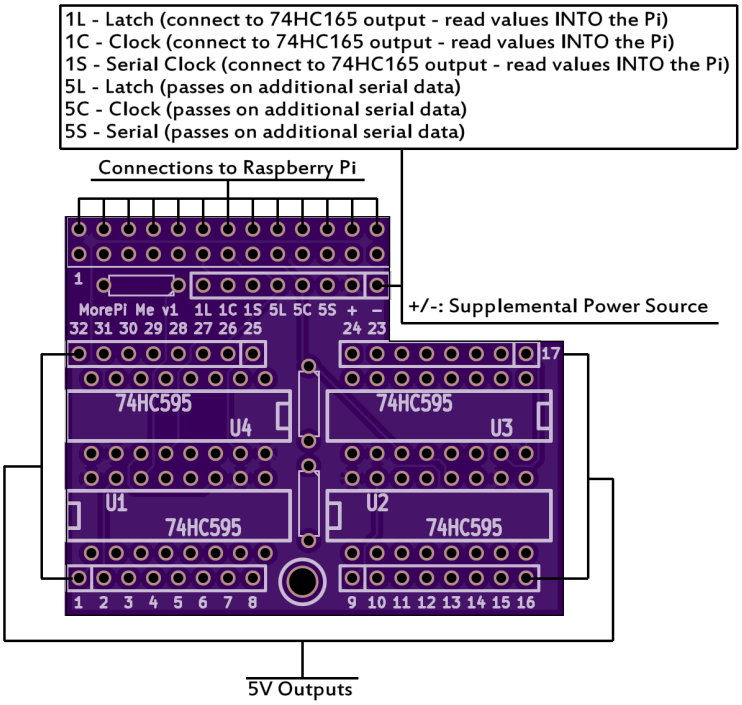
Mounting Holes:



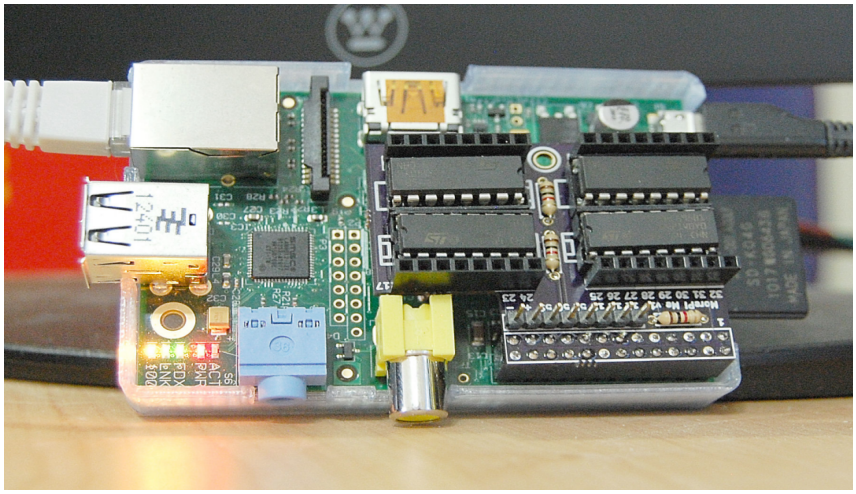
Additional physical/electrical specifications:

- Printed Circuit Board size: 2.00 x 1.60 x 0.063" (50.90 x 40.74 x 1.60mm)
- PCB thickness: 0.063" (1.60mm), not including any components
- PCB thickness: 0.472" (12mm), max height with all components
- Mounting holes: 1 holes provided. However, when it's placed on the Raspberry Pi, it will most likely not require anything else to keep it in place.

Board Connections:



Additional Photo:



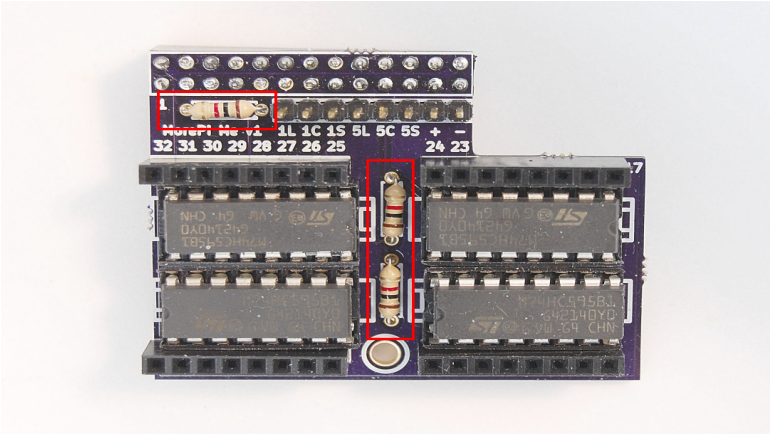
Assembled kit placed on the Raspberry Pi

Assembly Instructions

Build Notes:

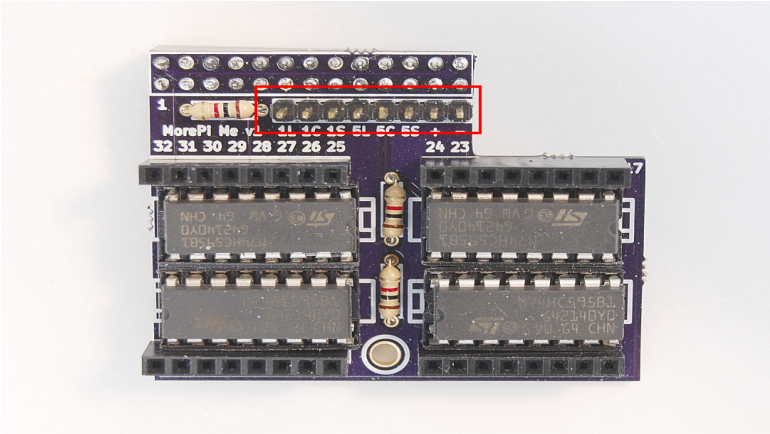
Note, the following instructions can be done in pretty much any order. I personally place all of the components on before soldering, but you are welcome to put in a component, solder it, then repeat with the rest of the components.

Step 1: Put in the components!



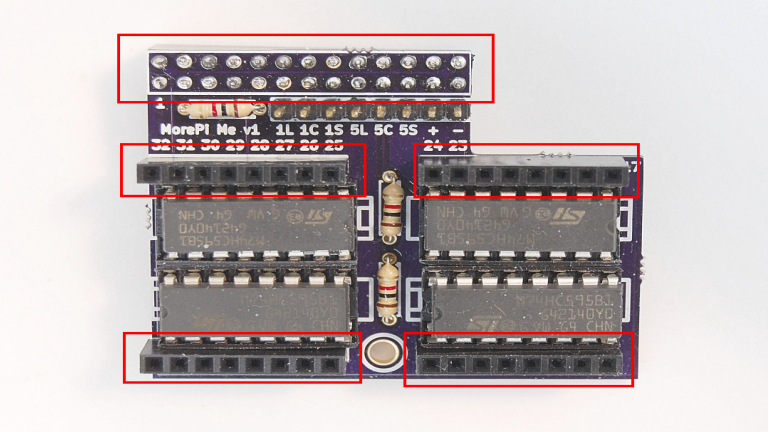
1k ohm Resistors

These 3 resistors go into the highlighted slots – polarity does not matter. (Brown-Black-Black-Brown-Brown)



1x8 Male Header

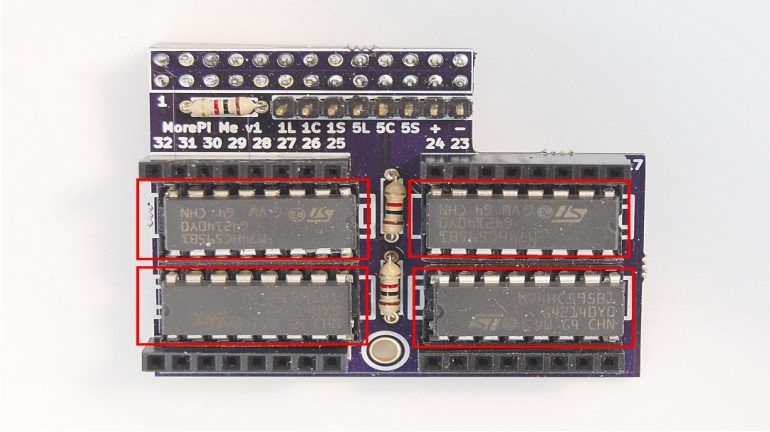
Place this header in the highlighted spot. The longer legs face upwards and can connect to our other shift register modules.



Female Headers

Solder in the female headers. The single row ones face upwards. These are the 5V outputs.

The 2x13 female header faces downwards. This is the header that connects to the Raspberry pi.



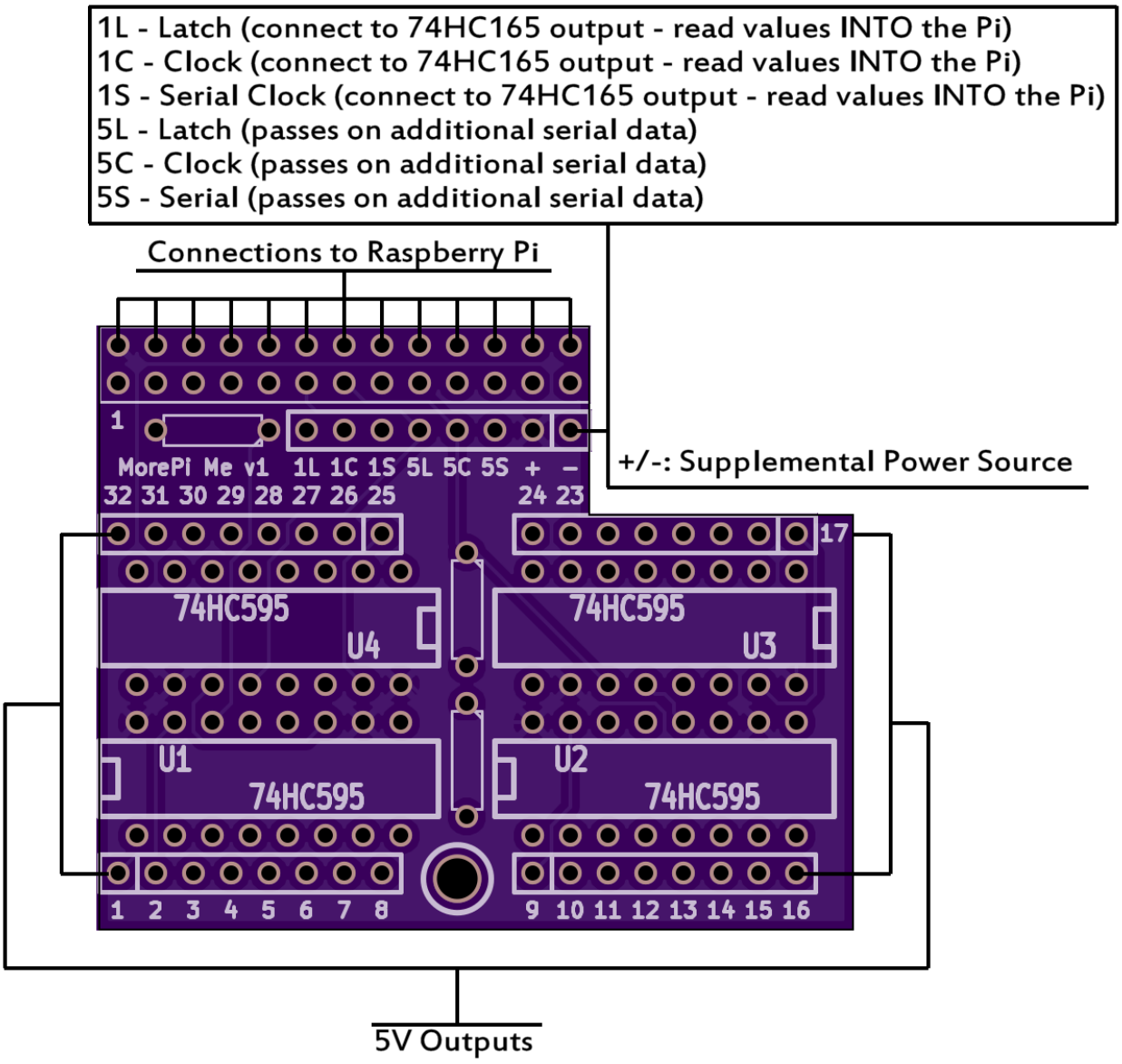
DIP Sockets and Shift Registers

Solder in the DIP sockets if you'd like (though these are optional). And then place the shift registers making sure to match the divot with the graphic on the board.

Step 2: Solder the electrical components in!

I use 60/40 0.38mm gauge solder for these pads. But also have 1.3mm gauge solder for the larger solder pads.

Use Notes



- The main function of this board is to provide 32 5V outputs to drive 32 items of your choice. A location for a supplemental power source is provided (+/-) for higher current needs (especially since each 74HC595 pin can supply 35mA with maximum per shift register of 70mA)
- 1L, 1C, and 1S – We added the additional ability for the board to take inputs from the output of our boards with 74HC165s. Your code would need to do something with that data if you wanted it to!
- 5L, 5C, and 5S – If you wanted to hook up additional modules after the MorePi Me, you could connect them to the kit via 5L, 5C, and 5S.

Various Write-ups

- **http://dangerousprototypes.com/2014/03/27/tymkrs-morepi-me-v1-pi-shift-register-board/**
- **Chasxmd's review:** http://iradan.com/?p=635

Example Code

Note: This code must be run with sudo

```
# MorePi Me v1 Demo by @tymkrs


def update():
        for index in range(0, 32):
                if state[32 - index]:
                        GPIO.output(DataPin, GPIO.HIGH)
                else:
                        GPIO.output(DataPin, GPIO.LOW)
                GPIO.output(ClockPin, GPIO.HIGH)
                GPIO.output(ClockPin, GPIO.LOW)
        GPIO.output(LatchPin, GPIO.HIGH)
        GPIO.output(LatchPin, GPIO.LOW)

def clear():
        for index in range(0, 32):
                state[index] = False

state = [False for index in range(33)]

LatchPin = 11
ClockPin = 13
DataPin = 15

import time
import RPi.GPIO as GPIO

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(LatchPin, GPIO.OUT) #latch
GPIO.setup(ClockPin, GPIO.OUT) #clock
GPIO.setup(DataPin, GPIO.OUT) #data

while True:
        for n in range(0, 8):
                clear()
                state[n + 1] = True
                update()
                time.sleep(.03)
        for n in range(0, 8):
                clear()
                state[8 - n] = True
                update()
                time.sleep(.03)
```