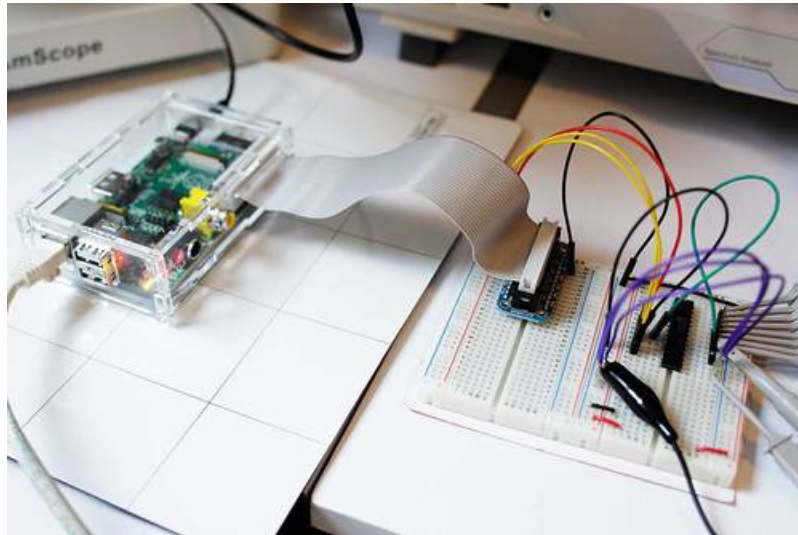


MCP230xx GPIO Expander on the Raspberry Pi

Created by Kevin Townsend



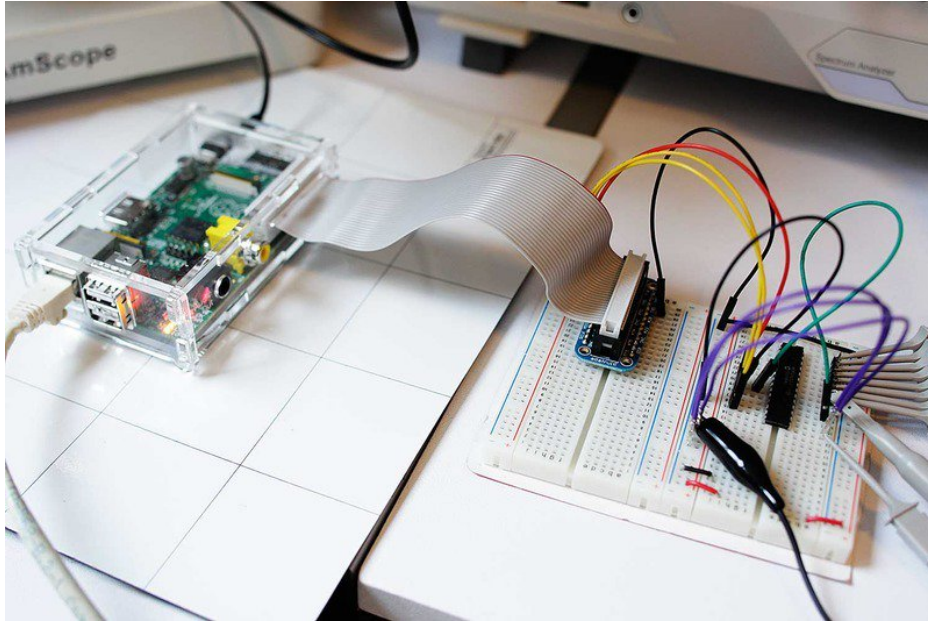
Last updated on 2018-10-16 09:35:37 PM UTC

Guide Contents

Guide Contents	2
Overview	3
What You'll Need	3
Hooking it all up	4
Using the library	6
Instantiating an instance of Adafruit_MCP230xx	7
Pin Numbering	7
Setting a pin as Input	7
Setting a pin as Output	8
Interrupts & Callbacks	8

Overview

The examples in this guide are no longer supported. Check out the MCP23xx guide for CircuitPython and Python usage: <https://learn.adafruit.com/using-mcp23008-mcp23017-with-circuitpython/overview>



While the Raspberry Pi packs and awful lot of punch for the price, and it's fairly flexible where HW expandability is concerned, there are situations where you might want a bit more basic digital IO. Thankfully, it's an easy problem to solve with an I2C-enabled device like the MCP23008 (for an extra 8 GPIO pins) or the MCP23017 (for an extra 16 GPIO pins). This tutorial will show you how you can get up and running quickly with either of these chips.

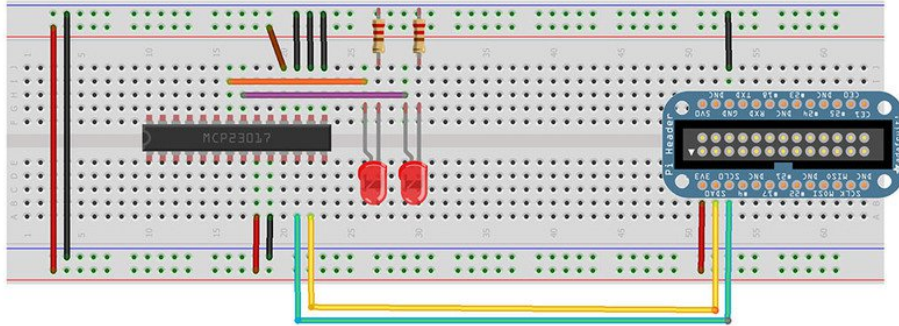
What You'll Need

- A [Raspberry Pi \(http://adafru.it/998\)](http://adafru.it/998) Model B
- A [Pi Cobbler Breakout \(http://adafru.it/914\)](http://adafru.it/914)
- An [MCP23017 \(http://adafru.it/732\)](http://adafru.it/732) or [MCP23008 \(http://adafru.it/593\)](http://adafru.it/593)
- And LED and a resistor to test with if you don't have a DMM or an oscilloscope
- If you're **not** using [Occidentalis \(https://adafru.it/aQx\)](https://adafru.it/aQx), Adafruit's own Raspberry Pi distro, you'll also need to [make sure your Pi is configured for I2C \(https://adafru.it/aTI\)](https://adafru.it/aTI) before running through this tutorial. (If you're using Occidentalis, I2C is already enabled, though, and you're ready to go!)

If you're not using Occidentalis, Adafruit's own Raspberry Pi distro, you'll also need to make sure your Pi is configured for I2C before running through our tutorial at <http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c> (If you're using Occidentalis, I2C is already enabled, though, and you're ready to go!)

Hooking it all up

The examples in this guide are no longer supported. Check out the MCP23xx guide for CircuitPython and Python usage: <https://learn.adafruit.com/using-mcp23008-mcp23017-with-circuitpython/overview>



The way that you hook the chip up to your breadboard will depend on the package you use (8-pin MCP23008 or 16-pin MCP23017). The pinouts are quite different between the two chips, so check the datasheet carefully first.

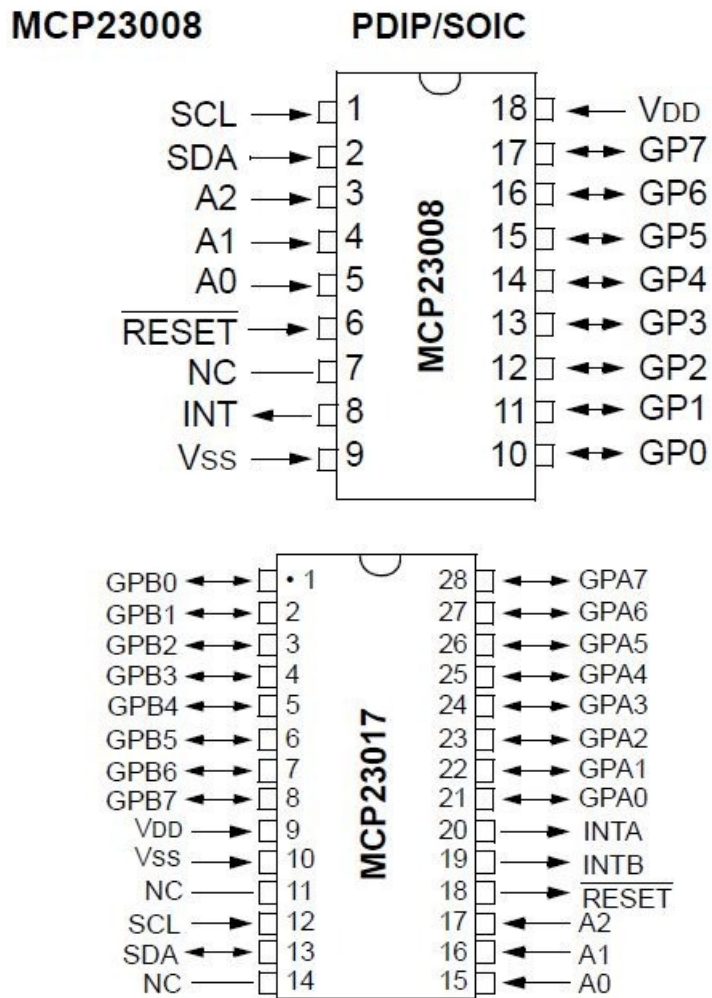
The MCP23017 is shown above with two LEDs connected, on GPA0 and GPA1.

1. The yellow line is SDA
2. The green line is SCL
3. The three black lines on top are the address pins
4. The brown pin is RESET which must be pulled high for normal operation
5. Red is 3.3V
6. Black is GND.

Since these io expander chips use i2c to communicate, you *can* theoretically power them from 5V while still connecting the i2c data lines to a 3.3V device like the pi. That's because the Pi has two i2c resistors that pull up SDA/SCL to 3.3V. Just make sure not to connect any resistors to SDA/SCL to 5V and you can power the chip from 5V (and have 5V input/output on the MCP chip). Its also fine of course to power the MCP chip from 3.3V but the 5V line on the Pi has more current capability so you might find its better to go that way.

BUT if your Pi power supply drifts a little higher than 5V, it might stop being able to register the 3.3V signal. So we recommend starting with 3.3V, and if you need 5V GPIO signalling on the MCP expander, try swapping the red wire to 5.0V

You can compare the two pinouts below to figure out how the 8-pin package should be hooked up depending on the pin names:



You're free to hook anything you want up to the 8 or 16 GPIO pins, but LEDs are used here since most people have one or two laying around and it's an easy way to verify the pin outputs. Be sure to connect a resistor in series to GND, though, to prevent the LED from burning out (if you don't know what value or the details of your LED try something large like 1K to start with).

Here's a quick video of the setup I was using during testing and development. An MCP23017 is used here, running out to a mixed-signal oscilloscope with an 8-channel logic analyzer (ergo the white clip-ons on all the GPIO pins).

Using the library

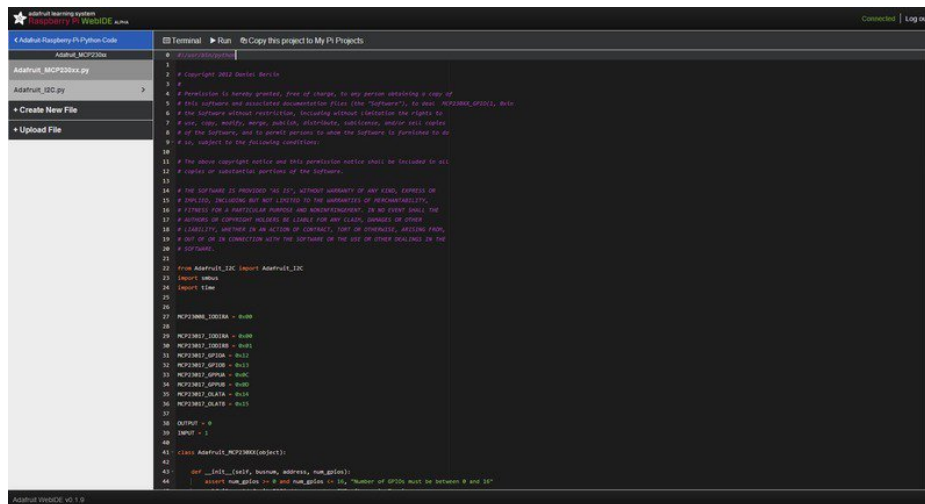
The examples in this guide are no longer supported. Check out the MCP23xx guide for CircuitPython and Python usage: <https://learn.adafruit.com/using-mcp23008-mcp23017-with-circuitpython/overview>

Never one to leave you with just a breakout board or an IC and a goodbye, Adafruit provides a library for the MCP23008 and MCP23017 in our [Pi repository on github \(https://adafru.it/r6A\)](https://adafru.it/r6A). The easiest way to use it is with our convenient [WebIDE \(https://adafru.it/aRn\)](https://adafru.it/aRn), which will automatically point to the Adafruit github repository.

Once you've opened up the WebIDE in the browser, you simply need to click in the left-hand navigation on the following folders and filenames:

- Adafruit-Raspberry-Pi-Python-Code
- Adafruit_MCP230xx
- Adafruit_MCP230xx.py

This should give you something similar to the following:



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

# Use busnum = 0 for older Raspberry Pi's (256MB)
mcp = Adafruit_MCP230XX(busnum = 0, address = 0x20, num_gpios = 16)
# Use busnum = 1 for new Raspberry Pi's (512MB with mounting holes)
# mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

# Set pins 0, 1 and 2 to output (you can set pins 0..15 this way)
mcp.config(0, OUTPUT)
mcp.config(1, OUTPUT)
mcp.config(2, OUTPUT)

# Set pin 3 to input with the pullup resistor enabled
mcp.pullup(3, 1)
# Read pin 3 and display the results
print "%d: %x" % (3, mcp.input(3) >> 3)

# Python speed test on output 0 toggling at max speed
while (True):
    mcp.output(0, 1) # Pin 0 High
    mcp.output(0, 0) # Pin 1 Low

```

This file contains both the base MCP230xx class that makes it easy to use the chip, along with a very simple demo that will toggle a single pin as fast as possible. The example code shows how you can set pins to both input and output:

Instantiating an instance of Adafruit_MCP230xx

To instantiate an instance of the wrapper class that allows you to access the MCP230xx, you need to uncomment one of the two lines at the top of the above code. There are two options because earlier versions of the Pi Model B (pre 512MB SDRAM) used I2C0, whereas the latest Model B devices (with 512MB SDRAM) use I2C1.

The address assumes you are using an MCP23017 with all three address pins set to GND. If you are using a different address pin configuration, you can open up the datasheet to see how the address scheme works ([MCP23017 datasheet \(https://adafru.it/aRo\)](https://adafru.it/aRo) or the [MCP23008 datasheet \(https://adafru.it/aRp\)](https://adafru.it/aRp).)

```

# Use busnum = 0 for older Raspberry Pi's (pre 512MB)
mcp = Adafruit_MCP230XX(busnum = 0, address = 0x20, num_gpios = 16)

# Use busnum = 1 for new Raspberry Pi's (512MB)
# mcp = Adafruit_MCP230XX(busnum = 1, address = 0x20, num_gpios = 16)

```

Pin Numbering

The MCP23008 has 8 pins - A0 thru A7. **A0** is called **0** in the library, and **A7** is called **7** (the rest follow the same pattern)

The MCP23017 has 16 pins - A0 thru A7 + B0 thru B7. **A0** is called **0** in the library, and **A7** is called **7**, then **B0** continues from there as is called **8** and finally **B7** is pin **15**

Setting a pin as Input

You can enable or disable the internal pullup resistor and set the pins as input with the following lines of code:

```
# Set pin 3 to input with the pullup resistor enabled
mcp.pullup(3, 1)

# Read pin 3 and display the results
print "%d: %x" % (3, mcp.input(3) >> 3)
```

The second line reads pin 3, and shifts the value left 3 bits so that it will equal 0 or 1 depending on whether the pin is high or low when it is sampled. This will result in output similar to the following: "3: 0" or "3: 1" (depending on the pin state).

Setting a pin as Output

To set a pin as output, you also need two lines of code:

```
# Set pin 0 to output (you can set pins 0..15 this way)
mcp.config(0, OUTPUT)

# Set pin 0 High
mcp.output(0, 1)

# Set pin 0 Low
mcp.output(0, 0)
```

That's all there is to it! The default sample code will toggle the GPIO pin as fast as possible, and if you hooked it up to an oscilloscope you'd end up with something similar to the following:

Interrupts & Callbacks

As it currently stands, the library does not support any sort of interrupt or call back functionality (there is a hardware interrupt pin on the MCP but we don't use it in this code). Only polling is currently supported!