**Teach-In 2014 with Raspberry Pi: Part 7**

by Mike and Richard Tooley

These are text files of the source code listings printed in EPE.

They appear in the same order as in the articles.

Separate listings are split by four empty lines.

Sn = 123

# Print the value of n using the default base 10

print(n)

123

n = 123

# Print the value of n in binary

print(bin(n))

0b1111011

n = 123

# Print the value of n in hexadecimal

print(hex(n))

0x7b

n = 123

# Print the value of n in octal

print(oct(n))

0o173

# Python 3.3

value\_input = input("Value to convert: ")

n = int(value\_input)

# Print eight bit binary number without 0b

# and pad with leading zeros if less than 8-bits

print('Binary = {:08b}'.format(n))

# Print three digit octal value without 0o

# and pad with leading zeros if less than 3-digits

print('Octal = {:03o}'.format(n))

# Print four digit hexadecimal without 0x

# and pad with leading zeros if less than 4-characters

print('Hex = {:04x}'.format(n))

sudo i2c detect –y 0

sudo i2c detect –y 1

import smbus

bus = smbus.SMBus(0)

bus = smbus.SMBus(1)

DEVICE = 0x20 # Device address when A0-A2 are reset

GPIOA = 0x12 # Port A data

GPIOB = 0x13 # Port B data

IODIRA = 0x00 # Port A direction

IODIRB = 0x01 # Port B direction

# MCP23017 for Python 2.x

# Import the required libraries

import time

import smbus

bus = smbus.SMBus(0) # change 0 to 1 for Rev.2 boards

# Define constants to use later on

DEVICE = 0x20 # Device address when A0-A2 are reset

GPIOA = 0x12 # Port A data

GPIOB = 0x13 # Port B data

IODIRA = 0x00 # Port A direction

IODIRB = 0x01 # Port B direction

# Initialise Port A as an input

bus.write\_byte\_data(DEVICE,IODIRA,0xff)

# Initialise Port B as an output

bus.write\_byte\_data(DEVICE,IODIRB,0x00)

# MCP23017 for Python 2.x

# Import the required libraries

import time

import smbus

bus = smbus.SMBus(0) # change 0 to 1 for Rev.2 boards

# Define constants to use later on

DEVICE = 0x20 # Port expander's I2C bus address

IODIRB = 0x01 # Direction register for Port B

OLATB = 0x15 # Output latch for Port B

GPIOB = 0x13 # Data register for Port B

# Initialise Port B as an output

bus.write\_byte\_data(DEVICE,IODIRB,0x00)

# Initialise Port B with all latches in the high state

bus.write\_byte\_data(DEVICE,OLATB,0xff)

while 1:

# Send 10101010 bit pattern to port B

bus.write\_byte\_data(DEVICE,GPIOB,0xaa)

# Wait 1 second

time.sleep(0.5)

# Send 01010101 bit pattern to port B

bus.write\_byte\_data(DEVICE,GPIOB,0x55)

# Wait 1 second

time.sleep(0.5)

# MCP23017 for Python 2.x

# Import the required libraries

import time

import smbus

bus = smbus.SMBus(0) # change 0 to 1 for Rev.2 boards

# Define constants to use later on

DEVICE = 0x20 # Port expander's I2C bus address

IODIRB = 0x01 # Direction register for Port B

OLATB = 0x15 # Output latch for Port B

GPIOB = 0x13 # Data register for Port B

# Initialise Port B as an output

bus.write\_byte\_data(DEVICE,IODIRB,0x00)

# Initialise Port B with all latches in the high state

bus.write\_byte\_data(DEVICE,OLATB,0xff)

count = 0

while 1:

# Shift bit pattern right

pattern=0b00000001 # Single bit set

count = 0 # Reset the count

while count < 8:

# Send bit pattern to port B

bus.write\_byte\_data(DEVICE,GPIOB,pattern)

pattern = 2 \* pattern

# Wait 0.1 second

time.sleep(0.1)

count = count + 1

count = 0 # Reset the count

pattern = pattern / 2

# Shift bit pattern left

while count < 8:

# Send bit pattern to port B

bus.write\_byte\_data(DEVICE,GPIOB,pattern)

pattern = pattern / 2

# Wait 0.1 second

time.sleep(0.1)

count = count + 1

# MCP23017 for Python 2.x

# Import the required libraries

import time

import smbus

bus = smbus.SMBus(0) # change 0 to 1 for Rev.2 boards

# Define constants for later use

DEVICE = 0x20 # Port expander's I2C bus address

IODIRA = 0x00 # Direction register for Port A

GPIOA = 0x12 # Data register for Port A

# Initialise Port A as an input

bus.write\_byte\_data(DEVICE,IODIRA,0xff)

while 1:

# Get input state of Port A

input\_state = bus.read\_byte\_data(DEVICE,GPIOA)

print('{0:08b}'.format(input\_state))

# Wait 1 second

time.sleep(1)

# MCP23017 for Python 2.x

# Import the required libraries

import time

import smbus

bus = smbus.SMBus(0) # change 0 to 1 for Rev.2 boards

# Define constants to use later on

DEVICE = 0x20 # Port expander's I2C bus address

IODIRA = 0x00 # Direction register for Port A

GPIOA = 0x12 # Data register for Port A

IODIRB = 0x01 # Direction register for Port B

GPIOB = 0x13 # Data register for Port B

OLATB = 0x15 # Output latch for Port B

# Initialise Port A as an input

bus.write\_byte\_data(DEVICE,IODIRA,0xff)

# Initialise Port B as an output

bus.write\_byte\_data(DEVICE,IODIRB,0x00)

# Initialise Port B with all latches in the high state

bus.write\_byte\_data(DEVICE,OLATB,0xff)

while 1:

# Get switch pattern from Port A

pattern = bus.read\_byte\_data(DEVICE,GPIOA)

# Write switch pattern to Port B

bus.write\_byte\_data(DEVICE,GPIOB,pattern)

# Wait 0.5 second

time.sleep(0.5)

sudo apt-get update

sudo apt-get install apache2

ip addr show

sudo apt-get install php5

<?php

phpinfo()

?>