```python
#!/usr/bin/python

import time
import math
from Raspi_I2C import Raspi_I2C

# ============================================================================
# Raspi PCA9685 16-Channel PWM Servo Driver
# ============================================================================

class PWM :
  # Registers/etc.
  __MODE1              = 0x00
  __MODE2              = 0x01
  __SUBADR1            = 0x02
  __SUBADR2            = 0x03
  __SUBADR3            = 0x04
  __PRESCALE           = 0xFE
  __LED0_ON_L          = 0x06
  __LED0_ON_H          = 0x07
  __LED0_OFF_L         = 0x08
  __LED0_OFF_H         = 0x09
  __ALL_LED_ON_L       = 0xFA
  __ALL_LED_ON_H       = 0xFB
  __ALL_LED_OFF_L      = 0xFC
  __ALL_LED_OFF_H      = 0xFD

  # Bits
  __RESTART            = 0x80
  __SLEEP              = 0x10
  __ALLCALL            = 0x01
  __INVRT              = 0x10
  __OUTDRV             = 0x04

  general_call_i2c = Raspi_I2C(0x00)

  @classmethod
  def softwareReset(cls):
    "Sends a software reset (SWRST) command to all the servo drivers on the bus"
    cls.general_call_i2c.writeRaw8(0x06)        # SWRST

  def __init__(self, address=0x40, debug=False):
    self.i2c = Raspi_I2C(address)
    self.i2c.debug = debug
    self.address = address
    self.debug = debug
    if (self.debug):
      print("Reseting PCA9685 MODE1 (without SLEEP) and MODE2")
    self.setAllPWM(0, 0)
    self.i2c.write8(self.__MODE2, self.__OUTDRV)
    self.i2c.write8(self.__MODE1, self.__ALLCALL)
    time.sleep(0.005)                                       # wait for oscillator

    mode1 = self.i2c.readU8(self.__MODE1)
    mode1 = mode1 & ~self.__SLEEP                 # wake up (reset sleep)
    self.i2c.write8(self.__MODE1, mode1)
    time.sleep(0.005)                             # wait for oscillator

  def setPWMFreq(self, freq):
    "Sets the PWM frequency"
    prescaleval = 25000000.0    # 25MHz
    prescaleval /= 4096.0       # 12-bit
    prescaleval /= float(freq)
    prescaleval -= 1.0
    if (self.debug):
      print("Setting PWM frequency to %d Hz" % freq)
      print("Estimated pre-scale: %d" % prescaleval)
```

```
68        prescale = math.floor(prescaleval + 0.5)
69        if (self.debug):
70          print("Final pre-scale: %d" % prescale)
71
72        oldmode = self.i2c.readU8(self.__MODE1);
73        newmode = (oldmode & 0x7F) | 0x10             # sleep
74        self.i2c.write8(self.__MODE1, newmode)        # go to sleep
75        self.i2c.write8(self.__PRESCALE, int(math.floor(prescale)))
76        self.i2c.write8(self.__MODE1, oldmode)
77        time.sleep(0.005)
78        self.i2c.write8(self.__MODE1, oldmode | 0x80)
79
80      def setPWM(self, channel, on, off):
81        "Sets a single PWM channel"
82        self.i2c.write8(self.__LED0_ON_L+4*channel, on & 0xFF)
83        self.i2c.write8(self.__LED0_ON_H+4*channel, on >> 8)
84        self.i2c.write8(self.__LED0_OFF_L+4*channel, off & 0xFF)
85        self.i2c.write8(self.__LED0_OFF_H+4*channel, off >> 8)
86
87      def setAllPWM(self, on, off):
88        "Sets a all PWM channels"
89        self.i2c.write8(self.__ALL_LED_ON_L, on & 0xFF)
90        self.i2c.write8(self.__ALL_LED_ON_H, on >> 8)
91        self.i2c.write8(self.__ALL_LED_OFF_L, off & 0xFF)
92        self.i2c.write8(self.__ALL_LED_OFF_H, off >> 8)
93
```