```
# -*- coding: utf-8 -*-
"""
Spyder Editor

@author Jason Grillo, Thomas Goehring, Trent Peterson
"""

import micropython
import ustruct

# BNO055 Registers
# - Referenced registers from Radomir Dopieralski's Circuit Python module

_CHIP_ID = micropython.const(0xa0)

CONFIG_MODE = micropython.const(0x00)
ACCONLY_MODE = micropython.const(0x01)
MAGONLY_MODE = micropython.const(0x02)
GYRONLY_MODE = micropython.const(0x03)
ACCMAG_MODE = micropython.const(0x04)
ACCGYRO_MODE = micropython.const(0x05)
MAGGYRO_MODE = micropython.const(0x06)
AMG_MODE = micropython.const(0x07)
IMU_MODE = micropython.const(0x08)
COMPASS_MODE = micropython.const(0x09)
M4G_MODE = micropython.const(0x0a)
NDOF_FMC_OFF_MODE = micropython.const(0x0b)
NDOF_MODE = micropython.const(0x0c)

_POWER_NORMAL = micropython.const(0x00)
_POWER_LOW = micropython.const(0x01)
_POWER_SUSPEND = micropython.const(0x02)

_MODE_REGISTER = micropython.const(0x3d)
_PAGE_REGISTER = micropython.const(0x07)
_TRIGGER_REGISTER = micropython.const(0x3f)
_POWER_REGISTER = micropython.const(0x3e)
_ID_REGISTER = micropython.const(0x00)

class bno055:
    """ This class implements a simple driver for the BNO055 Adafruit
    IMU. This IMU talk to the CPU over I<sup>2</sup>C.

#    An example of how to use this driver:
#    @code
#    imu = BNO055.bno055 (pyb.I2C (1, pyb.I2C.MASTER, baudrate = 100000),
```

```
#    imu.sys_status ()
#    imu.sys_error ()
#    imu.get_euler_pitch ()     # or other data...
#    @endcode
    The example code works for a BNO055 on a Adafruit<sup>TM</sup> breako
    board. """

    def __init__ (self, i2c, address):
        """ Initialize a BNO055 driver on the given I<sup>2</sup>C bus.
        @param i2c An I<sup>2</sup>C bus already set up in MicroPython
        @param address The address of the IMU on the I<sup>2</sup>C bus
        """
        self._address = address
        self._i2c = i2c
        #Select NDOF mode, @IMU Hard address, Set NDOF_Mode to MODE_REGIS
        self._i2c.mem_write(IMU_MODE, self._address, _MODE_REGISTER)
        self._i2c.mem_write(_POWER_NORMAL, self._address, _POWER_REGISTER
        # Define calibration values... init as zero to have no effect
        self._zeroes = [0,0,0]

    def get_euler_pitch(self):
        """ Get the absolute euler pitch of the IMU. (_zeroes[0])
        @return _pitch_value The calibrated absolute pitch of the IMU
        """
        #Read 2 Pitch start at pitch lsb
        self._pitch = self._i2c.mem_read(2, self._address, 0x1E)
        #Unpact struct to get pitch value
        self._pitch_decode = ustruct.unpack('<h',self._pitch)
        self._pitch_value = float(self._pitch_decode[0]/16)
        return (self._pitch_value - self._zeroes[0])

    def get_euler_roll(self):
        """ Get the absolute euler roll of the IMU. (_zeroes[1])
        @return _roll_value The calibrated absolute roll of the IMU
        """
        self._roll = self._i2c.mem_read(2, self._address, 0x1C)
        self._roll_decode = ustruct.unpack('<h',self._roll)
        self._roll_value = float(self._roll_decode[0]/16)
        return (self._roll_value - self._zeroes[1])

    def get_euler_yaw(self):
        """ Get the absolute euler yaw of the IMU. (_zeroes[2])
        @return _raw_value The calibrated absolute yaw of the IMU
        """
        self._yaw = self._i2c.mem_read(2, self._address, 0x1A)
        self._yaw_decode = ustruct.unpack('<h',self._yaw)
```

```
        self._yaw_value = float(self._yaw_decode[0]/16)
        return (self._yaw_value - self._zeroes[2])

    def sys_status(self):
        """ Get the IMU status to see if it is running or if there are er
        @return _status_value The absolute pitch of the IMU
        """
        self._status = self._i2c.mem_read(1,self._address,0x39)
        self._status_decode  = ustruct.unpack('b',self._status)
        self._status_value = int(self._status_decode[0])
        return self._status_value

    def sys_error(self):
        """ Obtain the error, if the IMU is not returning values.
        @return _error_value The
        """
        self._error = self._i2c.mem_read(1,self._address,0x3A)
        self._error_decode = ustruct.unpack('b',self._error)
        self._error_value = int(self._error_decode[0])
        return self._error_value

    def zero_Euler_vals(self):
        """ Zero the IMU for calibration purposes.
        @return _zeroes The calibration list for Euler angle outputs
        """
        self._zeroes[0] = self.get_euler_pitch()
        self._zeroes[1] = self.get_euler_roll()
        self._zeroes[2] = self.get_euler_yaw()
        return self._zeroes
```