```
"""
Waveshare AD/DA board, www.waveshare.com/high-precision-ad-da-board.htm
AD-converter ADS1256 (ti.com), first test with on-board trimpot
DA-converter DAC8534 (ti.com), not used
DDS-generator AD9952 (analog.com)
"""
import time
import pigpio

pi= pigpio.pi()

# set GPIO data direction          ("pin*" = 40pin only):
# pi.set_mode(0, pigpio.INPUT)     # pin*27 (Raspi-hat ID_SD)
# pi.set_mode(1, pigpio.INPUT)     # pin*28 (Raspi-hat ID_SC)
# pi.set_mode(2, pigpio.INPUT)     # pin  3 (I2C SDA, fixed 1k8 pull-up)
# pi.set_mode(3, pigpio.INPUT)     # pin  5 (I2C SCL, fixed 1k8 pull-up)
# pi.set_mode(4, pigpio.INPUT)     # pin  7 (free)
# pi.set_mode(5, pigpio.OUTPUT)      # pin*29 (free)
# pi.set_mode(6, pigpio.OUTPUT)      # pin*31 (free)
# pi.set_mode(7, pigpio.INPUT)     # pin 26 (main SPI CE1)
pi.set_mode(8, pigpio.OUTPUT)      # pin 24 AD9952 /CS (main SPI CE0)
pi.set_mode(9, pigpio.INPUT)       # pin 21 ADS1256/AD9952 DOUT
pi.set_mode(10, pigpio.OUTPUT)     # pin 19 ADS1256/AD9952 DIN
pi.set_mode(11, pigpio.OUTPUT)     # pin 23 ADS1256/AD9952 SCLK
pi.set_mode(12, pigpio.OUTPUT)     # pin*32 (free)
# pi.set_mode(13, pigpio.INPUT)    # pin*33 (free)
# pi.set_mode(14, pigpio.OUTPUT)   # pin  8 Serial TXD
# pi.set_mode(15, pigpio.INPUT)    # pin 10 Serial RXD
# pi.set_mode(16, pigpio.INPUT)    # pin*36 (aux SPI ce2)
pi.set_mode(17, pigpio.INPUT)      # pin 11 ADS1256 /DRDY (aux SPI ce1)
pi.set_mode(18, pigpio.OUTPUT)     # pin 12 ADS1256 /RESET (aux SPI ce0)
# pi.set_mode(19, pigpio.INPUT)    # pin*35 (aux SPI miso)
# pi.set_mode(20, pigpio.INPUT)    # pin*38 (aux SPI mosi)
# pi.set_mode(21, pigpio.INPUT)    # pin*40 (aux SPI sclk)
pi.set_mode(22, pigpio.OUTPUT)     # pin 15 ADS1256 /CS input
pi.set_mode(23, pigpio.OUTPUT)     # pin 16 DAC8532 /CS input
pi.set_mode(24, pigpio.OUTPUT)     # pin 18 AD9952 Reset)
pi.set_mode(25, pigpio.OUTPUT)     # pin 22 AD9952 I/O-Update)
# pi.set_mode(26, pigpio.INPUT)    # pin*37 (free)
pi.set_mode(27, pigpio.OUTPUT)     # pin 13 ADS1256 /PDWN input

# set GPIO outputs to start values
pi.write(23, 1)    # DAC8532 /CS high
pi.write(22, 1)    # ADS1256 /CS high
pi.write(11, 0)    # ADS1256 SCLK low (later overridden by spi_open)
pi.write(10, 0)    # ADS1256 DIN  low (later overridden by spi_open)
pi.write(27, 1)    # ADS1256 /PDWN high
pi.write(18, 1)    # ADS1256 /RESET high
pi.write(8, 1)     # AD9952  /CS high
pi.write(24, 0)    # AD9952  RESET low
pi.write(25, 0)    # AD9952  I/O-Update low

# Delay until DRDY line goes low, allowing for automatic calibration:
#https://raw.githubusercontent.com/heathsd/PyADS1256/master/pyads1256.py
def WaitDRDY():
    start = pi.get_current_tick()
    elapsed = pi.get_current_tick() - start
    # Waits for DRDY to go to zero or TIMEOUT seconds to pass
    drdy_level = pi.read(17) # ADS1256 /DRDY
    while (drdy_level == 1) and (elapsed < 500000):
        elapsed = pi.get_current_tick() - start
        drdy_level = drdy_level = pi.read(17)
    if elapsed >= 800000:
        print("WaitDRDY() Timeout\r\n")

# reset ADC:
pi.write(18, 0)    # ADS1256 /RESET low
time.sleep(0.001)  # wait 1 msec
pi.write(18, 1)    # ADS1256 /RESET high
time.sleep(0.5)    # wait 0.5 sec
# reset DDS:
pi.write(24, 1)    # AD9952  RESET high
time.sleep(0.001)  # wait 1 msec
pi.write(24, 0)    # AD9952  RESET low
time.sleep(0.5)    # wait 0.5 sec

# open main SPI for dds:
dds = pi.spi_open(0, 200000, 0x00000060)
# set to 0x00000060 = MSB-1st, 4-wire, main-SPI, disable CE0/1, mode0)

# open main SPI for adc:
ad_da = pi.spi_open(0, 200000, 0x00000061) # 0x61 = 97
```

```
# handle = spi_open(spi_channel, baud, spi_flags)

# start DDS:
pi.write(8, 0)      # AD9952 /CS low
pi.spi_write(dds, b'\x00\x00\x00\x02\x40') # CFR1 write ContrFunctReg1 (32bit)
pi.spi_write(dds, b'\x01\x18\x00\x00')     # CFR2 write ContrFunctReg2 (24bit).
pi.spi_write(dds, b'\x02\x00\x00')         # ASF  write Amplitude Scale Factor (16 bit)
pi.spi_write(dds, b'\x03\x00')             # ARR  write Amplitude Ramp Rate (8 bit)
pi.spi_write(dds, b'\x05\x00\x00')         # POW write Phase Offset Word (16 bit)

f_measure= 1000000 # measurement frequency set to 1 MHz
frq=f_measure+1 # start 1Hz above f_measure
ftw=((int((frq*21.47483648)+0.5)).to_bytes(4, byteorder='big'))
pi.spi_write(dds, b'\x04')
pi.spi_write(dds, ftw)  # set DDS to 1000001 Hz
# pi.spi_write(dds, b'\x04\x01\x47\xae\x14') # set DDS to 1 MHz
pi.write(25, 1)     # AD9952  I/O-Update high
pi.write(25, 0)     # AD9952  I/O-Update low
pi.write(8, 1)      # AD9952  /CS high
# time.sleep(2.0)     # wait 2 sec

# start ADC:
# set ADC registers to start values (other registers default values ok):
# select ADC:
pi.write(22, 0)     # ADS1256 /CS low
WaitDRDY()
pi.spi_write(ad_da, b'\xfe') # command 0xfe: soft-reset
time.sleep(0.5)     # wait 0.5 sec
# set register 00 (STATUS) reg.00,one byte,no autocal,no buffer
WaitDRDY()
pi.write(22, 0)     # ADS1256 /CS low
pi.spi_write(ad_da, b'\xfc\x00\x50\x00\x01')
pi.write(22, 1)     # ADS1256 /CS high
time.sleep(0.0001)  # wait 0.1 msec
# set register 02 (ADCON)
WaitDRDY()
pi.write(22, 0)     # ADS1256 /CS low
pi.spi_write(ad_da, b'\xfc\x00\x52\x00\x00')
pi.write(22, 1)     # ADS1256 /CS high
time.sleep(0.0001)  # wait 0.1 msec
# pi.set register 03 (DRATE) reg.03,one byte,10 samples per second
WaitDRDY()
pi.write(22, 0)     # ADS1256 /CS low
# pi.spi_write(ad_da, b'\xfc\x00\x53\x00\x23') # 10 samples per second
pi.spi_write(ad_da, b'\xfc\x00\x53\x00\x03') # 1 samples per second
pi.write(22, 1)     # ADS1256 /CS high
time.sleep(0.0001)  # wait 0.1 msec

# calibrate ADC
WaitDRDY()
pi.write(22, 0)     # ADS1256 /CS low
pi.spi_write(ad_da, b'\xf0') # command 0xf0: self offset/gain calibration
pi.write(22, 1)     # ADS1256 /CS high
time.sleep(1.0)     # wait 1 sec

# read ADC twice, discard first value
WaitDRDY()
pi.write(22, 0)     # ADS1256 /CS low
(count, databyte) = pi.spi_xfer(ad_da, b'\x51\x00\x08\xfc\x00\x01\x00\x00\x00')
WaitDRDY()
(count, databyte) = pi.spi_xfer(ad_da, b'\x51\x00\x08\xfc\x00\x01\x00\x00\x00')
pi.write(22, 1)     # ADS1256 /CS high
data = (databyte[6]<<16 | databyte[7]<<8 | databyte[8])


.....
```