User's Guide

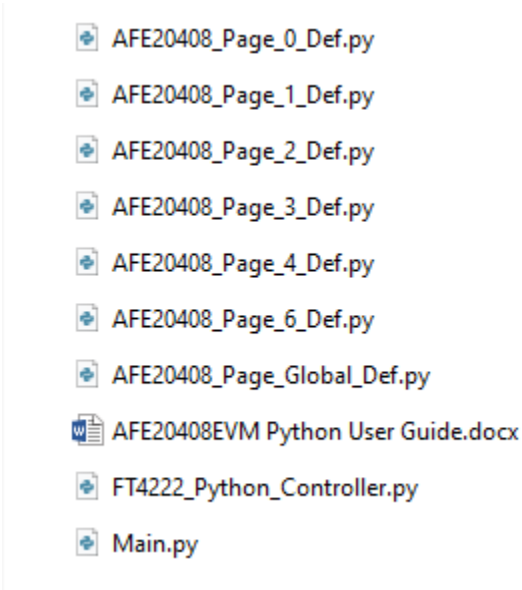
AFE20408EVM Python User's Guide

Erin Bowrie

This document describes the python code included with the AFE20408EVM software and shows usage examples.

The AFE20408EVM comes with an FT4222 module installed on the EVM. The FT4222 is a USB-to-I2C/SPI bridge that is used to communicate between the EVM GUI and the AFE20408. Scripts can be made utilizing the FT4222 to create custom commands and code.

Basic python code has been included with the EVM as a starting point for the user. This document will explain what each file does and how to utilize the starting code to create your own code.



The python folder contains nine python files. The files will be described in detail in the following sections.

1 FT4222_Python_Controller.py

This file contains the functions related to the FT4222 controller.

FtdiController()
Initializes the FTDI.

getDeviceInfo()
Gets FTDI device information.

closeHandle()
Close the FTDI handle once the program is complete. It is best practice to close the FTDI handle before reinitializing the FTDI.

1.1 I2C Commands

initializeI2C(speed)
Initializes the FTDI in I2C Mode.
Parameters:

Speed	Can be between 60k to 3400k. Note that the AFE20408 can only handle up to 400k bps

i2cWrite(Slave Address, Register Address, Register Data)
Sends an I2C write command to the EVM.
Parameters:

Slave Address	8-bit slave address. This address is configurable using EVM jumpers.
Register Address	8-bit register address
Register Data	16-bit register data

i2cRead(Slave Address, Register Address)
Sends an I2C read command to the EVM. Returns a 16-bit value.
Parameters:

Slave Address	8-bit slave address. This address is configurable using EVM jumpers.
Register Address	8-bit register address

Example:

```
ftdiObject = FtdiController()   # initialize the FTDI
ftdiObject.getFtdiDeviceInfo()  # Get FTDI info
ftdiObject.initializeI2C(400)   # initialize I2C with 400kbps

# Write to slave address 0x40, register address 0x08, with data 0x0001
ftdiObject.i2cWrite(0x40, 0x08, 0x0001)

# Read back from slave address 0x40, register address 0x08, and print the data.
print(ftdiObject.i2cRead(0x40, 0x08))

ftdiObject.closeHandle()    # close the FTDI when done.
```

1.2 SPI Commands

initializeSPI(Mode, Clock, CPOL, CPHA, Slave Select)
Initializes the FTDI in SPI Mode
Parameters:

Mode	Ft4222.SPIMaster.Mode.NONE Ft4222.SPIMaster.Mode.SINGLE Ft4222.SPIMaster.Mode.DUAL Ft4222.SPIMaster.Mode.QUAD	AFE20408EVM uses *SINGLE* mode. Do not use the other modes.
Clock	ft4222.SPIMaster.Clock.NONE ft4222.SPIMaster.Clock.DIV_2 ft4222.SPIMaster.Clock.DIV_4 ft4222.SPIMaster.Clock.DIV_8 ft4222.SPIMaster.Clock.DIV_16 ft4222.SPIMaster.Clock.DIV_32 ft4222.SPIMaster.Clock.DIV_64 ft4222.SPIMaster.Clock.DIV_128 ft4222.SPIMaster.Clock.DIV_256 ft4222.SPIMaster.Clock.DIV_512	Default clock is 60MHz. AFE20408 max clock is 20MHz.
CPOL	ft4222.SPI.Cpol.IDLE_LOW ft4222.SPI.Cpol.IDLE_HIGH	AFE20408 uses IDLE_LOW
CPHA	ft4222.SPI.Cpha.CLK_LEADING ft4222.SPI.Cpha.CLK_TRAILING	AFE20408 uses CLK_TRAILING
Slave Select	ft4222.SPIMaster.SlaveSelect.SS0 ft4222.SPIMaster.SlaveSelect.SS1 ft4222.SPIMaster.SlaveSelect.SS2 ft4222.SPIMaster.SlaveSelect.SS3	AFE20408EVM uses SS0

spiWrite(Register Address, Register Data)
Sends an SPI write command to the EVM.
Parameters:

Register Address	8-bit register address
Register Data	16-bit register data

spiRead(Register Address)
Sends an SPI read command to the EVM. Returns a 16-bit value.
Parameters:

Register Address	8-bit register address

Example:

```
    ftdiObject = FtdiController()   # initialize the FTDI
    ftdiObject.getFtdiDeviceInfo()  # Get FTDI info
    # Initialize SPI. Note the only changeable variable is the clock division. Everything else is static for the
EVM
    ftdiObject.initializeSPI(ft4222.SPIMaster.Mode.SINGLE, ft4222.SPIMaster.Clock.DIV_4,
ft4222.SPI.Cpol.IDLE_LOW, ft4222.SPI.Cpha.CLK_TRAILING, ft4222.SPIMaster.SlaveSelect.SS0)

    # Write to register address 0x08, with data 0x0001
    ftdiObject.spiWrite(0x08, 0x0001)

    # Read back from register address 0x08 and print the data.
    print(ftdiObject.spiRead(0x08))

    ftdiObject.closeHandle()    # close the FTDI when done.
```

2 AFE20408_Page_Global_Def.py

The Global page of the AFE20408 is always accessible – the PAGE register does not need to be changed to access any of the global registers. This file contains the global register addresses in a variable format. The file also contains variables for specific data combinations.
This page contains the following registers:

NOP	Software reset = 0x00AD
PAGE	PAGE 0 = GEN CONFIG PAGE 1 = ADC CONFIG PAGE 2 = ADC CCS CONFIG PAGE 3 = DAC CONFIG PAGE 4 = DAC BUFFER PAGE 6 = DAC ACTIVE
GEN_STATUS	Read only
ALARM_STATUS_0	Read only
ALARM_STATUS_1	Read only
PWR_STATUS_0	Read only
PWR_STATUS_1	Read only
PWR_EN	DAC Enable configurations
TRIGGER	ADC and DAC triggers
GPIO	
DRVEN_SW_EN	Enables Software DRVEN
DRVEN	Software DRVEN configurations
DAC_BCAST	DAC Broadcast data
GLOBAL_CFG	
ADC_SENSE0	SENSE0 readback
ADC_SENSE1	SENSE1 readback
ADC_ADC0	ADC0 readback
ADC_ADC1	ADC1 readback
ADC_TMP	Local temperature readback

Example:

```
    # Write to the DAC Power Enable address, enable all of the DACs

    # Example 1: Use the address variable, use raw data.
    ftdiObject.spiWrite(PWR_EN_ADDRESS, 0x01FF)

    # Example 2: Use the variables to set the data.
    ftdiObject.spiWrite(PWR_EN_ADDRESS, PWR_EN_PAON_En + PWR_EN_PDACB3_En +
PWR_EN_PDACB2_En + PWR_EN_PDACB1_En + PWR_EN_PDACB0_En + PWR_EN_PDACA3_En +
PWR_EN_PDACA2_En + PWR_EN_PDACA1_En + PWR_EN_PDACA0_En)
```

3 AFE20408_Page_0_Def.py

Page 0 of the AFE20408 contains basic configuration registers. The file contains variables for the registers and specific data combinations. You are required to write 0x0000 to the PAGE register before writing or reading from any register on this page.
This page contains the following registers:

CHIP_ID	Readback is 0x2480
CHIP_VER	
SDO_EN:	Required to enable SDO for SPI readback.
GEN_CFG_0	PAON and FLEXIO configurations
GEN_CFG_1	VSS Range and FLEXIO selection
ALARMOUT_SRC_0	
ALARMOUT_SRC_1	
ALARM_STATUS_0_BYP	
ALARM_STATUS_1_BYP	
PAON_SRC_0	
PAON_SRC_1	
RESET_FLAGS	

Example:

```
   #Example: Read chip ID
   ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_GEN_CONFIG)  #Set page
   ftdiObject.spiWrite(SDO_EN_ADDRESS, SDO_EN_FSDO_Dis + SDO_EN_SDOenabled)  #Enable
SDO
   print(ftdiObject.spiRead(CHIP_ID_ADDRESS))        #Read Chip ID
```

4 AFE20408_Page_1_Def.py

Page 1 of the AFE20408 contains ADC configuration registers. The file contains variables for the registers and specific data combinations. You are required to write 0x0001 to the PAGE register before writing or reading from any register on this page.
This page contains the following registers:

ADC_GEN_CFG	Contains False Alarm configurations, ADC Auto/direct mode, and Shunt Range
ADC_CONV_CFG_0	Contains conversion rate configurations
ADC_CONV_CFG_1	Contains sample size configurations
ADC_HYST_0	
ADC_HYST_1	
SENSE0_UP_THRESH	
SENSE0_LO_THRESH	
SENSE1_UP_THRESH	
SENSE1_LO_THRESH	
ADC0_UP_THRESH	
ADC0_LO_THRESH	
ADC1_UP_THRESH	
ADC1_LO_THRESH	
TMP_UP_THRESH	

Example:

```
    #Example: Configure ADC
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_ADC_CONFIG)  #Set ADC page

    #Set the following parameters:
        #ADC False Alarm = 16
        #Sense False Alarm = 32
        #Temp False Alarm = 4
        #ADC in automode
        #Shunt range 163.84mV
    ftdiObject.spiWrite(ADC_GEN_CFG_ADDRESS, ADC_GEN_CFG_FALR_ADC_16 +
ADC_GEN_CFG_FALR_SENSE_32 + ADC_GEN_CFG_FALR_TMP_4 +
ADC_GEN_CFG_AUTOMODE_Automode + ADC_GEN_CFG_SHUNT_RANGE_163p84mVrange)
    #Set the following parameters in the ADC_CONV_CFG_0 register:
        #ADC Conversion rate = 152us
        #SENSE Conversion rate = 282us
        #Temperature conversion rate = 4122us
    ftdiObject.spiWrite(ADC_CONV_CFG_0_ADDRESS, ADC_CONV_CFG_0_CONV_RATE_ADC_152us
+ ADC_CONV_CFG_0_CONV_RATE_SENSE_282us +
ADC_CONV_CFG_0_CONV_RATE_TMP_4122us)
    #Set the following parameters in the ADC_CONV_CFG_1 register:
        #ADC Samples = 16
        #SENSE Samples = 4
        #Temperature Samples = 1
    ftdiObject.spiWrite(ADC_CONV_CFG_1_ADDRESS, ADC_CONV_CFG_1_AVG_ADC_16samples +
ADC_CONV_CFG_1_AVG_SENSE_4samples + ADC_CONV_CFG_1_AVG_TMP_1sample)
```

5 AFE20408_Page_2_Def.py

Page 2 of the AFE20408 contains ADC Custom Channel Sequencing registers. The file contains variables for the registers and specific data combinations. You are required to write 0x0002 to the PAGE register before writing or reading from any register on this page.
This page contains the following registers:

ADC_CCS_IDS_n	Total of 63 CCS registers. Each register contains 2 indexes, for a total of 126 individual channels. Options for each channel include: GND SENSE0 SENSE1 ADC0 ADC1 TMP
ADC_CCS_CFG_0	Contains the Start and Stop index for the Sequencer

Example:

```
      #Example: Configure CCS. This will have the ADC iterate through only SENSE0 and ADC1.
      ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_ADC_CCS_CONFIG)  #Set ADC CCS page

   #Set the following parameters:
      #Index 0 = SENSE0
      #Index 1 = ADC1
   ftdiObject.spiWrite(ADC_CCS_IDS_0_ADDRESS, ADC_CCS_IDS_0_CCS_ID_0_SENSE0 +
ADC_CCS_IDS_0_CCS_ID_1_ADC1)
      ftdiObject.spiWrite(ADC_CCS_CFG_0_ADDRESS, 0x0001) #Start at index 0, end at index 1
```

6 AFE20408_Page_3_Def.py

Page 3 of the AFE20408 contains DAC configuration registers. The file contains variables for the registers and specific data combinations. You are required to write 0x0003 to the PAGE register before writing or reading from any register on this page.
This page contains the following registers:

DAC_CURRENT	Startup Current = 15mA Low Current = 30mA Normal Current = 90mA High Current = 120mA
DAC_SYNC_CFG	Synchronous and Broadcast configurations
DAC_CFG	OUT Pin CLAMP configuration
DAC_APD_EN	Alarm Power Down enable
DACA_APD_SRC_0	
DACA_APD_SRC_1	
OUTA_APD_SRC_0	
OUTA_APD_SRC_1	
DACB_APD_SRC_0	
DACB_APD_SRC_1	
OUTB_APD_SRC_0	
OUTB_APD_SRC_1	
DAC_CODE_LIMIT_0	
DAC_CODE_LIMIT_1	
DAC_CODE_LIMIT_2	
DAC_CODE_LIMIT_3	
DRVEN0_EN	Enable DRVEN0 for specific DACs
DRVEN1_EN	Enable DRVEN1 for specific DACs
FLEXIO_EN	

Example:

```
    #Example: Configure DACs
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_CONFIG)  #Set DAC Config page

    #Set all of the DAC currents to High mode
    ftdiObject.spiWrite(DAC_CURRENT_ADDRESS, DAC_CURRENT_DACA0_CURRENT_Normal +
DAC_CURRENT_DACA1_CURRENT_Normal + DAC_CURRENT_DACA2_CURRENT_Normal +
DAC_CURRENT_DACA3_CURRENT_Normal + DAC_CURRENT_DACB0_CURRENT_Normal +
DAC_CURRENT_DACB1_CURRENT_Normal + DAC_CURRENT_DACB2_CURRENT_High +
DAC_CURRENT_DACB3_CURRENT_High)
    #Set OUTA0 CLAMP to DACA1, OUTA2 CLAMP to VSSA, OUTB0 CLAMP to DACB1, and OUTB2
CLAMP to VSSB
    ftdiObject.spiWrite(DAC_CFG_ADDRESS,
DAC_CFG_CLAMP_SEL_OUTA0_OUTA0clampvoltageisDACA1 +
DAC_CFG_CLAMP_SEL_OUTA2_OUTA2clampvoltageisVSSA +
DAC_CFG_CLAMP_SEL_OUTB0_OUTB0clampvoltageisDACB1 +
DAC_CFG_CLAMP_SEL_OUTB2_OUTB2clampvoltageisVSSB)
    #Set OUTA0, DACA3, OUTB0, and DACB3 to toggle with DRVEN0
    ftdiObject.spiWrite(DRVEN0_EN_ADDRESS, DRVEN0_EN_DRVEN0_EN_DACA0_En +
DRVEN0_EN_DRVEN0_EN_DACA3_En + DRVEN0_EN_DRVEN0_EN_DACB0_En +
DRVEN0_EN_DRVEN0_EN_DACB3_En)
    #Set OUTA2 and OUTB2 to toggle with DRVEN1
    ftdiObject.spiWrite(DRVEN1_EN_ADDRESS, DRVEN1_EN_DRVEN1_EN_DACA2_En +
DRVEN1_EN_DRVEN1_EN_DACB2_En)
```

7 AFE20408_Page_4_Def.py

Page 4 of the AFE20408 contains the DAC Buffer registers. The file contains variables for the registers. You are required to write 0x0004 to the PAGE register before writing or reading from any register on this page.
This page contains the following registers:

DACA0
DACA1
DACA2
DACA3
DACB0
DACB1
DACB2
DACB3

Example:

```
#Example: Set DAC Data
ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_BUFFER)  #Set DAC Buffer page

#Set DACA0 to midscale code 0x1000
ftdiObject.spiWrite(DACA0_ADDRESS, 0x1000)
```

8 AFE20408_Page_6_Def.py

Page 6 of the AFE20408 contains the Active DAC registers. The file contains variables for the registers. You are required to write 0x0006 to the PAGE register before reading from any register on this page. These registers are Read Only. The registers contain the actual DAC value seen on the DAC output. This page contains the following registers:

DACA0_ACTIVE
DACA1_ACTIVE
DACA2_ACTIVE
DACA3_ACTIVE
DACB0_ACTIVE
DACB1_ACTIVE
DACB2_ACTIVE
DACB3_ACTIVE

Example:

```
#Example: Read DAC Data
ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_ACTIVE)  #Set DAC Buffer page

#Read the DACA0 data
print(ftdiObject.spiRead(DACA0_ACTIVE_ADDRESS))
```

9 Main.py

The Main.py file is an example of how to initialize the FTDI and write basic commands to the AFE20408. The code from Main.py is pasted here for convenience.

```
#Import all helper files
from FT4222_Python_Controller import *
from AFE20408_Page_0_Def import *
from AFE20408_Page_1_Def import *
from AFE20408_Page_2_Def import *
from AFE20408_Page_3_Def import *
from AFE20408_Page_4_Def import *
from AFE20408_Page_6_Def import *
from AFE20408_Page_Global_Def import *

if __name__ == '__main__':
    ftdiObject = FtdiController()   #initialize FTDI
    ftdiObject.getFtdiDeviceInfo()  #Get FTDI Info

    #SPI Communication Initialisation
    ftdiObject.initializeSPI(ft4222.SPIMaster.Mode.SINGLE, ft4222.SPIMaster.Clock.DIV_2,
ft4222.SPI.Cpol.IDLE_LOW, ft4222.SPI.Cpha.CLK_TRAILING, ft4222.SPIMaster.SlaveSelect.SS0)

    """
    TO WRITE TO AFE20408:
    ftdiObject.spiWrite([REGISTER ADDRESS], [DATA])

    TO READ FROM AFE20408:
    ftdiObject.spiRead([REGISTER ADDRESS])
    """

    """Page 0 Setup - GENERAL CONFIGURATION"""
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_GEN_CONFIG) #Set page to Gen Config, page 0
    ftdiObject.spiWrite(SDO_EN_ADDRESS, SDO_EN_SDOenabled) #enable SDO for reading
    print(ftdiObject.spiRead(CHIP_ID_ADDRESS))   #confirm AFE20408 is communicating

    #Setup PAON pin = Push-pull, Active High.
    ftdiObject.spiWrite(GEN_CFG_0_ADDRESS, GEN_CFG_0_FLEXIO_OUT_ODE_PushPull +
GEN_CFG_0_PAON_POLARITY_ActiveHigh)
    #Setup VSS range. This sets the Mid Range. Sets the FLEXIO function as RESET.
    ftdiObject.spiWrite(GEN_CFG_1_ADDRESS, GEN_CFG_1_VSSA_RANGE_MidRange +
GEN_CFG_1_VSSB_RANGE_MidRange + GEN_CFG_1_FLEXIO_FUNC_RESET)

    #Set up the ALARMOUT alarms. This only works if the FLEXIO pin is set to ALARMOUT.
    ftdiObject.spiWrite(ALARMOUT_SRC_0_ADDRESS, 0x0000) #Set to default
    ftdiObject.spiWrite(ALARMOUT_SRC_1_ADDRESS, 0x1833) #Set to default
    #Set up the PAON alarms. This does not matter if you do not use the PAON pin.
    ftdiObject.spiWrite(PAON_SRC_0_ADDRESS, 0x0133) #Set to all sources.
    ftdiObject.spiWrite(PAON_SRC_1_ADDRESS, 0x1833) #Set to default

    """Page 1 Setup - ADC CONFIGURATION"""
```

```
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_ADC_CONFIG)  #Set the page to ADC CONFIG, page 1
    #Set the False Alarm threshold, ADC Auto/Direct mode, and Shunt range.
    #This will set the False Alarm thresholds to 4 samples, the ADC to Automode, and the Shunt range to
163.84mV
    ftdiObject.spiWrite(ADC_GEN_CFG_ADDRESS, ADC_GEN_CFG_FALR_SENSE_4 +
ADC_GEN_CFG_FALR_ADC_4 + ADC_GEN_CFG_FALR_TMP_4 +
ADC_GEN_CFG_AUTOMODE_Automode + ADC_GEN_CFG_SHUNT_RANGE_163p84mVrange)
    #Set the conversion rate. This will set all channels to 152us.
    ftdiObject.spiWrite(ADC_CONV_CFG_0_ADDRESS, ADC_CONV_CFG_0_CONV_RATE_ADC_152us
+ ADC_CONV_CFG_0_CONV_RATE_SENSE_152us +
ADC_CONV_CFG_0_CONV_RATE_TMP_152us)
    #Set the sampling rate. This will set all channels to 1 sample.
    ftdiObject.spiWrite(ADC_CONV_CFG_1_ADDRESS, ADC_CONV_CFG_1_AVG_ADC_1sample +
ADC_CONV_CFG_1_AVG_SENSE_1sample + ADC_CONV_CFG_1_AVG_TMP_1sample)
    #Set the ADC Hysteresis. This sets to default.
    ftdiObject.spiWrite(ADC_HYST_0_ADDRESS, ADC_HYST_0_DEFAULT)
    ftdiObject.spiWrite(ADC_HYST_1_ADDRESS, ADC_HYST_1_DEFAULT)
    #Set the ADC channel thresholds for the Alarm configurations. These set to default.
    ftdiObject.spiWrite(ADC0_LO_THRESH_ADDRESS, ADC0_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(ADC1_LO_THRESH_ADDRESS, ADC1_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(ADC0_UP_THRESH_ADDRESS, ADC0_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(ADC1_UP_THRESH_ADDRESS, ADC1_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE0_LO_THRESH_ADDRESS, SENSE0_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE0_UP_THRESH_ADDRESS, SENSE0_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE1_LO_THRESH_ADDRESS, SENSE1_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE1_UP_THRESH_ADDRESS, SENSE1_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(TMP_UP_THRESH_ADDRESS, TMP_UP_THRESH_DEFAULT)

    """Page 2 Setup - ADC CCS CONFIGURATION"""
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_ADC_CCS_CONFIG) #Set the page to ADC CCS
CONFIG, Page 2
    #Example: Setup to read from SENSE0 after every other channel.
    ftdiObject.spiWrite(ADC_CCS_IDS_0_ADDRESS, ADC_CCS_IDS_0_CCS_ID_0_ADC0 +
ADC_CCS_IDS_0_CCS_ID_1_SENSE0)
    ftdiObject.spiWrite(ADC_CCS_IDS_1_ADDRESS, ADC_CCS_IDS_1_CCS_ID_2_ADC1 +
ADC_CCS_IDS_1_CCS_ID_3_SENSE0)
    ftdiObject.spiWrite(ADC_CCS_IDS_2_ADDRESS, ADC_CCS_IDS_2_CCS_ID_4_SENSE1 +
ADC_CCS_IDS_2_CCS_ID_5_SENSE0)
    ftdiObject.spiWrite(ADC_CCS_IDS_3_ADDRESS, ADC_CCS_IDS_3_CCS_ID_6_TMP +
ADC_CCS_IDS_3_CCS_ID_7_SENSE0)
    #Set the Start and Stop indices to 0 and 7
    ftdiObject.spiWrite(ADC_CCS_CFG_0_ADDRESS, 0x0007)

    """Page 3 Setup - DAC CONFIGURATION"""
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_CONFIG) #Set the page to DAC CONFIG, Page 3
    #Set the DAC current limits. This sets all the DACs to max current limit, 120mA
    ftdiObject.spiWrite(DAC_CURRENT_ADDRESS, 0xFFFF)
    #Set the OUT CLAMPs. This sets OUTA0 CLAMP = DACA1, OUTA2 CLAMP = DACA3, OUTB0
CLAMP = DACB1, OUTB2 CLAMP = DACB3
    ftdiObject.spiWrite(DAC_CFG_ADDRESS,
DAC_CFG_CLAMP_SEL_OUTA0_OUTA0clampvoltageisDACA1 +
```

```
DAC_CFG_CLAMP_SEL_OUTA2_OUTA2clampvoltageisDACA3 +
DAC_CFG_CLAMP_SEL_OUTB0_OUTB0clampvoltageisDACB1 +
DAC_CFG_CLAMP_SEL_OUTB2_OUTB2clampvoltageisDACB3)
    #Disables DAC broadcast and Syncronous modes.
    ftdiObject.spiWrite(DAC_SYNC_CFG_ADDRESS, 0x0000)
    #Set the DAC Alarm Powerdown settings. This sets the register to default - All DACs power down to
VSS during an Alarm.
    ftdiObject.spiWrite(DAC_APD_EN_ADDRESS, DAC_APD_EN_DEFAULT)
    #Configure what Alarms trigger each DACs. This enables all alarms)
    ftdiObject.spiWrite(DACA_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(DACA_APD_SRC_1_ADDRESS, DACA_APD_SRC_1_DEFAULT)
    ftdiObject.spiWrite(DACB_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(DACB_APD_SRC_1_ADDRESS, DACB_APD_SRC_1_DEFAULT)
    ftdiObject.spiWrite(OUTA_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(OUTA_APD_SRC_1_ADDRESS, OUTA_APD_SRC_1_DEFAULT)
    ftdiObject.spiWrite(OUTB_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(OUTB_APD_SRC_1_ADDRESS, OUTB_APD_SRC_1_DEFAULT)
    #Set the DRVEN pin configuration. This sets the OUT pins to toggle with DRVEN0, and disables
DRVEN1.
    ftdiObject.spiWrite(DRVEN0_EN_ADDRESS, DRVEN0_EN_DRVEN0_EN_DACA0_En +
DRVEN0_EN_DRVEN0_EN_DACA2_En + DRVEN0_EN_DRVEN0_EN_DACB0_En +
DRVEN0_EN_DRVEN0_EN_DACB2_En)
    ftdiObject.spiWrite(DRVEN1_EN_ADDRESS, 0x0000)

    """Setup Global Page"""
    #Disable Software DRVEN for the OUT pins
    ftdiObject.spiWrite(DRVEN_SW_EN_ADDRESS, 0x00AA)
    #Enable Software DRVEN on the odd DACs. REQUIRED for enabling DACA1/3, DACB1/3
    ftdiObject.spiWrite(DRVEN_ADDRESS, 0x00AA)
    #Enable all DACs
    ftdiObject.spiWrite(PWR_EN_ADDRESS, 0x00FF)

    """Set DAC voltage"""
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_BUFFER) #Set DAC Buffer page, Page 4
    #DAC voltage (Positive) = (DAC_CODE * 10)/2^13
    #DAC Voltage (Negative) = (DAC_CODE * 10)/2^13 - 10
    ftdiObject.spiWrite(DACA0_ADDRESS, 0x1570) # -3.3V or 6.7V depending on output range.
    ftdiObject.spiWrite(DACB0_ADDRESS, 0x0A8F) # 3.3V or -6.7V depending on output range.

    #Start ADC
    ftdiObject.spiWrite(TRIGGER_ADDRESS, TRIGGER_ADC_TRIG_EnableADC)

    print("SENSE0: " + ftdiObject.spiRead(0x18)) # ADC_SENSE0 channel readback
    print("SENSE1: " + ftdiObject.spiRead(0x19)) # ADC_SENSE1 channel readback
    print("ADC0: " + ftdiObject.spiRead(0x1A)) # ADC_ADC0 channel readback
    print("ADC1: " + ftdiObject.spiRead(0x1B)) # ADC_ADC1 channel readback
    print("TEMP: " + ftdiObject.spiRead(0x1C)) # ADC_TEMP channel readback

    #close the FTDI when done.
    ftdiObject.closeHandle()
```

10 Common Sequences

10.1 Reset Sequence

The software reset will reset all registers in the device to their default state.

```
    #RESET SEQUENCE
    ftdiObject.spiWrite(NOP_ADDRESS, NOP_SW_RST_SoftwareReset)
```

10.2 ADC Configuration

There are many ways to configure the ADC. Below is an example:

```
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_ADC_CONFIG)  #Set the page to ADC CONFIG, page 1
    #Set the False Alarm threshold, ADC Auto/Direct mode, and Shunt range.
    #This will set the False Alarm thresholds to 4 samples, the ADC to Automode, and the Shunt range to
163.84mV
    ftdiObject.spiWrite(ADC_GEN_CFG_ADDRESS, ADC_GEN_CFG_FALR_SENSE_4 +
ADC_GEN_CFG_FALR_ADC_4 + ADC_GEN_CFG_FALR_TMP_4 +
ADC_GEN_CFG_AUTOMODE_Automode + ADC_GEN_CFG_SHUNT_RANGE_163p84mVrange)
    #Set the conversion rate. This will set all channels to 152us.
    ftdiObject.spiWrite(ADC_CONV_CFG_0_ADDRESS, ADC_CONV_CFG_0_CONV_RATE_ADC_152us
+ ADC_CONV_CFG_0_CONV_RATE_SENSE_152us +
ADC_CONV_CFG_0_CONV_RATE_TMP_152us)
    #Set the sampling rate. This will set all channels to 1 sample.
    ftdiObject.spiWrite(ADC_CONV_CFG_1_ADDRESS, ADC_CONV_CFG_1_AVG_ADC_1sample +
ADC_CONV_CFG_1_AVG_SENSE_1sample + ADC_CONV_CFG_1_AVG_TMP_1sample)
    #Set the ADC Hysteresis. This sets to default.
    ftdiObject.spiWrite(ADC_HYST_0_ADDRESS, ADC_HYST_0_DEFAULT)
    ftdiObject.spiWrite(ADC_HYST_1_ADDRESS, ADC_HYST_1_DEFAULT)
    #Set the ADC channel thresholds for the Alarm configurations. These set to default.
    ftdiObject.spiWrite(ADC0_LO_THRESH_ADDRESS, ADC0_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(ADC1_LO_THRESH_ADDRESS, ADC1_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(ADC0_UP_THRESH_ADDRESS, ADC0_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(ADC1_UP_THRESH_ADDRESS, ADC1_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE0_LO_THRESH_ADDRESS, SENSE0_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE0_UP_THRESH_ADDRESS, SENSE0_UP_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE1_LO_THRESH_ADDRESS, SENSE1_LO_THRESH_DEFAULT)
    ftdiObject.spiWrite(SENSE1_UP_THRESH_ADDRESS, SENSE1_UP_THRESH_DEFAULT)
        ftdiObject.spiWrite(TMP_UP_THRESH_ADDRESS, TMP_UP_THRESH_DEFAULT)

    #Start ADC. TRIGGER then read the ADC outputs.
    ftdiObject.spiWrite(TRIGGER_ADDRESS, TRIGGER_ADC_TRIG_EnableADC)

    print("SENSE0: " + ftdiObject.spiRead(0x18)) # ADC_SENSE0 channel readback
    print("SENSE1: " + ftdiObject.spiRead(0x19)) # ADC_SENSE1 channel readback
    print("ADC0: " + ftdiObject.spiRead(0x1A)) # ADC_ADC0 channel readback
    print("ADC1: " + ftdiObject.spiRead(0x1B)) # ADC_ADC1 channel readback
    print("TEMP: " + ftdiObject.spiRead(0x1C)) # ADC_TEMP channel readback
```

10.3 DAC Configuration

There are many ways to configure the DAC. Below is an example:

```
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_CONFIG) #Set the page to DAC CONFIG, Page 3
    #Set the DAC current limits. This sets all the DACs to max current limit, 120mA
    ftdiObject.spiWrite(DAC_CURRENT_ADDRESS, 0xFFFF)
    #Set the OUT CLAMPs. This sets OUTA0 CLAMP = DACA1, OUTA2 CLAMP = DACA3, OUTB0
CLAMP = DACB1, OUTB2 CLAMP = DACB3
    ftdiObject.spiWrite(DAC_CFG_ADDRESS,
DAC_CFG_CLAMP_SEL_OUTA0_OUTA0clampvoltageisDACA1 +
DAC_CFG_CLAMP_SEL_OUTA2_OUTA2clampvoltageisDACA3 +
DAC_CFG_CLAMP_SEL_OUTB0_OUTB0clampvoltageisDACB1 +
DAC_CFG_CLAMP_SEL_OUTB2_OUTB2clampvoltageisDACB3)
    #Disables DAC broadcast and Syncronous modes.
    ftdiObject.spiWrite(DAC_SYNC_CFG_ADDRESS, 0x0000)
    #Set the DAC Alarm Powerdown settings. This sets the register to default - All DACs power down to
VSS during an Alarm.
    ftdiObject.spiWrite(DAC_APD_EN_ADDRESS, DAC_APD_EN_DEFAULT)
    #Configure what Alarms trigger each DACs. This enables all alarms)
    ftdiObject.spiWrite(DACA_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(DACA_APD_SRC_1_ADDRESS, DACA_APD_SRC_1_DEFAULT)
    ftdiObject.spiWrite(DACB_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(DACB_APD_SRC_1_ADDRESS, DACB_APD_SRC_1_DEFAULT)
    ftdiObject.spiWrite(OUTA_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(OUTA_APD_SRC_1_ADDRESS, OUTA_APD_SRC_1_DEFAULT)
    ftdiObject.spiWrite(OUTB_APD_SRC_0_ADDRESS, 0x0133)
    ftdiObject.spiWrite(OUTB_APD_SRC_1_ADDRESS, OUTB_APD_SRC_1_DEFAULT)
    #Set the DRVEN pin configuration. This sets the OUT pins to toggle with DRVEN0, and disables
DRVEN1.
    ftdiObject.spiWrite(DRVEN0_EN_ADDRESS, DRVEN0_EN_DRVEN0_EN_DACA0_En +
DRVEN0_EN_DRVEN0_EN_DACA2_En + DRVEN0_EN_DRVEN0_EN_DACB0_En +
DRVEN0_EN_DRVEN0_EN_DACB2_En)
        ftdiObject.spiWrite(DRVEN1_EN_ADDRESS, 0x0000)

    """Set DAC voltage"""
    ftdiObject.spiWrite(PAGE_ADDRESS, PAGE_DAC_BUFFER) #Set DAC Buffer page, Page 4
    #DAC voltage (Positive) = (DAC_CODE * 10)/2^13
    #DAC Voltage (Negative) = (DAC_CODE * 10)/2^13 - 10
    ftdiObject.spiWrite(DACA0_ADDRESS, 0x1570) # -3.3V or 6.7V depending on output range.
    ftdiObject.spiWrite(DACB0_ADDRESS, 0x0A8F) # 3.3V or -6.7V depending on output range.
```

11 Common Issues

ERROR: "No module named 'ft4222'"

Follow this guide to import the FT4222 driver to your PC. Importing Python Libraries in VS Code | by Gbemisola Adekoya | Medium

Use "pip install ft4222" in the terminal.