1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

PmodIA™ Library Reference Manual

Revised January 5, 2015
This manual applies to the PmodIA rev. A

Overview

The PmodIA Llibrary provides an interface to the AD5933 impedance converter system. The library initializes the AD5933 using I2C communication protocol and reads real and imaginary values from it after a frequency sweep has been performed.

1 Library Operation

The IA Library has two code sections, the header file and .cpp file. The header file WireIA.h defines all of the used register addresses, their sizes and the IA class which has member functions, member variables and type defined structures to interface with the registers. There are two private type defined structures that contain IA data and sweep parameter data. To instantiate an IA object, include the IA library in the same file directory of the project and call on the constructor function.

The library provides both a simple and detailed approach to performing an impedance calculation, which is through the functions in the library. There is a general function, PerformFrequencySweep(), that abstracts the step by step process of performing frequency sweep. However, you can use the member functions in class IA to perform a step by step configuration of the registers and perform a frequency sweep.

1.1 IA Initialization

The IA module is initialized by calling the IA () constructor function. The constructor initializes the I2C Wire communication. The wire library initializes the I2C1 controller only. The IA () constructor function initializes all of the register buffers and type defined structure variables with recommended values, specifically the programmable gain factor and the excitation voltage as shown below.

```
IA_DATA.range = 7; // recommended excitation voltage setting
IA_DATA.pga_gain = 1; //recommended programmable gain factor setting
```

2 Used Registers and Their Functions

These are the main registers used in IA libraries. A more in-depth view of these register functions can be found in the AD5933 datasheet. Any of these registers can be accessed by a user through the readRegisterValue and blockWrite functions. Some of the registers are read-only and hence can only be read and not modified.

Register Address	Register Name	Register Function	Register Type
0x80 0x81	Control	Sets the AD5933 control modes	Read/Write Read/Write
0x82 0x83 0x84	Start Frequency	Set the subsequent frequency where sweep is initiated	Read/Write Read/Write Read/Write
0x85 0x86 0x87	Frequency Increment	Sets the frequency increment between consecutive frequency points along the sweep	Read/Write Read/Write Read/Write
0x88 0x89	Number of increments	D15 to D8 D7 to D0	Read/Write Read/Write
0x8A 0x8B	Number of settling time cycles	D15 to D8 D7 to D0	Read/Write Read/Write
0x8F	Status	D7 to D0	Read only
0x92 0x93	Temperature Data	D15 to D8 D7 to D0	Read only Read only
0x94 0x95	Real Data	D15 to D8 D7 to D0	Read only Read only
0x96 0x97	Imaginary Data	D15 to D8 D7 to D0	Read only Read only

3 IA Library Functions

3.1 WireIA.h

The WireIA.h contains all the function declarations as well as other included libraries such as Wire.h, EEPROM.h, and WProgram.h. These included libraries contain definitions of other communication protocols used along with IA library, e.g., the Serial Interface to display data on a serial terminal. Constant definitions for register addresses and constants for computation are also declared in WireIA.h. Class IA is declared in WireIA.h and contains both private and public member variables, functions, and structures.

4 WireIA.cpp

4.1 Public Functions

IA()
Parameters:
- None

Return Value:
- None

Constructor for the IA object. Initializes register buffers and type defined structures that hold information about the IA system parameters.

setRegisterPointer(uint8_t RegAddress)
Parameters:
- Uint8_t RegAddress
  - 8 bit register address

Return Value:
- None

Sets the internal AD5933 register pointer to the address in the parameter. During a read or write action, the address that will first be addressed is set by this function.

blockWrite(int numBytes, uint8_t *data)
Parameters:
- Int numBytes
  - Number of bytes to be written to the AD5933
- Uint8_t *data
  - Data to be written to AD5933

Return Value:
- None

Writes a stream of bytes to slave using block write command through the I2C communication protocol. Set Register function needs to precede this function to set the beginning register where writing will begin.

blockRead(int numBytes, uint8_t *buffer, uint8_t RegAddress)
Parameters:
- Int numBytes
  - Number of bytes to read from AD5933
- Uint8_t *buffer
  - Buffer to store read data
- Uint8_t RegAddress
  - Register to start reading from

Return Value:
- None

Reads a stream of bytes from slave (AD5933). This function does not need to be preceded by set register function as it calls it within itself.

Uint8_t readRegisterValue(uint8_t RegAddress)

Parameters:

- Uint8_t RegAddress
  - Register address to read data

Return Value:

- uint8_t data
  - Byte of data read

Reads a byte of data from the register.

IA_init()

Parameters:

- None

Return Value:

- None

Initializes the IA I2C communication from the master to the slave.

setControlRegister(unsigned int function, unsigned int gain)

Parameters:

- Unsigned int function
  - Function mode of the AD5933
- Unsigned int gain
  - Amplifies the response signal into the ADC by 5 or 1

Return Value:

- None

This function sets the AD5933 control modes through the control register values.

setStartFrequency(unsigned int frequency)

Parameters:

- Unsigned int frequency
  - Start frequency

Return Value:

- None

This function sets the start frequency for the sweep in the Start Frequency Register. A call to this function performs the endian conversion and writes data through I2C to the start frequency register.

setFrequencyIncrement(unsigned int deltaFreq)

Parameters:

- Unsigned int deltaFreq
    - Delta frequency between excitation voltage points

Return Value:

- None

This function sets the frequency delta increment between two consecutive frequency points along the sweep.

setNumSamples(unsigned int samples)

Parameters:

- Unsigned int samples
    - Total number of samples

Return Value:

- None

This function sets the number of frequency points in the frequency sweep.

setSettlingTime(unsigned int settlingTime, unsigned int settlingFactor)

Parameters:

- Unsigned int settlingTime
    - Number of settling time cycles
- Unsigned int settlingFactor
    - Increases the settling time by 0, 2 or 4

Return Value:

- None

This function determines the number of output excitation cycles that are allowed to pass through the unknown impedance during a sweep point.

setRange(int option)

Parameters:

- Int option
    - Provides information of output voltage and feedback resistor to use

Return Value:

- None

This function gives a user an option to select the output excitation voltage and feedback resistor to use for a frequency sweep.

setSweepParameters(bool cal_Flag, int begin_Freq, int f_Step, int nSamples, bool r_Flag)

Parameters:

- Bool cal_Flag
  - Calibration sweep flag
- Int begin_Freq
  - Start Frequency
- Int f_Step
  - Delta Frequency
- Int nSamples
  - Number of sample points in a frequency sweep
- Bool r_Flag
  - repeat flag for a sweep point repeat

Return Value:

- None

This function initializes the sweep parameter struct with sweep parameters of the AD5933.

PerformFrequencySweep()

Parameters:

- None

Return Value:

- None

This function programs the frequency sweep parameters into relevant registers and then performs a frequency sweep through commands to the control register.

TwoPTCalibration()

Parameters:

- None

Return Value:

- None

This function performs system calibration to determine the gain factor and phase shift. These two system values are saved in the EEPROM.

CalculateGainFactor(int CalibrationIMP)

Parameters:

- Int CalibrationIMP
  - Known impedance to calibrate the system

Return Value:

- None

This function calculates the gain factor of the AD5933.

Impedance()

Parameters:

- None

Return Value:

- None

This function calculates impedance using a gain factor.

Phase()

Parameters:

- None

Return Value:

- None

This function calculates the phase shift from an impedance calculation.

ZReal()

Parameters:

- None

Return Value:

- None

This function calculates the real component of the impedance, $|Z_{real}| = |Z| * \cos(Z_{phi})$.

ZImaginary()

Parameters:

- None

Return Value:

- None

This function calculates the imaginary component of the Impedance, $|Z_{imaginary}| = |Z| * \cos(Z_{phi})$.

getIA_Data()

Parameters:

- None

Return Value:

- None

This function reads value from AD5933 registers and then populates the IA Data structure. This function requires a frequency sweep to be conducted.

Double getGF()
Parameters:

- None

Return Value:

- Gain value dependent on excitation voltage range

This function reads the Gain Factor from the EEPROM and saves it to the IA data structure.

Double getPhase()
Parameters:

- None

Return Value:

- The phase angle of last measured impedance

This function determines the phase angle of the impedance load.

Double getImpedance()
Parameters:

- None

Return Value:

- Impedance value

This function gets the last measured impedance.

5 Private Functions

EEPROMReadGF()
Parameters:

- None

Return Value:

- None

This function reads the gain factor stored in the EEPROM and saves the vale in the IA data structure. The value read in the EEPROM is dependent on the address as there are two gain factor values based on the feedback resistor being used for impedance measurement.

EEPROMWriteGF()
Parameters:

- None

Return Value:

- None

This function writes to the EEPROM the gain factor value calculated after a system calibration sweep. There are two gain factor values based on the feedback resistor being used and therefore there are two addresses where data can be written to.

6 Class Member Variables

All member variables in class IA are private.

Member Variable	Function
IA_STRUCT IA_DATA	Type defined structure that contains IA system data
SWEEP_PARAM SParam	Type defined structure that contains sweep parameter data
Uint8_t ControlData[2]	16 bit, control register buffer
Uint8_t StartFreqData[3]	24 bit, start frequency register buffer
Uint8_t FrequencyIncData[3]	24 bit, frequency increment register buffer
Uint8_t NumIncData[2]	16 bit, number of increments register buffer
Uint8_t SettlingTimeData[2]	16 bit, number of settling time cycles register buffer
Uint8_t realData[2]	16 bit, real data register buffer
Uint8_t imagData[2]	16 bit, imaginary data register buffer