AIM Software Interface Guide

Dec 2012

Some people may be interested in writing programs to interface with the AIM4170, the AIMuhf or the PowerAIM 120. A DLL with the necessary functions has been created to make integration of the AIM with other systems very straightforward. The main program running on the user's system can be written in any language.

There is a collection of files that demonstrate the DLL functions:
http://www.w5big.com/AIM_dll_863demo.zip

Put these files in a folder called : C**:\AIM_863_dll** for the demo program.
Later they can be put in any folder.

The lib file is normally in the same folder with the exe file that calls the DLL functions. Include this statement at the top of the program:

$use "AIM_863_Dll.lib"

if the lib file is not in the same folder as the exe file, the full path name should be used. For example,

$**use "c:\\AIM_863_dll\\AIM_863_Dll.lib"**

Library file: **AIM_863_DLL.lib**
DLL file: **AIM_863_DLL.dll**

The demo program shows the complete statements.

The following functions are declared in the user's program:
Note the function names are case sensitive when declared.

Use the program **AIM_860C.exe** to calibrate the AIM. This will save the cal data in the format that the DLL routines expect.

---

declare "AIM_863_dll",**AIM_InitVars(flag:int), int**

Initialize the variables used by the AIM. The flag is not used.
The returned value will always be zero.

---

declare "AIM_863_dll",**AIM_CommPort(flag:int,port:int, baud:int), int**

Initialize the comm port. The port number is specified and the baud rate is 57600 or 115000. If the flag=1, the port is initialized; if the flag=0, the handle associated with the port is released. The handle should be released at the end of the program.

The AIM hardware powers up with a default setting of 57600 baud. The pc should be initialized to 57600. Then a command to change to 115K can be sent .

The error code returned is zero if there was no problem.

---

declare "AIM_863_dll",**AIM_GetVersion(flag:int),string**

Read the version information stored in the firmware of the AIM. This is used to set up flags in the AIM program so one program can be used with all three instruments (AIM4170, AIMuhf, PowerAIM).

This function should be called after the comm port is initialized and before the cal file is loaded.

The length of the string that's returned is less than 255 bytes.
If its length is less than 10 bytes, it represents a numeric error code.

---

declare "AIM_863_dll",**AIM_GetSerialNum(flag:int),int**

Read the version information stored in the firmware of the AIM. This is used to set up flags in the AIM program so one program can be used with all three instruments (AIM4170, AIMuhf, PowerAIM). It is the same as the function called **AIM_GetVersion** discussed above.

This function should be called after the comm port is initialized and before the cal file is loaded. It can be used instead of the AIM_GetVersion function or both can be used in any combination.

The returned value is a six digit number representing the lower six digits of the unit's ID number. This is the number included in each calibration file to identify the file. *It is not the same as the serial number printed on the back of the case.* This ID number can be used when saving data for a particular AIM. For example, the name of the file can include this number.

If the integer that is returned is equal to "6", it indicates an error. All valid ID numbers will be much larger than six.

---

declare "AIM_863_dll",**AIM_LoadCalFile(_str : string), int**

Load a calibration file. The parameter called **_str** is the full path name of the cal file. This is equivalent to the regular AIM program **File -> Load Cal File** function.

The error code returned is zero if there was no problem.

---

declare "AIM_863_dll",**AIM_ProgRelay(flag: int),int**

Close the isolation relay if the flag=1 **and open the relay** if the flag=0.
The relay is closed when the red led on the front panel is on.

The relay should be left open except when a test is in progress.

If the AIM_ScanPoint function is called and the relay is open, it will be closed automatically before reading the impedance. It will be left closed after the reading.

Before the program is closed, be sure to program the relay to the open state. and release the handle for the comm port. This is illustrated in the **AIM_DLL 863demo** program

The error code returned is zero if there was no problem.

---

declare "AIM_863_dll",**AIM_ScanPoint(flag:int, freq_ : double,**
Rss: double byref,
Xss:double byref), int

Make an impedance reading at the specified frequency.
The freq value is in MHz.
The Resistive component (real part) and the Reactive component (imaginary part) of the impedance are returned in the double precision variables defined by the user as **Rss** and **Xss**. The values of the data in these variables don't matter when this function call is made.

If the flag=1, the data read by the AIM will be corrected using the present calibration data in memory. If the flag=0, the raw data will be returned without using the calibration correction.

The returned error code is zero if there was no problem.

---

declare "AIM_863_dll",**AIM_AvgReadings(avg : int), int**

Set the averaging value that is used by the AIM firmware to average the ADC readings. It can have a value from **0 to 128, 256, 512, or 1024**.

The error code returned is zero if there was no problem.

---

declare "AIM_863_dll", **AIM_Sync(flag:int) : int**

Output a Sync Pulse if the AIM has this hardware feature. The AIM4170B, AIM4170C, AIMuhf and PowerAIM have a connector on the rear panel for a sync output. This pulse is generated just before a data sample is taken. There are details in the AIM manual.

If **flag=0**, the sync pulse is disabled. (default condition is disabled after initialization)

If **flag=1**, output a single pulse the next time the AIM_ScanPoint function is called and then disable the sync.

If **flag=2**, output a pulse every time the AIM_ScanPoint function is called and leave it enabled.

An error = 0 is returned if the function is successful.
If the instrument does not have a sync output, error = 34524 is returned.

---

declare "AIM_863_dll",**AIM_ConstantFreq(flag:int, Constfreq:double),int**

If flag = 1, the AIM will set to a constant frequency and the relay will close automatically.

To turn the AIM off, set the same command with flag=0.

The error code returned is zero if there was no problem.

---

declare "AIM_863_dll",**AIM_GetTemperature(flag:int),double**

if flag=1, return the temperature in Fahrenheit.
if flag=0, return the temperature in Celcius.

if there is an error, the double precision variable will be equal to the error code.

Demo Program

Place all the files associated with the DLL demo in a folder named**: C:\AIM_863_dll**

Connect an AIM to the pc and make sure it is able to run with a regular AIM program, like AIM_860. Make a note of the comm port that is being used with the AIM program. To find the comm port, click **Setup -> Enter comm port**. The port will be the default value shown in the dialog box.

Close the AIM pc program. Turn the AIM power off and back on to initialize the firmware.

Start the DLL demo program called **AIM_dll_863demo.exe** and enter the comm port in the second text box shown in this window.

Click: **Initialize, Comm port, Get Version, Load Cal File in that order**.
The error code should be zero after each function.

Clicking the Relay button will turn the red led on and off.

Now the program is ready to read impedance data. Put a load on the RF port of the AIM. Enter the desired frequency in MHz in the first text box of the demo program.
Click the **Point Data** button.

The **frequency**, **R** and **X** will be displayed in the upper left corner of the window.
The red led will stay on.

If an error occurs, click the Err Code button to see a list of the error codes.

After the AIM is initialized, you can click the **Recycle** button to take impedance reading repetitively until the **Quit** button is clicked.

A tip for using the DLL with .NET :

This tip was found by a user of the AIM DLL:

It looks like the .NET runtime was releasing some global variables that the DLL was still trying to use. In case anybody wants to use the DLL in a .NET environment, the change is to remove the project reference called ***Microsoft.CSharp****. It makes some changes to the way .NET compiles code and interoperates with dynamic language runtime –*

Our thanks to JM for this feedback.

Sample Routine

The first four lines below only have to be executed when the program starts to run.
They don't have to be repeated before each test scan.

Be sure the AIM hardware is connected and powered up when the program start.

```
error= AIM_InitVars(0)  ;  // initialize data in the AIM, error will be zero

error=AIM_CommPort(1,port,baud)  ;  // Open comm port,  error=0 if successful.

s1=AIM_GetVersion(1)  ;    // Read version info in the AIM firmware.  Return a string.

error=AIM_LoadCalFile(filename)  ;  // user supplied file name for cal data.
                                    // This should be the complete path name.
                                    // error=0 if file loaded successfully.
```

```
Start_Freq=1 ;  // frequency values in MHz
End_Freq=20 ;
Delta_Freq=0.2 ;

num= 1 + (End_Freq - Start_Freq)/Delta_Freq   ;  // number of data points

error=AIM_AvgReadings(4)  ;  // set firmware averaging to 4x

error=AIM_ProgRelay(1)  ;  // close the isolation relay when ready to test

for (i=0 ;  i++ ; i < num)  // do a scan.
{

   freq= Start_Freq + i*Delta_Freq  ;  // see note 1 below

   error=AIM_ScanPoint(1,freq,Rss,Xss) ;  // values  of Rss, Xss don't matter here.
                                          // They provide the addresses where the data
                                          // will be returned.

   Rs[i]=Rss ;  // save the data.
   Xs[i]=Xss ;

}

error=AIM_ProgRelay(0)  ;  // open the isolation relay when test is complete
```

Note 1: When incrementing the frequency, its better to calculate the new frequency based on the index value times the Delta_Freq rather than simply adding Delta_Freq to the present value of the frequency. This reduces the effect of a small round off error due to the finite precision of floating point numbers. *The difference in program execution time is in the nanosecond range on modern pc's, consequently it is negligible.*

Equations:

Z = Rs + j*Xs *Rs and Xs correspond to the primary measurement data after being corrected by the cal data. This is the value of impedance at the point where the calibration loads were placed. All other parameters are calculated using these values.*

Zmag = sqrt(Rs*Rs+Xs*Xs) ;
Phase = atan(Xs/Rs) ;

Gamma = reflection coefficient = ((Rs + j*Xs) - Zo) / ((Rs + j*Xs) + Zo) ;
// Zo = reference impedance. It can be **anything**, including a complex number.
// Typically it is 50 or 75. Zo is defined in the user's program.
// Gamma is a complex number = RealPart + j*ImagPart
// *The reflection coefficient may be referred to as "**Rho**".*

// magnitude(gamma)= sqrt(RealPart*RealPart + ImagPart*ImagPart)

SWR = (1 + magnitude(gamma)) / (1 - magnitude(gamma)) ;

// SWR is not a complex number, it has only a magnitude.

Return Loss = -20*LOG(magnitude(gamma)) ;

// The series impedance Z = Rs + j*Xs, can be converted an equivalent parallel
// impedance with these equations:

Zmag_squared = Rs*Rs + Xs*Xs ; // Rs and Xs are the series circuit values.

Rp = Zmag_squared/Rs ; // Rp and Xp are the equivalent parallel circuit values.
Xp = Zmag_squared/Xs ;

Error Codes

6 => error trying to read serial number (ID number)
34502 => could not init the comm port
34503 => error closing the comm port
34505 => there was no communication between the AIM and the PC when asking for the version
34506 => error reading version, check comm port, cable and DC power.
34508 => Cal file format is not compatible
34509 => Cal file is not for this instrument
34510 => invalid temperature data
34511 => Invalid multipliers (UHF AIM)
34512 => Freq out of range
34513 => No data from AIM
34515 => This instrument cannot read temperature.
34516 => error trying to read temperature.
34517 => This instrument cannot read RF voltage.
34518 => error trying to read RF voltage.
34519 => No cal data is loaded
34520 => Cal file is for a different instrument
34521 => Cal file is for a different instrument
34522 => Cal file is not compatible with this program
34523 => Cal file was not found
34524 => This instrument does not have a sync output
34530 => Cal frequency range is not compatible with this instrument.