AIM4170 Programmer's Guide

Aug, 2008

Some people may be interested in writing programs to interface with the AIM4170, so the command interface details are included here.

The analyzer communicates with the PC by means of an RS232 data link. The port is set for 57,600 baud, 8 data bits, 1 stop bit, no parity. There is no handshaking; just two signal lines (send, receive) and a ground wire connect the analyzer and the PC.

When the AIM170 first powers-up, the internal controller sends out a test string that can be read or ignored. It's only purpose is to show the AIM4170 is functioning. The content of the string is (approximately):

"Antenna Analyzer AIM4170\n\r by W5BIG\n\n\r"

The AIM4170 does not expect any reply from the PC.

After power-up, the analyzer starts listening for a command character from the PC. This command character may be followed by one or more data characters. The data from the PC to the AIM4170 is usually sent as ASCII characters. In some cases, non-zero characters are sent as numbers without being converted to ASCII. For example, a numeric 1 can be sent as 0x01 without upsetting the PC's I/O routine. If the PC output routine sees a numeric zero, it thinks this is a null character indicating the end of a string.

While expecting data from the PC, the analyzer will wait as long as necessary for each character. After the last character in a command is received, the analyzer will execute the command and **immediately** start sending a reply, if required, back to the PC. The PC should be ready to receive the characters without interruption. Each character takes about 180 microseconds to send if the baud rate is 57.6K and 90 usec if the baud rate is 115K. The RS232 hardware in the PC will buffer several characters in a first-in-first-out buffer in case the PC takes more than 90 usec to process a character occasionally. Even slow (400MHz) computers are able to keep up with the data stream from the analyzer.

Commands:

B – read battery voltage. This can be used to read the voltage of the battery that is powering the AIM4170. The PC sends an ascii B and receives back two characters that form a 16 bit integer proportional to the battery voltage. If the characters received from the AIM4170 are called char0 and char1, the battery voltage calculation is:

```
Battery_voltage=(char0*256 + char1)/205  ; // full scale voltage is 19.9V
```

C – Change baud rate to 115K – The AIM powers up with its RS232 port set to 57.6K. When it receives a "C" command, it will immediate initialize the port to 115K. It will not go back to 57.6K until the power is cycled off/on. Note that a character corresponding to an ASCII C can be sent as part of a data string from the PC without being interpreted as a

command. A command character is the first character received when the AIM is in an idle mode.

D – Autopower-off enable/disable. "D1" enables the the autopower-off function in the controller. "D0" disables the autopower-off function (power will stay on all the time). The controller powers-up with autopower-off **enabled** so if battery power is

being used, it will turn off after 10 minutes of inactivity. A "D0" command from the PC sets this flag so the power **will not** turn off. This is usually the desired situation when the AIM4170 is operating from AC power.

F – take a measurement at the specified frequency.

The frequency to be used has to be expressed as a long integer (32 bits) for programming the DDS, which is the analyzer's frequency synthesizer. The conversion is as follows:

Let FREQ = frequency = a variable of type **double** in the PC program. This is the frequency value in megahertz we want to program.

The DDS clock frequency, **osc_freq,** is 400 MHz. This can be defined as a constant for convenience.

The calculations to convert from frequency to the **longint** program word are:

freq_conv = 4294967296 ; // = 2^32 this is a constant

u=FREQ/osc_freq ; // u is a dummy variable of type **double**

u=u*freq_conv + 0.5 ; // round off by adding 0.5

k=(longint) u ; // type cast to 32 bit integer

Now convert k to an ASCII string of exactly 8 characters. If necessary, it should be padded on the **left** with **leading** ASCII zeros (0x30).

The whole command string, s1, is then: s1 = "F"+freq_string.

Nine characters are sent to the analyzer for this command (command char + 8 data char).

The Windows command might be:

WriteFile(hcom,s1,9,written,0) ; // Send command

For example, to take a measurement at 7.1 MHz:

FREQ = 7.1

u= (7.1/400)* 4294967296 + 0.5 = 76235670

k=(longint)u = 76235670 = 0x48B4396

Converting k to a string: s1=”48B5396”

This string has a length of only 7 characters, so we add one more leading zero:

s1 = “048B5396”

Then concatenate this with the command character, F:

s1 = “F” + s1 = “F048B5396”

Now we have the nine-character command to send to the analyzer.

After sending this command, the PC should start listening for the data back from the analyzer. This return data will start as soon as the raw data has been collected by the analyzer (this takes a few milliseconds).

The returned data is 72 characters in all. These characters can be put into an array as they are received. Let’s assume the array is called **darray[]**. Note that these characters are **not** ASCII, they are pure data bytes. *The PC is able to receive bytes with a value of zero so we don’t have to worry about translating them to ASCII.* The format is:

Programmed Frequency =4 characters (4 bytes = longint). (This should be the same as the frequency word that was sent with the “F” command string)

16 amplitude values for the load port. These values represent the digitized sinusoidal waveform proportional to the load current. (32 characters in all)

17 amplitude values for the internal reference port. These values represent the digitized sinusoidal waveform proportional to the load voltage. (34 characters in all).

checksum – two characters (one 16 bit unsigned integer)

The checksum value should equal the summation of all the previous integer values, with the upper bits masked off and only the 16 low-order bits remaining.

For example:

```
cksum=0 ;

for (i=0; I<70; I+=2)       // Calculate cksum of received data

    cksum=cksum + darray[i]*256 + darray[i+1] ;

cksum = cksum & 0xFFFF ;
```

The last two bytes in the darray are used for the checksum received from the analyzer:

j=darray[70]*256 + darray[71] ; // this is the cksum calculated by the analyzer

j should be equal to cksum. If there is an error, the programmer has the option of retrying the data transfer or displaying an error message.

G – output a constant frequency. The analyzer will be programmed to the specified frequency but it does not take a measurement. The output will stay **on** so it can be used as a test signal for other equipment, e.g., a radio receiver. The output amplitude is about 40 millivolts rms into 50 ohms. The amplitude goes down slightly as the frequency goes up.

The format of the command string from the PC is the same as for the "F" command above.

For example: To program the analyzer to a constant frequency of 7.100000 MHz, the command would be: "G048B5396". The analyzer **does not** respond with any data to the PC. This frequency can be controlled by a user interface implemented on the PC so the frequency can be changed in increments as small as 1 Hz. The analyzer output continues indefinitely until the PC turns it off with a "K0" command (opening the relay).

J – average data values. This letter command is followed by a single character with a value N= 1 to 16. The AIM4170 controller will take N ADC readings at each frequency point, add them all together and return the sum to the PC. This reduces the noise in the data. Typically N=4, 8, 16. N=0 or 1 turns off averaging.

K – relay command. This controls the relay that is in series with the output.

Format: "K3" = close the relay prior to a test. This also turns on both DDS chips.

("K1" closes the relay but it only turns on one DDS chip in order to reduce power consumption when the analyzer is being used as a constant frequency signal source. Impedance readings cannot be obtained unless both DDS chips are on.)

"K0" = open the relay after a test.

For example:

WriteFile(hcom,"K3",2,written,0) ; // close the relay, Red LED comes on
Delay(100) ; // delay 100 msec after closing the relay before subsequent operations.

WriteFile(hcom,"K0",2,written,0) ; // open the relay, Red LED goes off
(no delay is necessary after opening the relay)

The simple K0 and K3 commands can be sent to the AIM4170 by a terminal emulation program, such as Hyperterminal, to see if the comm port is functioning.

When a rapid series of measurements will be done, such as scanning a band of frequencies, the relay is closed prior to the first measurement and it **remains closed**

throughout the scanning process. After the scan is complete, the relay should be **programmed open**.

It is very important that the relay be OPEN when the analyzer is not taking a measurement or being used as a signal source. This helps to protect the input circuitry from static electricity. The red LED on the analyzer front panel will be turned on or off by the controller when the relay is closed or open.

N – Band scan mode – several characters follow to set the start, stop and delta frequencies. The AIM returns a value corresponding to the measured signal strength and waits for a confirmation from the PC after each measurement.

P – enter programming mode. The controller will execute a section of its program that can download a new controller program from the PC using the Reprogramming utility supplied with the AIM4170. This process uses the RS232 port.

Q – turn the power off. The AIM4170 is turned off completely. It can be restarted using the push-button power switch.

R – resend the last raw data that was collected by the analyzer. This can be used to retry a data transmission or for a diagnostic to retrieve the results of an earlier measurement *without* executing a new measurement. The 'R' is sent by itself, with no frequency parameter. The AIM4170 responds with the raw data in the same format that it does when the 'F' command is used.

V – send program version. The analyzer responds with information about its present program version.

Command format:

```
WriteFile(hcom,"V",1,written,0) ; //  Send "V" command
```

Format of returned data:

First character is a non-zero value (1 byte) that is equal to the number of ASCII characters to follow.

Then the version string is sent. Each character is ASCII. This string can be saved in the PC and displayed if desired. The fields of the string are: version+ date+ time_of_day+@. The total number of bytes sent from the controller depends on the version code and other data, but it's always terminated by an @ sign.