AFC: Arduino Frequency Counter V1.0

By ON6MU, Guy Roels, (c)2012 (software)
Belgium

and VE7DXW, Alex Schwarz
(micro controller C code and counter preamp schematics)

MDSR homepage-> http://users.skynet.be/myspace/mdsr/
MDSR/DADP support group: http://groups.yahoo.com/group/mdsradio/
ON6MU homepage> http://www.qsl.net/on6mu
ARDUINO homepage: http://www.arduino.cc/

Freeware

Introduction

Arduino Frequency Counter Windows software

This program has been designed primarily for use with the Arduino chip and the Arduino C controller software.
In addition to being used with the Arduino chip and the Arduino C controller software, the Arduino Frequency Counter can also be used in conjunction with the MDSR software to display the measured frequency in the MDSR frequency window. In either case, an Arduino UNO controller PCB is required. The TTL counter input is located on this PCB on digital input pin 5. To be able to measure non TTL signals, a preamp is required (schematics are included in this document).
Note: An Arduino UNO controller with a clock speed of 16MHz will be able to measure frequencies of up to 8MHz at a 50-50 on-off cycle. For higher frequencies, a prescaler must be used.

Here are some of the functions of the AFC:

- reading the frequency coming from the Arduino chip
- defining up to four Arduino command buttons
- entering any Arduino command manually for testing
- setting text and background color of the display
- switching between Hz, kHz, MHz readings
- changing the polling interval, com-port, baud rate
- adding a calibration factor and offset frequency
- scanning to verify the AFC is running/reading
- viewing the raw data from the Arduino chip with a 3 line buffer
- measuring and displaying the frequency in the MDSR software

The AFC also features a hold button that stops the measurement refresh and holds the last reading.

All settings are saved in AFC.INI

INSTALLATION INSTRUCTIONS:

Install the Arduino drivers first, then connect the controller to the computer and load the frequency counter code on the Arduino chip (as explained in this document).
Run AFC.exe and follow the instructions. Select the COM port where the Arduino chip is connected. The baud rate has to be set to 9600.

SETUP:

Clicking “Setup” will provide access to the following settings:

COM Port
Choose the COM port where the Arduino chip is connected. If the COM port is not known it can be found in the devise manager of the Windows software under “ports”.

BAUDRATE
Change the communication speed with the Arduino chip to 9600 baud.

POLL
Typically, the poll rate should be set to 150ms. Some systems can experience timing issues; timing may need to be adjusted to obtain the best readability of the frequency display.

CORRECTION
The correction field allows the user to correct the frequency readout due to variances in the Arduino clock frequency. This value can be obtained by measuring a known frequency and dividing it with the displayed frequency. (expected values 0.900000 to 1.100000)

$$\frac{\text{Known\_Accurate\_Frequency}}{\text{Arduino\_Displayed\_frequency}}$$

If the correction is already set in the Arduino code and the readout is accurate, the value has to be set to 1. **Note: When entering the compensation value the decimal has to be a "." and not a ",".**

OFFSET
The offset field allows the user to add an offset to the measured frequency. The value of the offset can either be a positive or negative number. It can be used to compensate for IF frequency offsets.

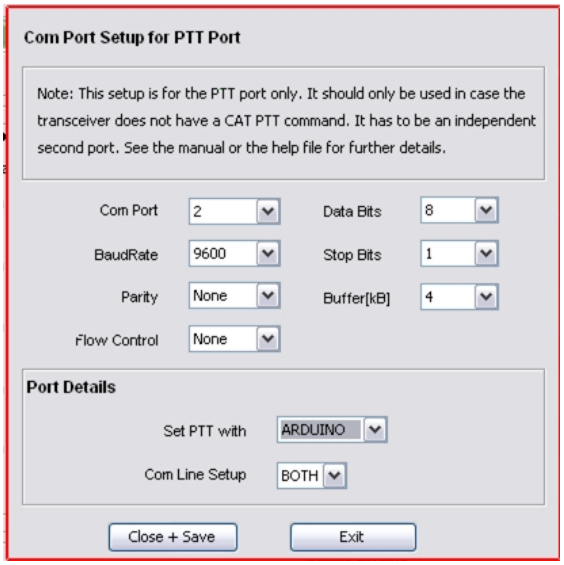
DISPLAY:

Stay on Top
Selecting this option will allow the frequency counter window to stay in front of all other windows that are on the screen.

Format Frequency
The frequency display is rounded when using kHz or MHz settings; if unchecked the raw input format is displayed.

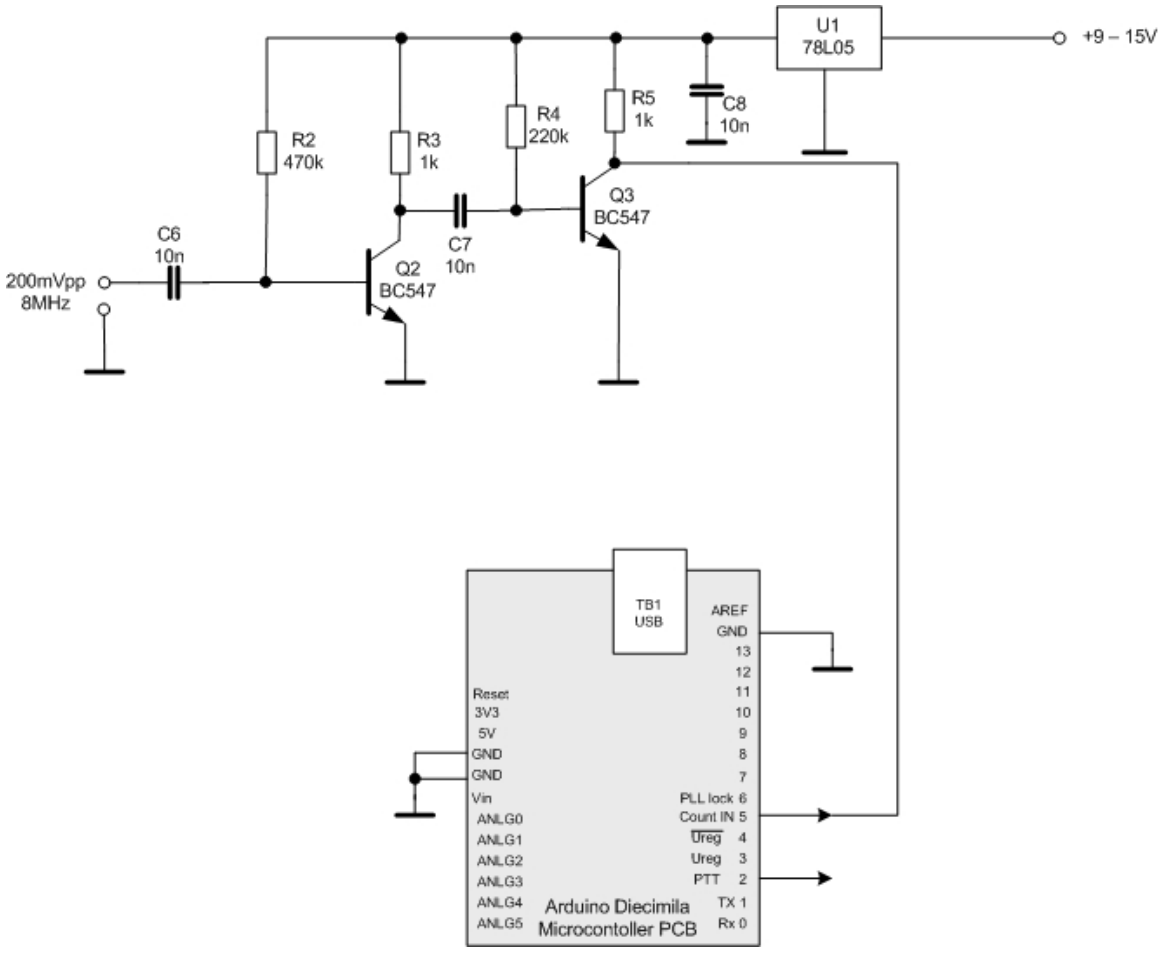
Displaying the Frequency in the MDSR software

The serial frequency data from the Arduino can be displayed in the MDSR frequency window. The MDSR does not provide any frequency correction or IF offset functionality. All these settings are stored in the Arduino chip and have to be set up correctly before the frequency display will be accurate and the IF shift is taken into account to display the receiving frequency.



After the Arduino controller has been connected to the computer open up the “PTT port setup” window in the MDSR software. Select the COM port of the Arduino controller (make sure that nothing else is connected to the controller). In “Port Details”, in the “Set PTT with” field, select “Arduino”. Click on “Close + Save” and restart the MDSR software.
Now the frequency display needs to be rerouted by the MDSR-SA software. Click on the “Wrench” icon of the MDSR-SA window and check the “Arduino Frequency Counter Mode” selection. Now the MDSR will display the frequency count of the Arduino controller.
Note: If the Arduino port is inactive, the “Arduino Frequency Counter Mode” will be greyed out.

Preamplifier and Arduino connection Schematics



Setting up the Arduino editor to compile and transfer the Frequency Counter code

All the files needed for the AFC and Arduino frequency counter setup, including the source code, can be found in the Arduino directory under the MDSR folder. The Arduino editor and the drivers can be downloaded from http://www.arduino.cc/. (size 85MB).

During the installation of the Arduino editor version 23, the following directory is created:

C:\Program Files\arduino-0023\libraries\

- Add a directory to the libraries folder called “FreqCounter”.
- Place the two files “FreqCounter.h” and “FreqCounter.cpp” into this folder (When the source code compiles, these files are necessary).

The project file is located in the following directory:

…\My Documents\Arduino\FreqCounter\

- Here, a FreqCounter folder will need to be created. Place the FreqCounter.pde file into it.
- Using the Arduino editor, open the FreqCounter.pde file and the source code will be visible .
- Press “ctrl R” to compile the code. If the setup is correct, the green bar below the editor window will read “Done compiling” and the black text area it will display the sketch size. If a problem occurs while compiling the code, an error message will be displayed. Review installation and ensure that all files are in the correct locations.
- If everything is in order, connect the Arduino PCB.
- Select the board and the serial port in the “Tools” menu. For “Board” select “Arduino UNO” and in “Serial” select the appropriate COM.
- To upload the firmware into the chip, press “ctrl U”. The TX and RX LED will light up as the data is transferring.
- To test the successful data transfer, go to the serial monitor by pressing “ctrl Shift M”. Make sure it selects the same port where the UNO resides. When the window comes up, the frequency data can be seen scrolling down. It will be 0 if no TLL signal is connected to input 5.

ARDUINO COMMANDS

The Arduino software will run as a frequency counter immediately after programming. But in order to calibrate and to make the software more flexible, a control routine set has been developed. The programming can be done either though the Arduino editor serial monitor or the AFC Arduino commands interface. It is important that the setup is done one step at a time because all values get stored in EEPROM memory and resetting the Arduino will not erase the values if they have not been entered properly. Sending a “V” to the Arduino will return all the programmed values from the Arduino to the serial monitor. Sending an “X” to the controller will return the frequency counter routine. **Note: All control characters are capitals.**

Commands:

Calibration: Command “C” – string length: 7 numeric digits
The controller returns “Calibration:” and waits for the calibration value. This calibration value has to be a 7 digit numeric value which is multiplied with the raw count to compensate for the local oscillator error. Without this calibration, the counter could be out as much as 2%. A frequency counter or a stable time base is needed to perform this correction. Before proceeding, set the calibration to 1 by entering “C send” and after the controller returns, “Calibration:” 1000000 (1 with six zeros) and then “send” again. If the controller returns an error send “X” to clear the error and the controller will return to the counting routine.

The correction value can be derived with the following formula:

$$\frac{\text{Known\_Accurate\_Frequency}}{\text{Arduino\_Displayed\_Frequency}}$$

Only type the first 7 digits from the left and leave out the decimal point, but include the 0 (zero) incase the value is smaller than 1. (e.g.: 0.99876348 → 0998763) After the number has been sent to the controller it will echo it back in 10s or, if no number has been entered, it will time out. This will set the calibration factor automatically to 1.000000. If an error condition persists, send a “C” and then the right correction string (7 numeric digits) as described earlier. Send “X” to start the counting routine again. Send “V” to check all the values stored in memory.
Now compare the two displays. If a difference remains, it can be fine-calibrated by entering a slightly higher or lower value. An accuracy of about 5Hz is obtainable.

Upper Limit: Command “U” – string length: 6 numeric digits
The upper limit specifies the highest frequency value the controller will return.
Type a capital “U” into the text field next to the send button and press “enter”. Now the frequency data will stop and you should see a line stating: “Upper Limit [kHz]:”
The processor expects a string with 6 numbers. For 10MHz enter “010000” which is 10000kHz. 4MHz would be “004000”. After ten seconds the controller times out and sometimes it returns erroneous data. But, if done correctly, it will program the controller accordingly and return the proper value. Once the string is sent across, it will take 10s until it echoes the value back for confirmation. If it times out or returns bad data, press capital “U” again and it will restart the setup. Enter a capital “X” into the send string field and the unit will return to being a frequency counter.

Lower Limit: Command “L” – string length: 6 numeric digits
The lower limit is the smallest number the controller will return. Upper and lower limits can be used to define a frequency band.
Type a capital “L” into the text field next to the send button and press “enter”. Now the frequency data will stop and you should see a line stating: “Lower Limit [kHz]:”
The same conditions apply for “Upper Limit”.

IF Offset: Command “I” – string length: 6 numeric digits
The IF Offset defines the value that is added or subtracted to match the oscillator frequency with the real RX frequency in case the oscillator is used as a mixing frequency in a heterodyne receiver.
Type a capital “I” into the text field next to the send button and press “enter”. Now the frequency data will stop and you should see a line stating: "IF offset [kHz]:"
The same conditions apply for “Upper Limit”.

IF Offset Direction: Command “D” – string length: 1 numeric digit

The offset direction allows the user to set the LO Offset to positive (1) or negative (0). If this value is set to “0” the IF Offset will be subtracted from the frequency count, if set to “1” the IF Offset will be added.
Note: In case the Offset value is bigger than the frequency count and the Offset direction is set to negative (0), the controller will return an error. It won’t display a negative frequency.
Type a capital “D” into the text field next to the send button and press “enter”. Now the frequency data will stop and you should see a line stating: "IF direction [- = 0, + = 1] :"
Enter the needed value to match the LO to the RX frequency.

Frequency Step Rate: (for future release)

Prescaler: Command “P” – string length: 2 numeric digits
The Prescaler value allows multiplying the frequency display to accommodate a prescaler. Since the hardware prescaler circuit divides the measured frequency, the controller has to multiply the actual count to match the counter display with the actual input. The prescaler values can be from 1 (no prescaler) to 99. To set the prescaler to 1, type in ”01”. The sent string has to be 2 digits in length (01,02,03….09,10,11…99).

PTT output: Command “T” to set the digital pin #2 into the low state (0V) and command “R” to return it to the high state (+5V). These control strings can also be linked to the PTT button in the MDSR software, if the PTT Port setup is set to use the “Arduino” for PTT.

Using the AFC program to control the Arduino Frequency counter

Control commands can also be set up in the AFC program. The AFC has four user defined buttons. By selecting the control command for each button, the Arduino controller will return the appropriate values as requested without having to use a serial monitor.
Up to 4 commands in pre-defined buttons can be set up for easy access. Use the right-mouse button to edit each button. An Arduino chip command can also be sent by typing it into the text field and pressing the “OK” button.

Raw Data Display:
The Raw Data Display shows the unformatted raw data coming from the Arduino chip with a 3-line buffer.

Resetting the AFC

By deleting the AFC.INI file and re-starting the AFC.EXE, the program is reset to default values.

LEGAL INFORMATION:

AFC and the Arduino counter code are NOT public domain, but they are free to use. There is no registration and no payment. The author keeps the copyright and all other rights.
Permission to sell AFC or the Arduino counter code for profit is prohibited. Please contact the authors for permission.
Any support you can provide for the projects and modules are greatly appreciated!
If you decide to make a contribution to the MDSR Project, please use the PayPal account at the website http://www.qsl.net/on6mu/

Please note:

It is prohibited to:

- Modify or patch the program, or in any way disassemble or change anything.
- Distribute the program without all the matching files.
- Distribute the program under another name other than AFC*.* (only the extension (*.ARJ, *.LZH, *.ZIP and others) may vary.)
- Ask for money other than the real costs of transport or postage.
- Ask for money for the program, part program, or copy of the program (under this concept, the 5-dollar-a-disk principle is allowed).

Also note:

By using this program, you accept these conditions.
The authors accept no responsibility for any unexpected negative outcomes resulting from the improper use of this program.
It is the users' responsibility to ensure that all local laws and requirements are met.

Conclusion

Thanks for using AFC! Please feel free to distribute this document and tell all your friends about our development and group. Suggestions, bug reports, tips and comments are very much appreciated and can be posted on the MDSR Yahoo user group forum.

For the latest version please visit the MDSR/DADP homepage:
http://users.skynet.be/myspace/mdsr/
http://groups.yahoo.com/group/mdsradio/

Thank you for your time and interest,

Guy, ON6MU
Alex, VE7DXW
and the MDSR Team