Sparton Inertial Systems

AHRS-M2 Quick Start Guide

Version 1.0

*Applicable to Sparton **AHRS-M2***

5612 Johnson Lake Rd • Deleon Springs, FL • 32130
800.824.0682 • 386.985.4631
www.sparton.com/ne • www.spartonnavex.com

Quick Start

This quick start is intended for use by someone who wishes to obtain data from the inertial system. This document provides examples of the NorthTek protocol as it is the easiest to begin programming with the **AHRS-M2** Sparton Sensor.

The documents referenced in this Quick Start may be found on the inertial system CD or https://spartonnavex.com/technical-support/downloads/.

The inertial system supports two protocols where one is binary and one is ASCII. The binary protocol is RFS (Remote Function Select) and the ASCII protocol is NorthTek.

This Quick Start provides an example of the method to get processed 3-axis magnetometer, accelerometer and gyroscope information as well as pitch, roll, and yaw.

Prerequisites

It is assumed that the instructions in the Spartacus Quick Start Guide have been followed to set up the inertial system hardware such that it can communicate with the Spartacus Host Application. A terminal emulator is required to follow this Quick Start procedure. See Appendix A for set-up examples or use the terminal emulator that is built into the Spartacus Host Application. This Quick Start will assume that the Spartacus Host Application Terminal Emulator is being used but other terminal emulators could be used as well.

NorthTek Protocol

NorthTek is a programming language based on ANSI Forth. With NorthTek, the user has a lot of control in how the inertial system outputs its data using interpretive command line type instructions or by creating text files that define a NorthTek program script. NorthTek is case sensitive and lines are terminated with a carriage return.

Mask Outputs (Easy Access to Pertinent data)

There are a few commands that will enable the user to get quick and easy output data from the compass. We call these the mask variables. At 115200 baud, each character takes about 0.1 ms which limits output to about 100 characters at 100Hz.

- compass_mask - Data is time (ms), pitch (deg), roll (deg), yaw (deg)
  - *boresightMatrix is applied to this output*
- magp_mask - Data is time (ms), magp X, Y, Z (mGauss)
  - *boresightMatrix is NOT applied to this output*
- accelGyrop_mask - Data is time (ms), accelp X, Y, Z (mg), gyrop X, Y, Z (rads/sec), temperature (degC)
  - *boresightMatrix is NOT applied to this output*

For each of these mask commands there will be a *d.on* command to turn on the output and a *d.off* command to turn off the output. To simply stop the streaming to be resumed later, simply send the CTRL-S command. To restart the streaming, send the CTRL-Q command.

Enter:

accelGyrop_mask d.on

Response example: [AGp,time_ms,accelp_X,accelp_Y,accelp_Z,gyrop_X,gyrop_Y,gyrop_Z,temp]

AGp,154261571, 171.02, 618.89,-751.83,2.425e-02,2.691e-02,-9.859e-03, 26.55

Enter:

compass_mask d.on

Response example: [C,time_ms, pitch_deg,roll_deg,yaw_deg]

C,11590797, 17.890,-43.611,231.619

Enter:

magp_mask d.on

Response example: [magp,time_ms,magp_X,magp_Y,magp_Z]

magp,11636044, -415.8, 192.3, 425.0

NorthTek Display Command Word "di."

The "di." command can be used to display any of the variables listed in the long list of available variables (**reference**: the *Variable Summary Table* and the *Variable Detailed Descriptions* table in the ***AHRS-M2*** *Software Interface Control Document - ICD*).

This command will tell you the date the code was compiled on!

Enter:

VERSION di.

Response example:

VERSION = $ Version: 1.1.10
OK

NorthTek Display Command Word "di.choice"

The "di.choice" command can be used to display any of the variable choices listed in the long list of available variables. The number after the text in the [x] is used to set the variable.

Enter:

clearFieldCal di.choice

Response example:

ClearDone[0]
ClearCmd[1]**OK**

The correct command sequence to set the ***clearFieldCal*** to ***ClearCmd*** is shown below.

Enter:

clearFieldCal 1 set drop

BitStream ASCII

Reference: *Custom Combination Print Streaming* section in the *IMU-10 Software Interface User's Manual.*

This capability allows the user to select a custom combination of sensor data. In this example, suppose we want to have cputime, processed magnetometer/accelerometer/gyroscope.

Note that, the "drop" is needed to clear the status result of the "set" off of the stack. **Reference**: *NPL Basics* section in the *NorthTek System Programming Manual*

This will enable the user to output the cputime, processed gyro and accel data at 100Hz.

```
// This line sets clears all the output enables by setting them to zero
chan0Enables m[ 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]m set drop

// This line sets the output preamble to "x:"
chan0Preamble " x:" set drop
```

```
// This line sets the output postamble to ";"
chan0Postamble " ;" set drop

// The lines below use the appropriate variable names to be turned on in the
output stream
// These lines will be output in their VID numbered order
chan0EnableBit cputime dvid@ set drop // VID == 83
chan0EnableBit accelp dvid@ set drop // VID == 31
chan0EnableBit gyrop dvid@ set drop // VID == 33

chan0Format 3 set drop // Sets the output stream to BitstreamAscii
chan0Trigger 5 set drop // Sets the Trigger to CompassData
chan0TriggerDivisor 1 set drop // Sets the divisor to (ODR / TriggerDivisor)
```

Response example (repeats):

```
x:112.914749,25.390560,1001.462280,2.398158e-03,1.012556e-02,-2.931082e-03,181241,;
```

In the above response, the "x:" identifies the type of output. It is followed by the timestamp (milliseconds since power up), and the variable data is in VID ordered from lowest number to highest.

Note that if we print high rate data at a slower baud rate, it may not be possible to output all of the data due to the serial communication bandwidth limitations. In this case, entire lines of data will be dropped to ensure that the lines that do get output are complete. The data will still be computed at the normal rate within the inertial system.

To stop the streaming output, enter:

```
<ctrl-s>
chan0TriggerDivisor 0 set drop<cr>
<ctrl-q>
```

BitStream Ascii Prefix and Postfix

The BitStreamAscii format prepends a custom prefix and appends a custom postfix to each line of data. The above example shows the default prefix of "$CUS0" and the default postfix of "*". These can be customized by setting the string variable chanNPreamble and chanNPostamble. Each of these has an 80 character limit. Here is an example to change them for channel 0 (there must be a space after the first quote for any strings):

```
chan0Preamble " Accels:" set drop
chan0Postamble " ;" set drop
```

Response:

```
Accels:16.522532,-10.340560,1026.988770,30.937500,;
```

```
Accels:18.240021,-14.073236,1024.551147,30.875000,;
```

The BitStreamAscii format can be customized to output a CRC or checksum. See Appendix C for source code containing the CRC and Checksum used.

To add a CRC, include the character '\0x01' in the chanNPreamble which indicates the start of the CRC. To mark the end of the CRC'd bytes, include the character '\x02' in the chanNPostamble.

To select a checksum, include the character '\0x03' in the chanNPostamble and to select a CRC, include the character '\0x04' in the chanNPostamble. The CRC or checksum will appear in the place of the '\0x01'. For example:

```
// Show the use of embedded format characters to put CRC/checksums in user
output

// This one does checksum

// specify where to start the checksum
chan0Preamble "$\x01CUS0," set drop

// specify where to end checksum and where to put it
chan0Postamble "\x02*\x03" set drop

chan0Format 3 set drop
chan0Trigger 4 set drop

// This one does CRC

// specify where to start the CRC
chan1Preamble "$\x01CUS1," set drop

// specify where to end CRC and where to put it
chan1Postamble "\x02*\x04" set drop

chan1Format 3 set drop
chan1Trigger 4 set drop

// Demonstrate a test

// Clear any previous data selections for channels 0 and 1
chan0Enables array[ 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]array set drop
chan1Enables array[ 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]array set drop

// Select the data for channel 0
chan0EnableBit accelp dvid@ set drop
chan0EnableBit cputime dvid@ set drop

// Select the data for channel 1
chan1EnableBit gyrop dvid@ set drop
```

```
chan1EnableBit cputime dvid@ set drop

// Slow down the output to 2 Hz (100 Hz/50 = 2 Hz)
chan0TriggerDivisor   50 set drop
chan1TriggerDivisor   50 set drop
```

Example output from above script:

```
$CUS0,2.612818e-01,-1.617098e-02,10.077161,3177458,*63
$CUS1,1.715285e-02,2.853286e-02,-4.727963e-02,3177478,*C4B4
$CUS0,2.249641e-01,2.542049e-03,10.094501,3177958,*49
$CUS1,2.079850e-02,-5.304283e-03,-3.368925e-02,3177978,*F0FF
…
```

BitStream NMEA

The Bit Stream NMEA format allows the user to set up a trigger to periodically invoke an inertial sensor implemented NMEA command. A NMEA command can be hooked up to a Bit Stream channel by setting the corresponding database string variable chanNCommand where N is 0, 1 or 2. The period is defined by the trigger for the Nth channel allowing the user to process freshly computed data. For example (note space after first quote):

```
chan1Trigger 4 set drop       // trigger when conditioned data ready
chan1TriggerDivisor 100 set drop  // output at 100 Hz/100  = 1 Hz
chan1Format 4 set drop    // select NMEA protocol
chan1Command " $PSPA,A\n" set drop
```

To stop the stream (must include **CRLF** after drop)
chan1TriggerDivisor 0 set drop

BitStream Binary

Reference: *Custom Combination Print Streaming* section in the *Software Interface User's Manual.*

For BitStream Binary, the output values are packed binary and are sorted in order of VID number. Floats are represented by the IEEE-754 32-bit format. It is up to the user to be aware of the size of each object being packed into the packet. Ordinals consume 1 byte, Int32's and Float32's take 4, arrays take multiples of 4, etc. The ordering of the data can be determined as described above using the "chanNEnables" command. The data is sent in a SAPP packet using a protocol identifier of 9 (see Appendix E Standard Asynchronous Packet Protocol in the RFS Protocol Suite document). The first byte of the SAPP packet payload is the channel number.

This will enable the user to output the processed gyro and accel data at 100Hz.

```
// This line sets clears all the output enables by setting them to zero
chan0Enables m[ 0 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]m set drop

// The lines below use the appropriate variable names to be turned on in the
output stream
// These lines will be output in their VID numbered order
chan0EnableBit cputime dvid@ set drop // VID == 83
chan0EnableBit accelp dvid@ set drop // VID == 31
chan0EnableBit gyrop dvid@ set drop // VID == 33

chan0Format 2 set drop // Sets the output stream to BitstreamBinary
chan0Trigger 5 set drop // Sets the Trigger to CompassData
chan0TriggerDivisor 1 set drop // Sets the divisor to (ODR / TriggerDivisor)
```

The array output will be sent as an RFS packet:

```
01 20 69 00 5A 4B E3 42 E1 85 BA 41 76 10 95 79 44 E4 8D 51 BB A9 E5 25 3C 2C 6F 94 BB 0B 30 0E 00 B7 CB 03
01 20 69 00 D9 F4 E6 42 E1 8B B9 41 26 4C 79 44 E4 8D 51 BA 88 FE 32 3C E4 8D 51 BB 10 95 30 0E 00 09 82 03
01 20 69 00 5A C8 E3 42 E0 6D BE 41 96 E6 78 44 E4 8D D1 3A 88 FE 32 3C 61 04 63 BB 1F 30 0E 00 BD CB 03
01 20 69 00 59 BC E5 42 E2 B5 B2 41 E6 2C 79 44 E4 8D 51 3A 49 43 2A 3C AE F8 82 BB 29 30 0E 00 27 2B 03
01 20 69 00 5A 51 E2 42 E2 BB B1 41 56 44 79 44 6B 2A 1D 3B 88 FE 32 3C E4 8D 51 BB 33 30 0E 00 89 B8 03
01 20 69 00 DA 95 E1 42 E0 6D BE 41 46 1D 79 44 66 17 40 3B 27 5C 37 3C E4 8D 51 BB 3D 30 0E 00 CE E7 03
01 20 69 00 5A 45 E4 42 DF 55 C2 41 46 1D 79 44 66 17 40 3B 88 FE 32 3C 61 04 63 BB 47 30 0E 00 A5 C3 03
01 20 69 00 D9 77 E6 42 E0 73 BD 41 76 10 95 79 44 A0 BF EB 3B E8 A0 2E 3C DF 7A F4 BA 51 30 0E 00 3C 4F 03
01 20 69 00 59 BC E5 42 DE 2B C9 41 96 E6 78 44 DF 7A 74 3B E8 A0 2E 3C 66 17 40 BB 5B 30 0E 00 6C 9D 03
01 20 69 00 DA 00 E5 42 DF 43 C5 41 D6 05 79 44 ED B3 8B 3A 88 FE 32 3C 66 17 40 BB 65 30 0E 00 10 83 CD 03
01 20 69 00 5A CE E2 42 E1 8B B9 41 A6 0D 79 44 E4 8D D1 BA 49 43 2A 3C E4 8D 51 BB 6F 30 0E 00 39 A0 03
```

Note that if we print high rate data at a slower baud rate, it may not be possible to output all of the data due to the serial communication bandwidth limitations. In this case, entire lines of data will be dropped to ensure that the lines that do get output are complete. The data will still be computed at the normal rate within the inertial system.

To stop the streaming output, enter:

```
chan0TriggerDivisor 1 set drop
```

NMEA

Reference: *NMEA* section in the *Software Interface User's Manual* for an introduction.

The acronym NMEA is used here to represent NMEA 0183 which is a communication standard for marine electronic devices. This standard is defined by the National Marine Electronics Association.

NMEA Set-up

Turn on key stroke "local echo" if you are using a terminal emulator **OR** issue the following command (once per session):

```
nmeaecho 1 set drop
```

If you are using a terminal emulator other than the one built into the NDS-2 Host Application, set up the terminal emulator to transmit a carriage return and line feed (i.e. <cr><lf> or hex 0d0a) upon hitting enter **OR** some terminal emulators like Tera Term will do so if you type <ctrl-j>.

Standard NMEA command ($xxHDM):

There is a 5 second timeout on entering each character in a NMEA command to prevent the inertial system from getting stuck in the protocol.

To get heading using the standard NMEA command, enter:

```
$PSPA,M
```

Response example:

```
$PSPA,Mx=25.7,My=238.2,Mz=110.0,Mt=263.6*27
```

The 2E is the checksum and the 'M' indicates magnetic heading. **Reference**: *NMEA* section in the *Software Interface User's Manual* for details on interpreting the response and for other NMEA protocol commands.

To get repeated values every 0.1 seconds, enter the command:

```
$xxHDM,RPT=0.1
```

To pause the output, enter:

```
<ctrl-s>
```

Enter any NMEA command to cancel the repeat, for example:

```
$xxHDM
<ctrl-q>
```

If you forget to enter the <ctrl-q>, then the next streaming command won't output.

Sparton Custom NMEA command:

A NMEA command has been created that allows access to almost any of a long list of available inertial system variables (**reference**: the *Variable Summary Table* and the *Variable Detailed Descriptions* table in the *Software Interface User's Manual*). For our processed gyro readings example, the corresponding variable name is "gyrop".

SETUP for this example should follow **NMEA Set-up** listed above:

Send Example (*some ENTER keys are not setup properly for this example*):

```
$PSRFS,gyrop,get // Must use a linefeed CTRL-J
```

Response example:

```
$PSRFS,gyrop,-0.005596,0.003464,-0.010392*2A
```

Appendix A

Tera Term setup:

To set up Tera Term install the product into whichever location you wish. Open Tera Term to start a session. Go to ‘Setup’ and select ‘Serial Port’.

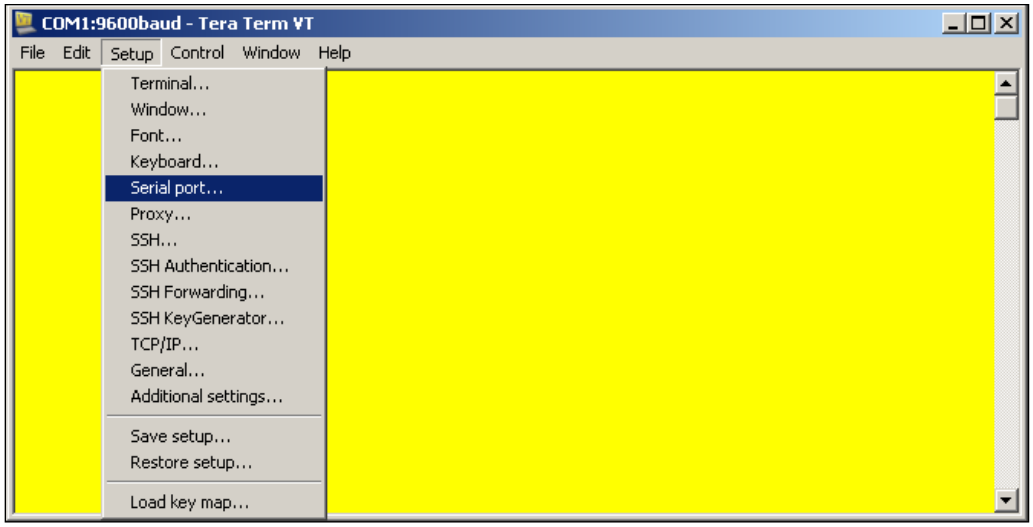

Figure 1 - Main Tera Term Window

Once the window in Figure 2 appears, select the appropriate baud rate for the product. Make sure the ‘Data’, ‘Parity’, ‘Stop’, ‘Flow control’, and ‘msec/line’ fields all match the values in the Figure 2.

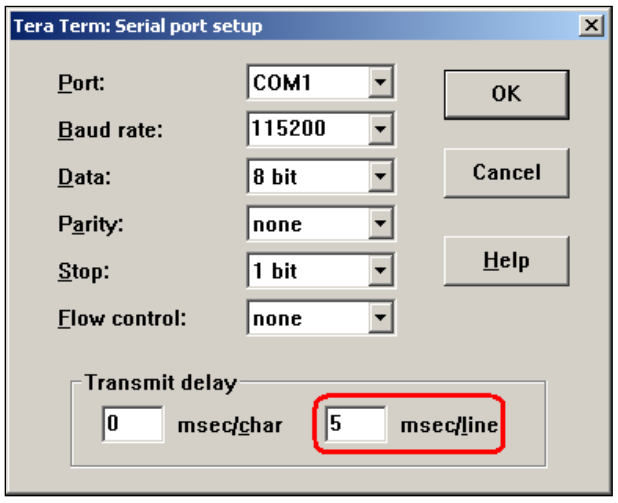

Figure 2 - Tera Term Serial port setup Window

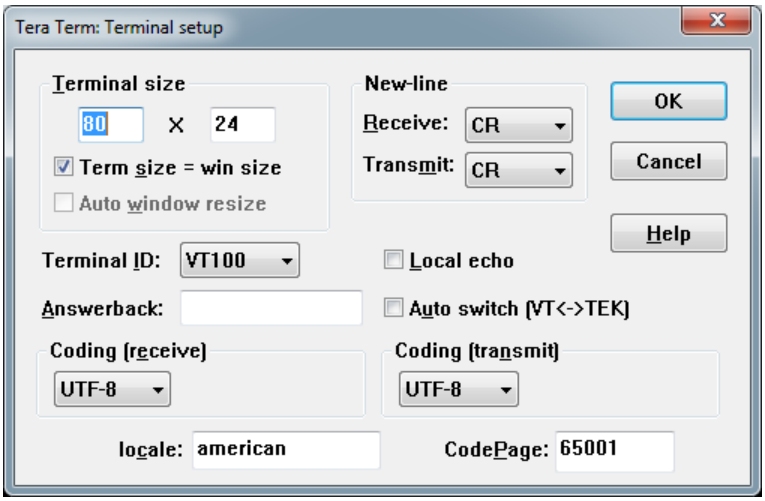

Figure 3 -Tera Term Serial port setup Window

HyperTerminal Setup:

To set up HyperTerminal install the product into whichever location you wish. Open HyperTerminal to start a session. Go to 'File' and select 'Properties'.

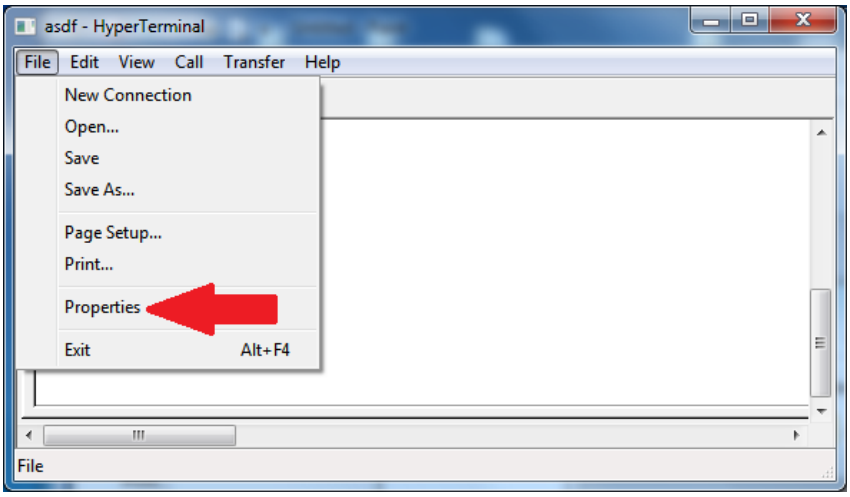

Figure 4 - Main HyperTerminal Window

When Figure 4 appears navigate to the 'Settings" tab and click the 'ASCII Setup' button.

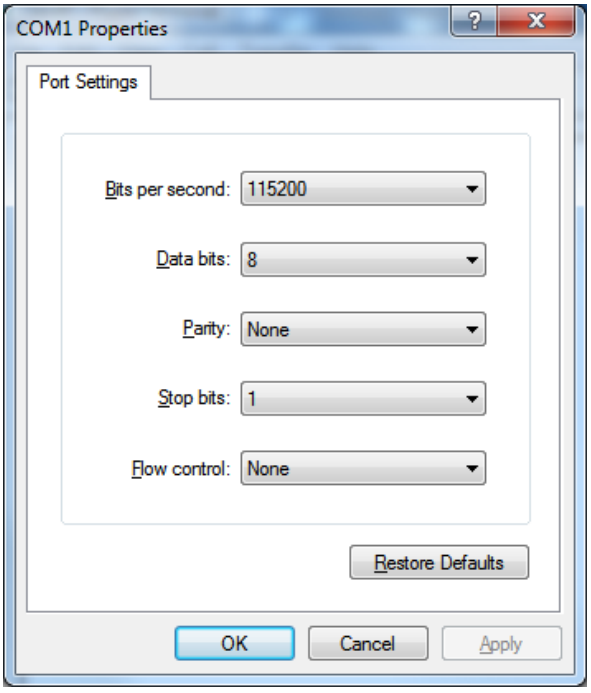

Figure 5 - COM Properties

When Figure 5 appears navigate to the 'Settings" tab and click the 'ASCII Setup' button.

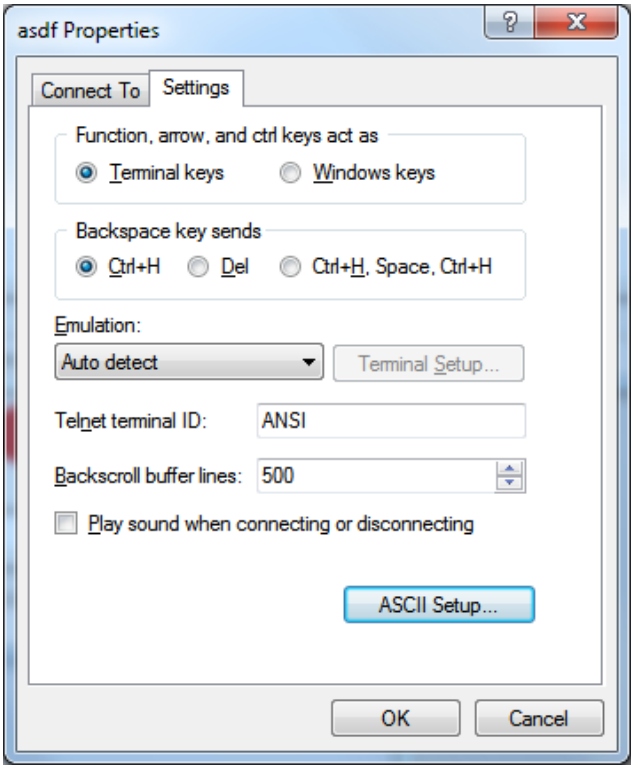

Figure 6 - HyperTerminal Properties Window

When the window in Figure 6 appears, set the 'Line delay' field to 5 'milliseconds'.

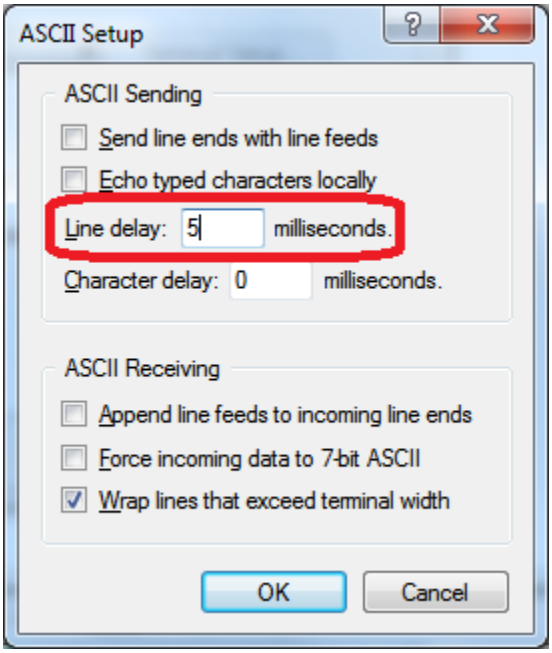

Figure 7 - - HyperTerminal ASCII Setup Window