USBee AX Test Pod Users Manual

CWAV
www.usbee.com

USBee AX Test Pod
Users Manual

CWAV
www.usbee.com
(951) 693-3065
support@usbee.com

USBee AX License Agreement

The following License Agreement is a legal agreement between you (either an individual or entity), the end user, and CWAV. You have received the USBee Package, which consists of the USBee Pod, USBee Software and Documentation. If you do not agree to the terms of the agreement, return the unopened USBee Pod and the accompanying items to CWAV for a full refund. Contact support@usbee.com for the return address.

By opening and using the USBee Pod, you agree to be bound by the terms of this Agreement.

Grant of License
CWAV provides royalty-free Software, both in the USBee Package and on-line at www.usbee.com, for use with the USBee Pod and grants you license to use this Software under the following conditions: a) You may use the USBee Software only in conjunction with the USBee Pod, or in demonstration mode with no USBee Pod connected, b) You may not use this Software in conjunction with any pod providing similar functionality made by other than CWAV, and c) You may not sell, rent, transfer or lease the Software to another party.

Copyright
No part of the USBee Package (including but not limited to manuals, labels, USBee Pod, or accompanying diskettes) may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of CWAV, with the sole exception of making backup copies of the diskettes for restoration purposes. You may not reverse engineer, decompile, disassemble, merge or alter the USBee Software or USBee Pod in any way.

Limited Warranty
The USBee Package and related contents are provided “as is” without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, with the sole exception of manufacturing failures in the USBee Pod or diskettes. CWAV warrants the USBee Pod and physical diskettes to be free from defects in materials and workmanship for a period of 12 (twelve) months from the purchase date. If during this period a defect in the above should occur, the defective item may be returned to the place of purchase for a replacement. After this period a nominal fee will be charged for replacement parts. You may, however, return the entire USBee Package within 30 days from the date of purchase for any reason for a full refund as long as the contents are in the same condition as when shipped to you. Damaged or incomplete USBee Packages will not be refunded.

The information in the Software and Documentation is subject to change without notice and, except for the warranty, does not represent a commitment on the part of CWAV. CWAV cannot be held liable for any mistakes in these items and reserves the right to make changes to the product in order to make improvements at any time.

IN NO EVENT WILL CWAV BE LIABLE TO YOU FOR DAMAGES, DIRECT, INDIRECT, INCIDENTAL OR CONSEQUENTIAL, INCLUDING DAMAGES FOR ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE SUCH USBEE POD, SOFTWARE AND DOCUMENTATION, EVEN IF CWAV HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR FOR ANY CLAIM BY ANY OTHER PARTY. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU. IN NO EVENT WILL CWAV’S LIABILITY FOR DAMAGES TO YOU OR ANY OTHER PERSON EVER EXCEED THE AMOUNT OF THE PURCHASE PRICE PAID BY YOU TO CWAV TO ACQUIRE THE USBEE, REGARDLESS OF THE FORM OF THE CLAIM.

Term
This license agreement is effective until terminated. You may terminate it at any time by returning the USBee Package (together with the USBee Pod, Software and Documentation) to CWAV. It will also terminate upon conditions set forth elsewhere in this agreement or if you fail to comply with any term or condition of this agreement. You agree that upon such termination you will return the USBee Package, together with the USBee Pod, Software and Documentation, to CWAV.

Table of Contents

1 Introducing the USBee AX Pod

The USBee AX Test Pod is a PC-based programmable multifunction digital storage oscilloscope, logic analyzer and digital signal generator in a single compact and easy to use device. It is the ideal bench tool for engineers, hobbyists and students.

Connecting to your PC, the USBee AX Test Pod uses the power and speed of the USB 2.0 bus to capture and control analog and digital information from your own hardware designs. The USBee AX takes advantage of already existing PC resources by streaming data over the High-Speed USB 2.0 bus to and from the PC. This allows the PC to perform all of the triggering and data storing and makes possible an affordable USBee AX, while pushing the sample storage capabilities orders of magnitudes beyond that of traditional dedicated oscilloscopes, logic analyzers or signal generators. The USBee AX Test Pod can utilize available PC memory as the sample buffer, allowing selectable sample depths from one to many hundreds of millions of samples.

The USBee AX Test Pod can capture and generate samples up to a maximum of 24 million samples per second depending on the PC configuration. The USBee AX Auto-Calibration feature automatically reduces the sample rate to ensure accurate and reliable timing, even on systems with slower processor and USB bus speeds. The USBee AX Test Pod perfectly merged features and functions to provide exactly the performance needed for hardware and microprocessor designs such as BASIC Stamp and PIC systems to ensure an affordable and compact unit.

The USBee AX Test Pod does not need an external power supply. The USB bus supplies the power to the pod, so your PC will be supplying the power. The Pod does, however, require a self powered hub (not bus powered) if a hub is used between the PC and Pod.

WARNING

As with all electronic equipment where you are working with live voltages, it is possible to hurt yourself or damage equipment if not used properly. Although we have designed the USBee AX pod for normal operating conditions, you can cause serious harm to humans and equipment by using the pod in conditions for which it is not specified.

Specifically:

- ALWAYS connect at least one GND line to your circuits ground
- NEVER connect the digital signal lines (0 thru 7, TRG and CLK) to any voltage other than between 0 to 5 Volts
- NEVER connect the analog signal lines (CH1 and CH2) to any voltage other than between -10 and +10 Volts
- The USBee AX actively drives Pod signals 0 through 7 in some applications. Make sure that these pod test leads are either unconnected or connected to signals that are not also driving. Connecting these signals to other active signals can cause damage to you, your circuit under test or the USBee AX test pod, for which CWAV is not responsible.
- Plug in the USBee AX Pod into a powered PC BEFORE connecting the leads to your design.

The USBee AX is available in three configurations to balance your test needs with your budget. These are the AX-Standard, AX-Plus and AX-Pro. The following table shows which features come with each of these packages.

	USBee AX-Standard	USBee AX-Plus	USBee AX-Pro
Oscilloscope	✔	✔	✔
Logic Analyzer	✔	✔	✔
Mixed Signal Oscope/Logic Analyzer	✔	✔	✔
Digital Voltmeter	✔	✔	✔
USB (Low and Full Speed) Decoder		✔	✔
I2C Decoder		✔	✔
SPI Decoder		✔	✔
Async Decoder		✔	✔
Signal Generator		✔	✔
Data Logger			✔
Frequency Counter			✔
Remote Controller			✔
PWM Controller			✔
Frequency Generator			✔
I2C Controller			✔
Pulse Counter			✔
USBee Toolbuilder Source Code			✔

In this manual, features that operate with the various USBee AX pod types are highlighted with the following symbols:

AX-Standard, AX-Plus and AX-Pro

AX-Plus and AX-Pro Only

AX-Pro Only

The USBee AX system is also expandable by simply adding more USBee AX pods for more channels and combined features.

1.1 PC System Requirements

The USBee AX Test Pod requires the following minimum PC features:

- Windows® XP or Windows® 2000 operating system
- Pentium or higher processor
- One USB2.0 High Speed enabled port. It will not run on USB 1.1 Full Speed ports.
- 32MBytes of RAM
- 125MBytes of Hard disk space
- Internet Access (for software updates and technical support)

1.2 Each Package Includes

The USBee AX contains the following in each package:

- USBee AX Universal Serial Bus Pod
- Set of 14 multicolored test leads and high performance miniature test clips
- Getting Started Guide
- USB Cable (A to Mini-B)
- USBee AX Test Pod CD-ROM

1.3 Hardware Specifications

Connection to PC	USB 2.0 High Speed (required)
Power	via USB cable
Test Leads	14 9" leads with 0.025" square sockets
USB Cable Length	6 Feet
Dimensions	2.1" x 1.3" x 0.5"
Minigrip Test Clips	14

The maximum sample rate for any mode depends on your PC hardware CPU speed and USB 2.0 bus utilization. For the fastest possible sample rates, follow these simple steps:

- Disconnect all other USB devices not needed from the PC
- Do not run other applications while capturing or generating samples.

The maximum sample buffer size also depends on your PC available RAM at the time the applications are started.

1.4 Software Installation

Each USBee AX pod is shipped with an installation CD that contains the USBee AX software and manuals. You can also download the software from the software from our web site at www.usbee.com. Either way, you must install the software on each PC you want to use the USBee AX on before you plug in the device.

To install the software:

- Download the USBee EX 2.0 Software from http://www.usbee.com/download.htm and unzip into a new directory. Or insert the USBee AX CD in your CD drive. Unzip the downloaded file into a new directory.
- From the "Start|Run" Windows® menu, run the SETUP.EXE.
- Follow the instructions on the screen to install the USBee AX software on your hard drive. This may take several minutes.
- Now, plug a USB A to USB Mini-B cable in the USBee AX and the other end into a free USB 2.0 High Speed port on your computer.
- You will see a dialog box indicating that it found new hardware and is installing the software for it. Follow the on screen directions to finish the driver install.
- The USBee AX Software is now installed.
- Run any of the applications by going to the Start | Program Files | USBee AX Test Pod and choosing the application you want to run.

1.5 Calibration

Since electronic components vary values slightly over time and temperature, the USBee AX Pod requires calibration periodically to maintain accuracy. The USBee AX has been calibrated during manufacturing and should maintain accuracy for a long time, but in case you want to recalibrate the device, follow these steps. The calibration values are stored inside the USBee AX pod. Without calibration the measurements of the oscilloscope may not be accurate as the pod ages.

To calibrate your USBee AX Pod you will need the following equipment:

- External Voltage Source (between 5V and 9V)
- High Precision Multimeter

When you are ready to calibrate the USBee AX Pod, plug in the pod and run either the Oscilloscope or the Mixed Signal Oscilloscope application. Then go to the menu item Setup | Calibrate. You will be

asked to confirm that you really want to do the calibration. If so, press Yes, otherwise press No. Then follow these steps:

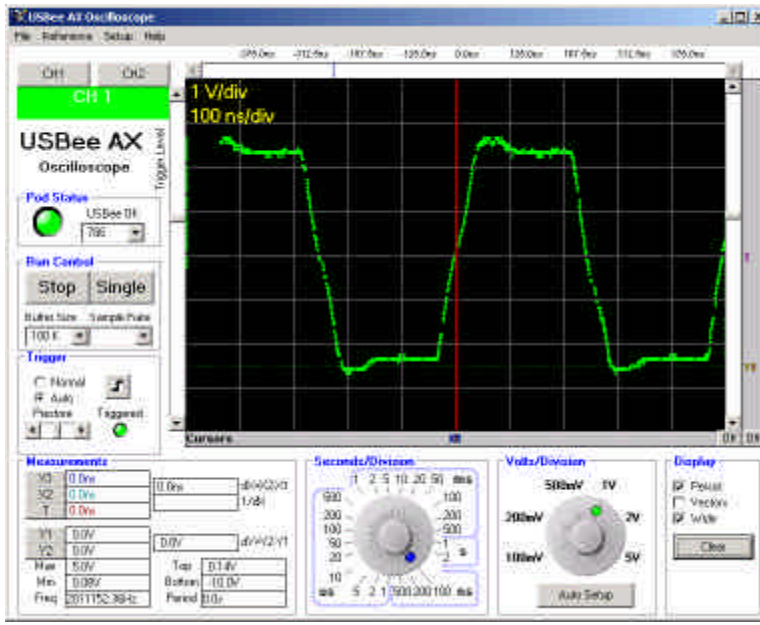
- Connect the CH1 signal to the GND signal using the test leads and press OK. A measurement will be taken.
- Connect the CH2 signal to the GND signal using the test leads and press OK. A measurement will be taken.
- Connect the GND signal to the ground and the CH1 signal to the positive connection of the External Voltage Source using the test leads and press OK. A measurement will be taken.
- With the Multimeter, measure the actual voltage between the GND signal and the CH1 signal and enter this value in the dialog box.
- Connect the GND signal to the ground and the CH2 signal to the positive connection of the External Voltage Source using the test leads and press OK. A measurement will be taken.
- With the Multimeter, measure the actual voltage between the GND signal and the CH2 signal and enter this value in the dialog box.
- The calibration is now complete. The calibration values have been saved inside the pod.

The analog measurements of your USBee AX pod are only as accurate as the voltages supplied and measured during calibration.

2 Oscilloscope

AX-Standard, AX-Plus and AX-Pro

This section details the operation of the Oscilloscope application that comes with the USBee AX. Below you see the application screen.



The USBee AX Oscilloscope functions as a standard Digital Storage Oscilloscope, which is a tool used to measure and display analog signals in a graphical format. It displays what the analog signal input is doing over time.

2.1 Oscilloscope Specifications

Analog Inputs	2
Analog Channels	1
Maximum Analog Sample Rate [1]	16 Msps
Analog Bandwidth	3 MHz
Input Impedance	1M Ohm/30 pF
Input Voltage Range	-10V to +10V
Analog Sensitivity	78mV
Analog Resolution	256 steps
Channel Buffer Depth [2]	>1 Million
Volts per Division Settings	100mV to 5V in 6 steps
Time per Division Settings	100ns to 2s in 23 steps
Trigger Modes	Auto, Normal, Single
Trigger Voltage	Between -10V and +10V
Cursors	2 Time and 2 Voltage
Voltage Display Offset	Up to maximum inputs
Time Display Offset	Up to available buffer depth
Trigger Position Setting	10% to 90%
Measurements	Min, Max, Top Bottom, Freq, Period
Reference Waveform	Save and compare

2.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view an analog waveform.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the CH1 pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to your circuit you would like to test. You can either use the socket to plug onto a header post, or

connect it to one of the mini-grabber clips and then attach it to your signal of choice.

- Run the Oscilloscope Application.
- Press the Auto Setup button. This will select the best Seconds/Division, Volts/Division and Trigger Level for the signal you have connected.
- You can then scroll the display, either by using the slider bars, or by clicking and dragging on the waveform itself. You can also change the knobs to zoom the waveform.
- You can make simple measurements by using the Cursors area (gray bars under and along side the wave). Click the left mouse button to place one cursor and click the right mouse button to place the second. The resulting measurements are then displayed in the Measurements section of the display.

2.3 Features

2.3.1 Pod Status

The Oscilloscope display shows a current USBee AX **Pod Status** by a red or green LED. When a USBee AX is connected to the computer, the Green LED shows and the list box shows the available **Pod ID List** for all of the USBee Ax's that are connected. You can choose which one you want to use. The others will be unaffected. If a USBee AX is not connected, the LED will glow red and indicate that there is no pod attached.

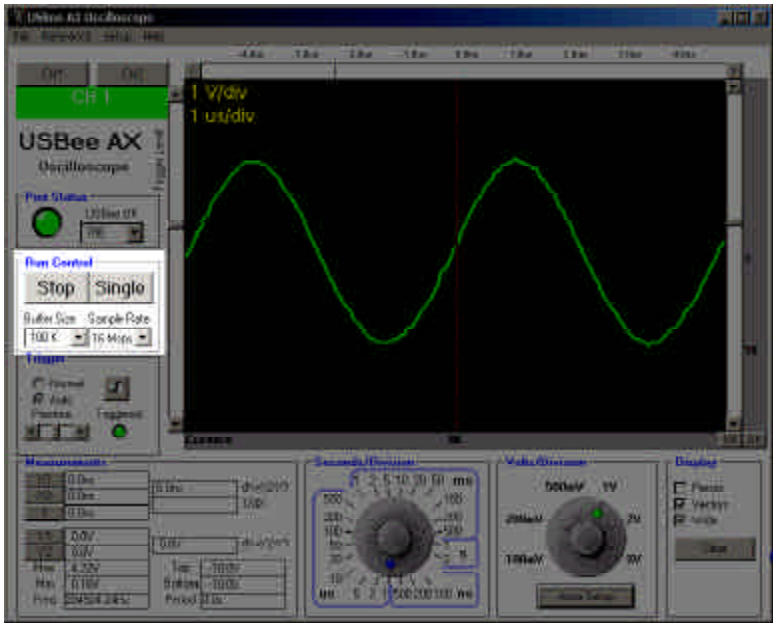
If you run the software with no pod attached, it will run in demonstration mode and simulate data so that you can still see how the software functions.

2.3.2 Channel Control

You can choose which channel will be captured and displayed by pressing the CH1 or CH2 button. The next trace shown will be from that new analog channel.

2.3.3 Run Control

The Oscilloscope captures the behavior of analog signals and displays them as a "trace" in the waveform window. The Run Control section of the display lets you choose how the traces are captured. Below is the Run Control section of the display.



The left button is the **Run/Stop** control. When the oscilloscope is first started, the Run button is pressed. This Run mode performs an infinite series of analog traces, one after the other. This lets you see frequent updates of what the actual signal is doing in real time. If you would like to stop the updating, just press the Stop button and the updating will stop. This run mode is great for signals that repeat over time.

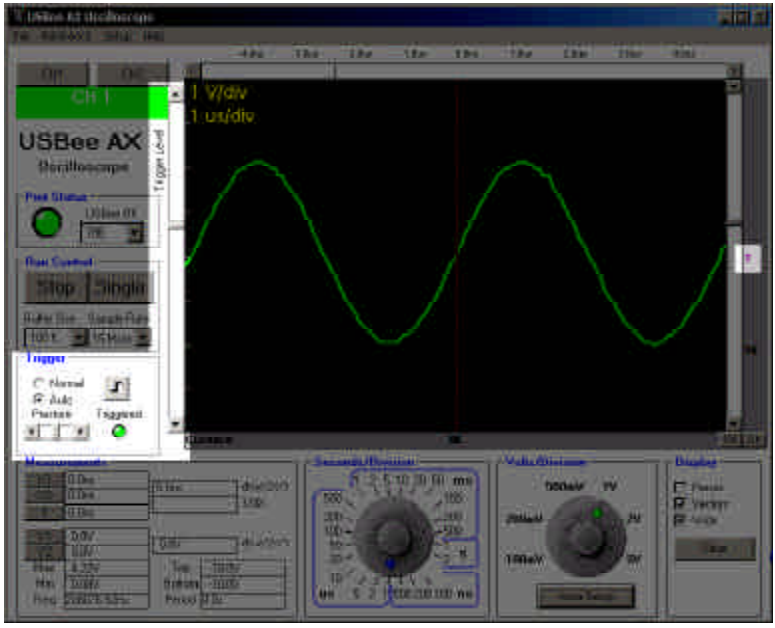
The **Single** button captures a single trace and stops. This mode is good for detailed analysis of a single event, rather than one that occurs repeatedly.

The **Buffer Size** lets you select the size of the Sample Buffer that is used. For each trace, the buffer is completely filled, and then the waveform is displayed. You can choose buffers that will capture the information that you want to see, but remember that the larger the buffer, the longer it will take to fill.

You can also choose the S**ample Rate** that you want samples taken. You can choose from 1Msps (samples per second) to up to 16 Msps. The actual maximum sample rate depends on your PC configuration. You can run the menu item Setup | Sample Rate Test to determine the maximum sample rate for your system.

2.3.4 Trigger Settings

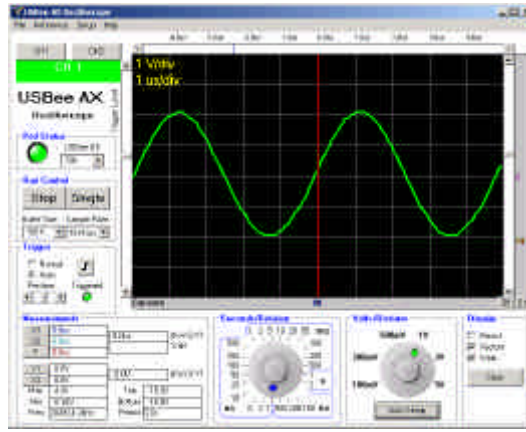
The Oscilloscope uses a Trigger mechanism to allow you to capture just the data that you want to see. You can specify the trigger voltage level (-10V to +10V) by using the slider on the left hand side of the waveform display. A red line that indicates the trigger level will momentarily be shown as you scroll this level. A small T will also be shown on the right hand side of the screen (in the cursors bar) that shows where this level is set to.



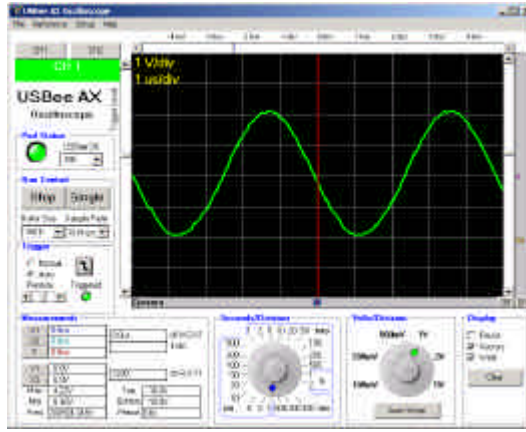
The waveforms are shown with a trigger position which represents where the trigger occurred. This sample point is marked on the waveform display with a Vertical red dotted line and a "T" in the horizontal cursors bar.

This trigger position is where the waveform crossed the **Trigger Voltage** level that you have set. To move the trigger voltage level, just move the slider on the left of the waveform.

You can also specify if you want the oscilloscope to trigger on a R**ising or Falling Edge**. Below shows a trace captured on each of the edges.



Trigger Slope = Rising Edge



Trigger Slope = Falling Edge

The Trigger position is placed where the actual signal crosses the trigger voltage with the proper slope. The USBee AX allows for huge sample buffers, which means that you can capture much more data than can be shown on a single screen. Therefore you can scroll the waveform back and forth on the display to see what happened before or after the trigger.

You can use the **Prestore** setting to specify how much of the data that is in the sample buffer comes before the actual trigger position. If you place the Prestore all the way to the left, most of the samples taken will be after the trigger position. If you place Prestore all the way to the right, most of the samples taken will be before the Trigger position.

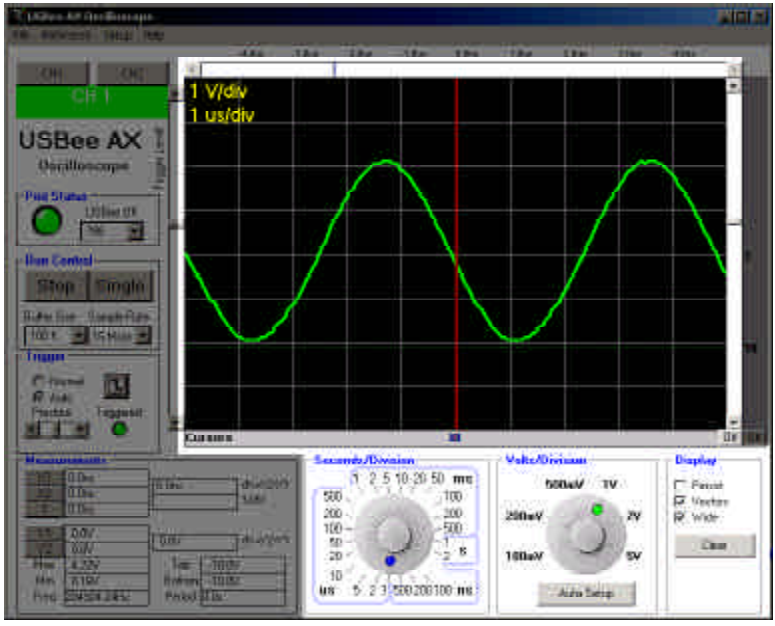
This control lets you see what actually happened way before or way after the trigger occurred.

The Auto and Normal modes specify how the screen is to behave if the trigger voltage level you set is outside the range of the actual signal you are measuring. In **Normal** mode, the screen will only update when the measured signal actually crosses the trigger level. In **Auto** mode, the display will periodically update even if the waveform does not cross the trigger level. This allows you to see what is happening on the display even if you have an incorrect trigger level set. If the trigger level is set to a level that is within the actual range of the measured signal, then Auto and Normal function the same.

The little **Triggered LED** on the display will glow green when the trigger condition has been met. It will glow red when the trigger condition has not been met.

2.3.5 Waveform Display and Zoom Settings

The Waveform display area is where the measured signal information is shown. It is displayed with time increasing from left to right and voltage increasing from bottom to top. The screen is divided into **Divisions** to help in measuring the waveform. The amount of Volts per division and the amount of Seconds per Division are displayed in the top left of the display.



The position of the waveform defaults to show the actual trigger position in the center of the screen. However, you can move the display to see what happened before or after the trigger position.

To **Scroll the Waveform in Time** left and right, you can use the scroll bar at the top of the waveform display, or you can simply click and drag the waveform itself.

To **Scroll the Waveform in Voltage** up and down, you can use the scroll bar at the right of the waveform display, or you can simply click and drag the waveform itself.

To change the number of S**econds per Division** or the number of V**olts per Division**, use the knobs at the bottom of the display. Simply click the knob and drag to the desired setting. You can also zoom in and out in time by clicking on the waveform. To zoom in, click the left mouse on the waveform window. To zoom out in time, click the right mouse button on the waveform window.

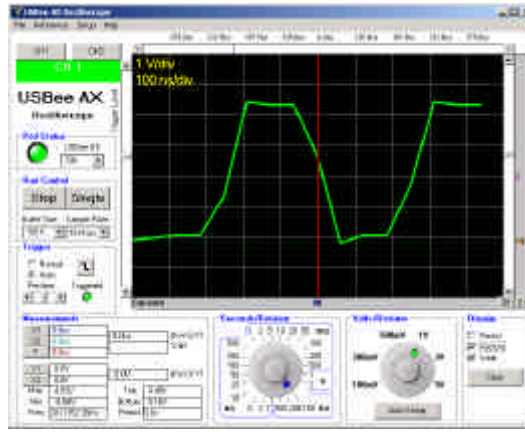
The Display section of the screen shows three selections that affect the way the waveform is displayed.

The **Wide** setting shows the wave using a wider pixel setting. This makes the wave easier to see.

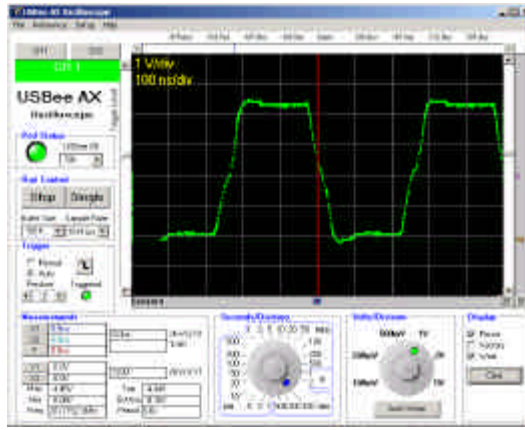
The **Vectors** setting draws the waveform as a line between adjacent samples. With this mode turned off, the samples are shown simply as dots on the display at the sample position.

The **Persist** mode does not clear the display and writes one trace on top of the other trace.

The benefits of these display modes can be seen when you are measuring fast signals and want to get more resolution out of the oscilloscope than the maximum sample rate allows. See the below traces to see the difference. Each trace is taken of the same signal, but the right one shows much more wave detail over a short time of display updates.



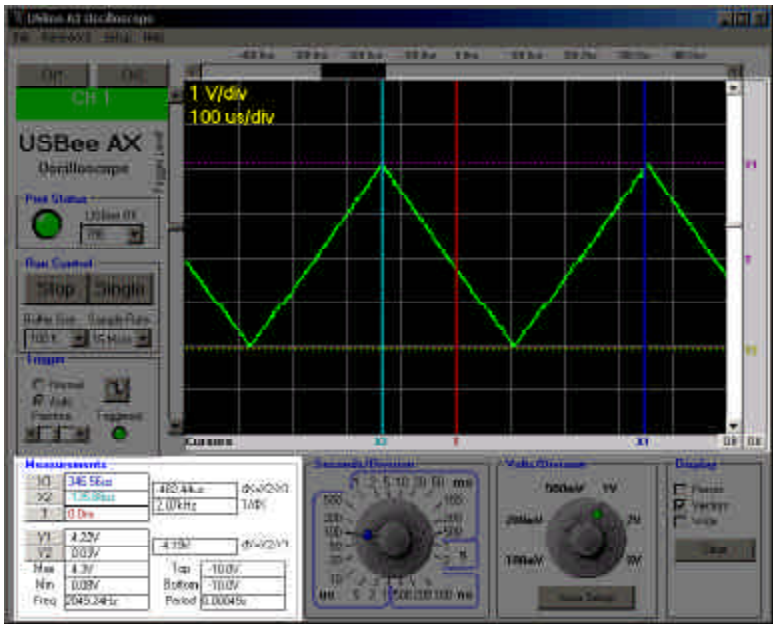
Persist = OFF, Vectors = ON, Wide = ON



Persist = ON, Vectors = OFF, Wide = ON

2.3.6 Measurements and Cursors

The main reason for using an oscilloscope is to measure the various parts of a waveform. The USBee AX uses cursors to help in these measurements.



The **X1 and X2 Cursors** are placed on any horizontal sample time. This lets you measure the time at a specific location or the time between the two cursors. To place the X cursors, move the mouse to the gray box just below the waveform. When you move the mouse in this window, you will see a temporary line that indicates where the cursors will be placed. Place the X1 cursor by left clicking the mouse at the current location. Place the X2 cursor by right clicking the mouse at the current location.

The **Y1 and Y2 Cursors** are placed on any vertical voltage level. This lets you measure the voltage at a specific location or the difference in voltage between the two cursors. To place the Y cursors, move the mouse to the gray box just to the right of the scroll bar to the right of the waveform. When you move the mouse in this window, you will see a temporary line that indicates where the cursors will be placed. Place the Y1 cursor by left clicking the mouse at the current location. Place the Y2 cursor by right clicking the mouse at the current location.

In the Measurement window, you will see the various measurements made off of these cursors.

- **X1 Position** – time at the X1 cursor relative to the trigger position
- **X2 Position** – time at the X2 cursor relative to the trigger position
- **dX** – time difference between X1 and X2 cursors
- **1/dX** – the frequency or the period between X1 and X2 cursors
- **Y1 Position** – voltage at the Y1 cursor relative to Ground
- **Y2 Position** – voltage at the Y2 cursor relative to Ground
- **dY** – voltage difference between Y1 and Y2 cursors

There are also a set of automatic measurements that are made on each trace. These are calculated without the use of the cursors. These are:

- **Max** – the maximum voltage of all samples in the current trace
- **Min** – the minimum voltage of all samples in the current trace
- **Top** – the average of the top of the waveform
- **Bottom** – the average of the bottom of the waveform
- **Freq** – the frequency of the signal currently shown on the screen
- **Period** – the period of the signal currently shown on the screen

2.3.7 File Save, Open and Export

Using the File menu functions, you can save, open or export. a current set of configuration and trace sample data.

Choose the menu item File | Save As to save the current configuration and sample data to a binary ULC file. The format of this ULC file follows.

To load a previously saved waveform and display it, choose File | Open and specify the filename to load. This waveform will then be displayed as it was saved.

You can also export a specific portion of the sample data by placing the X1 and X2 cursors. When you choose File | Export to Text the samples between the X1 and X2 cursors will be written to a file in comma delimited text format as below.

2.3.7.1 Export to Text Format

```
Signal 0: Signal 0
Signal 1: Signal 1
Signal 2: Signal 2
Signal 3: Signal 3
Signal 4: Signal 4
Signal 5: Signal 5
Signal 6: Signal 6
Signal 7: Signal 7
Sample Rate: 16 Msps
Number Of Samples: 17
Pod ID: 786
38665, FC,   1.875,
38666, FC,   1.797,
38667, FC,   1.875,
38668, FC,   1.797,
38669, FC,   1.875,
38670, FC,   1.797,
38671, FC,   1.875,
38672, FC,   1.875,
```

2.3.7.2 ULC File Format

Bytes	Description
20	Signal 0
20	Signal 1
20	Signal 2
20	Signal 3
20	Signal 4
20	Signal 5
20	Signal 6
20	Signal 7
1	Sample Rate (247, 167,...)
4	Buffer Size in bytes (BufSize)
1	SG Trigger Mode (don't care, high, low, rising, falling)
1	Loop
1	LA Trigger Setting Signal 0 Number 1
1	LA Trigger Setting Signal 1 Number 1
1	LA Trigger Setting Signal 2 Number 1
1	LA Trigger Setting Signal 3 Number 1
1	LA Trigger Setting Signal 4 Number 1
1	LA Trigger Setting Signal 5 Number 1
1	LA Trigger Setting Signal 6 Number 1
1	LA Trigger Setting Signal 7 Number 1
1	LA Trigger Setting Signal 0 Number 2
1	LA Trigger Setting Signal 1 Number 2
1	LA Trigger Setting Signal 2 Number 2
1	LA Trigger Setting Signal 3 Number 2
1	LA Trigger Setting Signal 4 Number 2
1	LA Trigger Setting Signal 5 Number 2
1	LA Trigger Setting Signal 6 Number 2
1	LA Trigger Setting Signal 7 Number 2
1	LA Trigger Setting Signal 0 Number 3
1	LA Trigger Setting Signal 1 Number 3
1	LA Trigger Setting Signal 2 Number 3
1	LA Trigger Setting Signal 3 Number 3
1	LA Trigger Setting Signal 4 Number 3
1	LA Trigger Setting Signal 5 Number 3
1	LA Trigger Setting Signal 6 Number 3
1	LA Trigger Setting Signal 7 Number 3
1	LA Trigger Setting Signal 0 Number 4
1	LA Trigger Setting Signal 1 Number 4
1	LA Trigger Setting Signal 2 Number 4
1	LA Trigger Setting Signal 3 Number 4
1	LA Trigger Setting Signal 4 Number 4
1	LA Trigger Setting Signal 5 Number 4
1	LA Trigger Setting Signal 6 Number 4
1	LA Trigger Setting Signal 7 Number 4
1	LA Clocking Mode (Internal/External)
4	Prestore Setting
4	Trigger Position (sample number at trigger)

4 Center Display Position (sample number at center of screen)
4 Scale Value
4 SubScale Value
4 X1Cursor Position (sample number at X1Cursor)
4 X2Cursor Position (sample number at X2Cursor)
4 Pod ID used
4 Volts Per Division Factor
4 Voltage display offset
4 Y1Cursor Position (voltage factor at Y1Cursor)
4 Y2Cursor Position (voltage factor at Y2Cursor)
4 Scope Trigger Voltage Level factor
1 Trigger Slope
BufSize Digital channel samples(one byte per sample - each bit is a signal - bit 0 = signal 0)
BufSize Analog channel samples(one byte per sample – 0 = -10V, 255 = +10V)

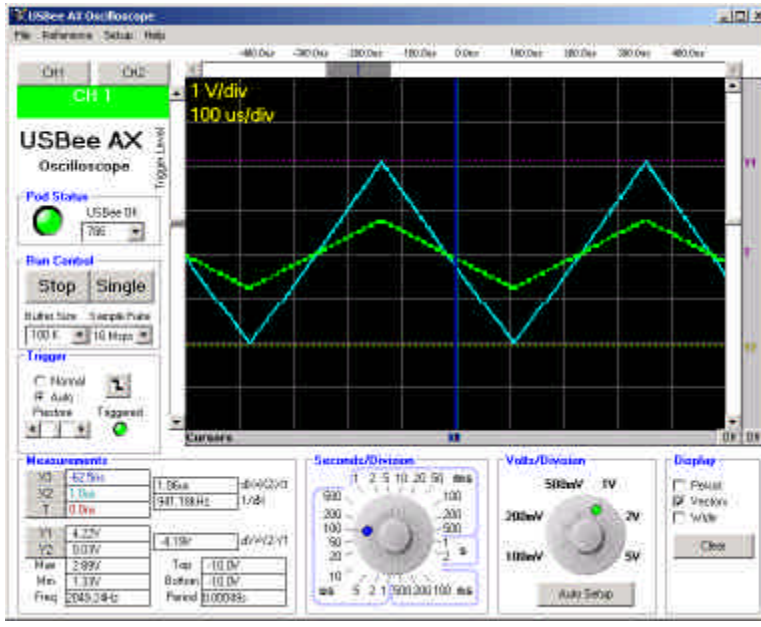
2.3.8 Printing

You can print the current screen to any printer by choosing the File | Print menu item.

2.3.9 Reference Waveform

To compare two signals that are taken at separate times, you can save one as a reference waveform and display it behind the currently active waveform at the same time. To save the current screen as the reference waveform, choose the menu item Reference | Save Reference Waveform. To display this waveform behind the current waveform, choose the menu item File | Show Reference Waveform.

Below is a screen shot showing the use of the reference waveform. The reference waveform is in Cyan.



2.3.10 Calibration

Since electronic components vary values slightly over time and temperature, the USBee AX Pod requires calibration periodically to maintain accuracy. The USBee AX has been calibrated during manufacturing and should maintain accuracy for a long time, but in case you want to recalibrate the device, follow these steps. The calibration values are stored inside the USBee AX pod. Without calibration the measurements of the oscilloscope may not be accurate as the pod ages.

To calibrate your USBee AX Pod you will need the following equipment:

- External Voltage Source (between 5V and 9V)
- High Precision Multimeter

When you are ready to calibrate the USBee AX Pod, go to the menu item Setup | Calibrate. You will be asked to confirm that you really want to do the calibration. If so, press Yes, otherwise press No. Then follow these steps:

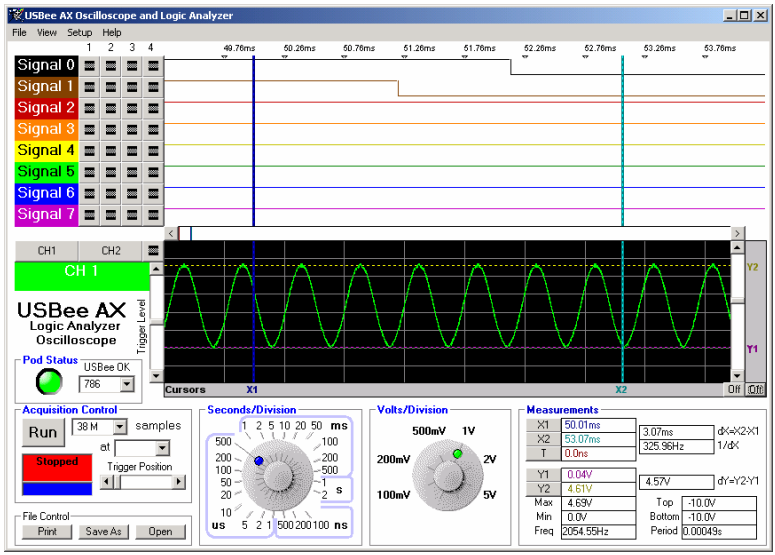
- Connect the CH1 signal to the GND signal using the test leads and press OK. A measurement will be taken.
- Connect the CH2 signal to the GND signal using the test leads and press OK. A measurement will be taken.
- Connect the GND signal to the ground and the CH1 signal to the positive connection of the External Voltage Source using the test leads and press OK. A measurement will be taken.
- With the Multimeter, measure the actual voltage between the GND signal and the CH1 signal and enter this value in the dialog box.
- Connect the GND signal to the ground and the CH2 signal to the positive connection of the External Voltage Source using the test leads and press OK. A measurement will be taken.
- With the Multimeter, measure the actual voltage between the GND signal and the CH2 signal and enter this value in the dialog box.
- The calibration is now complete. The calibration values have been saved inside the pod.

The analog measurements of your USBee AX pod are only as accurate as the voltages supplied and measured during calibration.

3 Mixed Signal Oscilloscope

AX-Standard, AX-Plus and AX-Pro

This section details the operation of the Mixed Signal Oscilloscope application that comes with the USBee AX. Below you see the application screen.



The USBee AX Mixed Signal Oscilloscope functions as a standard Digital Storage Oscilloscope combined with a Digital Logic Analyzer, which is a tool used to measure and display analog signals in a graphical format. It displays what the analog and digital input signals do over time. The digital and analog samples are taken at the same time and can be used to debug mixed signal systems.

3.1 Mixed Signal Oscilloscope/Logic Analyzer Specifications

Analog Inputs	2
Analog Channels	1
Maximum Analog Sample Rate [1]	16 Msps
Analog Bandwidth	3 MHz
Input Impedance	1M Ohm/30 pF
Analog Input Voltage Range	-10V to +10V
Analog Sensitivity	78mV
Analog Resolution	256 steps
Channel Buffer Depth [2]	>1 Million
Volts per Division Settings	100mV to 5V in 6 steps
Time per Division Settings	100ns to 2s in 23 steps
Trigger Modes	Auto, Single, Digital Triggers
Analog Trigger Voltage	Between -10V and +10V
Cursors	2 Time and 2 Voltage
Voltage Display Offset	Up to maximum inputs
Time Display Offset	Up to available buffer depth
Trigger Position Setting	10% to 90%
Measurements	Min, Max, Top Bottom, Freq, Period
Digital Channels	8
Maximum Digital Sample Rate [1]	16 Msps
Internal Clocking	Yes
External Clocking	No
Digital Trigger Levels	4
Digital Trigger Qualifiers	Rising Edge, Falling Edge, High,Low
Trigger Prestore	Yes
Trigger Poststore	Yes
Sample Clock Output	Yes
Maximum Digital Input Voltage	+5.5V

Digital Input Low Level	< 0.8V
Digital Input High Level	> 1.4V

3.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view a mixed signal (analog and digital) waveform trace.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the CH1 pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire. Connect the other end of the wire to your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to your signal of choice.
- Connect any of the digital inputs 0 thru 7 on the USBee AX pod to one of the signal wires using the small socket on the end of the wire. Connect the other end of the wire to your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to your signal of choice.
- Run the Mixed Signal Oscilloscope Application.
- Press the Run button. This will capture and display the current activity on all of the signals.
- You can then scroll the display, either by using the slider bars, or by clicking and dragging on the waveform itself. You can also change the knobs to zoom the waveform.
- You can make simple measurements by using the Cursors area (gray bars under and along side the waves). Click the left mouse button to place one cursor and click the right mouse button to place the second. The resulting measurements are then displayed in the Measurements section of the display.

3.3 Features

3.3.1 Pod Status

The Mixed Signal Oscilloscope display shows a current USBee AX **Pod Status** by a red or green LED. When a USBee AX is connected to the computer, the Green LED shows and the list box shows the available **Pod ID List** for all of the USBee Ax's that are connected. You can choose which one you want to use. The others will be unaffected. If a

USBee AX is not connected, the LED will glow red and indicate that there is no pod attached.

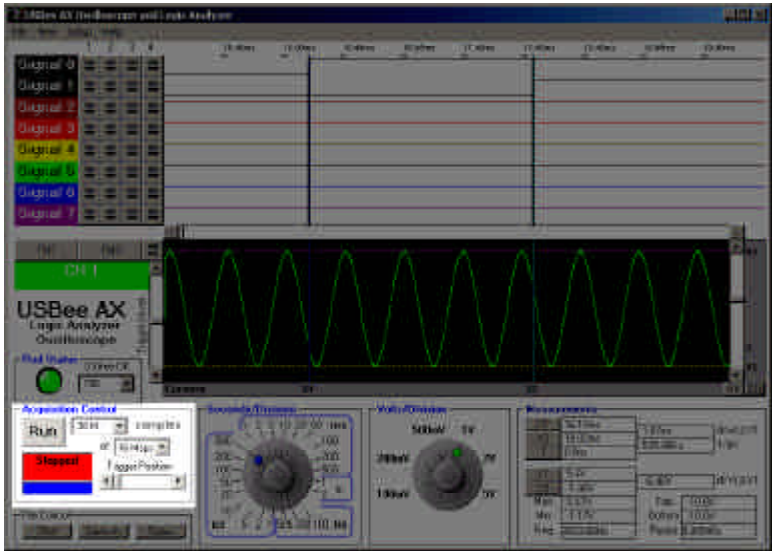
If you run the software with no pod attached, it will run in demonstration mode and simulate data so that you can still see how the software functions.

3.3.2 Analog Channel Control

You can choose which channel will be captured and displayed by pressing the CH1 or CH2 button. The next trace shown will be from that new analog channel. All 8 digital lines are always sampled every trace.

3.3.3 Acquisition Control

The Mixed Signal Oscilloscope captures the behavior of analog and digital signals and displays them as a “trace” in the waveform window. The Acquisition Control section of the display lets you choose how the traces are captured. Below is the Acquisition Control section of the display.



The button is the **Run / Stop** control. When the mixed signal oscilloscope is first started, the Run button is not pressed and is waiting for you to start a capture. The Run button captures a single trace and stops. This mode is good for detailed analysis of a single event.

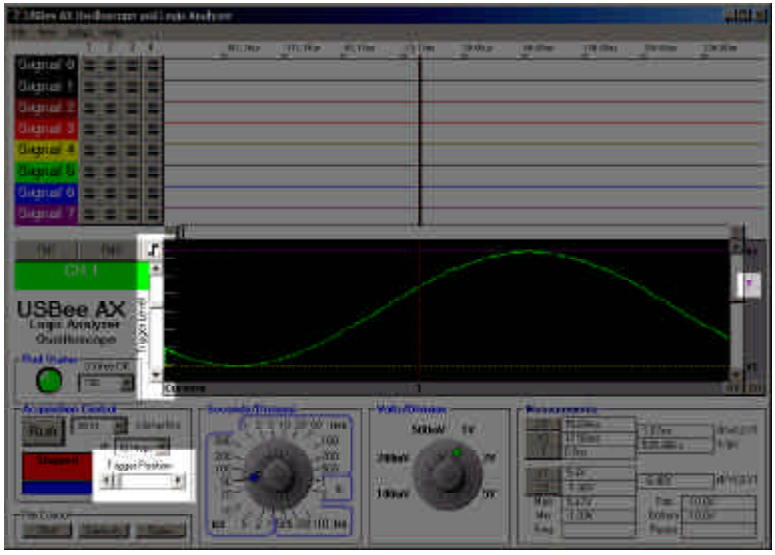
The **Buffer Size** lets you select the size of the Sample Buffer that is used. For each trace, the buffer is completely filled, and then the waveform is displayed. You can choose buffers that will capture the information that you want to see, but remember that the larger the buffer, the longer it will take to fill.

You can also choose the S**ample Rate** that you want samples taken. You can choose from 1 Msps (samples per second) to up to 16 Msps. The actual maximum sample rate depends on your PC configuration. You can run the menu item Setup | Sample Rate Test to determine the maximum sample rate for your system.

3.3.4 Trigger Settings

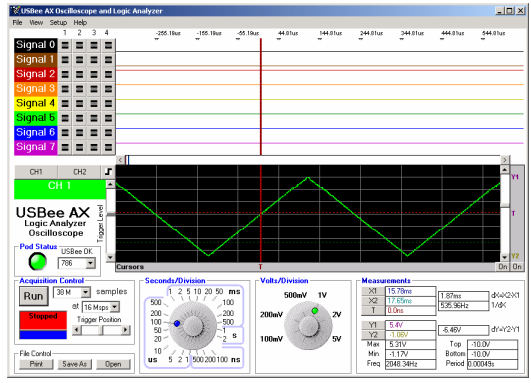
The Mixed Signal Oscilloscope uses a Trigger mechanism to allow you to capture just the data that you want to see. You can use either a digital channel trigger (as in the logic analyzer), or an analog trigger (as in the oscilloscope). You can not use a combination of analog and digital.

For an **Analog trigger**, you can specify the trigger voltage level (-10V to +10V) by using the slider on the left hand side of the analog waveform display. A red line that indicates the trigger level will momentarily be shown as you scroll this level. A small T will also be shown on the right hand side of the screen (in the cursors bar) that shows where this level is set to.

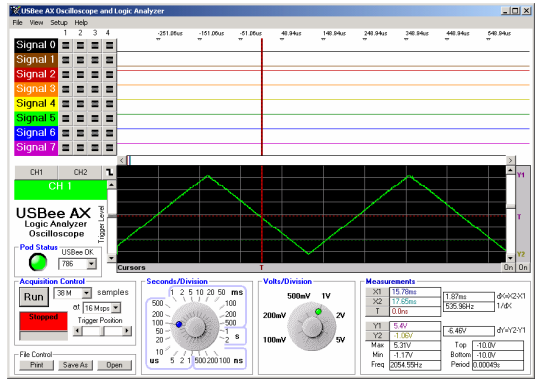


For an analog trigger, the trigger position is where the waveform crossed the **Trigger Voltage** level that you have set at the specified slope. To move the trigger voltage level, just move the slider on the left of the waveform. To change the slope, press the button to the right of the CH1 and CH2 buttons.

You can also specify if you want the oscilloscope to trigger on a **Rising or Falling Edge**. Below shows a trace captured on each of the edges.



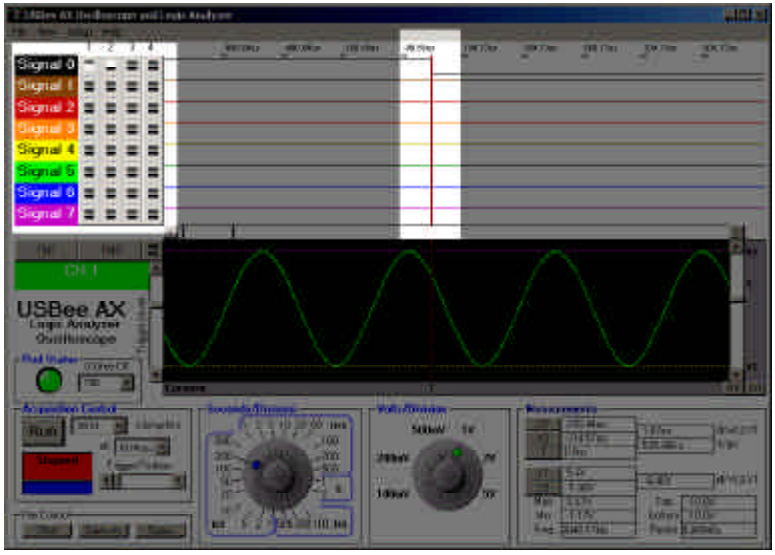
Analog Trigger Slope = Rising Edge



Analog Trigger Slope = Falling Edge

The Trigger position is placed where the actual signal crosses the trigger voltage with the proper slope. The USBee AX allows for huge sample buffers, which means that you can capture much more data than can be shown on a single screen. Therefore you can scroll the waveform back and forth on the display to see what happened before or after the trigger.

For a **Digital trigger**, you can specify the digital states for any of the 8 signals that must be present on the digital lines before it will trigger. Below shows the trigger settings (to the right of the Signal labels). This example shows that we want to trigger on a falling edge of Signal 0, which is represented by a high level followed by a low level. To change the level of any of the trigger settings, just click the level button to change from don't care to high to low.



The digital trigger condition is made up of up to 4 sequential states of any of the 8 signals. Each state for a single signal can be high, low or don't care. This allows you to trigger on rising edges, falling edges, edges during another signals constant level, or one edge followed by another edge.

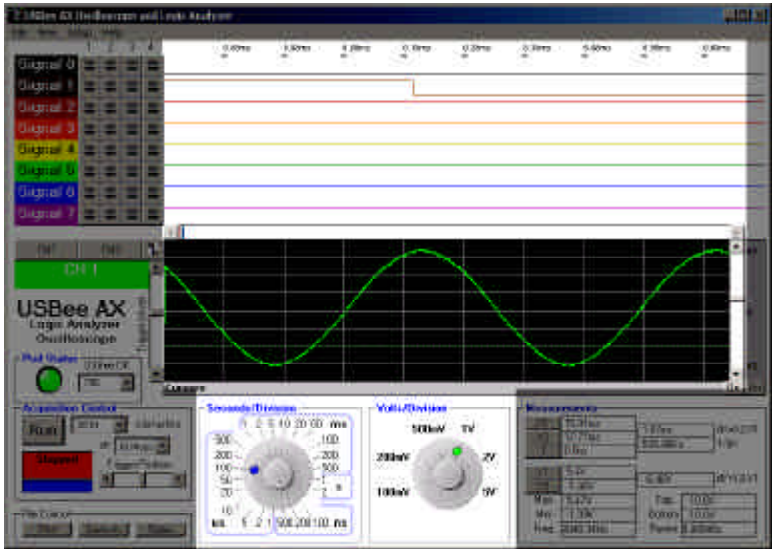
The waveforms are shown with a trigger position which represents where the trigger occurred. This sample point is marked on the waveform display with a Vertical red dotted line and a "T" in the horizontal cursors bar.

You can use the **Trigger Position** setting to specify how much of the data that is in the sample buffer comes before the actual trigger position. If you place the Trigger Position all the way to the left, most of the samples taken will be after the trigger sample. If you place Trigger Position all the way to the right, most of the samples taken will be before the Trigger sample. This control lets you see what actually happened way before or way after the trigger occurred.

The status box on the display will show red when the unit is not acquiring samples, flash blue when it is waiting for a trigger, and glow green when the trigger condition has been met. It will glow red again when the capture is completed.

3.3.5 Waveform Display and Zoom Settings

The Waveform display area is where the measured signal information is shown. It is displayed with time increasing from left to right and voltage increasing from bottom to top. The screen is divided into **Divisions** to help in measuring the waveforms.



The position of the waveform defaults to show the actual trigger position in the center of the screen after a capture. However, you can move the display to see what happened before or after the trigger position.

To **Scroll the Waveforms in Time** left and right, you can use the scroll bar at the top of the analog waveform display (between the analog and digital waveforms), or you can simply click and drag the waveform itself.

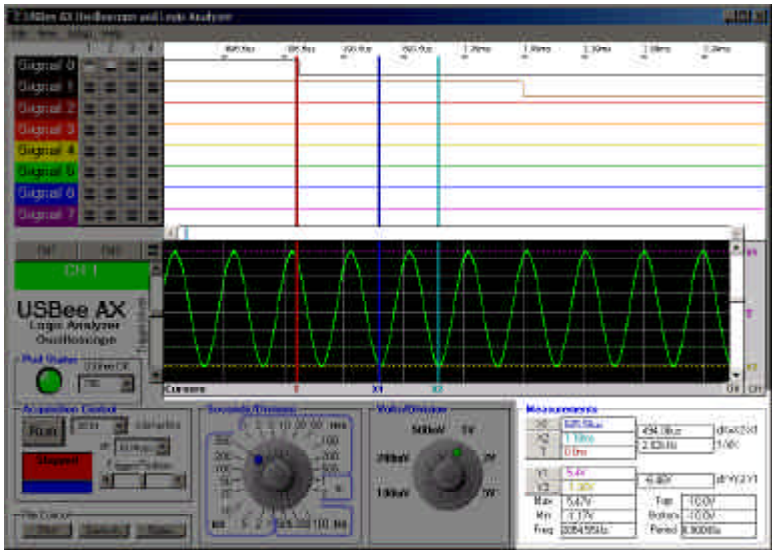
To **Scroll the Analog Waveform in Voltage** up and down, you can use the scroll bar at the right of the waveform display, or you can simply click and drag the waveform itself.

To change the number of **Seconds per Division** or the number of **Volts per Division** for the analog channel, use the knobs at the

bottom of the display. Simply click the knob and drag to the desired setting. You can also zoom in and out in time by clicking on the waveform. To zoom in, click the left mouse on the waveform window. To zoom out in time, click the right mouse button on the waveform window.

3.3.6 Measurements and Cursors

The main reason for using an oscilloscope or logic analyzer is to measure the various parts of a waveform. The USBee AX uses cursors to help in these measurements.



The **X1 and X2 Cursors** are placed on any horizontal sample time. This lets you measure the time at a specific location or the time between the two cursors. To place the X cursors, move the mouse to the gray box just below the waveform. When you move the mouse in this window, you will see a temporary line that indicates where the cursors will be placed. Place the X1 cursor by left clicking the mouse at the current location. Place the X2 cursor by right clicking the mouse at the current location.

The **Y1 and Y2 Cursors** are placed on any vertical voltage level. This lets you measure the voltage at a specific location or the difference in voltage between the two cursors. To place the Y cursors, move the mouse to the gray box just to the right of the scroll bar to the right of the waveform. When you move the mouse in this window, you will see a temporary line that indicates where the cursors will be placed. Place

the Y1 cursor by left clicking the mouse at the current location. Place the Y2 cursor by right clicking the mouse at the current location.

In the Measurement window, you will see the various measurements made off of these cursors.

- **X1 Position** – time at the X1 cursor relative to the trigger position
- **X2 Position** – time at the X2 cursor relative to the trigger position
- **dX** – time difference between X1 and X2 cursors
- **1/dX** – the frequency or the period between X1 and X2 cursors
- **Y1 Position** – voltage at the Y1 cursor relative to Ground
- **Y2 Position** – voltage at the Y2 cursor relative to Ground
- **dY** – voltage difference between Y1 and Y2 cursors

There are also a set of automatic measurements that are made on the analog waveform for each trace. These are calculated without the use of the cursors. These are:

- **Max** – the maximum voltage of all samples in the current trace
- **Min** – the minimum voltage of all samples in the current trace
- **Top** – the average of the top of the waveform
- **Bottom** – the average of the bottom of the waveform
- **Freq** – the frequency of the signal currently shown on the screen
- **Period** – the period of the signal currently shown on the screen

3.3.7 File Save, Open and Export

Using the File menu functions, you can save, open or export. a current set of configuration and trace sample data.

Choose the menu item File | Save As to save the current configuration and sample data to a binary ULC file. The format of this ULC file follows.

To load a previously saved waveform and display it, choose File | Open and specify the filename to load. This waveform will then be displayed as it was saved.

You can also export a specific portion of the sample data by placing the X1 and X2 cursors. When you choose File | Export to Text the samples between the X1 and X2 cursors will be written to a file in comma delimited text format as below.

3.3.7.1 Export to Text Format

```
Signal 0: Signal 0
Signal 1: Signal 1
Signal 2: Signal 2
Signal 3: Signal 3
Signal 4: Signal 4
Signal 5: Signal 5
Signal 6: Signal 6
Signal 7: Signal 7
Sample Rate: 16 Msps
Number Of Samples: 17
Pod ID: 786
38665, FC,   1.875,
38666, FC,   1.797,
38667, FC,   1.875,
38668, FC,   1.797,
38669, FC,   1.875,
38670, FC,   1.797,
38671, FC,   1.875,
38672, FC,   1.875,
```

3.3.7.2 ULC File Format

Bytes	Description
20	Signal 0
20	Signal 1
20	Signal 2
20	Signal 3
20	Signal 4
20	Signal 5
20	Signal 6
20	Signal 7
1	Sample Rate (247, 167,...)
4	Buffer Size in bytes (BufSize)
1	SG Trigger Mode (don't care, high, low, rising, falling)
1	Loop
1	LA Trigger Setting Signal 0 Number 1
1	LA Trigger Setting Signal 1 Number 1
1	LA Trigger Setting Signal 2 Number 1
1	LA Trigger Setting Signal 3 Number 1
1	LA Trigger Setting Signal 4 Number 1
1	LA Trigger Setting Signal 5 Number 1
1	LA Trigger Setting Signal 6 Number 1
1	LA Trigger Setting Signal 7 Number 1
1	LA Trigger Setting Signal 0 Number 2
1	LA Trigger Setting Signal 1 Number 2
1	LA Trigger Setting Signal 2 Number 2
1	LA Trigger Setting Signal 3 Number 2
1	LA Trigger Setting Signal 4 Number 2
1	LA Trigger Setting Signal 5 Number 2
1	LA Trigger Setting Signal 6 Number 2
1	LA Trigger Setting Signal 7 Number 2
1	LA Trigger Setting Signal 0 Number 3
1	LA Trigger Setting Signal 1 Number 3
1	LA Trigger Setting Signal 2 Number 3
1	LA Trigger Setting Signal 3 Number 3
1	LA Trigger Setting Signal 4 Number 3
1	LA Trigger Setting Signal 5 Number 3
1	LA Trigger Setting Signal 6 Number 3
1	LA Trigger Setting Signal 7 Number 3
1	LA Trigger Setting Signal 0 Number 4
1	LA Trigger Setting Signal 1 Number 4
1	LA Trigger Setting Signal 2 Number 4
1	LA Trigger Setting Signal 3 Number 4
1	LA Trigger Setting Signal 4 Number 4
1	LA Trigger Setting Signal 5 Number 4
1	LA Trigger Setting Signal 6 Number 4
1	LA Trigger Setting Signal 7 Number 4
1	LA Clocking Mode (Internal/External)
4	Prestore Setting

4 Trigger Position (sample number at trigger)
4 Center Display Position (sample number at center of screen)
4 Scale Value
4 SubScale Value
4 X1Cursor Position (sample number at X1Cursor)
4 X2Cursor Position (sample number at X2Cursor)
4 Pod ID used
4 Volts Per Division Factor
4 Voltage display offset
4 Y1Cursor Position (voltage factor at Y1Cursor)
4 Y2Cursor Position (voltage factor at Y2Cursor)
4 Scope Trigger Voltage Level factor
1 Trigger Slope
BufSize Digital channel samples(one byte per sample - each bit is a signal - bit 0 = signal 0)
BufSize Analog channel samples(one byte per sample – 0 = -10V, 255 = +10V)

3.3.8 USB, I2C, Async, and SPI Decoders

Using these View menu functions, you can decode these serial busses and extract the actual data transmitted via the protocols. These features are detailed in a later section of this document will only function on the USBee AX-Plus and USBee AX-Pro models.

3.3.9 Calibration

Since electronic components vary values slightly over time and temperature, the USBee AX Pod requires calibration periodically to maintain accuracy. The USBee AX has been calibrated during manufacturing and should maintain accuracy for a long time, but in case you want to recalibrate the device, follow these steps. The calibration values are stored inside the USBee AX pod. Without calibration the measurements of the oscilloscope may not be accurate as the pod ages.

To calibrate your USBee AX Pod you will need the following equipment:

- External Voltage Source (between 5V and 9V)
- High Precision Multimeter

When you are ready to calibrate the USBee AX Pod, go to the menu item Setup | Calibrate. You will be asked to confirm that you really want to do the calibration. If so, press Yes, otherwise press No. Then follow these steps:

- Connect the CH1 signal to the GND signal using the test leads and press OK. A measurement will be taken.
- Connect the CH2 signal to the GND signal using the test leads and press OK. A measurement will be taken.
- Connect the GND signal to the ground and the CH1 signal to the positive connection of the External Voltage Source using the test leads and press OK. A measurement will be taken.
- With the Multimeter, measure the actual voltage between the GND signal and the CH1 signal and enter this value in the dialog box.
- Connect the GND signal to the ground and the CH2 signal to the positive connection of the External Voltage Source using the test leads and press OK. A measurement will be taken.
- With the Multimeter, measure the actual voltage between the GND signal and the CH2 signal and enter this value in the dialog box.
- The calibration is now complete. The calibration values have been saved inside the pod.

The analog measurements of your USBee AX pod are only as accurate as the voltages supplied and measured during calibration.

4 Digital Voltmeter (DVM)

AX-Standard, AX-Plus and AX-Pro

This section details the operation of the Digital Voltmeter (DVM) application that comes with the USBee AX. Below you see the application screen.

4.1 Digital Voltmeter Specifications

Analog Channels Displayed	2
Analog Input Voltage Range	-10V to +10V
Minimum Measurable Resolution	78mV
Analog Resolution	256 steps
Update Rate	3 samples per second
Logging Function	Store data to text file

4.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to measure two analog voltages.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the CH1 pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire. Connect the other end of the wire to your circuit you would like to test.
- Connect the CH2 pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire. Connect the other end of the wire to your circuit you would like to test.
- Run the DVM Application.
- The voltages of the CH1 and CH2 signal will be displayed and updated about once every second.

4.3 Features

4.3.1 Pod Status

The DVM display shows a current USBee AX **Pod Status** by a red or green LED. When a USBee AX is connected to the computer, the Green LED shows and the list box shows the available **Pod ID List** for all of the USBee Ax's that are connected. You can choose which one you want to use. The others will be unaffected. If a USBee AX is not connected, the LED will glow red and indicate that there is no pod attached.

If you run the software with no pod attached, it will run in demonstration mode and simulate data so that you can still see how the software functions.

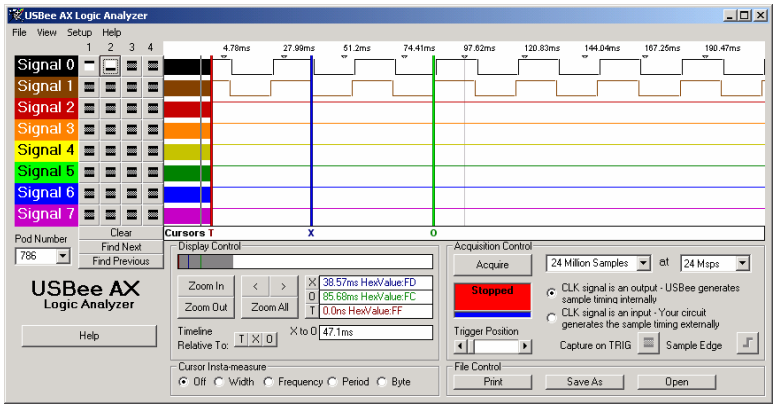
4.3.2 Voltage Measurement

The DVM takes a 250 msec measurement of each of the channels and displays the average voltage over that time period. Although the resolution of each individual sample is 78.125mV, the averaged values are far more accurate.

5 Logic Analyzer

AX-Standard, AX-Plus and AX-Pro

This section details the operation of the Logic Analyzer application that comes with the USBee AX. Below you see the application screen.



5.1 Logic Analyzer Specifications

Digital Channels	8
Maximum Digital Sample Rate [1]	24 Msps
Internal Clocking	Yes
External Clocking	Yes
Trigger Levels	4
Trigger Qualifiers	Rising Edge, Falling Edge, High,Low
Number of Samples [2]	1 million samples up to PC RAM
Sample Rates [1]	1Msps to 24 Msps
Trigger Prestore	Yes
Trigger Poststore	Yes
Sample Clock Output	Yes
Maximum Input Voltage	+5.5V
Input Low Level	< 0.8V

Input High Level	> 1.4V
Cursors	Trigger position, X and O
Measurements	Hex value, Period, Frequency

5.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view the 8 digital signals.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 thru Signal 7 lines pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to your circuit you would like to test.
- Run the Logic Analyzer Application.
- Press the Run button. This will capture a trace of the activity on the 8 digital lines.
- You can then scroll the display, either by using the slider bars, or by clicking and dragging on the waveform itself. You can also change the knobs to zoom the waveform.
- You can make simple measurements by using the Cursors area (gray bars under the wave). Click the left mouse button to place one cursor and click the right mouse button to place the second. The resulting measurements are then displayed in the Measurements section of the display.

5.3 Features

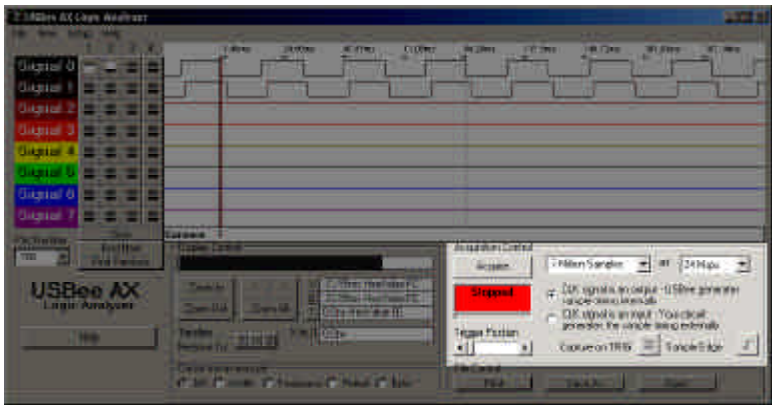
5.3.1 Pod Status

The Logic Analyzer display shows a list with the available **Pod ID List** for all of the USBee Ax's that are connected to your PC. You can choose which one you want to use. The others will be unaffected. If a USBee AX is not connected, the list box will read Demo to indicate that there is no pod attached.

If you run the software with no pod attached, it will run in demonstration mode and simulate data so that you can still see how the software functions.

5.3.2 Acquisition Control

The Logic Analyzer captures the behavior of digital signals and displays them as a “trace” in the waveform window. The Acquisition Control section of the display lets you choose how the traces are captured. Below is the Acquisition Control section of the display.



The **Acquire** button starts and stops a capture. When the logic analyzer is first started, the Acquire button is not pressed and is waiting for you to start a capture. The Acquire button captures a single trace and stops. This mode is good for detailed analysis of a single event.

The **Buffer Size** lets you select the size of the Sample Buffer that is used. For each trace, the buffer is completely filled, and then the waveform is displayed. You can choose buffers that will capture the information that you want to see, but remember that the larger the buffer, the longer it will take to fill.

You can also choose the **Sample Rate** that you want samples taken. This uses an **Internal Clock** at that sample rate you choose. You can choose from 1 Msps (samples per second) to up to 24 Msps. The actual maximum sample rate depends on your PC configuration. You can run the menu item Setup | Sample Rate Test to determine the maximum sample rate for your system.

The USBee AX can also use an **External Clock** as the sample clock via the CLK line. This is selected by the radio button that reads “CLK signal is an input…”. You can also then choose which **Sampling Edge** that the samples will be taken on: the rising or falling edge of the external clock using the pushbutton. In external timing mode, you can also use the **External TRG** signal to qualify the capture of the traces. Set this toggle pushbutton to the state of TRG that you want samples

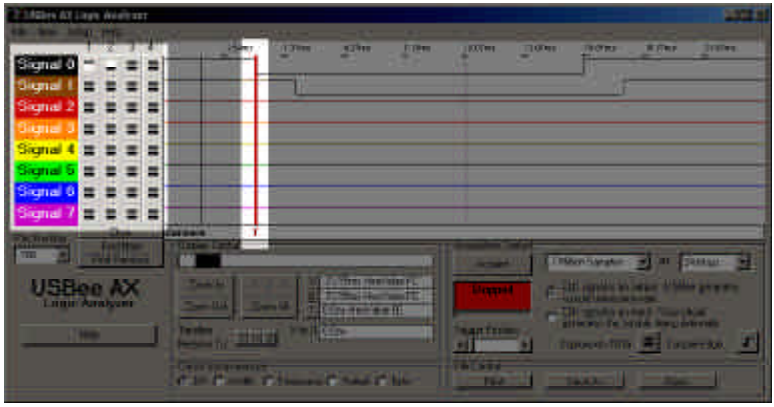
to be taken. Samples occurring during invalid TRG times will not be stored.

The **Status Box** on the display will show red when the unit is not acquiring samples, flash blue when it is waiting for a trigger, and glow green when the trigger condition has been met. It will glow red again when the capture is completed.

5.3.3 Trigger Settings

The Logic Analyzer uses a Trigger mechanism to allow you to capture just the data that you want to see.

You can specify the digital states for any of the 8 signals that must be present on the digital lines before it will trigger. Below shows the trigger settings (to the right of the Signal labels). This example shows that we want to trigger on a falling edge of Signal 0, which is represented by a high level followed by a low level. To change the level of any of the trigger settings, just click the level button to change from don't care to high to low.



The Trigger position is placed where the actual signal crosses the trigger voltage with the proper slope. The USBee AX allows for huge sample buffers, which means that you can capture much more data than can be shown on a single screen. Therefore you can scroll the waveform back and forth on the display to see what happened before or after the trigger.

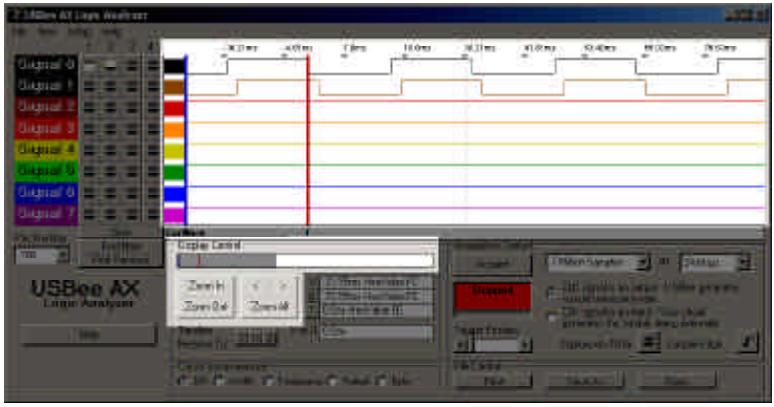
The digital trigger condition is made up of up to 4 sequential states of any of the 8 signals. Each state for a single signal can be high, low or don't care. This allows you to trigger on rising edges, falling edges, edges during another signals constant level, or one edge followed by another edge.

The waveforms are shown with a trigger position which represents where the trigger occurred. This sample point is marked on the waveform display with a Vertical red dotted line and a “T” in the horizontal cursors bar.

You can use the **Trigger Position** setting to specify how much of the data that is in the sample buffer comes before the actual trigger position. If you place the Trigger Position all the way to the left, most of the samples taken will be after the trigger sample. If you place Trigger Position all the way to the right, most of the samples taken will be before the Trigger sample. This control lets you see what actually happened way before or way after the trigger occurred.

5.3.4 Waveform Display and Zoom Settings

The Waveform display area is where the measured signal information is shown. It is displayed with time increasing from left to right and voltage increasing from bottom to top. The screen is divided into **Divisions** to help in measuring the waveforms.



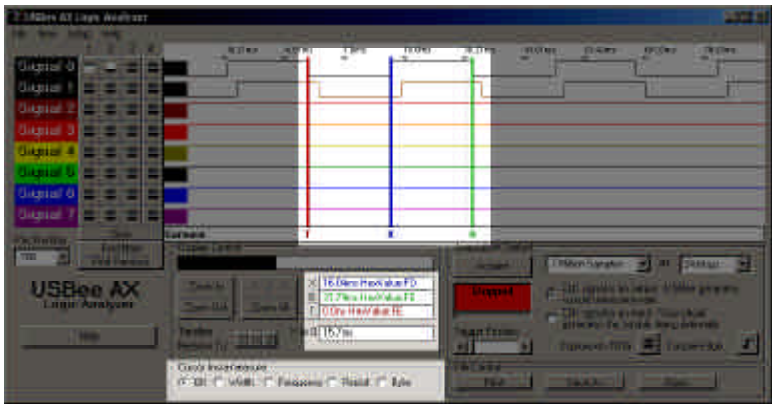
The position of the waveform defaults to show the actual trigger position in the center of the screen after a capture. However, you can move the display to see what happened before or after the trigger position.

To **Scroll the Waveforms in Time** left and right, you can use the left and right arrows highlighted above, click and drag the Overview Bar (right under the Display Control title), or you can simply click and drag the waveform itself.

To change the zoom ratio for the time, click the Zoom In or Zoom Out buttons. You can also zoom in and out in time by clicking on the waveform. To zoom in, click the left mouse on the waveform window. To zoom out in time, click the right mouse button on the waveform window.

5.3.5 Measurements and Cursors

The main reason for using a logic analyzer is to measure the various parts of a waveform. The USBee AX uses cursors to help in these measurements.



The **X and O Cursors** are placed on any horizontal sample time. This lets you measure the time at a specific location or the time between the two cursors. To place the X and O cursors, move the mouse to the white box just below the waveform. When you move the mouse in this window, you will see a temporary line that indicates where the cursors will be placed. Place the X cursor by left clicking the mouse at the current location. Place the O cursor by right clicking the mouse at the current location.

In the Measurement window, you will see the various measurements made off of these cursors. To change the selected relative cursor, click the T,X or O buttons next to the “Timeline Relative To” text.

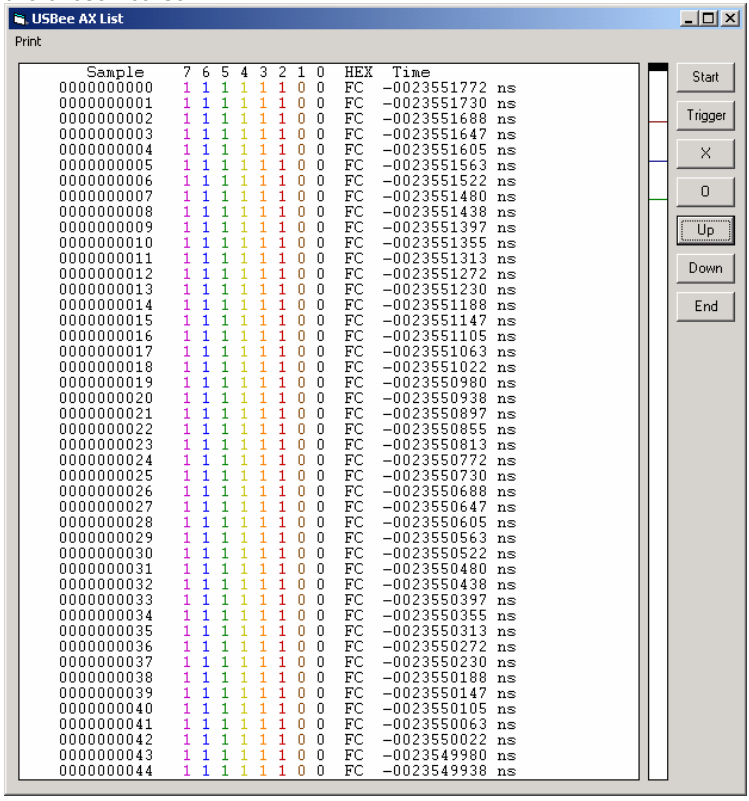
- **X Position** – time at the X1 cursor relative to the selected cursor
- **O Position** – time at the X2 cursor relative to the selected cursor
- **X to O -** difference between X and O cursors

There are also a set of automatic measurements that are made on the waveform by moving the mouse pointer over the waves with the following modes turned on. These are calculated without the use of the cursors. These are:

- **Width** – the width of a pulse
- **Frequency** – the frequency of an periodic cycle
- **Period** – the period of periodic cycle
- **Byte** – the hex value of all 8 signals at a given sample

5.3.6 List Display

You can also display the captured data in a list format that details each sample that was taken by breaking it down by digital value (0 or 1), sample number, 8-bit Hex representation and sample time relative to the chosen cursor.



Sample	7	6	5	4	3	2	1	0	HEX	Time
0000000000	1	1	1	1	1	1	0	0	FC	-0023551772 ns
0000000001	1	1	1	1	1	1	0	0	FC	-0023551730 ns
0000000002	1	1	1	1	1	1	0	0	FC	-0023551688 ns
0000000003	1	1	1	1	1	1	0	0	FC	-0023551647 ns
0000000004	1	1	1	1	1	1	0	0	FC	-0023551605 ns
0000000005	1	1	1	1	1	1	0	0	FC	-0023551563 ns
0000000006	1	1	1	1	1	1	0	0	FC	-0023551522 ns
0000000007	1	1	1	1	1	1	0	0	FC	-0023551480 ns
0000000008	1	1	1	1	1	1	0	0	FC	-0023551438 ns
0000000009	1	1	1	1	1	1	0	0	FC	-0023551397 ns
0000000010	1	1	1	1	1	1	0	0	FC	-0023551355 ns
0000000011	1	1	1	1	1	1	0	0	FC	-0023551313 ns
0000000012	1	1	1	1	1	1	0	0	FC	-0023551272 ns
0000000013	1	1	1	1	1	1	0	0	FC	-0023551230 ns
0000000014	1	1	1	1	1	1	0	0	FC	-0023551188 ns
0000000015	1	1	1	1	1	1	0	0	FC	-0023551147 ns
0000000016	1	1	1	1	1	1	0	0	FC	-0023551105 ns
0000000017	1	1	1	1	1	1	0	0	FC	-0023551063 ns
0000000018	1	1	1	1	1	1	0	0	FC	-0023551022 ns
0000000019	1	1	1	1	1	1	0	0	FC	-0023550980 ns
0000000020	1	1	1	1	1	1	0	0	FC	-0023550938 ns
0000000021	1	1	1	1	1	1	0	0	FC	-0023550897 ns
0000000022	1	1	1	1	1	1	0	0	FC	-0023550855 ns
0000000023	1	1	1	1	1	1	0	0	FC	-0023550813 ns
0000000024	1	1	1	1	1	1	0	0	FC	-0023550772 ns
0000000025	1	1	1	1	1	1	0	0	FC	-0023550730 ns
0000000026	1	1	1	1	1	1	0	0	FC	-0023550688 ns
0000000027	1	1	1	1	1	1	0	0	FC	-0023550647 ns
0000000028	1	1	1	1	1	1	0	0	FC	-0023550605 ns
0000000029	1	1	1	1	1	1	0	0	FC	-0023550563 ns
0000000030	1	1	1	1	1	1	0	0	FC	-0023550522 ns
0000000031	1	1	1	1	1	1	0	0	FC	-0023550480 ns
0000000032	1	1	1	1	1	1	0	0	FC	-0023550438 ns
0000000033	1	1	1	1	1	1	0	0	FC	-0023550397 ns
0000000034	1	1	1	1	1	1	0	0	FC	-0023550355 ns
0000000035	1	1	1	1	1	1	0	0	FC	-0023550313 ns
0000000036	1	1	1	1	1	1	0	0	FC	-0023550272 ns
0000000037	1	1	1	1	1	1	0	0	FC	-0023550230 ns
0000000038	1	1	1	1	1	1	0	0	FC	-0023550188 ns
0000000039	1	1	1	1	1	1	0	0	FC	-0023550147 ns
0000000040	1	1	1	1	1	1	0	0	FC	-0023550105 ns
0000000041	1	1	1	1	1	1	0	0	FC	-0023550063 ns
0000000042	1	1	1	1	1	1	0	0	FC	-0023550022 ns
0000000043	1	1	1	1	1	1	0	0	FC	-0023549980 ns
0000000044	1	1	1	1	1	1	0	0	FC	-0023549938 ns

5.3.7 File Save and Open

Using the File menu functions, you can save and open a current set of configuration and trace sample data.

Choose the menu item File | Save As to save the current configuration and sample data to a binary ULB file. The format of this ULB file follows.

To load a previously saved waveform and display it, choose File | Open and specify the filename to load. This waveform will then be displayed as it was saved.

5.3.7.1 ULB File Format

Bytes	Description
20	Signal 0
20	Signal 1
20	Signal 2
20	Signal 3
20	Signal 4
20	Signal 5
20	Signal 6
20	Signal 7
1	Sample Rate (247, 167,...)
4	Buffer Size in bytes (BufSize)
1	SG Trigger Mode (don't care, high, low, rising, falling)
1	Loop
1	LA Trigger Setting Signal 0 Number 1
1	LA Trigger Setting Signal 1 Number 1
1	LA Trigger Setting Signal 2 Number 1
1	LA Trigger Setting Signal 3 Number 1
1	LA Trigger Setting Signal 4 Number 1
1	LA Trigger Setting Signal 5 Number 1
1	LA Trigger Setting Signal 6 Number 1
1	LA Trigger Setting Signal 7 Number 1
1	LA Trigger Setting Signal 0 Number 2
1	LA Trigger Setting Signal 1 Number 2
1	LA Trigger Setting Signal 2 Number 2
1	LA Trigger Setting Signal 3 Number 2
1	LA Trigger Setting Signal 4 Number 2
1	LA Trigger Setting Signal 5 Number 2
1	LA Trigger Setting Signal 6 Number 2
1	LA Trigger Setting Signal 7 Number 2
1	LA Trigger Setting Signal 0 Number 3
1	LA Trigger Setting Signal 1 Number 3
1	LA Trigger Setting Signal 2 Number 3
1	LA Trigger Setting Signal 3 Number 3
1	LA Trigger Setting Signal 4 Number 3
1	LA Trigger Setting Signal 5 Number 3
1	LA Trigger Setting Signal 6 Number 3
1	LA Trigger Setting Signal 7 Number 3
1	LA Trigger Setting Signal 0 Number 4
1	LA Trigger Setting Signal 1 Number 4
1	LA Trigger Setting Signal 2 Number 4
1	LA Trigger Setting Signal 3 Number 4
1	LA Trigger Setting Signal 4 Number 4
1	LA Trigger Setting Signal 5 Number 4
1	LA Trigger Setting Signal 6 Number 4
1	LA Trigger Setting Signal 7 Number 4
1	LA Clocking Mode (Internal/External)
4	Prestore Setting
4	Trigger Position (sample number at trigger)

4 Center Display Position (sample number at center of screen)
4 Scale Value
4 SubScale Value
4 X1Cursor Position (sample number at X1Cursor)
4 OCursor Position (sample number at OCursor)
4 Pod ID used
BufSize Digital channel samples(one byte per sample - each bit is a signal - bit 0 = signal 0)

5.3.8 Printing

You can print the current screen to any printer by choosing the File | Print menu item.

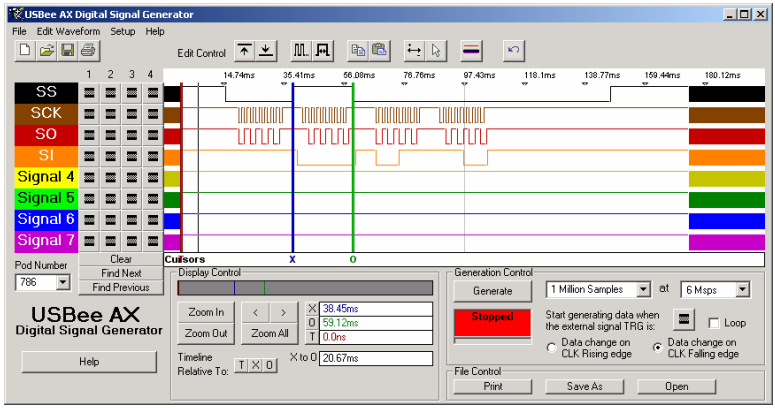
5.3.9 USB, I2C, Async, and SPI Decoders

Using these View menu functions, you can decode these serial busses and extract the actual data transmitted via the protocols. These features are detailed in a later section of this document will only function on the USBee AX-Plus and USBee AX-Pro models.

6 Digital Signal Generator

AX-Plus and AX-Pro Only

This section details the operation of the Digital Signal Generator application that comes with the USBee AX. Below you see the application screen.



6.1 Digital Signal Generator Specifications

Digital Output Channels	8
Maximum Digital Sample Rate [1]	24 Msps
Internal Clocking	Yes
External Clocking	No
Number of Samples [2]	1 million samples up to PC RAM
Sample Rates [1]	1Msps to 24 Msps
Sample Clock Output	Yes
Channel Output Drive Current	4mA
Output Low Level	< 0.8V
Output High Level	> 2.4V
Looping	Yes
External Trigger Signal	Yes

6.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to generate a set of digital waveforms.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect any of the Signal 0 thru 7 pins on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to your circuit you would like to actively drive.
- Run the Signal Generator Application.
- Draw a waveform you want to generate using the waveform edit controls at the top of the waveform window.
- Press the Generate button. This will generate the waveform you just drew on the pod signals.

6.3 Features

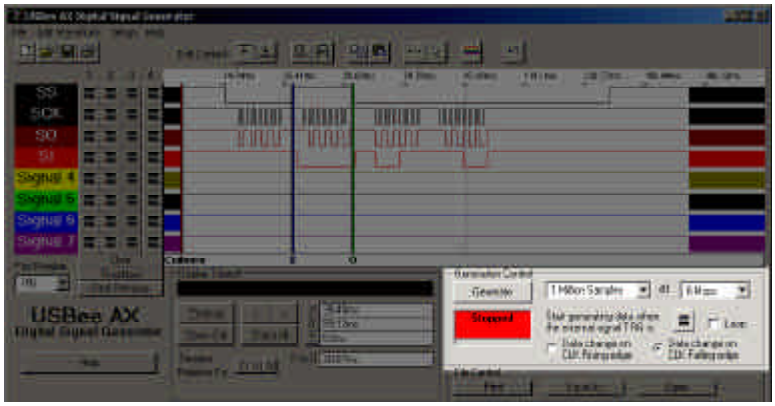
6.3.1 Pod Status

The Signal Generator display shows a list with the available **Pod ID List** for all of the USBee Ax's that are connected to your PC. You can choose which one you want to use. The others will be unaffected. If a USBee AX is not connected, the list box will read Demo to indicate that there is no pod attached.

If you run the software with no pod attached, it will run in demonstration mode so that you can still see how the software functions.

6.3.2 Generation Control

The Signal Generator lets you draw the behavior of digital signals and the generates them as a "trace" on the pod signals. The Generation Control section of the display lets you choose how the traces are generated. Below is the Generation Control section of the display.



The **Generate** button starts and stops a data output. When the signal generator is first started, the Generate button is not pressed and is waiting for you to draw a waveform. The Generate button outputs a single trace and stops, unless you check the **Loop** box.

The **Buffer Size** lets you select the size of the Sample Buffer that is used. For each trace, the buffer is completely played back. No partial buffers can be generated. You can choose buffers that will hold the information that you want to output, but remember that the larger the buffer, the longer it will take to generate.

You can also choose the **Sample Rate** that you want samples to be aligned to. This uses an **Internal Clock** at that sample rate you choose. You can choose from 1 Msps (samples per second) to up to 24 Msps. The actual maximum sample rate depends on your PC configuration. You can run the menu item Setup | Sample Rate Test to determine the maximum sample rate for your system.

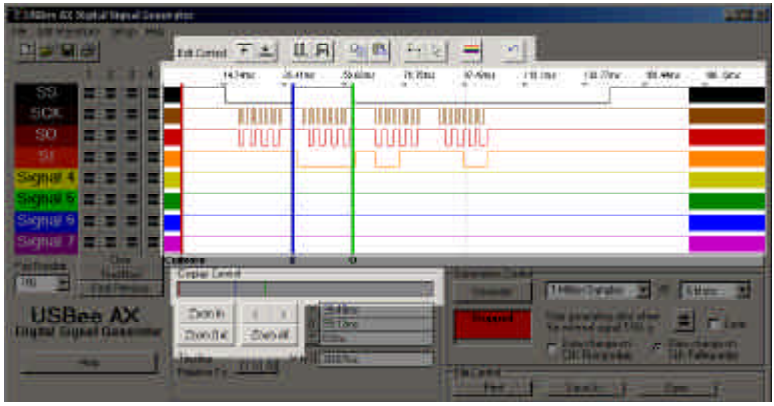
While the pod is generating the waveform on the pod signals, the CLK line is an output and toggles once for each of the samples provided. You can specify the **CLK Edge** that the output data changes on using the two radio buttons above.

The TRG signal can be used as an **External Trigger** for the pattern generation. Select the state of the TRG signal you want to start the output on by pressing the toggle pushbutton above.

The **Status Box** on the display will show red when the unit is not outputting samples, flash blue when it is waiting for a trigger, and glow green when the trigger condition has been met. It will glow red again when the generation is completed.

6.3.3 Waveform Edit, Display and Zoom Settings

The Waveform display area is where the signal information is shown. It is displayed with time increasing from left to right and voltage increasing from bottom to top. The screen is divided into **Divisions** to help in measuring the waveforms.



To **Scroll the Waveforms in Time** left and right, you can use the left and right arrows highlighted above, click and drag the Overview Bar (right under the Display Control title), or you can simply click and drag the waveform itself.

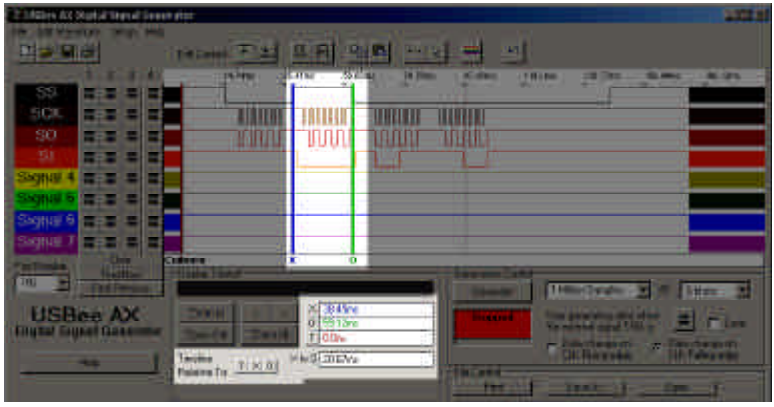
To change the zoom ratio for the time, click the **Zoom In** or **Zoom Out** buttons. You can also zoom in and out in time by clicking on the waveform. To zoom in, click the left mouse on the waveform window. To zoom out in time, click the right mouse button on the waveform window.

The cursor in the waveform window can be in one of two modes: **Pan and Zoom**, or **Select**. In pan and zoom, you can click and drag the waveform around on the screen. In Select, you click and drag to select a portion of the waveform to edit. Change modes by clicking the left-right arrow (pan and zoom), or the standard arrow (select).

Editing the Waveform is done by selecting the portion of the waveform by clicking and dragging to highlight a section, and then pressing one of the Edit Control buttons at the top. You can set the specified samples to a high level, low level, create a clock on that signal, create a single pulse, or copy and paste. You can also **Undo** the last change if needed.

6.3.4 Measurements and Cursors

To help you create time accurate waveforms, the cursors can be used to get exact timing.



The **X and O Cursors** are placed on any horizontal sample time. This lets you measure the time at a specific location or the time between the two cursors. To place the X and O cursors, move the mouse to the white box just below the waveform. When you move the mouse in this window, you will see a temporary line that indicates where the cursors will be placed. Place the X cursor by left clicking the mouse at the current location. Place the O cursor by right clicking the mouse at the current location.

In the Measurement window, you will see the various measurements made off of these cursors. To change the selected relative cursor, click the T,X or O buttons next to the "Timeline Relative To" text.

- **X Position** – time at the X1 cursor relative to the selected cursor
- **O Position** – time at the X2 cursor relative to the selected cursor
- **X to O -** difference between X and O cursors

6.3.5 File Save and Open

Using the File menu functions, you can save and open a current set of configuration and trace sample data.

Choose the menu item File | Save As to save the current configuration and sample data to a binary ULB file. The format of this ULB file follows.

To load a previously saved waveform and display it, choose File | Open and specify the filename to load. This waveform will then be displayed as it was saved.

6.3.5.1 ULB File Format

Bytes	Description
20	Signal 0
20	Signal 1
20	Signal 2
20	Signal 3
20	Signal 4
20	Signal 5
20	Signal 6
20	Signal 7
1	Sample Rate (247, 167,...)
4	Buffer Size in bytes (BufSize)
1	SG Trigger Mode (don't care, high, low, rising, falling)
1	Loop
1	LA Trigger Setting Signal 0 Number 1
1	LA Trigger Setting Signal 1 Number 1
1	LA Trigger Setting Signal 2 Number 1
1	LA Trigger Setting Signal 3 Number 1
1	LA Trigger Setting Signal 4 Number 1
1	LA Trigger Setting Signal 5 Number 1
1	LA Trigger Setting Signal 6 Number 1
1	LA Trigger Setting Signal 7 Number 1
1	LA Trigger Setting Signal 0 Number 2
1	LA Trigger Setting Signal 1 Number 2
1	LA Trigger Setting Signal 2 Number 2
1	LA Trigger Setting Signal 3 Number 2
1	LA Trigger Setting Signal 4 Number 2
1	LA Trigger Setting Signal 5 Number 2
1	LA Trigger Setting Signal 6 Number 2
1	LA Trigger Setting Signal 7 Number 2
1	LA Trigger Setting Signal 0 Number 3
1	LA Trigger Setting Signal 1 Number 3
1	LA Trigger Setting Signal 2 Number 3
1	LA Trigger Setting Signal 3 Number 3
1	LA Trigger Setting Signal 4 Number 3
1	LA Trigger Setting Signal 5 Number 3
1	LA Trigger Setting Signal 6 Number 3
1	LA Trigger Setting Signal 7 Number 3
1	LA Trigger Setting Signal 0 Number 4
1	LA Trigger Setting Signal 1 Number 4
1	LA Trigger Setting Signal 2 Number 4
1	LA Trigger Setting Signal 3 Number 4
1	LA Trigger Setting Signal 4 Number 4
1	LA Trigger Setting Signal 5 Number 4
1	LA Trigger Setting Signal 6 Number 4
1	LA Trigger Setting Signal 7 Number 4
1	LA Clocking Mode (Internal/External)
4	Prestore Setting
4	Trigger Position (sample number at trigger)

4 Center Display Position (sample number at center of screen)
4 Scale Value
4 SubScale Value
4 X1Cursor Position (sample number at X1Cursor)
4 OCursor Position (sample number at OCursor)
4 Pod ID used
BufSize Digital channel samples(one byte per sample - each bit is a signal - bit 0 = signal 0)

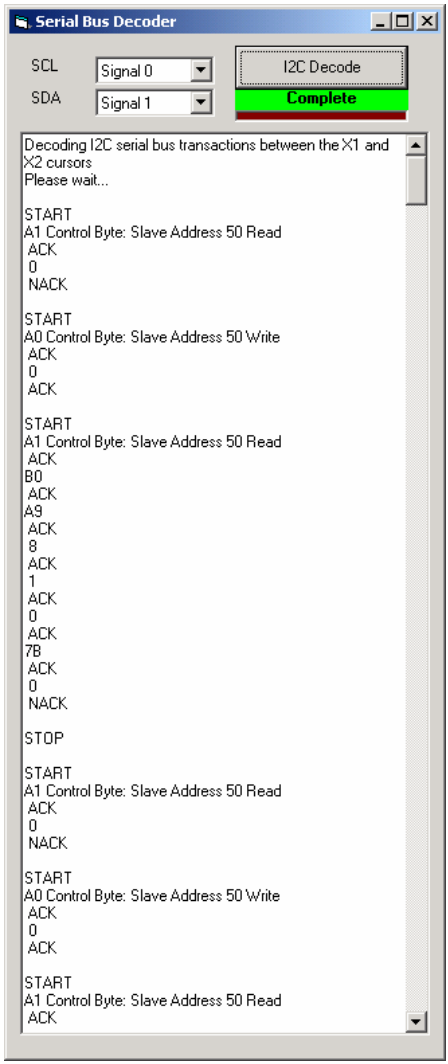
6.3.6 Printing

You can print the current screen to any printer by choosing the File | Print menu item.

7 I2C Decoder

AX-Plus and AX-Pro Only

This section details the operation of the I2C Decoder function that is part of the Logic Analyzer application that comes with the USBee AX. Below you see the application screen.



7.1 I2C Decoder Specifications

I2C Clock Speed	up to 12MHz
I2C Data Decoded	Start,Stop,Ack,Nak,Data
Decoder Output Format	Text File
Decoder Location	Logic Analyzer and Mixed Signal Scope

7.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view I2C data from your design.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 pin on the USBee AX pod to your circuit SDA line.
- Connect the Signal 1 pin on the USBee AX pod to your circuit SCL line.
- Run the Logic Analyzer Application.
- Press the Run Button when your I2C bus is running.
- Position the X and O (or X1 and X2) cursors at the beginning and end of where you want to decode.
- Press View | I2C Decoder in the menu.
- Select the correct SDA and SCL lines in the dropdown box.
- Press the Decode Button.
- The Text Box will be filled with the decoded bus data.

7.3 Decoder Details

The I2C bus decoder is part of the Logic Analyzer or Mixed Signal Oscilloscope applications. You can access these through the View menu.

You first must capture a trace of data that contains the I2C bus signals, SDA and SCL. The I2C bus is Open Collector, meaning that you must have an external pull-up resistor on each of the SDA and SCL lines. You most likely already have one on your circuit design.

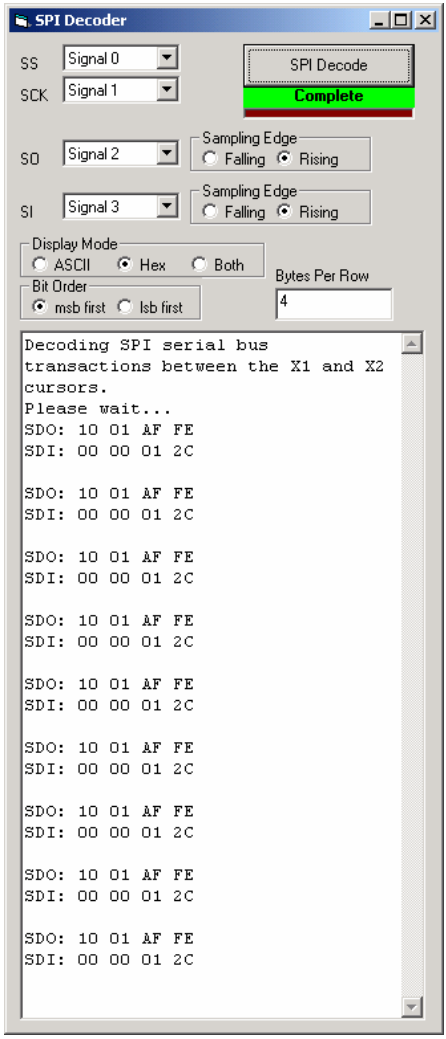
Once you have the trace captured, you can then run the decoder to extract the data. You can select any of the eight lines for the SDA or SCL lines.

The protocol and data is then extracted to the text box. You can then select, copy and paste this data into any other program for formatting or processing.

8 SPI Decoder

AX-Plus and AX-Pro Only

This section details the operation of the SPI Decoder function that is part of the Logic Analyzer application that comes with the USBee AX. Below you see the application screen.



8.1 SPI Decoder Specifications

SPI Clock Speed	up to 12MHz
Async Baud Rate	up to 12Mbaud in 1baud steps
Decoder Output Format	Text File
Decoder Location	Logic Analyzer and Mixed Signal Scope

8.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view SPI data from your design.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 pin on the USBee AX pod to your circuit Slave Select line.
- Connect the Signal 1 pin on the USBee AX pod to your circuit SPI Clock line.
- Connect the Signal 2 pin on the USBee AX pod to your circuit SPI Output Data line.
- Connect the Signal 3 pin on the USBee AX pod to your circuit SPI Input Data line.
- Run the Logic Analyzer Application.
- Press the Run Button when your SPI bus is running.
- Position the X and O (or X1 and X2) cursors at the beginning and end of where you want to decode.
- Press View | SPI Decoder in the menu.
- Select the correct SS, SCK, SI and SO lines in the dropdown boxes.
- Press the Decode Button.
- The Text Box will be filled with the decoded bus data.

8.3 Decoder Details

The SPI bus decoder is part of the Logic Analyzer or Mixed Signal Oscilloscope applications. You can access these through the View menu.

You first must capture a trace of data that contains the SPI bus signals, SS, SCK, SI and SO. Once you have the trace captured, you can then run the decoder to extract the data. You can select any of the eight lines for the four protocol lines.

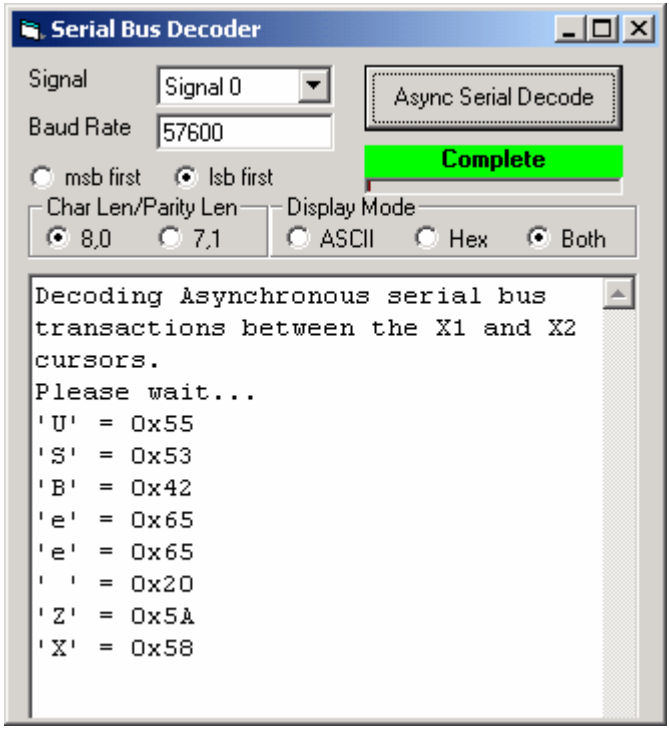
You can specify which edge the data lines are to be sampled on (opposite of the data changing edge). You can also choose the output data format and how many bytes to write out per line.

The protocol and data is then extracted to the text box. You can then select, copy and paste this data into any other program for formatting or processing.

9 Async Serial Decoder

AX-Plus and AX-Pro Only

This section details the operation of the Async Decoder function that is part of the Logic Analyzer application that comes with the USBee AX. Below you see the application screen.



9.1 Async Serial Decoder Specifications

Async Baud Rate	up to 12Mbaud in 1baud steps
Decoder Output Format	Text File
Decoder Location	Logic Analyzer and Mixed Signal Scope

9.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view Async data from your design.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 pin on the USBee AX pod to your circuit Async line.
- Run the Logic Analyzer Application.
- Press the Run Button when your Async bus is running.
- Position the X and O (or X1 and X2) cursors at the beginning and end of where you want to decode.
- Press View | Async Decoder in the menu.
- Select the correct Async line in the dropdown boxes, enter the baud rate, select the bit order and data bits/parity.
- Press the Decode Button.
- The Text Box will be filled with the decoded bus data.

9.3 Decoder Details

The Async bus decoder is part of the Logic Analyzer or Mixed Signal Oscilloscope applications. You can access these through the View menu.

You first must capture a trace of data that contains the Async bus signal. Once you have the trace captured, you can then run the decoder to extract the data. You can select any of the eight lines for the decode.

You can specify the baud rate to 1 baud, select msbit or lsbit first, data/parity combinations and output display format of your choosing.

Once the decode button is pressed, the protocol and data is then extracted to the text box. You can then select, copy and paste this data into any other program for formatting or processing.

10 USB Decoders

AX-Plus and AX-Pro Only

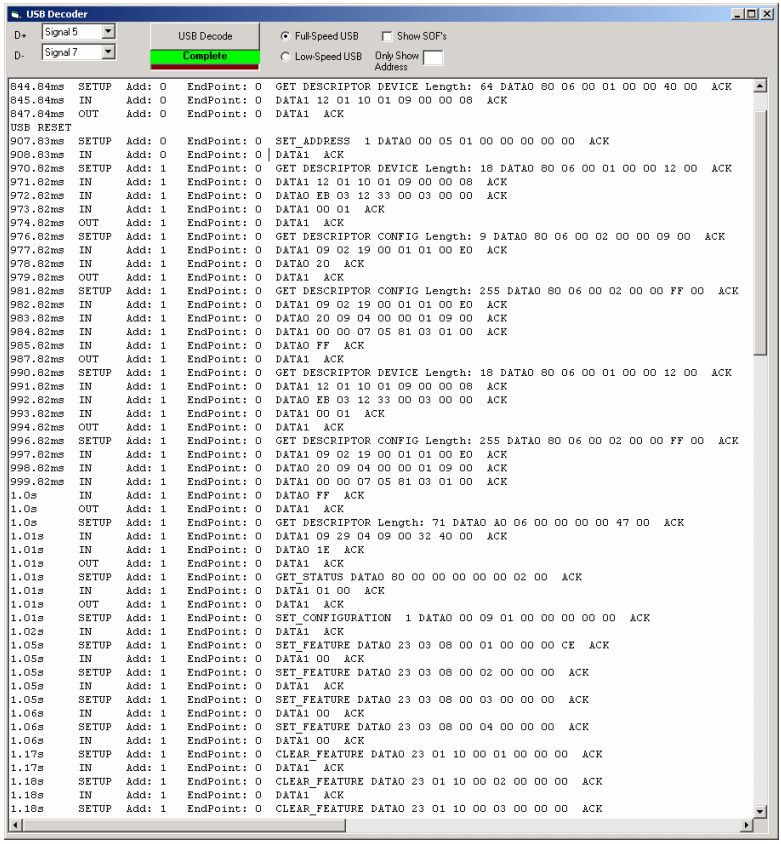
This section details the operation of the USB Decoder functions that are part of the Logic Analyzer and Mixed Signal Oscilloscope application that comes with the USBee AX.

The USB Decoder that is part of the **Logic Analyzer** can decode **Full and Low Speed USB**. Since Full Speed USB is 12Mbps signaling, the Logic Analyzer must run at 24Msps for the decoder to work.

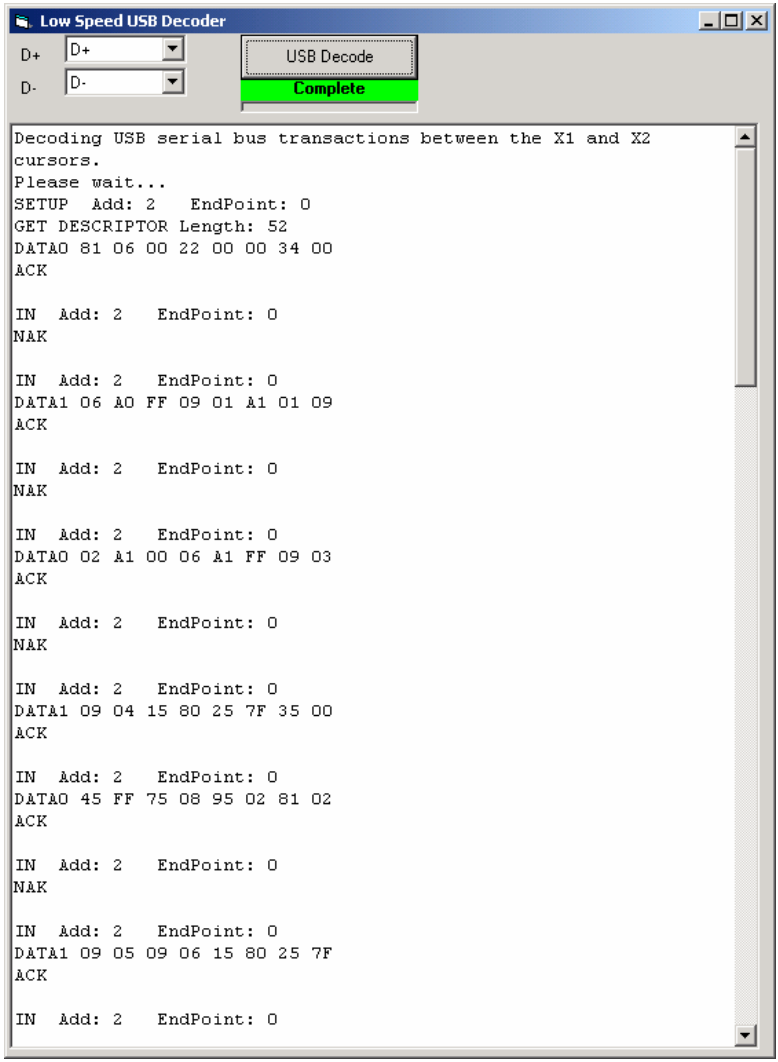
You use the X and O cursors to select the area you want to decode, select Full or Low Speed and press USB Decode. It will then extract the protocol and show what is happening on the bus. You can choose to hide SOFs by unchecking the **Show SOF** checkbox. You can also filter the display to only show a certain Device Address transfers. Enter the device address you want to see in the **Only Show Address** box. Nothing or zero in this box show all transfers.

The time for the start of each of the transfers is shown at the left of the output screen. This time correlates to the waveform screen so that you can lineup the events.

Below you see the application screen for the Full and Low Speed USB Decoder.



The USB Decoder that is part of the **Mixed Storage Oscilloscope** is limited to the **Low Speed USB** which runs at 1.5Mbps since the maximum sample rate of the digital signals is 16Msps. This is too slow for Full-Speed USB sampling. The decoding is the same as the above function except that it only operates at Low Speed USB. Below you see the application screen.


Low Speed USB Decoder
D+
D+
D-
D-
USB Decode
Complete
Decoding USB serial bus transactions between the X1 and X2
cursors.
Please wait...
SETUP Add: 2 EndPoint: 0
GET DESCRIPTOR Length: 52
DATA0 81 06 00 22 00 00 34 00
ACK
IN Add: 2 EndPoint: 0
NAK
IN Add: 2 EndPoint: 0
DATA1 06 A0 FF 09 01 A1 01 09
ACK
IN Add: 2 EndPoint: 0
NAK
IN Add: 2 EndPoint: 0
DATA0 02 A1 00 06 A1 FF 09 03
ACK
IN Add: 2 EndPoint: 0
NAK
IN Add: 2 EndPoint: 0
DATA1 09 04 15 80 25 7F 35 00
ACK
IN Add: 2 EndPoint: 0
DATA0 45 FF 75 08 95 02 81 02
ACK
IN Add: 2 EndPoint: 0
NAK
IN Add: 2 EndPoint: 0
DATA1 09 05 09 06 15 80 25 7F
ACK
IN Add: 2 EndPoint: 0

10.1 USB Low Speed Decoder Specifications

USB Bus Speed	Low Speed Only
USB Data Decoded	Setup,Data0,Data1,Ack,Nak,In,Out,Ep,Address
Decoder Output Format	Text File
Decoder Location	Mixed Signal Scope

10.2 USB Full and Low Speed Decoder Specifications

USB Bus Speed	Full and Low USB Speed Only
USB Data Decoded	Setup,Data0,Data1,Ack,Nak,In,Out,Ep,Address, Standard Device Requests
Decoder Output Format	Text File
Decoder Location	Logic Analyzer (Must run at 24Msps for Full Speed USB decoding)

10.3 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to view USB data from your design.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 pin on the USBee AX pod to your circuit D+ line.
- Connect the Signal 1 pin on the USBee AX pod to your circuit D- line.
- Run the Logic Analyzer Application.
- Set the sample rate to 24Msps for Full Speed or > 3Msps for Low Speed

- Press the Run Button when your USB bus is running.
- Position the X and O (or X1 and X2) cursors at the beginning and end of where you want to decode.
- Press View | USB Decoder in the menu.
- Select the correct D+ and D- lines in the dropdown boxes.
- Press the Decode Button.
- The Text Box will be filled with the decoded bus data.

10.4 Decoder Details

The USB bus decoder is part of the Logic Analyzer or Mixed Signal Oscilloscope applications. You can access these through the View menu.

You first must capture a trace of data that contains the USB bus signals, D+ and D-. Once you have the trace captured, you can then run the decoder to extract the data. You can select any of the eight lines for the two protocol lines.

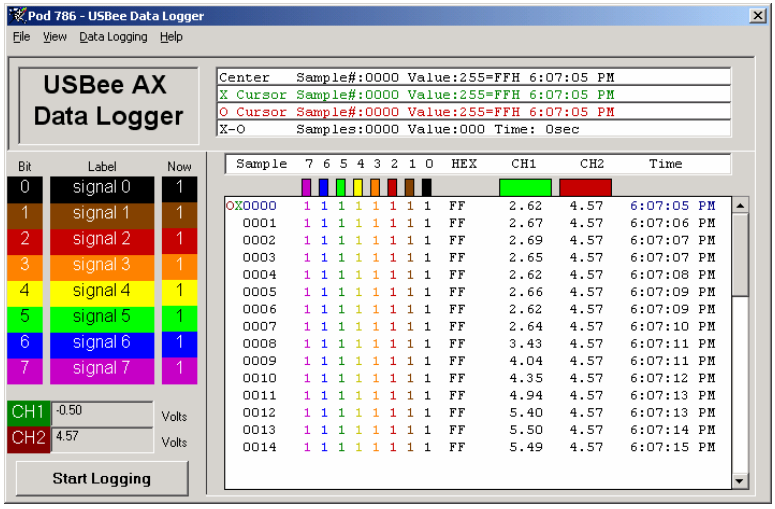
Once you press the Decode button, the protocol and data is then extracted to the text box. You can then select, copy and paste this data into any other program for formatting or processing.

The USB decoder parses out the Endpoints, Data PIDs, Device ID's and decodes the standard Endpoint 0 device requests.

11 Data Logger

AX-Pro Only

This section details the operation of the Data Logger application that comes with the USBee AX. Below you see the application screen.



11.1 Data Logger Specifications

Digital Channels Logged	8
Analog Channels Logged	2
Sample Rates	250ms to 300sec

11.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to log analog and digital data.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a

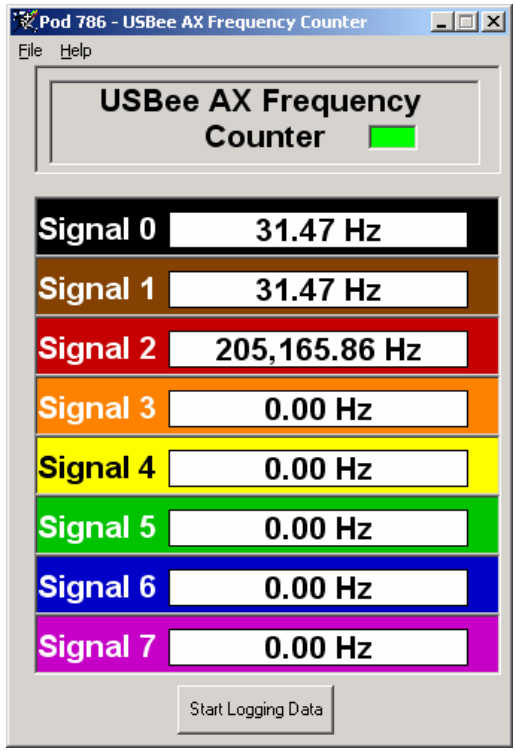
header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.

- Connect the CH1 and/or CH2 pins on the USBee AX pod to one of the signal wires you would like to test.
- Connect the digital Signal 0 thru 7 pins on the USBee AX pod to one of the signal wires you would like to test.
- Run the Data Logger Application.
- Press the Start Logging button. Select the sample rate from the list box and press OK. Select the filename for the logged data to be exported to and press OK.
- This will start the logging process. When you are finished, press the Stop Logging button.
- The data is then displayed in the list format for review. You can also post process the text based log file using other programs.

12 Frequency Counter

AX-Pro Only

This section details the operation of the Frequency Counter application that comes with the USBee AX. Below you see the application screen.



12.1 *Frequency Counter Specifications*

Digital Channels Measured	8
Analog Channels Measured	0
Maximum Measured Frequency [1]	12MHz
Maximum Digital Input Voltage	+5.5V
Resolution	1Hz
Gate Time	1 sec

12.2 Quick Start

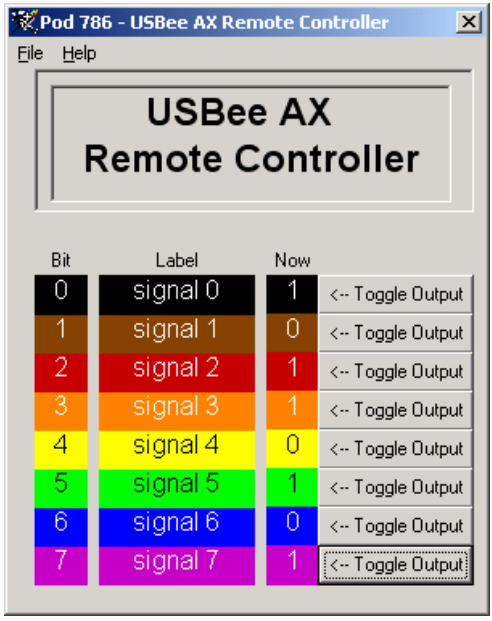
In order to quickly get up and running using this application, here is a step by step list of the things you need to do to measure the frequency of a digital signal.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 thru 7 signals on the USBee AX pod to your circuit you would like to test.
- Run the Frequency Counter Application.
- The frequency of each of the 8 signal lines will then be displayed.
- You can log the frequency data to a file by pressing the “Start Logging Data” button.

13 Remote Controller

AX-Pro Only

This section details the operation of the Remote Controller application that comes with the USBee AX. Below you see the application screen.



13.1 Remote Controller Specifications

Digital Channels Controlled	8
Analog Channels Controlled	0
Control Mechanism	Toggle Button per channel
Channel Output Drive Current	4mA
Output Low Level	< 0.8V
Output High Level	> 2.4V

13.2 Quick Start

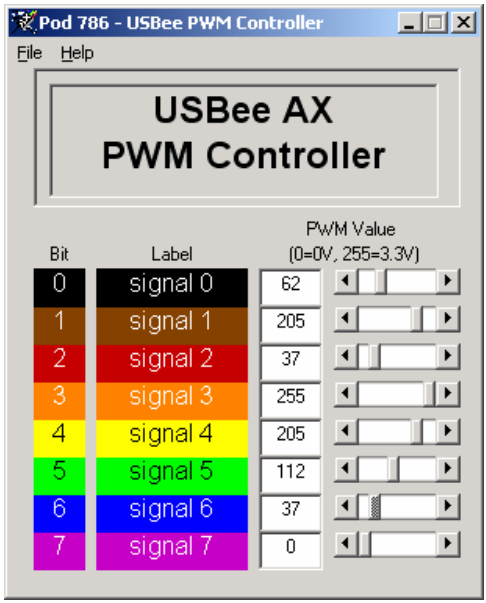
In order to quickly get up and running using this application, here is a step by step list of the things you need to do to control the output of each of the digital signal lines.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 thru 7 lines on the USBee AX pod to your circuit you would like to actively drive.
- Run the Remote Controller Application.
- Press any of the Toggle buttons and the level of the output will toggle (Low to High, High to Low)..

14 PWM Controller

AX-Pro Only

This section details the operation of the Pulse Width Modulator application that comes with the USBee AX. Below you see the application screen.



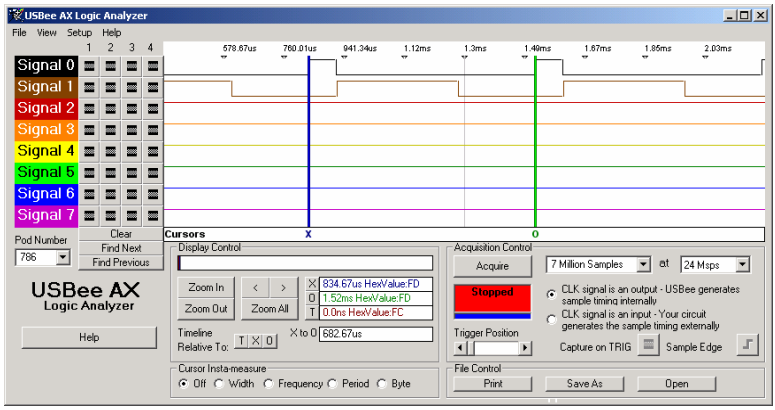
14.1 PWM Controller Specifications

Digital Channels Controlled	8
Analog Channels Controlled	0
Resolution	256 steps
PWM Frequency	1.46kHz
Control Mechanism	Slider Switch
Channel Output Drive Current	4mA
Output Low Level	< 0.8V
Output High Level	> 2.4V

14.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to create 8 PWM signals.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 thru 7 lines on the USBee AX pod to your circuit you would like to actively drive with a PWM signal.
- Run the PWM Controller Application.
- Use the scroll bars to set the desired PWM level, with 0 being all low and 255 being all high outputs.

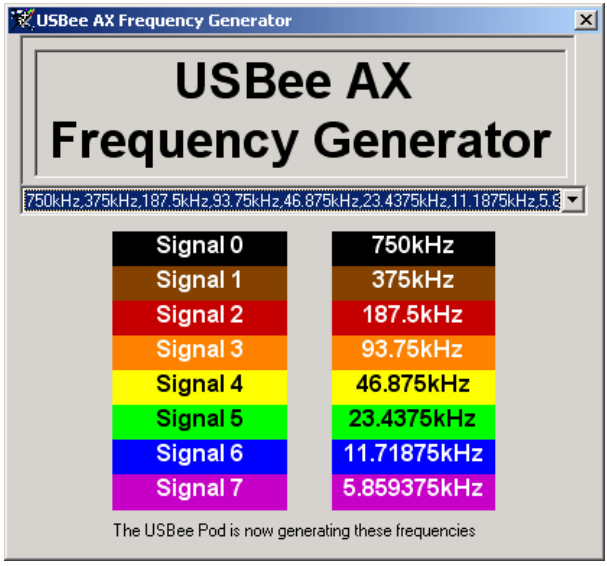


The above shows 2 outputs of the PWM Controller. Signal 0 shows the PWM value set to 31 (out of 255) and Signal 1 shows the PWM value of 137. A value of 0 is all low, and a value of 255 is all high.

15 Frequency Generator

AX-Pro Only

This section details the operation of the Frequency Generator application that comes with the USBee AX. Below you see the application screen.



15.1 Frequency Generator Specifications

Digital Channels Controlled	8
Analog Channels Controlled	0
Sets of Frequencies	6
Set 1	1MHz, 500kHz, 250kHz, 62.5kHz,31.25kHz, 15.625kHz, 7.8125kHz
Set 2	32kHz, 16kHz, 8kHz, 4kHz, 2kHz, 1kHz, 500Hz, 250Hz
Set 3	750kHz, 375kHz, 187.5kHz, 93.75kHz, 46.875kHz, 23.4375kHz, 11.1875kHz,

	5.5893kHz
Set 4	19.2kHz, 9600Hz, 4800Hz, 2400Hz, 1200Hz, 600Hz, 300Hz, 150Hz
Set 5	64Hz, 32Hz, 16Hz, 8Hz, 4Hz, 2Hz, 1Hz, 0.5Hz
Set 6	1920Hz, 960Hz, 480Hz, 240Hz, 120Hz, 60Hz, 30Hz, 15Hz
Channel Output Drive Current	4mA
Output Low Level	< 0.8V
Output High Level	> 2.4V

15.2 Quick Start

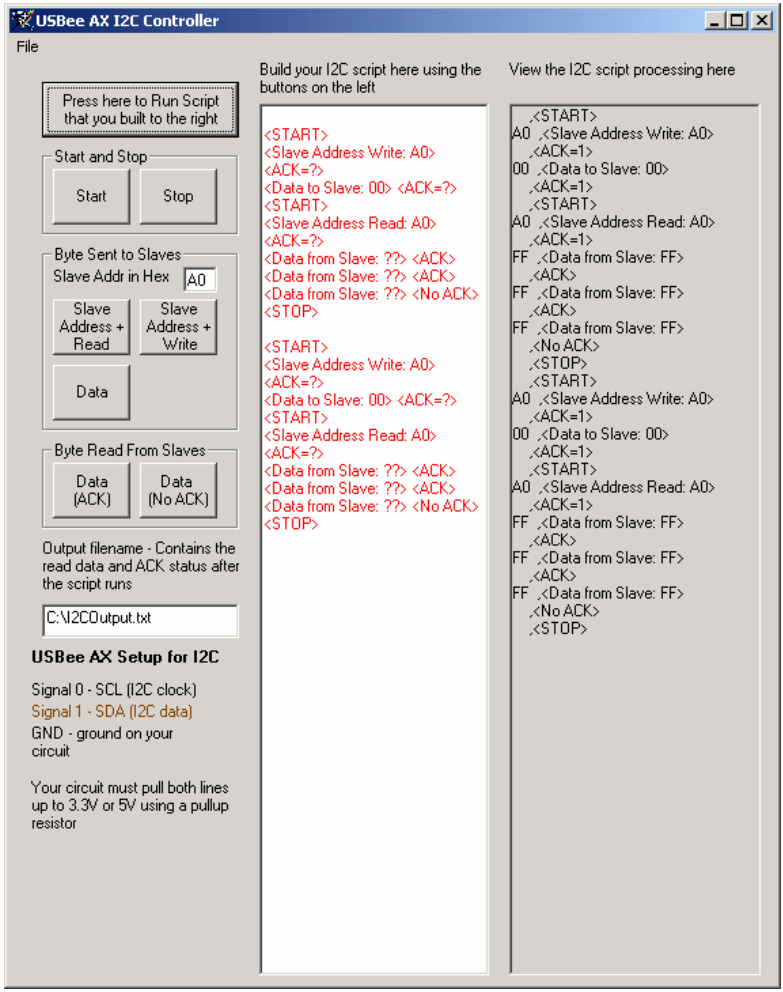
In order to quickly get up and running using this application, here is a step by step list of the things you need to do to generate one of the fixed sets of frequencies on the digital lines.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 thru 7 lines on the USBee AX pod to your circuit you would like to actively drive.
- Run the Frequency Generator Application.
- From the dropdown list, select the set of frequencies that you want to generate out the pod.
- These frequencies are now being generated on the pod digital signals.

16 I2C Controller

AX-Pro Only

This section details the operation of the I2C Controller application that comes with the USBee AX. Below you see the application screen.



16.1 I2C Controller Specifications

I2C Clock Speed	2.2 KHz average
I2C Control Method	Text Script
I2C Script Tokens	Start, Stop, Ack, Nak, Read, Write, Data
Script Edit Functions	Cut, Copy, Paste, Save, Open, New
I2C Output Format	Text File (includes read data and Ack state)
Channel Output Drive Current	4mA
Output Low Level	< 0.8V
Output High Level	Open Collector (requires external pull-up resistor)

16.2 Quick Start

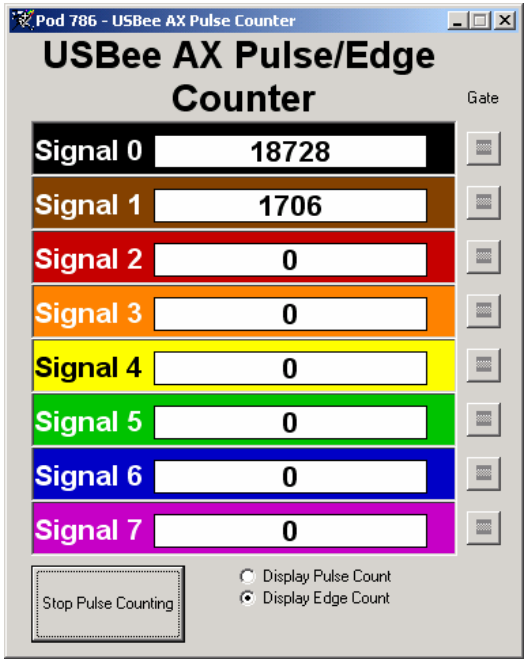
In order to quickly get up and running using this application, here is a step by step list of the things you need to do to generate I2C transactions.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 pin on the USBee AX pod to your circuit SDA line.
- Connect the Signal 1 pin on the USBee AX pod to your circuit SCL line.
- Run the I2C Controller Application.
- Press the buttons to create a script of the I2C transaction you want to run.
- Press the Run Script button to generate the I2C transaction.
- The transaction result is written to the output window (and text file) including and read data and ACK states..

17 Pulse Counter

AX-Pro Only

This section details the operation of the Pulse Counter application that comes with the USBee AX. Below you see the application screen.



17.1 Pulse Counter Specifications

Digital Channels Measured	8
Analog Channels Measured	0
Minimum Pulse Width [1]	83.3nS
Pulse Count Control	Clear, Start and Stop
Display Mode	Pulse or Edge Count
External Gate Signals	up to 7
Gate Conditions	High or Low

17.2 Quick Start

In order to quickly get up and running using this application, here is a step by step list of the things you need to do to count the number of edges or pulses of a digital signal.

- Connect the GND pin on the USBee AX pod to one of the signal wires using the small socket on the end of the wire.
- Connect the other end of the wire to the Ground of your circuit you would like to test. You can either use the socket to plug onto a header post, or connect it to one of the mini-grabber clips and then attach it to the Ground.
- Connect the Signal 0 thru 7 signals on the USBee AX pod to your circuit you would like to test.
- Run the Pulse Counter Application.
- Press the Start Counting button.
- The number of pulses one each of the 8 digital signals is displayed.
- You can use any of the 8 lines as a gate to enable the counting during specified times. For example, you can count pulses only when Signal 0 is high by setting the Signal 0 Gate to High. Pulses that occur when Signal 0 is low are not counted.

18 USBee Toolbuilder

AX-Pro Only

18.1 Overview

The USBee AX Test Pod System consists of the USBee AX Test Pod connected to a Windows® 2000 or Windows® XP PC High Speed USB 2.0 port through the USB cable, and to your circuit using the multicolored test leads and clips. Once connected and installed, the USBee can then be controlled using either the USBee AX Windows Software or your own USBee AX Toolbuilder software.

The USBee AX system is also expandable by simply adding more USBee AX pods for more channels and combined features.

The USBee AX Test Pod is ideal for students or designers that need to get up and running with High Speed USB immediately. With a mini-B USB connector on one end and signal pin headers on the other, this simple pod will instantly USB 2.0 High-Speed enable your design. Then using the source code libraries, drivers and DLL's that are included here you can write your own Visual Basic or C PC application to control and monitor the signal pins on the pod.

The USBee AX has headers that are the interface to your circuits. The signals on these headers represent an 8 bit data bus, a Read/Write#/TRG signal and a clock line. Using the libraries and source code provided you can do byte-wide reads and writes to these signals. The USBee AX acts as the master, driving the Read/Write#/TRG signals and Clock lines to your circuit.

There are six modes of data transfers that you can use depending on your system needs.

- Voltmeter Mode
- Mixed Signal Scope Capture
- Digital Logic Analyzer Capture
- Digital Signal Generator
- Bi-Directional “bit-bang” mode
- Uni-Directional High Speed mode

18.1.1 Voltmeter Mode

The simplest of the analog functions is the DVM (Digital Voltmeter) routine. It simply measures the voltage on a specified channel. This measurement is taken over a quarter second an the average is returned.

The routine GetAnalogAverageCount() samples the specified channel and returns the measurement.

18.1.2 Mixed Signal Scope Capture

The USBee AX has the ability to capture samples from the 8 digital signals and one of the two analog channels at the same time. Each analog sample is time synchronized with the corresponding digital samples.

In mixed signal mode, there are two sample buffers, one for the digital samples and one for the analog samples. Each buffer is 8 bits/sample. The Digital samples are represented by each bit in the byte. So Digital Signal 0 is bit 0 of each byte. The Analog sample is the 8-bit ADC value taken during that sample period. The samples range from 0 (at -10.0V) to 255 (at +10.0V). Each count of the ADC equates to 78.125mV, which is the lowest resolution possible on the USBee AX without averaging.

The maximum sample rate that is possible in Mixed Signal mode is 16Msps. This value can depend on your PC system and available processing speed.

The method for performing a single data capture, or sampling, using the Mixed Signal routines is as follows:

- Allocate the sample buffers (MakeBuffer16() and MakeBufferScope())
- Start the capture running (StartCaptureMSO(…))
- Monitor the capture in progress to determine if it is triggered, filling, or completed. (CaptureStatus()).
- End the capture when it is finished. (StopCaptureMSO())
- Process the sample data that is now contained in the sample buffers.

The Mixed Signal functions are the only method of doing oscilloscope traces with the USBee AX. This means that if you need just an oscilloscope trace, then you will get the digital samples at the same time.

18.1.3 Digital Logic Analyzer Capture

The USBee AX has the ability to capture samples from the 8 digital signals at up to 24Msps in Logic Analyzer mode. In this mode, each digital signal is represented by each bit in the byte that is stored in the sample buffer. Therefore, digital Signal 0 is bit 0 of each byte. The maximum sample rate can depend on your PC system and available processing speed.

The method for performing a single data capture, or sampling, using the Logic Analyzer routines is as follows:

- Allocate the sample buffer (MakeBuffer())

- Start the capture running (StartCapture(…))
- Monitor the capture in progress to determine if it is triggered, filling, or completed. (CaptureStatus()).
- End the capture when it is finished. (StopCapture())
- Process the sample data that is now contained in the sample buffer.

18.1.4 Digital Signal Generator

The USBee AX has the ability to generate (output) samples from the 8 digital signals at up to 24Msps in Signal Generator mode. In this mode, each digital signal is represented by each bit in the byte that is stored in the sample buffer. Therefore, digital Signal 0 is bit 0 of each byte. These samples can then be generated on command. The maximum sample rate can depend on our PC system and available processing speed.

The method for generating a single output pattern using the Signal Generator routines is as follows:

- Allocate the sample buffer (MakeBuffer())
- Fill the sample buffer with the pattern data you want to generate.
- Start the generation running (StartGenerate (…))
- Monitor the generation in progress to determine if it is triggered, filling, or completed. (GenerateStatus()).
- Terminate the generation. (StopGenerate())

The USBee AX can not generate analog output voltages using this mode. Variable analog outputs are possible using the PWM Controller and an external RC circuit.

18.1.5 Bi-Directional and Uni-Directional Modes

These two modes allow bit-level data transfers to and from the USBee AX pod. The first offers complete flexibility of the 8 digital signal lines, while the other gives you very high transfer rates.

In the Bi-Directional Mode, each of the 8 data signals can be independently setup as inputs or outputs. When sending data to the pod, only the lines that are specified as outputs will be driven. When reading data from the pod, all 8 signals lines will return the actual value on the signal (whether it is an input or an output)

In the High-Speed Mode, all of the 8 data signal lines are setup in the same direction (as inputs or outputs) at the same time. When sending data to the pod, all signals become outputs. When reading data from the pod, all signals become inputs.

Also in High Speed mode, you can specify the CLK rate. Available CLK rates are 24MHz, 12MHz, 6MHz, 3MHz, and 1MHz. For slower rates you can use the bi-directional mode

In each of the modes you can specify the polarity of the CLK line. You can set the CLK line to change data on the falling edge and sample on the rising edge, or visa versa.

The routines used to read and write the data to the pod are the same for both modes. You call the SetMode function to specify the mode you want to use. All subsequent calls for data transfers will then use that mode of transfer.

The following table shows the possible transfer rates for the various modes. This assumes that your USB 2.0 host controller can achieve these rates. USB 2.0 Host controllers can vary greatly.

Mode	Transfer Type	Burst Rate	Sustained Average Rate
Bi-Directional	Write-SetSignals	300k Bytes/sec	~300k Bytes/sec
Bi-Directional	Read-GetSignals	175k Bytes/sec	~175k Bytes/sec
High-Speed	Write-SetSignals	24M Bytes/sec	~20M Bytes/sec
High-Speed	Read-GetSignals	16M Bytes/sec	~13M Bytes/sec

18.2 System Software Architecture

The USBee AX Pod is controlled through a set of Windows DLL function calls. These function calls are defined in following sections and provide initialization and data transfer routines. This DLL can be called using a variety of languages, including C and Visual Basic. We have included sample applications in C and Visual BASIC that show how you can use the calls to setup and control the pod.

After installing the software on your computer, you can then plug in the USBee AX pod. Immediately after plugging in the pod, the operating system finds the USBEEAX.INF file in the \Windows\INF directory. This file specifies which driver to load for that device, which is the USBEEAX.SYS file in the \Windows\System32\Driver directory. This driver then remains resident in memory until you unplug the device.

Once you run your USBee Toolbuilder application, it will call the functions in the USBEEAX.DLL file in the \Windows\System32 directory. This DLL will then make the correct calls to the USBEEAX.SYS driver to perform the USB transfers that are required by the pod.

18.3 The USBee AX Pod Hardware

The USBee AX has two sets of header pins that can be connected to a standard 0.025" square socketed wire. One section of pins is for the digital interface and the other is for the analog channels. Below is the pinout for these two interfaces.

Digital 11 pin Header Pinout: (0-5V MAX input levels)

Pin 1	Data In/Out Bit 0
Pin 2	Data In/Out Bit 1
Pin 3	Data In/Out Bit 2
Pin 4	Data In/Out Bit 3
Pin 5	Data In/Out Bit 4
Pin 6	Data In/Out Bit 5
Pin 7	Data In/Out Bit 6
Pin 8	Data In/Out Bit 7
Pin 9	Read/Write# Output (bit-bang mode),TRG (Signal Generator Mode) (R/W#/TRG)
Pin 10	Clock Output (CLK)
Pin 11	Ground

Analog 3 pin Header Pinout: (-10V to +10V MAX input levels)

Pin 1	Analog Channel 1 Input
Pin 2	Ground
Pin 3	Analog Channel 2 Input

Each of the calls to the USBee AX interface libraries operate on a byte wide transfer. For each byte that is sent out the signal pins or read into the signal pins, the R/W#/TRG line is set and the CLK line toggles to indicate the occurrence of a new sample. Each of the bits in the byte transferred maps to the corresponding signal on the AX pod. For example, if you send out a byte 0x80 to the pod, first the Read/Write# line will be driven low, then the signal on Pin 8 will go high and the others (pin 1-7) will go low. Once the data is on the pins, the Clock line is toggled to indicate that the new data is present.

18.4 Installing the USBee AX Toolbuilder

Do not plug in the USBee AX pod until after you install the software.

The USBee AX Toolbuilder software is included as part of the installation with the USBee AX Installation CD and can be downloaded

from www.usbee.com. Run the setup.exe install program in the downloaded file to install from the web. The install program will install the following USBee Toolbuilder files and drivers into their correct location on your system. Other files will also be installed, but are not necessary for Toolbuilder operation.

18.4.1 USBee AX Toolbuilder Project Contents

Contents of the USBee AX Toolbuilder Visual C Program
(contained in the \Program Files\USBee AX\USBeeAXToolbuilder\HostInC directory after the install).

USBeeAX.dsp	Visual C Project File
USBeeAX.dsw	Visual C Workspace File
USBeeAX.cpp	Visual C program
UsbAXla.lib	USBee AX Interface library file

Contents of the USBee AX Toolbuilder Visual Basic Program
(contained in the \Program Files\USBee AX\USBeeAXToolbuilder\HostInVB directory after the install).

USBeeAXTB.vbp	Visual Basic Project File
USBeeAXTB.vbw	Visual Basic Workspace File
AX.bas	Visual Basic program including DLL declarations
USBeeAXTB.frm	Visual Basic main form

The USBee AX Toolbuilder also depends on the following files for proper operation. These files will be installed in the following directories prior to plugging in the USBee AX pod to USB.

- USBAXLA.DLL in the Windows/System32 directory
- USBEEAX.INF in the Windows/INF directory
- USBEEAX.SYS in the Windows/System32/Drivers directory

Once the above files are in the directories, plugging in the USBee AX pod into a high speed USB port will show a “New Hardware Found” message and the drivers will be loaded.

18.5 USBee AX Toolbuilder Functions

This section details the functions that are available in the usbaxla.dll and defines the parameters to each call.

18.5.1 Initializing the USBee AX Pod

18.5.1.1 EnumerateAXPods

This routine finds all of the USBee AX pods that are attached to your computer and returns an array of the Pod IDs.

Calling Convention

```
int EnumerateAxPods(unsigned int *PodID);
```

where `PodID` is a pointer to the list of Pod IDs found.

Return Value:

Number of USBee AX Pods found

18.5.1.2 InitializeAXPod

This routine initializes the Pod number PodNumber. This routine must be called before calling any other USBee AX functions.

Calling Convention

```
int InitializeAXPod(unsigned int PodNumber);
```

where `PodNumber` is the Pod ID of the pod used found on the back of the unit.

Return Value:

0 = Pod Not Found

1 = Pod Initialized

18.5.1.3 SpeedTest

This routine tests the connection to the initialized AX pod to determine the maximum sample rate for the current PC configuration for the Logic Analyzer and Signal Generator functions.

Calling Convention

```
int SpeedTest( void )
```

Return Value:

247 = 24MHz

167 = 16MHz

127 = 12MHz

87 = 8MHz

67 = 6MHz

47 = 4MHz

37 = 3MHz

27 = 2MHz

17 = 1MHz

18.5.1.4 SpeedTest16

This routine tests the connection to the initialized AX pod to determine the maximum sample rate for the current PC configuration for the Mixed Signal Oscilloscope functions.

Calling Convention

```
int SpeedTest16( void )
```

Return Value:

167 = 16MHz

127 = 12MHz

87 = 8MHz

67 = 6MHz

47 = 4MHz

37 = 3MHz

27 = 2MHz

17 = 1MHz

18.5.2 Bit Bang-Modes

18.5.2.1 SetMode

This routine sets the operating mode for the Pod number PodNumber. This routine must be called before calling the SetSignals or GetSignals functions.

Calling Convention

```
int SetMode (int Mode);
```

- Mode is the type of transfers that you will be doing and includes a number of bit fields.
    - Bit 0 – High Speed or Bi-Directional mode
        - Bit 0 = 0 specifies independent Bi-Directional transfer mode. In this mode, each of the 8 data signals can be independently setup as inputs or outputs. When sending data to the pod, only the lines that are specified as outputs will be driven. When reading data from the pod, all 8 signals lines will return the actual value on the signal (whether it is an input or an output).
        - Bit 0 = 1 specifies high speed all-input or all-output transfer mode. In this mode, all of the 8 data signal lines are setup in the same direction (as inputs or outputs). When sending data to the pod, all signals become outputs. When reading data from the pod, all signals become inputs.
    - Bit 1 – CLK mode
        - Bit 1 = 0 specifies that data changes on the Rising edge and data is sampled on the Falling edge of CLK.
        - Bit 1 = 1 specifies that data changes on the Falling edge and data is sampled on the Rising edge of CLK.
    - Bits 4,3,2 – High Speed CLK rate (don't care in bi-directional mode)
    - Bits 4,3,2 = 0,0,0 CLK=24MHz
    - Bits 4,3,2 = 0,0,1 CLK=12MHz

- Bits 4,3,2 = 0,1,0 CLK=6MHz
- Bits 4,3,2 = 0,1,1 CLK=3MHz
- Bits 4,3,2 = 1,0,0 CLK=1MHz

Return Value:

0 = Pod Not Found

1 = Pod Initialized

18.5.2.2 SetSignals - Setting the USBee AX Output Signals

Calling Convention

```
int SetSignals (  unsigned char State,
                  unsigned int length,
                  unsigned char *Bytes)
```

- State is not used for High-Speed Mode. In Bi-Directional mode, State is the Input/Output state of each of the 8 USBee signals (0 through 7). A signal is an Input if the corresponding bit is a 0. A signal is an Output if the corresponding bit is a 1.
- length is the number of bytes in the array Bytes() that will be shifted out the USBee pod. The maximum length is 32767.
- Bytes() is the array that holds the series of bytes that represent the levels driven on the output signals. When set as an output, a signal is driven high (3.3V) if the corresponding bit is a 1. A signal is driven low (0V) if the corresponding bit is a 0. In Bi-Directional mode, if a signal is set to be an Input in the State parameter, the associated signal is not driven. The Read/Write#/TRG line is set low prior to data available, and the CLK line toggles for each output byte (Length times).

Return Value:

1 = Successful

0 = Failure

18.5.2.3 GetSignals - Reading the USBee AX Input Signals

Calling Convention

```
int GetSignals   (   unsigned char State,
                     unsigned int length,
                     unsigned char *Bytes)
```

- State is not used for High-Speed Mode. In Bi-Directional mode, State is the Input/Output state of each of the 8 USBee signals (0 through 7). A signal is an Input if the corresponding bit is a 0. A signal is an Output if the corresponding bit is a 1.
- length is the number of bytes in the array Bytes() that will be read from the USBee pod. The maximum length is 32767.
- Bytes() is the array that will hold the series of bytes that represent the levels read on the input signals. The Read/Write# line is set high prior to data available, and the CLK line toggles for each input byte (Length times).
- Return Value is the digital level of all 8 USBee pod Signals (bit 0 is signal 0, bit 7 is signal 7)

18.5.3 Logic Analyzer Function

The following API describes the routines that control the Logic Analyzer functionality of the USBee AX Test Pod.

18.5.3.1 MakeBuffer

This routine creates the sample buffer that will be used to store the acquired samples.

Calling Convention

```
unsigned char *MakeBuffer( unsigned long Size )
```

where `Size` is the number of samples (Bytes) to allocate.

Return Value:

0 = Failed to allocate the buffer

other = pointer to allocated buffer

18.5.3.2 DeleteBuffer

This routine releases the sample buffer that was used to store the acquired samples.

Calling Convention

```
unsigned char *DeleteBuffer(  unsigned char
                              *buffer)
```

where `buffer` is the pointer to the allocated buffer.

Return Value:

0 = Failed to deallocate the buffer

other = Success

18.5.3.3 StartCapture

This routine starts the pod capturing data at the specified trigger and sample rates.

Calling Convention

```
int StartCapture( unsigned int SampleRate,
                  unsigned int ClockMode,
                  unsigned char *Triggers,
                  signed int TriggerNumber,
                  unsigned char *buffer,
                  unsigned long length,
                  unsigned long poststore);
```

- SampleRate is as follows:
    - 247 = 24MHz
    - 167 = 16MHz
    - 127 = 12MHz
    - 87 = 8MHz
    - 67 = 6MHz
    - 47 = 4MHz
    - 37 = 3MHz
    - 27 = 2MHz
    - 17 = 1MHz
- ClockMode:
    - 2 = Internal Clocking Mode
    - 4 = External Timing – sample on CLK rising edge
    - 5 = External Timing – sample on CLK falling edge
    - 6 = External Timing – sample on CLK rising edge AND TRIG signal high
    - 7 = External Timing – sample on CLK falling edge AND TRIG signal high
    - 8 = External Timing – sample on CLK rising edge AND TRIG signal low
    - 9 = External Timing – sample on CLK falling edge AND TRIG signal low
- Triggers: array of Mask/Value byte pairs. Mask is a bit mask that indicates which bit signals to observe. 1 in a bit position means to observe that signal, 0 means to ignore it. Value is the actual value of the bits to compare against. If a bit is not used in the Mask, make sure that the corresponding bit is a 0 in Value.
- TriggerNumber: the number of pairs of Mask/Value in the above Triggers Array.
- Buffer: pointer to the sample buffer to store the acquired data into. This buffer must be created using the MakeBuffer routine.
- Length: The total number of samples to acquire. This value must be a multiple of 65536.

- Poststore: The total number of bytes to store after the trigger event happens. If the trigger happens early, the samples are stored until the buffer is full.

Return Value:

0 = Failed

1 = Success

18.5.3.4 CaptureStatus

This routine checks the status of the data capture in progress.

Calling Convention

```
int CaptureStatus(  char *breaks,
                    char *running,
                    char *triggered,
                    long *start,
                    long *end,
                    long *trigger,
                    char *full )
```

- Break: The number of breaks that have occurred in the data sampling since the start of the acquisition. This value is zero (0) if the acquisition has been continuous. If the value is 1 or greater, there was a break in the capture for some reason. If breaks occur repeatedly, your PC is not capable of the sample rate you've chosen and a lower sample rate is needed to achieve continuous sampling.
- Running: 1 = Acquisition is still running, 0 = Acquisition has completed
- Triggered: 1 = Trigger has occurred, 0 = still waiting for the trigger
- Start: Sample Number of the start of the buffer. 0 unless there is an error.
- End: The sample number of the last sample.
- Trigger: The sample number at the point of trigger.
- Full: The percentage of the buffer that is currently filled. Ranges from 0 to 100.

Return Value:

Number of breaks in the sampling

18.5.3.5 StopCapture

This routine terminates a pending capture.

Calling Convention

```
int StopCapture(void)
```

Return Value:

1 = Capture Stopped

0 = Stop Failed

18.5.3.6 LoggedData

This routine returns the value of a particular sample.

Calling Convention

```
long LoggedData( unsigned long index )
```

Index: sample number to return

Return Value:

Value of the given sample

18.5.4 Digital Signal Generator Function

The following API describes the routines that control the Signal Generator functionality of the USBee AX Test Pod.

18.5.4.1 SetData

This routine sets the value of a given 8-bit sample to the value specified. You can also write directly to the allocated buffer after calling MakeBuffer().

Calling Convention

```
long SetData(     unsigned long index,
                  unsigned char value);
```

- Index: sample number to change
- Value: byte value to store in that sample

Return Value:

0 = Set failed

1 = Set successful

18.5.4.2 StartGenerate

This routine starts the pod generating data with the specified trigger, sample rates, and data.

Calling Convention

```
int StartGenerate(  unsigned int SampleRate,
                    unsigned char triggermode,
                    unsigned char *buffer,
                    unsigned long length);
```

- SampleRate is as follows:
    - 247 = 24MHz
    - 167 = 16MHz
    - 127 = 12MHz
    - 87 = 8MHz
    - 67 = 6MHz
    - 47 = 4MHz
    - 37 = 3MHz
    - 27 = 2MHz
    - 17 = 1MHz
- TriggerMode: Indicates the value on the external TRG signal that must occur before the waveforms are generated. 0 = Don't Care, 1 = rising edge, 2 = falling edge, 3 = high level, 4 = low level
- Buffer: pointer to the sample that holds the data to generate. This buffer must be created using the MakeBuffer routine.
- Length: The total number of samples to generate. This value must be a multiple of 65536.

Return Value:

0 = Failed

1 = Success

18.5.4.3 GenerateStatus

This routine checks the status of the data generation in progress.

Calling Convention

```
int GenerateStatus(  char *breaks,
                     char *running,
                     char *triggered,
                     char *complete );
```

- Breaks: The number of breaks that have occurred in the data generating since the start of the generation. This value is zero (0) if the sample timing has been continuous. If the value is 1 or greater, there was a break in the generation for some reason. If breaks occur repeatedly, your PC is not capable of the sample rate you've chosen and a lower sample rate is needed to achieve continuous sample timing.
- Running: 1 = Generation is still running, 0 = Generation has completed

- Triggered: 1 = Trigger has occurred, 0 = still waiting for the trigger
- Complete: The percentage of the buffer that has been generated. Ranges from 0 to 100.

Return Value:

0 = Status Failed

1 = Status Successful

18.5.4.4 StopGenerate

This routine stops a signal generation in progress and terminates a generation cycle.

Calling Convention

```
int StopGenerate(void );
```

Return Value:

0 = Stop Failed

1 = Stop Successful

18.5.5 Mixed Signal Oscilloscope/Logic Analyzer Function

The following API describes the routines that control the Mixed Signal Oscilloscope functionality of the USBee AX Test Pod.

18.5.5.1 MakeBuffer16

This routine creates the sample buffer to store the acquired digital samples. This resulting buffer is two times the Size specified for operational purposes during the capture. Upon completion of the capture (after calling StopCapture16()) only the first Size bytes are the digital sample data. The remaining bytes are then unused.

Calling Convention

```
unsigned char *MakeBuffer16(  unsigned long
                                Size )
```

where Size is the number of samples to allocate. The actual size of the buffer is two times this value, but only the first Size bytes hold the digital sample data.

Return Value:

0 = Failed to allocate the buffer

other = pointer to allocated buffer

18.5.5.2 MakeBufferScope

This routine creates the sample buffer to store the acquired analog samples. Upon completion of the capture (after calling StopCapture16()) this buffer holds the sampled analog data.

The values stored in this buffer are in ADC counts, which go from 0 at -10V to 255 at +10V (78.125mV per count).

Calling Convention

```
unsigned char *MakeBufferScope( unsigned long
                                Size )
```

where `Size` is the number of samples to allocate.

Return Value:

0 = Failed to allocate the buffer

other = pointer to allocated buffer

18.5.5.3 DeleteBuffer16

This routine releases the sample buffers that was used to store the acquired digital and analog samples.

Calling Convention

```
unsigned char *DeleteBuffer(  unsigned char
                              *buffer)
```

where `buffer` is the pointer to the allocated digital buffer. The buffer allocated by the MakeBufferScope() call is also released.

Return Value:

0 = Failed to deallocate the buffer

other = Success

18.5.5.4 StartCaptureMSO

This routine starts the pod capturing digital and analog samples at the specified trigger and sample rates.

Calling Convention

```
int StartCaptureMSO( unsigned int Channel,
                     unsigned int Slope,
                     unsigned int Level,
                     unsigned int SampleRate,
                     unsigned int ClockMode,
                     unsigned char *Triggers,
                     signed int TriggerNumber,
                     unsigned char *Dbuffer,
                     unsigned char *Abuffer,
                     unsigned long Length,
                     unsigned long Poststore);
```

- Channel is as follows:
    - 1 = CH1
    - 2 = CH2
- Slope is as follows:
    - 0 = Analog Slope for Trigger is Don't Care. Uses Digital Triggers instead.
    - 1 = Analog Slope for Trigger is Rising Edge. Ignores digital triggers.
    - 2 = Analog Slope for Trigger is Falling Edge. Ignores digital triggers.
- Level: if Slope is not 0, this value specifies the analog trigger level. This value is in ADC counts, which go from 0 at -10V to 255 at +10V (78.125mV per count).
- SampleRate is as follows:
    - 167 = 16MHz
    - 127 = 12MHz
    - 87 = 8MHz
    - 67 = 6MHz
    - 47 = 4MHz
    - 37 = 3MHz
    - 27 = 2MHz
    - 17 = 1MHz
- ClockMode:
    - Does not matter – It is forced to use Internal Timing for MSO functions.
- Triggers: array of Mask/Value byte pairs used for triggering on the digital samples. Mask is a bit mask that indicates which bit signals to observe. 1 in a bit position means to observe that signal, 0 means to ignore it. Value is the actual value of the bits to compare against. If a bit is not used in the Mask, make sure that the

corresponding bit is a 0 in Value. These triggers are only in effect if the Slope is 0.

- TriggerNumber: the number of pairs of Mask/Value in the above Triggers Array.
- Dbuffer: pointer to the sample buffer to store the acquired digital data into. This buffer must be created using the MakeBuffer16 routine.
- Abuffer: pointer to the sample buffer to store the acquired analog data into. This buffer must be created using the MakeBufferScope routine.
- Length: The total number of samples to acquire. This value must be a multiple of 65536.
- Poststore: The total number of bytes to store after the trigger event happens. If the trigger happens early, the samples are stored until the buffer is full.

Return Value:

0 = Failed

1 = Success

18.5.5.5 CaptureStatus

This routine checks the status of the data capture in progress.

Calling Convention

```
int CaptureStatus(  char *breaks,
                    char *running,
                    char *triggered,
                    long *start,
                    long *end,
                    long *trigger,
                    char *full )
```

- Break: The number of breaks that have occurred in the data sampling since the start of the acquisition. This value is zero (0) if the acquisition has been continuous. If the value is 1 or greater, there was a break in the capture for some reason. If breaks occur repeatedly, your PC is not capable of the sample rate you've chosen and a lower sample rate is needed to achieve continuous sampling.
- Running: 1 = Acquisition is still running, 0 = Acquisition has completed
- Triggered: 1 = Trigger has occurred, 0 = still waiting for the trigger
- Start: Sample Number of the start of the buffer. 0 unless there is an error.
- End: The sample number of the last sample.
- Trigger: The sample number at the point of trigger.
- Full: The percentage of the buffer that is currently filled. Ranges from 0 to 100.

Return Value:

Number of breaks in the sampling

18.5.5.6 StopCaptureMSO

This routine terminates a pending Mixed Storage Oscilloscope capture that was started by StartCaptureMSO(). After execution of this routine, the buffers contain the acquired data.

The values stored in the buffer made by MakeBufferScope() are in ADC counts, which go from 0 at -10V to 255 at +10V (78.125mV per count). The values stored in the buffer made by MakeBuffer16() are in 8 bit values, with one bit per digital line (bit 0 is digital Signal 0).

Calling Convention

```
int StopCaptureMSO(void)
```

Return Value:

1 = Capture Stopped

0 = Stop Failed

18.5.6 Digital Voltmeter (DVM) Function

The following API describes the routine that samples the analog voltages.

18.5.6.1 GetAnalogAverageCount

This routine reads the average analog voltage at the specified channel.

Calling Convention

```
unsigned long *GetAnalogAverageCount(
                              unsigned long Channel)
```

where `Channel` is the channel number to sample (1 or 2).

Return Value:

Average ADC count times 0x100000. ADC counts go from 0 at -10V to 255 at +10V (78.125mV per count). This return value ranges from 0x00000000 (-10V) to 0x0FF00000 (+10V).

18.6 Example C Code

File USBeeAX.cpp

```
//****************************************************************************
// USBee AX Toolbuilder Sample Application
//
// This file contains sample C code that accesses the USBee AX Toolbuilder functions
// that are contained in the USBAXLA.DLL file.  These routines are detailed in the
// USBee AX Toolbuilder document which includes the available routines and
// associated parameters.
//
// Copyright 2005, CWAV - All rights reserved.
// www.usbee.com
//****************************************************************************

#include "stdio.h"
#include "conio.h"
#include "windows.h"

#define CWAV_API __stdcall
#define CWAV_IMPORT __declspec(dllimport)

// AX DLL Routine Declarations

// Basic Bit-Bang I/O Routines
CWAV_IMPORT int CWAV_API SetSignals (unsigned char State, unsigned int length, unsigned char
                                                                                *Bytes);
            // Sets the Digital signals
CWAV_IMPORT int CWAV_API GetSignals (unsigned char State, unsigned int length, unsigned char
                                                                                *Bytes);
            // Reads the Digital I/O signals
CWAV_IMPORT int CWAV_API SetMode (int Mode);    // Sets the I/O Mode

// SetMode definitions
#define FAST_ONEWAY_DATA            1
#define SLOW_TWOWAY_DATA            0

#define DATA_CHANGES_ON_RISING_EDGE       2
#define DATA_CHANGES_ON_FALLING_EDGE            0
#define DATA_IS_SAMPLED_ON_RISING_EDGE          0
#define DATA_IS_SAMPLED_ON_FALLING_EDGE         2

#define _24MHz          (0 << 2)
#define _12MHz          (1 << 2)
#define _6MHz           (2 << 2)
#define _3MHz           (3 << 2)
#define _1MHz           (4 << 2)

// Buffer Routines
CWAV_IMPORT unsigned char * CWAV_API MakeBuffer( unsigned long Size );
            // Makes a Logic Analyzer or Signal Generator buffer
CWAV_IMPORT unsigned char * CWAV_API MakeBuffer16( unsigned long Size );
            // Makes the digital buffer for mixed signal capture
CWAV_IMPORT unsigned char * CWAV_API MakeBufferScope( unsigned long Size );
            // Makes the associated Scope buffer (must be used after MakeBuffer16)
CWAV_IMPORT int CWAV_API DeleteBuffer( unsigned char *buffer );

            // Deletes the associated buffer
CWAV_IMPORT int CWAV_API DeleteBuffer16( unsigned char *buffer );

            // Deletes the digital and scope buffers
CWAV_IMPORT long CWAV_API SetData( unsigned long index, unsigned char value);
            // Sets the data in the logic buffer

// Logic Analyzer Declarations
CWAV_IMPORT int CWAV_API EnumerateAXPods( unsigned int *Pods );

            // Find all USBee AX-Pro pods attached to this computer
CWAV_IMPORT int CWAV_API InitializeAXPod(unsigned int PodNumber);

            // Inits the specified Pod.  This must be done before operation.
CWAV_IMPORT int CWAV_API StartCapture(    unsigned int SampleRate, unsigned int ClockMode,
                                          unsigned char *Triggers, signed int TriggerNumber,
                                          unsigned char *buffer, unsigned long length,
                                          unsigned long poststore);
            // Start a Logic Analyzer trace

CWAV_IMPORT int CWAV_API StopCapture(void);
            // End a Logic Analyzer trace
CWAV_IMPORT int CWAV_API CaptureStatus( char *breaks, char *running, char *triggered,
                                        long *start, long *end, long *trigger, char *full );
            // Monitor the capture in progress
CWAV_IMPORT int CWAV_API SpeedTest( void );
```

```
            // Determines the max bandwidth your PC configuration can achieve.

// Signal Generator Declarations
CWAV_IMPORT int CWAV_API StartGenerate(unsigned int SampleRate, unsigned char triggermode,
                                       unsigned char *buffer, unsigned long length);
            // Signal Generator start
CWAV_IMPORT int CWAV_API GenerateStatus( char *breaks, char *running, char *triggered, char
                                                                                  *complete );
            // Generation Status
CWAV_IMPORT int CWAV_API StopGenerate( void );
            // Stops the Generation in progress

// StartGenerate External Trigger Settings
#define DONT_CARE_TRIGGER 0
#define RISING_EDGE_TRIGGER 1
#define FALLING_EDGE_TRIGGER 2
#define HIGH_LEVEL_TRIGGER 3
#define LOW_LEVEL_TRIGGER 4

// MSO/Scope Functions
CWAV_IMPORT int CWAV_API StartCaptureMSO(unsigned int Channel, unsigned int Slope,
                         unsigned int Level, unsigned int SampleRate,
                         unsigned int ClockMode, unsigned char *Triggers,
                         signed int TriggerNumber, unsigned char *buffer,
                         unsigned char *obuffer,unsigned long length,
                         unsigned long poststore);
            // Starts a Mixed Signal Capture
CWAV_IMPORT int CWAV_API StopCaptureMSO( void );
            // Stops a mixed signal capture

#define CHANNEL1  1
#define CHANNEL2  2

#define DONT_CARE_SLOPE 0
#define RISING_EDGE_SLOPE     1
#define FALLING_EDGE_SLOPE    2

// DVM Functions
CWAV_IMPORT unsigned long CWAV_API GetAnalogAverageCount( unsigned long channel );
            // Reads the analog channel average voltage


unsigned char VoltsToCounts( float Volts )           // Converts Volts into ADC counts
{
      unsigned char counts;
      counts = (char) ((Volts + 10.0) / 0.078125);
      return(counts);
}
float CountsToVolts( float Counts )             // Converts ADC counts into Volts
{
      double Volts;
      Volts = (float)((double)Counts * 0.078125) - 10.0;
      return((float)Volts);
}

int main(int argc, char* argv[])
{
      unsigned char DataInBuffer[65536], DataOutBuffer[65536];
      unsigned int PodNumber, PodID[10], NumberOfPods;
      int ReturnVal;
      unsigned long x;
      unsigned char ch;


      printf("Sample USBee AX Toolbuilder application in C\n");

      //**********************************
      // Pod Initializations Functions - must call InitializeAXPod before using any functions
      //**********************************
      printf("Getting the PodIDs available\n");
      NumberOfPods = EnumerateAXPods(PodID);
      if (NumberOfPods == 0) {
            printf("No USBee AX-Pro Pods found\n");
            getch();
            return 0;
      }

      PodNumber = PodID[0];   // Use the first one we find.  Change this to address your pod

      printf("Initializing the Pod\n");
      ReturnVal = InitializeAXPod(PodNumber);
      if (ReturnVal != 1) {
            printf("Failure Initializing the Pod\n");
            getch();
            return 0;
      }

      //**********************************
```

```
// Basic I/O Functions
//***********************************


printf("Setting the Mode to fast mode\n");
ReturnVal = SetMode(FAST_ONEWAY_DATA | DATA_CHANGES_ON_RISING_EDGE | _24MHz );

if (ReturnVal != 1) {
      printf("Failure setting the mode\n");
      getch();
      return 0;
}

// Make some data to send out the pod signals
for(x=0;x<65536;x++) DataOutBuffer[x]= (char)x;

printf("Sending 163,835 bytes out the pod\n");
for (x = 0; x < 5; x++)
{
      DataOutBuffer[0]=ch++;
      SetSignals (0xFF /* Don't Care */, 32767, DataOutBuffer);
}

printf("Reading 163,835 bytes from the pod signals\n");
for (x = 0; x < 5; x++)
{
      GetSignals (0x00 /* Don't Care */, 32767, DataInBuffer);
}

printf("Setting the Mode to bi-directional mode\n");
ReturnVal = SetMode(SLOW_TWOWAY_DATA | DATA_IS_SAMPLED_ON_RISING_EDGE );

if (ReturnVal != 1) {
      printf("Failure setting the mode\n");
      getch();
      return 0;
}

printf("Sending 32767 bytes out the pod\n");

DataOutBuffer[0]=ch++;
SetSignals (0xFF, 32767, DataOutBuffer);

printf("Reading 32767 bytes from the pod signals\n");

GetSignals (0x00, 32767, DataInBuffer);


//***********************************
// Logic Analyzer Functions
//***********************************

printf("\nSample USBee AX Logic Analyzer Toolbuilder application in C\n");

printf("Start Capturing Data from Pod\n");

unsigned char Rate = 247;                // Sample Rate = 24Msps
unsigned char ClockMode = 2;             // Internal Timing
unsigned char Triggers[4];
Triggers[0] = 0;                    // Trigger Mask = Don't Care
Triggers[1] = 0;                    // Trigger Value
char NumberOfTriggers = 1;
long SampleBufferLength = 16 * 65536;          // 1Meg Sample Buffer
unsigned char *SampleBuffer = MakeBuffer(SampleBufferLength);
long PostStore = SampleBufferLength;

ReturnVal = StartCapture(Rate, ClockMode, Triggers, NumberOfTriggers, SampleBuffer,
                         SampleBufferLength, PostStore);

if (ReturnVal != 1) {
      printf("Failure Starting Capture\n");
      getch();
      return 0;
}

printf("Waiting for data to be captured...");

char Breaks;
char Running;
char Triggered;
long Start;
long End;
long Trigger;
char Full;

do {
```

```
      Sleep(500); // This is required to put pauses between the status requests,
                  // otherwise will eat into the USB bandwidth.

      ReturnVal = CaptureStatus(&Breaks, &Running, &Triggered, &Start, &End, &Trigger,
                                &Full);
      printf(".");
      if (Running && (Breaks != 0)) {
            printf("LA Sample Rate too high\n");
            break;
      }

} while (Running && (Breaks == 0));
printf("\n");

StopCapture();


//***********************************
// Signal Generator Functions
//***********************************

printf("Sample USBee AX Signal Generator Application in C\n");

// Make some data
for (long y = 0; y < SampleBufferLength; y++)
      SampleBuffer[y] = y & 255;

ReturnVal = StartGenerate (247, DONT_CARE_TRIGGER, SampleBuffer, SampleBufferLength);

printf("Waiting for generate to finish.");

Running = 1;

while (Running)
{
      GenerateStatus( &Breaks, &Running, &Triggered, &Full );
      Sleep(400);
      printf(".");
      if (Breaks) break;
}
printf("\nBreaks= %d\n", Breaks);
printf("Running= %d\n", Running);
printf("Triggered= %d\n", Triggered);
printf("Complete= %d\n", Full);

printf("Stopped\n");
StopGenerate();

DeleteBuffer(SampleBuffer);

//***********************************
// MSO Functions
//***********************************

printf("\nSample USBee AX Mixed Signal Scope Toolbuilder application in C\n");

printf("Start Capturing Data from Pod\n");

Rate = 87;              // Sample Rate = 8Msps
ClockMode = 2;                // Internal Timing
Triggers[0] = 0;              // Trigger Mask = Don't Care
Triggers[1] = 0;              // Trigger Value
NumberOfTriggers = 1;
SampleBufferLength = 16 * 65536;    // 1Meg Sample Buffer
SampleBuffer = MakeBuffer16(SampleBufferLength);   // Where to put the Digital samples
unsigned char *ScopeBuffer = MakeBufferScope(SampleBufferLength);
                                                   // Where to put the Scope Samples
unsigned char Slope = DONT_CARE_SLOPE;
unsigned char Level = VoltsToCounts(0.5);   // Analog Trigger Level in ADC Counts
unsigned char Channel = CHANNEL2;           // Ch1 = 1, Ch2 = 2
PostStore = SampleBufferLength;

ReturnVal = StartCaptureMSO(Channel, Slope, Level, Rate, ClockMode, Triggers,
                            NumberOfTriggers, SampleBuffer, ScopeBuffer,
                            SampleBufferLength, PostStore);

if (ReturnVal != 1) {
      printf("Failure Starting MSO Capture\n");
      getch();
      return 0;
}

printf("Waiting for data to be captured...");

do {

      Sleep(500); // This is required to put pauses between the status requests,
                  // otherwise will eat into the USB bandwidth.
```

```
            ReturnVal = CaptureStatus(&Breaks, &Running, &Triggered, &Start, &End, &Trigger,
                                      &Full);
            printf(".");
            if (Running && (Breaks != 0)) {
                  printf("LA Sample Rate too high\n");
                  break;
            }

      } while (Running && (Breaks == 0));
      printf("\n");

      StopCaptureMSO();

      // The data is now available to read
      for( x = 0; x < 15; x++)
      {
            printf("Sample %d: Signal[7..0] = %02X  AnalogChannel = %7.3g\n",
                                    x, SampleBuffer[x], CountsToVolts(ScopeBuffer[x]));
      }

      DeleteBuffer16(SampleBuffer); // Also deletes the Scope Buffer

      //************************************
      // DVM Functions   -  WARNING: This routine sets up a new sample buffer.
      // Make sure you delete any previous sample buffer before running this.
      //************************************

      float CH1Volts = CountsToVolts( (float)((float)GetAnalogAverageCount( 1 ) / 0x100000));
      printf("DVM CH1 = %7.3g\n", CH1Volts);

      float CH2Volts = CountsToVolts((float)((float)GetAnalogAverageCount( 2 ) / 0x100000));
      printf("DVM CH2 = %7.3g\n", CH2Volts);

      printf("Hit any key to continue...\n");

      getch();

      return 0;
}
```

18.6.1 Performance Analysis of the "Bit-Bang" Routines

The following logic analyzer capture shows the timing of the execution of the first part of the above example (The SetSignals and Get Signals section). The Clock line (CLK) is the strobe for each of the bytes transferred and the Data line (DATA) represents the data on each of the 8 pod signal lines. The R/W# indicates if it is a read or a write.

As you can see, the entire program takes about 325msec to execute. In this time we perform:

- Initializing the Pod
- Setting the Mode to High Speed mode
- Sending 163,835 bytes out the pod using High Speed mode
- Reading 163,835 bytes from the pod signals using High-Speed mode
- Setting the Mode to bi-directional mode
- Sending 32767 bytes out the pod using bi-directional mode
- Reading 32767 bytes from the pod signals using bi-directional mode

As a comparison between the modes, all transfers in high speed mode (all 327,670 bytes) occur before the first cursor on the logic analyzer

trace below. The Bi-Directional write (32767 bytes) occur between the cursors, and the bi-direction reads occur after the second cursor.

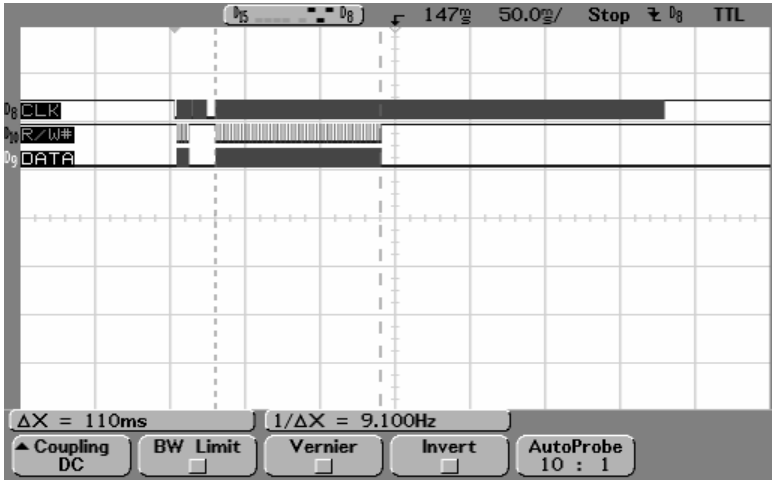


Figure 1. All USBee AX transfers for USBeeAX.c

We will now break down the transfers above so that you can see the exact timing of the signals in the various modes.

The following trace shows the High-Speed Writes (163,835 bytes) followed by Reads (163,835 bytes). We first send out 5 blocks of 32767 bytes which take about 8.2msec. Then we follow with reads of 5 blocks of 32767 bytes which take about 11.6msec.

For writes, each of the blocks of 32768 bytes is bursted at 24Mbytes/sec. The Reads are slower at about 16MHz. The time between bursts is the time it takes for the PC to queue up the next USB transfer. This time may vary depending on your processor speed. Even with these delays in the data stream, the overall bandwidth achieved for writes here is over 20Mbytes/sec and 13.4bytes/sec for reads.

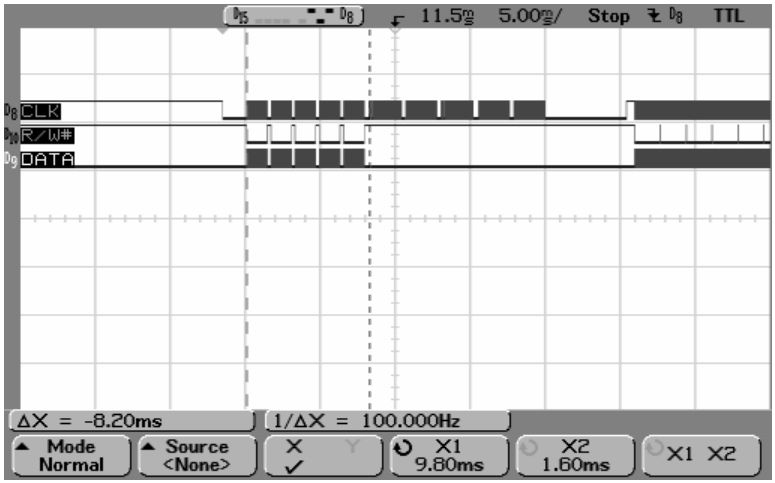


Figure 2. High-Speed Mode – 5 SetSignals followed by 5 GetSignals (each 32767 bytes)

The following trace shows the signal byte transfer timing using the High-Speed mode. You can see that the Clock period is 24MHz and each data line changes close to the rising edge of the clock.

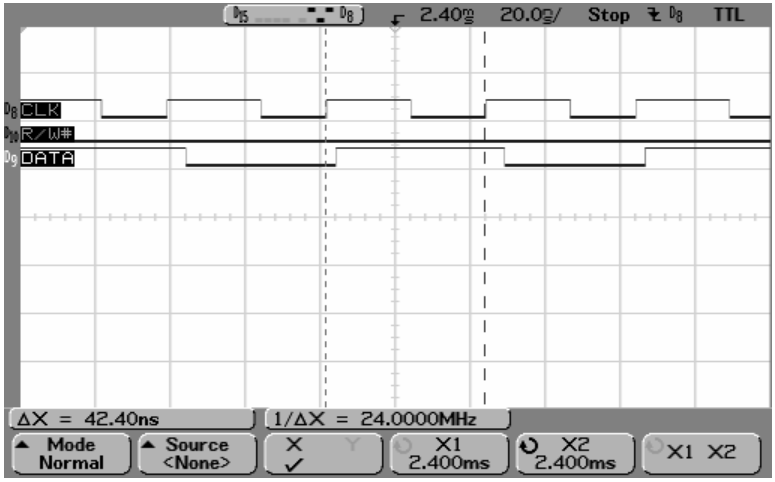


Figure 3. Data and Clock timing in High-Speed mode (SetSignal)

The following traces show the low level timing for the Bi-Directional Mode SetSignal and GetSignal calls.

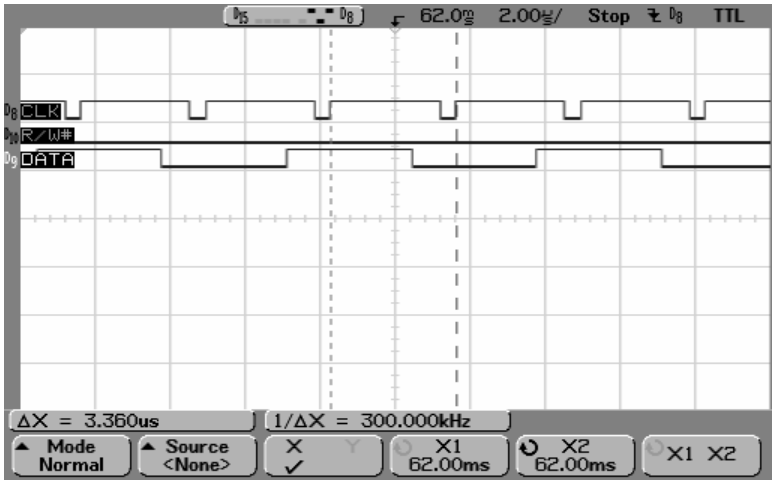


Figure 4. Bi-Directional mode SetSignal byte timing

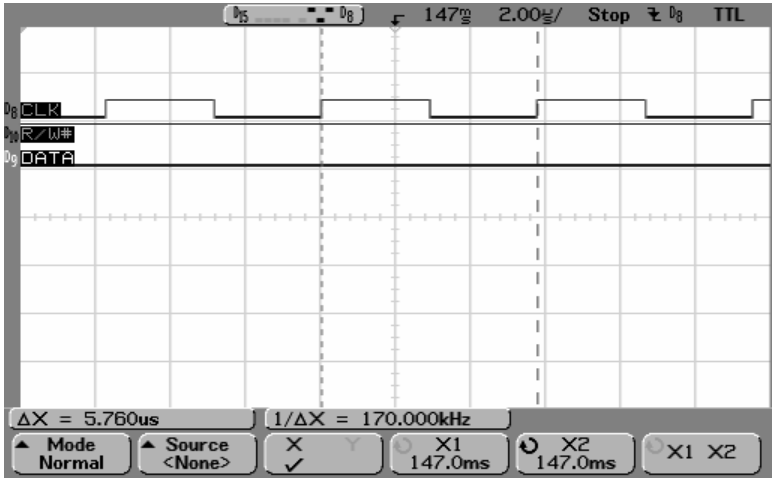


Figure 5. Bi-Directional mode GetSignal byte timing

The data is sampled in the middle of the low clock period.

All of the above traces can have the opposite polarity for the CLK line by setting the appropriate bit in the SetMode parameter.

18.7 Example Visual Basic Code

```
'''//****************************************************************************
'''// USBee AX Toolbuilder Sample Application
'''//
'''// This file contains sample Visual Basic code that uses USBee AX Toolbuilder functions
'''// that are contained in the USBAXLA.DLL file.  These routines are detailed in the
'''// USBee AX Toolbuilder document which includes the available routines and
'''// associated parameters.
'''//
'''// Copyright 2005, CWAV - All rights reserved.
'''// www.usbee.com
'''//****************************************************************************
Declare Function SetSignals Lib "usbaxla" Alias "?SetSignals@@YGHEIPAE@Z" ( _
      ByVal State As Byte, ByVal Length As Long, ByRef Buffer As Byte) As Long
Declare Function GetSignals Lib "usbaxla" Alias "?GetSignals@@YGHEIPAE@Z" ( _
      ByVal State As Byte, ByVal Length As Long, ByRef Buffer As Byte) As Long
Declare Function SetMode Lib "usbaxla" Alias "?SetMode@@YGHH@Z" ( _
      ByVal mode As Byte) As Long
Declare Function MakeBuffer Lib "usbaxla" Alias "?MakeBuffer@@YGPAEK@Z" ( _
      ByVal Size As Long) As Long
Declare Function MakeBuffer16 Lib "usbaxla" Alias "?MakeBuffer16@@YGPAEK@Z" ( _
      ByVal Size As Long) As Long
Declare Function MakeBufferScope Lib "usbaxla" Alias "?MakeBufferScope@@YGPAEK@Z" ( _
      ByVal Size As Long) As Long
Declare Function DeleteBuffer Lib "usbaxla" Alias "?DeleteBuffer@@YGHPAE@Z" ( _
      ByVal Buffer As Long) As Long
Declare Function DeleteBuffer16 Lib "usbaxla" Alias "?DeleteBuffer16@@YGHPAE@Z" ( _
      ByVal Buffer As Long) As Long
Declare Function SetData Lib "usbaxla" Alias "?SetData@@YGJKE@Z" (ByVal Index As Long, _
      ByVal Value As Byte) As Long
Declare Function EnumerateAXPods Lib "usbaxla" Alias "?EnumerateAXPods@@YGHPAI@Z" ( _
      ByRef Source As Long) As Long
Declare Function InitializeAXPod Lib "usbaxla" Alias "?InitializeAXPod@@YGHI@Z" ( _
      ByVal PodID As Long) As Long
Declare Function StartCapture Lib "usbaxla" Alias "?StartCapture@@YGHIIPAEH0KK@Z" ( _
      ByVal SampleRate As Long, ByVal ClockMode As Long, ByRef Triggers As Byte, _
      ByVal TriggerNumber As Long, ByVal Buffer As Long, ByVal Length As  Long, _
      ByVal PostStore As Long) As Long
Declare Function StopCapture Lib "usbaxla" Alias "?StopCapture@@YGHXZ" () As Long
Declare Function CaptureStatus Lib "usbaxla" Alias "?CaptureStatus@@YGHPAD00PAJ110@Z" ( _
      ByRef Breaks As Byte, ByRef Running As Byte, ByRef Triggered As Byte, _
      ByRef StartP As Long, ByRef EndP As Long, ByRef Trigger As Long, ByRef Full As Byte) _
      As Long
Declare Function SpeedTest Lib "usbaxla" Alias "?SpeedTest@@YGHXZ" () As Long
Declare Function SpeedTest16 Lib "usbaxla" Alias "?SpeedTest16@@YGHXZ" () As Long
Declare Function GenerateStatus Lib "usbaxla" Alias "?GenerateStatus@@YGHPAD000@Z" ( _
      ByRef Breaks As Byte, ByRef Running  As Byte, ByRef Triggered As Byte, _
      ByRef Complete As Byte) As Long
Declare Function StopGenerate Lib "usbaxla" Alias "?StopGenerate@@YGHXZ" () As Long
Declare Function StartGenerate Lib "usbaxla" Alias "?StartGenerate@@YGHIEPAEK@Z" ( _
      ByVal SampleRate As Long, ByVal TriggerMode As Byte, ByVal Buffer As Long, _
      ByVal Length As Long) As Long
Declare Function StartCaptureMSO Lib "usbaxla" Alias "?StartCaptureMSO@@YGHIIIIIPAEH00KK@Z" _
       (ByVal Channel As Long, ByVal Slope As Long, ByVal Level As Long, _
      ByVal SampleRate As Long, ByVal ClockMode As Long, ByRef Triggers _
      As Byte, ByVal TriggerNumber As Long, ByVal Buffer As Long, ByVal ScopeBuffer As Long, _
      ByVal Length As Long, ByVal PostStore As Long) As Long
Declare Function StopCaptureMSO Lib "usbaxla" Alias "?StopCaptureMSO@@YGHXZ" () As Long
Declare Function GetAnalogAverageCount Lib "usbaxla" Alias "?GetAnalogAverageCount@@YGKK@Z" _
       (ByVal Channel As Long) As Long
Declare Function ScopeData Lib "usbaxla" Alias "?ScopeData@@YGJJ@Z" (ByVal Index As Long) _
      As Long
Declare Function LoggedData Lib "usbaxla" Alias "?LoggedData@@YGJK@Z" (ByVal Index As Long) _
      As Long

Declare Function GetTickCount Lib "kernel32" () As Long

' GetSignal and SetSignal Mode definitions
Global Const FAST_ONEWAY_DATA = 1
Global Const SLOW_TWOWAY_DATA = 0

Global Const DATA_CHANGES_ON_RISING_EDGE = 2
Global Const DATA_CHANGES_ON_FALLING_EDGE = 0
Global Const DATA_IS_SAMPLED_ON_RISING_EDGE = 0
Global Const DATA_IS_SAMPLED_ON_FALLING_EDGE = 2

Global Const C_24MHz = (0 * 4)
Global Const C_12MHz = (1 * 4)
Global Const C_6MHz = (2 * 4)
Global Const C_3MHz = (3 * 4)
Global Const C_1MHz = (4 * 4)

Global Const DONT_CARE_TRIGGER = 0
Global Const RISING_EDGE_TRIGGER = 1
Global Const FALLING_EDGE_TRIGGER = 2
Global Const HIGH_LEVEL_TRIGGER = 3
```

```
Global Const LOW_LEVEL_TRIGGER = 4

Global Const CHANNEL1 = 1
Global Const CHANNEL2 = 2

Global Const DONT_CARE_SLOPE = 0
Global Const RISING_EDGE_SLOPE = 1
Global Const FALLING_EDGE__SLOPE = 2

Global Rate As Byte         ' Sample Rate
Global Slope As Long        ' Analog Slope Mode (Rising, Falling, Don't care)
Global Level As Long        ' Analog trigger level ( in ADC counts * &H100000 )
Global NumberOfSamples As Long ' Number of Samples to store
Global BigBuffer As Long       ' Pointer to the Digital Sample Data
Global ScopeBigBuffer As Long  ' Pointer to the Analog Sample Data
Global Triggers(256) As Byte   ' Holds the digital trigger states
Global TriggerStates As Long   ' Number of digital trigger states
Global ReturnVal As Long    ' Status return value
Global TRunning As Byte     ' Are we currently capturing data?
Global TBreaks As Byte      ' Were there breaks in the data? (only sample if still running)
Global TTriggered As Byte   ' Are we triggered yet?
Global TStartP As Long      ' What is the starting pointer
Global TEndP As Long        ' What is the ending pointer
Global TTrigger As Long     ' The trigger position
Global TFull As Byte        ' How full are we (0 to 100%)

Sub DelayMS(delay As Long)

    OldTime = GetTickCount

    While (GetTickCount - OldTime) < delay
        DoEvents
    Wend

End Sub
Function VoltsToCounts(Volts As Single) As Byte
    Dim Counts As Byte

    Counts = ((Volts + 10#) / 0.078125)

    VoltsToCounts = Counts

End Function
Function CountsToVolts(Counts As Single) As Single
    Dim Volts As Double

    Volts = (Counts * 0.078125) - 10#

    CountsToVolts = Volts

End Function

Sub DoIt()

    Dim DataInBuffer(65536) As Byte
    Dim DataOutBuffer(65536) As Byte
    Dim ReturnVal As Integer
    Dim X As Long
    Dim ch As Byte
    Dim NumberOfSamples As Long
    Dim PodNumber As Long
    Dim PodID(10) As Long
    Dim NumberOfPods As Long


    Mainform.OutText.Cls

    Mainform.OutText.Print "Sample USBee AX Toolbuilder application in Visual Basic"

    ''//***********************************
    ''// Pod Initializations Functions - must call InitializeAXPod before using any functions
    ''//***********************************
    Mainform.OutText.Print "Getting the PodIDs available"
    NumberOfPods = EnumerateAXPods(PodID(0))
    If (NumberOfPods = 0) Then
        Mainform.OutText.Print "No USBee AX-Pro Pods found"
        Exit Sub
    End If

    PodNumber = PodID(0)       ''// Use the first one we find.  Change this to address your pod

    Mainform.OutText.Print "Initializing the Pod"
    ReturnVal = InitializeAXPod(PodNumber)
    If (ReturnVal <> 1) Then
        Mainform.OutText.Print "Failure Initializing the Pod"
        Exit Sub
    End If
```

```
'//************************************
'// Basic I/O Functions
'//************************************


Mainform.OutText.Print "Setting the Mode to fast mode"
ReturnVal = SetMode(FAST_ONEWAY_DATA Or DATA_CHANGES_ON_RISING_EDGE Or C_24MHz)
If (ReturnVal <> 1) Then
    Mainform.OutText.Print "Failure setting the mode"
    Exit Sub
End If

'// Make some data to send out the pod signals
For X = 0 To 65535
    DataOutBuffer(X) = X And 255
Next X

Mainform.OutText.Print "Sending 163,835 bytes out the pod"
For X = 0 To 4
    DataOutBuffer(0) = ch
    ch = ch + 1
    SetSignals &HFF, 32767, DataOutBuffer(0)
Next X

Mainform.OutText.Print "Reading 163,835 bytes from the pod signals"
For X = 0 To 4
    GetSignals &H0, 32767, DataInBuffer(0)
Next X

Mainform.OutText.Print "Setting the Mode to bi-directional mode"
ReturnVal = SetMode(SLOW_TWOWAY_DATA Or DATA_IS_SAMPLED_ON_RISING_EDGE)
If (ReturnVal <> 1) Then
    Mainform.OutText.Print "Failure setting the mode"
    Exit Sub
End If

Mainform.OutText.Print "Sending 32767 bytes out the pod"

DataOutBuffer(0) = ch
ch = ch + 1
SetSignals &HFF, 32767, DataOutBuffer(0)

Mainform.OutText.Print "Reading 32767 bytes from the pod signals"

GetSignals &H0, 32767, DataInBuffer(0)


'//************************************
'// Logic Analyzer Functions
'//************************************

Mainform.OutText.Print "USBee AX Logic Analyzer Toolbuilder application in Visual Basic"

Mainform.OutText.Print "Start Capturing Data from Pod"

Rate = 247                          '// Sample Rate = 24Msps
ClockMode = 2                       '// Internal Timing
Triggers(0) = 0                     '// Trigger Mask = Don't Care
Triggers(1) = 0                     '// Trigger Value
NumberOfTriggers = 1
SampleBufferLength = 16 * 65536     '// 1Meg Sample Buffer
SampleBuffer = MakeBuffer(SampleBufferLength)
PostStore = SampleBufferLength

ReturnVal = StartCapture(Rate, ClockMode, Triggers(0), NumberOfTriggers, SampleBuffer, _
                         SampleBufferLength, PostStore)

If (ReturnVal <> 1) Then
    Mainform.OutText.Print "Failure Starting Capture"
    Exit Sub
End If

Mainform.OutText.Print "Waiting for data to be captured..."

Do

    DelayMS 100 '// This is required to put pauses between the status requests, otherwise
                '// the CaptureStatus will eat into the USB bandwidth.

    ReturnVal = CaptureStatus(TBreaks, TRunning, TTriggered, TStartP, TEndP, TTrigger, _
                              TFull)
    Mainform.OutText.Print ".";
    If (TRunning And (TBreaks <> 0)) Then
        Mainform.OutText.Print "LA Sample Rate too high"
        Exit Do
    End If

Loop While (TRunning And (TBreaks = 0))
```

```
    Mainform.OutText.Print ""

    StopCapture


    '//***********************************
    '// Signal Generator Functions
    '//***********************************

    Mainform.OutText.Print "Sample USBee AX Signal Generator Application in C"

    '// Make some data
    For y = 0 To SampleBufferLength - 1
        SetData y, y And 255
    Next y

    ReturnVal = StartGenerate(247, DONT_CARE_TRIGGER, SampleBuffer, SampleBufferLength)

    Mainform.OutText.Print "Waiting for generate to finish."

    Running = 1

    Do While (Running)
        GenerateStatus TBreaks, TRunning, TTriggered, TFull
        DelayMS 100

        Mainform.OutText.Print ".";

        If (TBreaks) Then Exit Do
    Loop

    Mainform.OutText.Print "Breaks= " & TBreaks
    Mainform.OutText.Print "Running= " & TRunning
    Mainform.OutText.Print "Triggered= " & TTriggered
    Mainform.OutText.Print "Complete= " & TFull
    Mainform.OutText.Print "Stopped"

    StopGenerate

    DeleteBuffer (SampleBuffer)

    '//***********************************
    '// MSO Functions
    '//***********************************

    Mainform.OutText.Print "Sample USBee AX Mixed Signal Scope Toolbuilder application in
Visual Basic"

    Mainform.OutText.Print "Start Capturing Data from Pod"

    Rate = 87                           '// Sample Rate = 8Msps
    ClockMode = 2                       '// Internal Timing
    Triggers(0) = 0                     '// Trigger Mask = Don't Care
    Triggers(1) = 0                     '// Trigger Value
    NumberOfTriggers = 1
    SampleBufferLength = 16 * 65536   '// 1Meg Sample Buffer
    SampleBuffer = MakeBuffer16(SampleBufferLength)      '// Where to put the Digital samples
    ScopeBuffer = MakeBufferScope(SampleBufferLength)    '// Where to put the Scope Samples
    Slope = DONT_CARE_SLOPE
    Level = VoltsToCounts(0.5)                           '// Analog Trigger Level in ADC Counts
    Channel = CHANNEL2                                   '// Ch1 = 1, Ch2 = 2
    PostStore = SampleBufferLength

    ReturnVal = StartCaptureMSO(Channel, Slope, Level, Rate, ClockMode, Triggers(0), _
                                NumberOfTriggers, SampleBuffer, ScopeBuffer, _
                                SampleBufferLength, PostStore)

    If (ReturnVal <> 1) Then
        Mainform.OutText.Print "Failure Starting MSO Capture"
        Exit Sub
    End If

    Mainform.OutText.Print "Waiting for data to be captured..."

    Do

        DelayMS 100    '// This is required to put pauses between the status requests,
                       '// otherwise the CaptureStatus will eat into the USB bandwidth.

        ReturnVal = CaptureStatus(TBreaks, TRunning, TTriggered, TStartP, TEndP, TTrigger, _
                                  TFull)
        Mainform.OutText.Print ".";
        If (TRunning And (TBreaks <> 0)) Then
            Mainform.OutText.Print "LA Sample Rate too high"
            Exit Do
        End If

    Loop While (TRunning And (TBreaks = 0))
```

```
    Mainform.OutText.Print ""

    StopCaptureMSO

    '// The data is now available to read
    For X = 0 To 15
        Mainform.OutText.Print "Sample " & X & ": Signal[7..0] = " & LoggedData(X) & " _
                                  AnalogChannel = " & CountsToVolts(ScopeData(X))
    Next X

    DeleteBuffer16 SampleBuffer  '// Also deletes the Scope Buffer

    '//**********************************
    '// DVM Functions   -  WARNING: This routine sets up a new sample buffer.
    '// Make sure you delete any previous sample buffer before running this.
    '//**********************************

    CH1Volts = CountsToVolts(GetAnalogAverageCount(1) / &H100000)
    Mainform.OutText.Print "DVM CH1 = " & CH1Volts

    CH2Volts = CountsToVolts(GetAnalogAverageCount(2) / &H100000)
    Mainform.OutText.Print "DVM CH2 = " & CH2Volts


End Sub
```

Printed in the USA

Version 1.1