Institute of Neuroinformatics
UNI - ETH Zurich

Architectura y
Technologia de Computadores
Universidad de Sevilla

# USBAERmini2 userguide

**Author:**
**Raphael Berner**

**October 10, 2008**

# Contents

## List of Figures



## List of Tables

# 1 Setting Up The Host Computer

For jAER software setup and driver installation, see the jAER wiki at `http://jaer.wiki.sourceforge.net`.

# 2 Setting Up Matlab

Make sure Matlab uses Java Runtime Environment 1.5 by typing `version -java` in the command prompt. If not, create a windows environment variable called `MATLAB_JAVA` which points to the JRE 1.5 home directory.

Start matlab and edit the file librarypath.txt by typing `edit librarypath.txt` in the command prompt. Add the path to the folder `\host\java\jars` to this file. If you want to use devices based on the Silicon Labs C8051F320 as well, add also the path to `\host\java\jars\SiLabsNativeWindows`.
The file will then look something like this:

```
##
## FILE: librarypath.txt
##
## Entries:
##    o path_to_jnifile
##    o [alpha,glnx86,sol2,unix,win32,mac]=path_to_jnifile
##    o $matlabroot/path_to_jnifile
##    o $jre_home/path_to_jnifile
##
$matlabroot/bin/$arch
C:\Users\raphael\jAER\host\java\jars
```

At this point you have to restart Matlab, otherwise the change in the librarypath.txt file does not become active.

Navigate to the folder `\host\matlab\` and run the file `startup.m`. This adds `jaer.jar` and `usbio.jar` to the dynamic matlab path. Uncommented in this file is also a line which instantiates the hardware factory, through which the devices can be accessed.

Matlab is now ready to use the USBAERmini2 boards. To verify the installation, connect the sequencer output of a board directly to its monitor input and sequence a few hundred events using the script `aemonseq.m`. See section 7 for details on this script.

# 3 Setting Up The Hardware Connections

Connect the USBAERmini2 boards to USB2.0 ports of your computer. If you wish to capture synchronised data from several boards, connect the

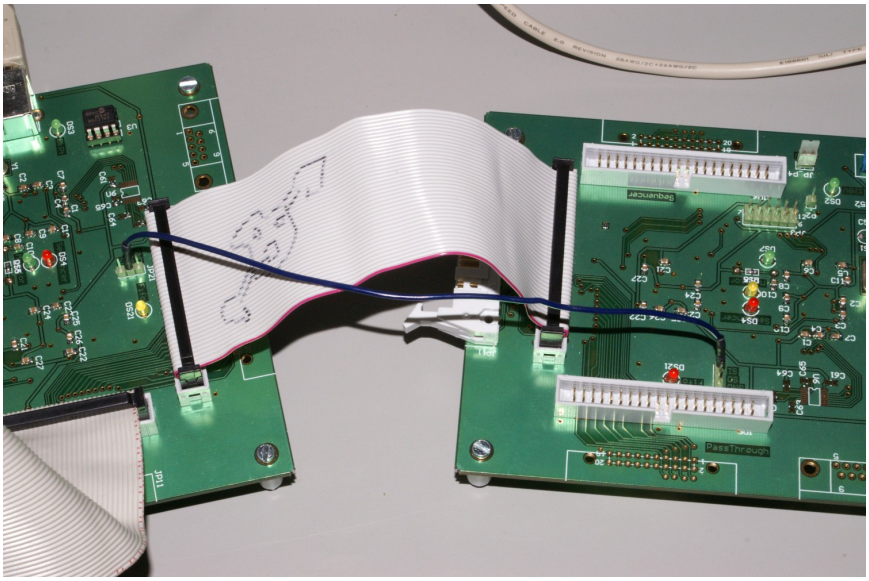

Figure 1: Two USBAERmini2 boards with synchronisation connection

synchronisation output pin (labelled SO) from the desired master board to the synchronisation input pins (labelled SI) of the slave boards as shown in figure 1.

If the device is used in terminal mode, i.e. the monitor port is connected, but the pass-through port is not, it is best to connect request and acknowledge lines of the pass-through port. The device can detect if a receiver is present on the pass-through port, but only while monitoring. When monitoring is inactive, request and acknowledge lines of monitor and pass-through port are connected directly, therefore transmission is blocked if no device is present and monitoring is not active.

# 4 LEDs

**DS1/3.3V** Power LED. Should always be turned on while connected.

**DS2/1.8V** Power LED. Should always be turned on while connected.

**DS3/BkPt** Breakpoint LED. Only active in debugger mode.

**DS4/Sequencer** Turned on while sequencer is running in host trigger mode.

**DS5/Monitor** Turned on while monitor is running.

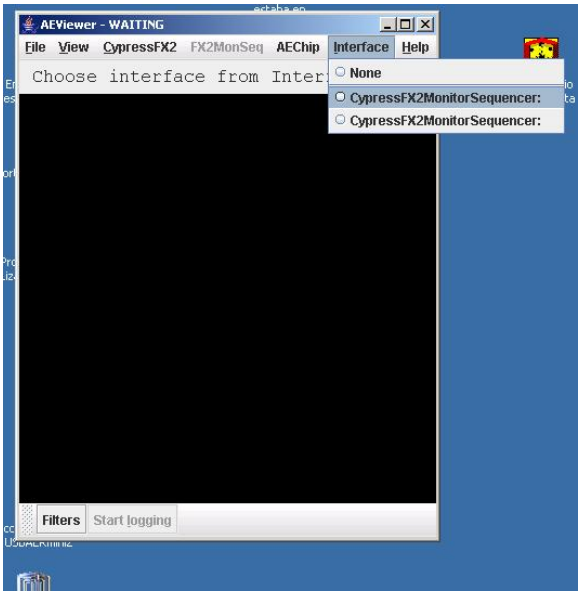


Figure 2: jAER Window, selecting an interface

**DS7** Heartbeat. Should always be blinking while connected, except while downloading to the EEPROM. If it stops, the microprocessor is stuck and the device has to be reset by unconnecting.

**DS21/Fifo** Master/Slave. Turned on as long as SI input is low, i.e. when the device is acting as timestamp master.

# 5 The jAER Viewer

jAER is a Java application to view real time data form several kinds of AER devices like the USBAERmini2, the USB2AERmapper and the TmpDiff128 retina, to record data from these devices, sequence to USBAERmini2 boards or to view recorded data.

## 5.1 Starting and Live Displaying

Starting jAER opens a window like the one shown in figure 2 should show up.

Choose the appropriate chip-type from the menu AEChip and then select the interface to display from the Interface menu, see figure 2.

As soon as an interface is selected, event acquisition is started and chip type, interface type and name are displayed in the titlebar, as can be seen in figure 3.

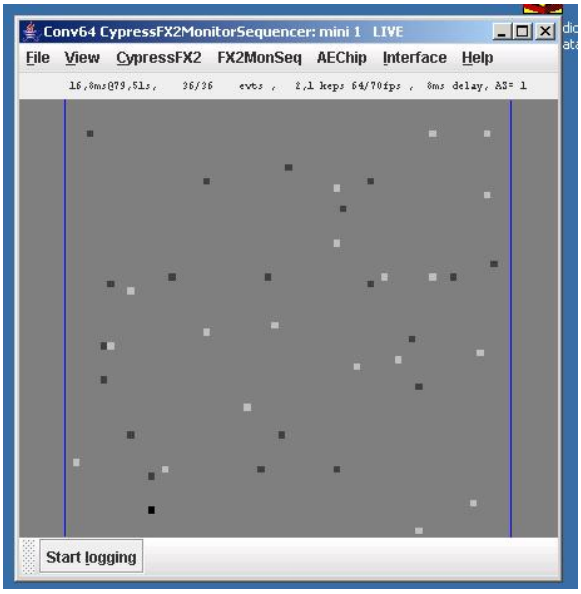


Figure 3: jAER Window, LIVE view. Chip-type and interface are displayed in the titlebar

If the selected interface is an USBAERmini2, the FX2MonSeq menu will become active. In this menu, you select a `.dat` file to be sequenced, you can change the way the device behaves when it can not write data to the FX2 FIFOs and you can change the timestamp tick. Please be aware that jAER handles the timestamps as $1\mu s$ regardless of the tick you select, so if you select $0.2\mu s$, the clock will advance faster.

If the display of the events seems jerky, this may be due to a low event rate, taking a long time to fill the driver's buffers until it passes them to the application. In this case, it is best to reduce the size of the buffers. Figure 4 shows how to do this.

## 5.2 Recording Events

Recording of events can be started by pressing the "Start logging" button in the lower left corner and stopped by pressing it again.

### Synchronized Recording of Multiple Devices

Make sure the item *Synchronize Viewers* in the *File* menu is enabled. There are two types of synchronization available:

- Software synchronization
- Hardware (electrical) synchronization

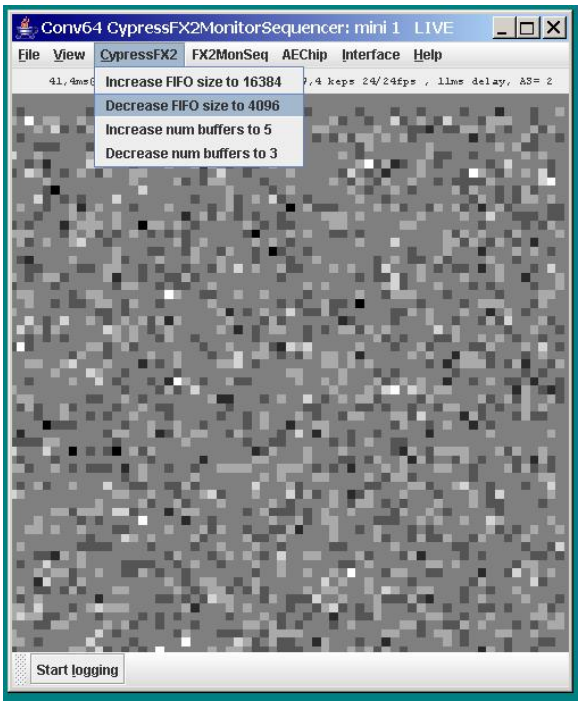


Figure 4: Reducing the driver buffer size in the CaviarViewer

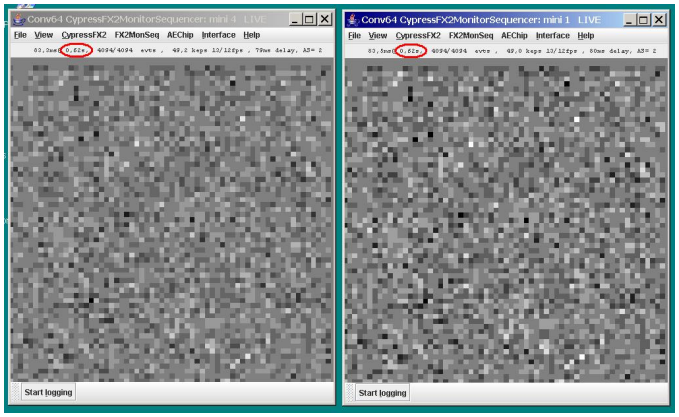


Figure 5: Check if the time is reset in all windows.

Software synchronization resets the timestamps on the devices by sending a vendor request to each device in turn. This is not absolutely synchronous, but does not require the synchronization pins to be connected. The devices are reset by pressing the zero key in any window.

The electrical synchronization is much more accurate, because all the devices are reset and clocked from the master device. To enable electrical synchronization, connect the SO pin of the desired master device to the SI pins of the slave devices. Then press the zero key in any window to reset and synchronize all the devices. Always check if the time really is reset in all the windows.

### 5.3 Sequencing Recorded Events

The USBAERmini2 can be used to stimulate other devices with previously recorded `.dat` files. The command can be found in the MonSeq menu. Please note that the USBAERmini2 currently only supports interspike intervals up to 65ms (across the whole address space, single addresses can of course have longer interspike intervals), longer interspike intervals are shortened modulo 65ms. The device will loop the selected file until it is told to stop.

### 5.4 How the device handles blocked FX2 FIFOs

When the data rate is high and the host computer is not fast enough in collecting the data from the device, the FIFOs in the FX2 can fill up, so that the CPLD can not write more events. The device offers two ways how this is handled.

The default setting is blocking the AER transmission by holding ACK low and waiting until a FIFO is available. This will block the AER sender but no events are lost.

The other setting is to discard events as long as the FIFOs are unavailable. The device offers a rough estimate on how many events have been lost.

To change this setting, check or uncheck the *Enable missed events* box in the *MonSeq* menu.

### 5.5 Playing Back Recorded Files

You can view either single data files (.dat) or synchronised sets of files (.index). You can drag and drop either type of file onto a fresh jAER AEViewer window. Or you can select the file using menu item File/Open... (shortcut "o"). If you want to select an .index file, then you need to change the file type in the file chooser; due to a bug in the graphics it only shows you one choice and doesn't show the Open button until you hover over it. This is a byproduct of using a fast native "heavyweight" Canvas to render the events.

Examine the menus for help; almost all menu items have single-key shortcuts. (e.g. r=rewind, f=faster, s=slower, etc).

## 6 Accessing The Java Classes From Matlab

Java classes and methods can be accessed directly from Matlab without any wrapper functions. For example the following calls instantiate a hardware factory and get a reference to the first USBAERmini2 device.

```
>> factory = ch.unizh.ini.caviar.hardwareinterface.usb.
CypressFX2Factory.instance();
>> usb0=factory.getInterface(0)
```

Note that the instantiation of the factory is not necessary, as this is already done by `startup.m`. See section 7 for details.

### 6.1 Important Java Methods

Please see the javadoc for more details and a complete list of methods.

#### CypressFX2Factory.listDevices()

List the available USBAERmini2 devices. Ignore the following errors, they are due to rapidly opening and closing the devices. *error binding to pipe for EP1 for device status: Error code 0x00000006: Windows system error code. can't set pipe parameters: Error code 0xE0001000: Operation failed.*

#### CypressFX2Factory.getInterface(int interfaceNumber)

This method returns a reference to a USBAERmini2 device. Parameter is the interface number, which has to be chosen from the output of `factory.listDevices`.

### CypressFX2MonitorSequencer.open()

Opens the device.

### CypressFX2MonitorSequencer.setOperationMode(int mode)

Sets operation mode, which includes timestamp tick and trigger mode. Valid modes are shown in table 1.

| Mode | Tick | Trigger |
|---|---|---|
| 0 | 1us | Master (Host) |
| 1 | 0.033us | Master (Host) |
| 2 | 1us | Slave |
| 3 | 0.033us | Slave |

Table 1: USBAERmini2 operation modes

In host trigger mode, event acquisition and sequencing is started and stopped when the commands from the host are received through USB. With this mode, it is not possible to start several devices synchronously.

In slave trigger mode, event acquisition and sequencing is started and stopped when the master device starts or stops. Therefore, slave mode works only when the SI pin is connected to the SO pin of the master device and the master device receives the start or stop command from the host. This mode is recommended to use for the slave devices when using `multi_monitor.m` or `multi_monitor_seq.m`. These matlab scripts are described in section 7.

### CypressFX2MonitorSequencer.getOperationMode()

This method returns the timestamp tick and displays the operation mode.

### CypressFX2MonitorSequencer.setContinuousSequencingEnabled(bool)

Disables or enables continuous event sequencing. If enabled, the AEWriter thread rewinds at the end of the packet of events it has to send and sends these events over and over again. If disabled, the AEWriter thread send the events only once. The default state is that events are sequenced only once.

This method can be used in combination with the scripts `aemonseq.m` and `multi_monitor_seq.m`. The scripts `aeseq.m` and `aeseq_cont.m` already call this method with the suitable parameter.

### CypressFX2.setAEReaderFifoSize(int size)

Sets the buffer size of the event-capturing thread. In general, a buffer size of at least 8kB leads to the highest possible event rates, however, when the

| | |
|---|---|
| startup.m | adds classes to the dynamic path and instantiates the hardware factory |
| aemon.m | monitoring a single device |
| aemonseq.m | monitoring and sequencing with a single device |
| aeseq.m | sequencing from a single device |
| aeseq_cont.m | continuous sequencing from a single device |
| aeseq_cont_stop.m | stops continuous sequencing |
| multi_monitor.m | monitoring from multiple devices |
| multi_monitor_seq.m | sequencing from one device and monitoring from multiple devices |

Table 2: Matlab scripts

CaviarViewer is used to monitor a low eventrate connection, smaller buffer sizes are needed to produce suitable frame rates.

#### CypressFX2MonitorSequencer.getNumMissedEvents()

This method returns an estimation of the number of events the device has missed because of the Cypress FX2 fifos being full when the host does not collect the events fast enough.

#### CypressFX2MonitorSequencer.writeMonitorSequencerFirmware()

This methods writes the firmware, which is saved in the usb2aemon.jar files as well, to the EEPROM. Use this function when a new firmware version is available.

#### CypressFX2MonitorSequencer.setDeviceName(String name)

Sets a new serial number. Parameter is the new serial number string. Be advised that after you plug in a device with a new serial number, the Windows New Hardware Installation Wizard will show up. The string length is limited to eight characters.

## 7 Matlab Scripts

Table 2 gives an overview over the matlab scripts available in `\host\matlab\monitor_sequencer`.

### 7.1 Important Note on Sequencing and Monitoring in Matlab

Be advised that the sequencer needs *interspike intervals* for correct sequenc- IMPORTANT!

ing of spike trains, but the monitor returns *absolute* timestamps. Therefore recorded spike trains have to be processed (usually by the Matlab function `diff`) before they can be sent to the sequencer again.

### 7.2 startup.m

This script adds the path to the jar-files to the dynamic matlab path and instantiates the hardware factory, through which the devices can be accessed. This script has to be called at startup. Please note that this script is located in the parent folder, i.e. in `CAVIAR\wp5\USBAER\INI-AE-Biasgen\host\matlab\`.

### 7.3 aemon.m

`[inaddr,ints,tick]=aemon(usbinterface,monitortime)`
This is a script to monitor events with one USBAERmini2 device.

**Parameters**

**usbinterface** Reference to USBAERmini2 device. Get a reference to a USBAERmini2 device using for example `usb0=factory.getInterface(0)`.

**monitortime** How long the monitoring is active. This time is measured in seconds.

**Returns**

**inaddr** Address array returned from device.

**ints** Timestamps array returned from device.

**tick** Timestamp tick used in timestamps vector.

### 7.4 aemonseq.m

`[inaddr,ints,tick]=aemonseq(usbinterface,addr,ts,monitortime)`
Script to sequence and monitor events with one USBAERmini2 device.

**Parameters**

**usbinterface** Reference to USBAERmini2 device. Get a reference to a USBAERmini2 device using for example `usb0=factory.getInterface(0)`.

**addr** Array of addresses to be sent to device.

**ts** Array of interspike intervals. Note that no interspike interval should be bigger than $2^{16} - 1$. Also note that you have to set them according to the timestamp tick used on the device.

**monitortime** How long the monitoring is active. This time is measured in seconds.

**Returns**

**inaddr** Address array returned from device.

**ints** Timestamps array returned from device.

**tick** Timestamp tick used in timestamps vector.

### 7.5 aeseq.m

`aeseq(usbinterface,addresses,timestamps)`

Function to sequence events with one USBAERmini2 device. The device will stop sequencing when it has sequenced all events. Use `aeseq_cont.m` if you want to sequence continuously.

**Parameters**

**usbinterface** Reference to USBAERmini2 device. Get a reference to a USBAERmini2 device using for example `usb0=factory.getInterface(0)`.

**addr** Array of addresses to be sent to device.

**ts** Array of interspike intervals. Note that no interspike interval should be bigger than $2^{16} - 1$. Also note that you have to set them according to the timestamp tick used on the device.

### 7.6 aeseq_cont.m

`aeseq_cont(usbinterface,addr,ts)`

Function to continuously sequence events with one USBAERmini2 device. The device will rewind if it reaches the end of the arrays. This function is non-blocking. Call `aeseq_cont_stop.m` to stop sequencing. Use `aeseq.m` if you don't want to sequence continuously.

**Parameters**

**usbinterface** Reference to USBAERmini2 device.

**addr** Array of addresses to be sent to device.

**ts** Array of interspike intervals. Note that no interspike interval should be bigger than $2^{16} - 1$. Also note that you have to set them according to the timestamp tick used on the device.

### 7.7 aeseq_cont_stop.m

`aeseq_cont_stop(usbinterface)`

Stops a continuous sequencing device and releases it, so it can be accessed again from other processes.

**Parameters**

**usbinterface** Reference to the sequencing USBAERmini2 device.

### 7.8 multi_monitor.m

`[addr,isi,timestamps,tick]=multi_monitor(devices,monitortime)`

Function to monitor events with one or more USBAERmini2 devices.

**Parameters**

**devices** Array of references to monitor devices.

**monitortime** How long the monitoring is active. Time in seconds.

**Returns**

**addr** Cell array of address arrays. Same order as in monitors array.

**isi** Cell array of interspike intervals in nanoseconds.

**timestamps** Cell array of timestamps, unprocessed.

**tick** Timestamp tick used in timestamps arrays.

Make sure all the monitoring devices use the same timestamp tick! For synchronising the monitoring devices, connect SO pin of the desired master device to the SI pins of the slave devices, and set operation mode accordingly using `usbinterface.setOperationMode(mode)`. See 6 for more details.

### 7.9 multi_monitor_seq.m

```
[addr,isi,timestamps,tick]=
      multi_monitor_seq(sequencer,monitors,addr,ts,monitortime)
```

Function to sequence events with a USBAERmini2 device and monitor with one or more other USBAERmini2 devices.

**Parameters**

**sequencer** Reference to sequencer device.

**monitors** Array of references to monitor devices.

**addr** Array of addresses to be sent to device.

**ts** Array of interspike intervals. Note that no interspike interval should be bigger than $2^{16} - 1$. Also note that you have to set them according to the timestamp tick used on the sequencer device.

**monitortime** How long the monitoring is active. Time in seconds.

**Returns**

**addr** Cell array of address arrays. Same order as in monitors array.

**isi** Cell array of interspike intervals in nanoseconds.

**timestamps** Cell array of timestamps, unprocessed.

**tick** Timestamp tick used in timestamps arrays.

Make sure all the monitoring devices use the same timestamp tick! For synchronising the monitoring devices, connect SO pin of the desired master device to the SI pins of the slave devices, and set operation mode accordingly using `usbinterface.setOperationMode(mode)`. Use one of the monitoring devices as master and use the sequencing device in slave mode and don't forget to connect it's SI pin. If you use it master mode, it will start sequencing a few milliseconds before the other devices start monitoring.

See 6 for more details on the operation modes.

# 8 Importing Recorded Data To Matlab

To import data into Matlab that was recorded with the jAER viewer, use the script `loadaerdat.m` located in `\host\matlab\`. A window will pop up where you can navigate to the desired file. The function returns two vectors, one containing the addresses, one containing the timestamps. Remember that the timestamp vector represents absolute timestamps, which are not suited directly for sequencing! Create a suitable vector with the following Matlab command.

```
isi = [timestamps(1); diff(timestamps)];
```

# 9 Updating Firmware

The Cypress firmware can be updated from the jAER by launching the CypressFX2 EEPROM utility in the CypressFX2 menu. There are now two firmware versions available: a new one with JTAG support and an older one for the few boards out there which use the FX2 instead of the FX2LP (the FX2 has only 8k RAM which is not enough to run the new firmware). Before downloading the new firmware with JTAG support, please make sure your board uses the FX2LP, the device number of the FX2LP is CY7C68013A or 014A, the one of the FX2 is without A.

To program the CPLD from the FX2, you have to add jumpers on the JTAG connector JP3 to connect the FX2 pins to the Coolrunner JTAG ports. On the connector, connect pin 6 to pin 12, 5 to 11, 4 to 10 and

3 to 9. You can now download the latest CPLD firmware using the CypressFX2EEPROM utility or directly from the *Update firmware...* command in the *CypressFX2* menu.

To update the CPLD firmware on the boards with an FX2, a Xilinx download cable and the Xilinx Impact application are required. The Impact project file is stored in the repository under `\deviceFirmwarePCBLayout\USBAERmini2CPLD\USB` `USBAERmini2\USBAERmini2.ipf`. Please note that this project file stores the location of the firmware file with an absolute path, you will have to edit it with a text editor to change the path to your checkout directory.

The current PCB unfortunately doesn't have printed information on how to connect the Xilinx JTAG download cable. The JTAG connector is JP3, the pin layout is the following:

| Pin | Description |
|---|---|
| 7 | VCC |
| 8 | GND |
| 9 | TCK |
| 10 | TDO |
| 11 | TDI |
| 12 | TMS |

## 10 Using A Device Without Firmware

This section describes how to handle a device without EEPROM or with an EEPROM that has not been programmed yet. This would be the case for a newly built device.

Plug the device into your computer and follow the driver installation instructions found in the jAER wiki. After successful driver installation, the device will show up as *CypressFX2Blank* in the device manager. Now start the jAER application. When the jAER application finds a blank FX2 device, it will automatically download the firmware for the TmpDiff128 retina to its RAM. Therefore you will be prompted again to install the driver. When the device is successfully installed as TmpDiff128 device, you can run the CypressFX2EEPROM utility from the CypressFX2 menu of the jAER application.

Press *Scan for device* to open the blank device, and then press *EEPROM Monitor/Sequencer JTAG firmware* to download the USBAERmini2 firmware to the device.

Now you can follow the instructions in section 9 to program the CPLD.

## 11 Power Issues

The USBAERmini2 was actually designed to be used with the Cypress FX2LP, which is an improved version of the FX2 and needs less power.

Some of the boards built in Sevilla still use the FX2 though, which draws more than 100mA from the USB bus during enumeration, therefore some hosts may disable the device, as this is not within the USB specifications. If this is the case and the USBAERmini2 is not recognised, please try another USB port on your computer.