



**OXFORD  
SEMICONDUCTOR**

**UG-0004 Apr 04**

# **Oxford Semiconductor Uploader User Guide**

**Oxford Semiconductor Limited  
25 Milton Park  
Abingdon  
Oxfordshire  
OX14 4SH, UK  
(44) 1235 824900**

**<http://www.oxsemi.com>**

*All trademarks are the property of their respective owners*

© Oxford Semiconductor Limited 2004

The content of this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Oxford Semiconductor Limited. Oxford Semiconductor Limited assumes no responsibility or liability for any errors or inaccuracies that may appear in this book.

<b>Contents</b> .....	<b>-iii</b>
<b>Preface</b> .....	<b>v</b>
Revision Information .....	v
Typographic conventions .....	v
Reference Documents .....	v
Contacting Oxford Semiconductor .....	vi
<b>Glossary of Terms</b> .....	<b>vii</b>
<b>Chapter 1 Introduction</b> .....	<b>1</b>
<b>Chapter 2 Using the Uploader</b> .....	<b>3</b>
Uploader Main Screen .....	3
Typical Uploading Process .....	6
Viewing the Log .....	7
Get Image & the Cloning Process .....	8
<b>Chapter 3 Selecting a Flash Memory</b> .....	<b>9</b>
Using a New Flash Type .....	11
<b>Chapter 4 Initializing the Uploader</b> .....	<b>13</b>
Uploader Settings .....	13
Modifying the Initialization Settings .....	14

<b>Chapter 5 Modifying the Configuration Settings</b> .....	<b>17</b>
Updating the Configuration Settings .....	17
Advanced Options .....	20
Uploading Changes .....	26
<b>Chapter 6 Uploading a Binary Image</b> .....	<b>27</b>
Program Size Limits .....	28
The Uploading Process .....	28
<b>Chapter 7 Adding a Flash Memory Type</b> .....	<b>31</b>
Adding a New Flash Memory Type .....	31
Using a Flash Memory .....	35
OXFW900 .....	37
OXFW911 .....	37
<b>Appendix A Force-Flash Mode</b> .....	<b>37</b>
OXUF922 .....	38
OXAV940 .....	38
OXFW912, OXFW911plus, OXFW911-ATAPI .....	40
OXFW970 .....	41
Upgrading the Uploader .....	43
Installing the Uploader .....	43
<b>Appendix B Installing the Uploader</b> .....	<b>43</b>
<b>Appendix C Troubleshooting</b> .....	<b>47</b>
MAC Uploader Debug Information .....	49
Assumptions .....	50

This manual documents the Oxford Semiconductor Uploader utility.


## Revision Information

[Table I](#) documents the revisions of this manual

<i>Table I Revision Information</i>	
Revision	Modification
February 2004	First publication
April 2004	Revised to coincide with an uploader upgrade

## Typographic conventions

In this manual, the conventions listed in [Table II](#) apply.

<i>Table II Typographic Conventions</i>	
Convention	Meaning
<i>Italic Letters With Initial Capital Letters</i>	A cross-reference to another publication
Courier Font	Software code, or text typed in via a keyboard
1, 2, 3	A numbered list where the order of list items is significant
■	A list where the order of items is not significant
"Title"	Cross-refers to another section within the document
	Significant additional information

## Reference Documents

The following documents are referenced in this manual:

*Serial Bus Protocol 2 Specification (SBP-2)*

ANSI NCITS 325-1998  
ANSI, 20 November 1998

## Contacting Oxford Semi- conductor

Oxford Semiconductor contact details:

**Oxford Semiconductor Ltd.**  
25 Milton Park  
Abingdon  
Oxfordshire  
OX14 4SH

United Kingdom

Website: <http://www.oxsemi.com>

Telephone: +44 (0) 1235 824900

Fax: +44 (0) 1235 821141

Email: [sales@oxsemi.com](mailto:sales@oxsemi.com)

Alternatively, you can contact your local representative.

**Table I** summarizes the terms and abbreviations used in the *Oxford Semiconductor Uploader User Guide*.

<b>Table I Glossary of Terms &amp; Abbreviations</b>	
<b>Term</b>	<b>Meaning</b>
Force-flash mode	The internal processor of a device in force-flash mode is disabled and its flash port is enabled so that a binary image can be loaded directly into flash memory. The means of putting a device into force-flash mode depends on its type; see Appendix A <a href="#">Force-Flash Mode</a> for details
JVM	Java virtual machine
Tag	A marker in the binary image for either USB, FireWire or audio interfaces. A tag contains the relative address in flash memory of configuration information for the interface, which the uploader modifies before uploading the binary image
SBP-2 Unit directory	<p>The <i>Serial Bus Protocol 2 Specification (SBP-2)</i> (ANSI NCITS 325-1998) specification states that configuration ROM for targets must contain at least one unit directory in the format specified by the SBP-2 standard, containing Unit_Spec_ID &amp; Unit_SW_Version entries, as specified by ISO/IEC 13213, and a Management_Agent entry, as specified by this SBP-2 standard.</p> <p>Targets must implement at least one logical unit: logical unit zero. Additional logical units may be implemented.</p> <p>A logical unit is described by entries in the unit directory or by entries in a logical unit directory depending on the unit directory, or by entries taken in combination from both. The properties of logical units are established by Command_Set_Spec_ID, Command_Set, Command_Set_Revision and Unit_Characteristics entries. An instance of a specific logical unit is established by a Logical_Unit_Number entry.</p> <p>The unit directory may also contain a Unit_Unique_ID entry</p>

This page is intentionally blank

# Introduction

The Oxford Semiconductor uploader is a GUI utility which is designed to be used from a host platform, to allow users to program blank Oxford Semiconductor devices or upgrade existing firmware. The uploader runs on the following platforms:

- Linux
- Macintosh
- Windows

The uploader detects the status of devices attached to the host platform on the 1394 bus. It readily identifies blank devices, but where there is existing firmware the uploader tries to identify the flash memory type and firmware version. For more recent devices, the uploader can also identify the Oxford Semiconductor device type.

The uploader works with a variety of flash memory components, which allow it to handle the full range of board designs used with the Oxford Semiconductor device family. Its facilities also allow users to define new flash memory components to extend the uploader's capabilities and increase its flexibility.

The uploader can be used to view or modify a device's configuration and initialization settings and review the flash programming command sequence for a selected flash memory type.

The default mode of the uploader is to upload firmware to a single device, but if used in conjunction with the Oxford Semiconductor cloning utility, it can upload identical firmware to a batch of devices.

- ⚡ For more about the cloner utility, see *Using the Oxford Semiconductor Cloning Utility*.
- ⚡ For details of how to install the Oxford Semiconductor uploader, see Appendix B [Installing the Uploader](#).

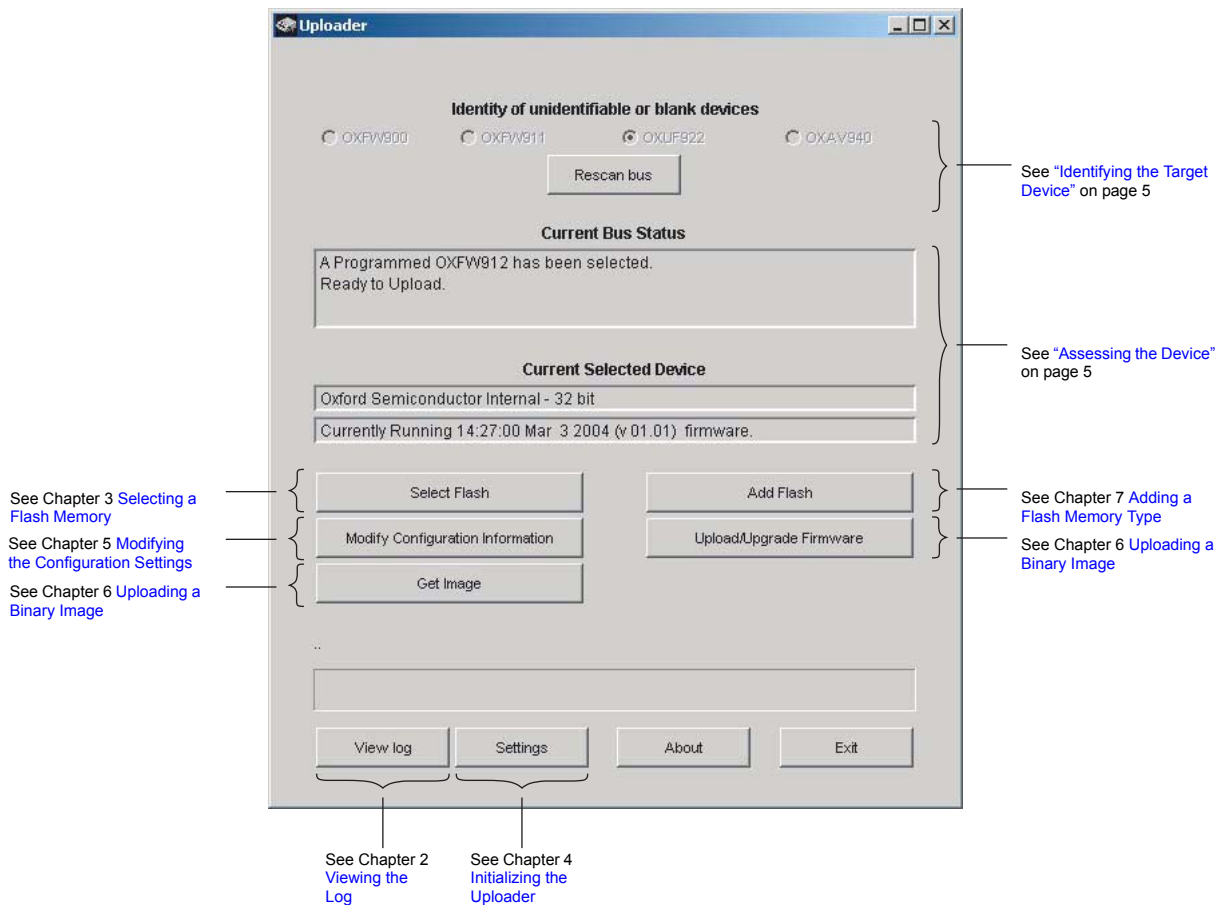
This page is intentionally blank

# Using the Uploader

To use the Oxford Semiconductor uploader to program a device, the development board containing the device must be connected to the 1394 bus on the host machine and powered up before starting the uploader.

## **Uploader Main Screen**

[Figure 2-1](#) on page 4 shows the uploader's main screen display.

**Figure 2-1 Uploader Main Screen**

When the uploader starts up, it populates the main screen with a combination of details from the initialization settings file and data read from the device on the 1394 bus.



Information in the uploader initialization file includes parameter settings that enable operational features of the uploader such as adding new flash memory types. For further details, see Chapter 4 [Initializing the Uploader](#).

If there are problems with detecting the current firmware on a device, the uploader delivers a warning that it will be necessary to use force-flash mode to program it. Appendix A [Force-Flash Mode](#) covers force-flash mode in greater detail and explains how to put a device into force-flash mode.

## Identifying the Target Device

The uploader scans the 1394 bus for an attached device and tries to identify it. By default, the uploader can automatically identify the most recent additions to the Oxford Semiconductor device range, and proceeds to assess the firmware without pausing. However, the uploader cannot detect the following device types if they are not running Oxford Semiconductor firmware:

- OXFW900
- OXFW911
- OXUF922
- OXAV940

For these devices, select the appropriate radio button from those shown under **Identity of unidentifiable or blank devices** to allow the device assessment to continue.

If you swap the device attached to the 1394 bus for a different device of the types listed, clicking the **Rescan bus** button prompts the uploader to scan the bus again.



Identifying the device types of recent Oxford Semiconductor devices automatically can be time-consuming. It is possible to turn this facility off; see Chapter 4 [Initializing the Uploader](#) for details.

## Assessing the Device

The uploader examines the device to determine whether it has previously been programmed; the text box **Current Bus Status** shows the device identity and its programming state. **Current Selected Device** displays the type of flash memory on the device, and also the firmware version and the date when it was built.

If the uploader cannot establish the current status of the attached device, force-flash mode is required to program the device; see Appendix A [Force-Flash Mode](#) for details.

## Other Uploader Buttons

Buttons on the main uploader window are occasionally disabled, depending on the status of the device attached. This information is given in [Table 2-1](#) on page 6, which documents the buttons in the uploader main screen.

Button	Enabled?	Description
Rescan bus	Always	Prompts the uploader to scan the bus again. used when the bus device is replaced
Select Flash	Always	Used to specify which type of flash memory is used on the attached device; see Chapter 3 <a href="#">Selecting a Flash Memory</a>
Add Flash	Depends on uploader initialization file setting	Used to define a new type of flash memory and add it to the flash memory list; see Chapter 7 <a href="#">Adding a Flash Memory Type</a>
Modify Configuration Information	Always	Used to edit the ROM and USB descriptors for the attached device; see Chapter 5 <a href="#">Modifying the Configuration Settings</a>
Upload/Upgrade Firmware	When the uploader can successfully assess the attached device	Used to upgrade the firmware on the attached device; see Chapter 6 <a href="#">Uploading a Binary Image</a>
Get Image	Depends on uploader initialization file setting	Used to copy the binary image so that it can be cloned to a succession of devices of the same type
View Log	When the uploader can successfully identify the binary image on the attached device	Displays log details of the most recent upload process for the attached device; see “ <a href="#">Viewing the Log</a> ” on page 7
Settings	Always	Documents which tags are used for the attached device; see Chapter 4 <a href="#">Initializing the Uploader</a>

## Typical Uploading Process

The main use for the uploader is to transfer a binary image from the host PC to a single device attached to the host’s 1394 bus, irrespective of whether the device has been programmed previously.

Firmware cannot be uploaded until details of the recipient device, such as flash memory type and storage disk I/O modes, have been defined, so for new devices it is necessary to specify the following details first:

- Flash memory type (you might have to add a new flash memory type first) and system clock settings; Chapter 7 [Adding a Flash Memory Type](#) explains how to add a new flash memory type and selecting a flash memory is covered in Chapter 3 [Selecting a Flash Memory](#)
- Configuration ROM & USB settings, bus master & slave device types & I/O modes. See Chapter 5 [Modifying the Configuration Settings](#) (standard settings for each device are included with the uploader)
- Configuration tag information; see Chapter 4 [Initializing the Uploader](#)

The uploader facilities are used to supply this information. Although it is mandatory for first-time devices, it can be modified as required before subsequent uploads, so that users can iteratively refine applications under development.

Clicking the **Upload/Upgrade Firmware** button on the uploader main window initiates a firmware upload. There are two main parts to the upload process: in the first, an accelerator code fragment is temporarily copied to the device's flash memory. In the second, following a device reset the accelerator code fragment is executed to copy the configuration file and binary image into flash memory as it is transmitted by the host.

Firmware uploading is covered in greater detail in Chapter 6 [Uploading a Binary Image](#).

## Viewing the Log

When the programming procedure for a device has completed successfully, a log of the upload process can be displayed immediately afterwards. Clicking the **View Log** button displays the log, including details of the configuration file and tags used, the firmware file identity and its length. A typical log is shown in [Figure 2-2](#).

**Figure 2-2 Log Details**



The log is only available while the device is connected to the bus. Removing the device deletes the log.

The configuration file can be modified before uploading the binary image; see Chapter 5 [Modifying the Configuration Settings](#) for details. Similarly, it is possible to specify a different type of flash memory, as explained in Chapter 7 [Adding a Flash Memory Type](#), or edit the device ROM and USB descriptors, which are used to identify the location of configuration information in the binary image. This is covered in Chapter 4 [Initializing the Uploader](#).

Finally, Appendix C [Troubleshooting](#) offers suggestions on overcoming some of the problems occasionally encountered during firmware uploading.

## Get Image & the Cloning Process

The Get Image facility is used in the cloning process. It allows the user to replicate a binary image to any number of other identical devices. As part of the upload process, the chip ID of each recipient device is incremented.



For details of the cloning process and how to use the cloner, see the *Oxford Semiconductor Cloner User Guide*.

# Selecting a Flash Memory

The flash memory on an unprogrammed device must be specified before the Oxford Semiconductor uploader can copy a binary image to it. The uploader is supplied with a file of currently-supported flash memory devices, **flashes.dat**, from which you can select the appropriate flash memory for a device. Alternatively, you can use the uploader to define new memory types, which are added to **flashes.dat**.

⚡ It is only possible to specify a flash memory type for devices with an external flash memory. If a device has an internal flash memory, such as the OXFW912, selecting a flash memory type has no effect.

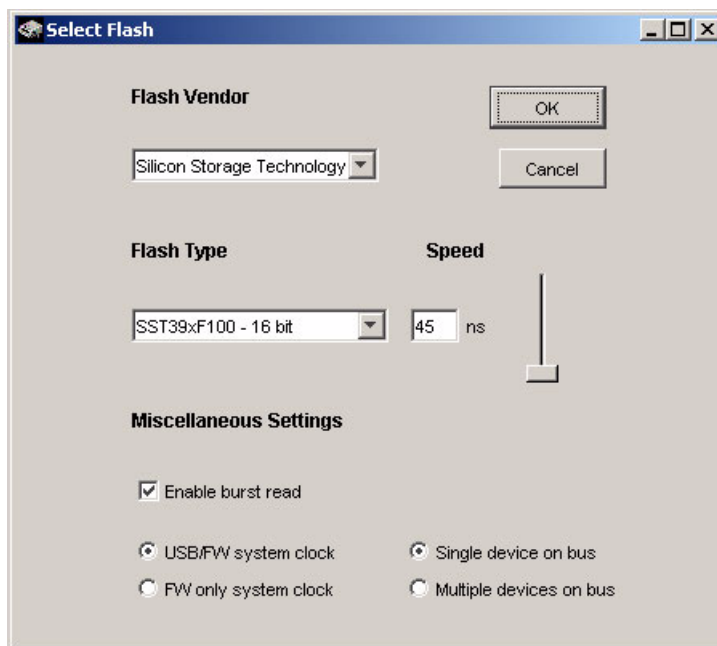
⚡ To define a new flash memory type, see Chapter 7 [Adding a Flash Memory Type](#).

To select the flash memory for a device, click the **Select Flash** button on the Uploader main screen to invoke the Select Flash screen.

⚡ If you are modifying the flash specification for a device that is already programmed, the uploader prompts for confirmation before displaying the Select Flash screen and then displays a warning to indicate the type flash memory in the device.

[Figure 3-1](#) on page 10 shows a typical Select Flash screen.

**Figure 3-1 Select Flash Screen**



In addition to specifying a flash memory type and its read mode, the uploader uses the other settings on this screen to calculate the wait states necessary for the processor to read from the flash device.

Table 3-1 summarizes the fields and buttons on the Select Flash screen

Group Field	Device	Field	Description
	All	Flash Vendor	Select the flash vendor from the drop-down box of all vendors currently supported
	All	Flash Type	Select the flash type from the drop-down box of all flash memories for this vendor. If the flash type is not present, you can add it; see <a href="#">Chapter 7 Adding a Flash Memory Type</a>
Speed	All	Text box	Either enter a speed or use the slider bar to adjust the flash read access time. Do not enter a value less than the speed specified in the flash memory data sheet
		Slider bar	
Miscellaneous Settings	All	Enable Burst Read	Check this to enable the facility; burst reads can be used to accelerate the execution of the firmware. Consult the data sheet to ensure that the flash memory can handle burst transfers
	OXUF922	USB/FW System Clock	Set this to match the device configuration. The clock source is usually the 48-MHz source, which is used as both the system clock and USB clock, with a separate crystal for the 1394 PHY. However, there is an additional mode in which the 1394 clock replaces the 48-MHz clock, but this precludes the use of the USB
		FW only system clock	
	All	Single device on bus	Select one of these to signify whether the flash memory is the only device on the static bus. This setting may have an impact on wait-state calculation. If multiple devices are attached, the flash memory is always assumed to be chip-select 0
Multiple device on bus			

The flash memory type and other settings specified on this screen are not written to the device until you either upload a binary image or upload changes having modified the configuration information. See Chapter 5 [Modifying the Configuration Settings](#) for more details.

## Using a New Flash Type

If the flash memory type on a device is not included in the **flashes.dat** file provided on the installation CD, you must specify it before you can reference it in the uploader. This is covered in Chapter 7 [Adding a Flash Memory Type](#). Although you can then select the new flash type as part of the programming procedure, the uploader displays the following warning message:

```
The current device has been programmed with a uncontrolled flash type.  
Verify that the reported flash device is correct before continuing
```

This display only appears when you have added new entries to the flash memory types file provided on the installation CD. It is a fail-safe mechanism to prevent device corruption in situations where several host platforms have been used to upload programs and each could be using a differently-modified **flashes.dat** file. You need to confirm that the correct flash type has been specified in the display. Click **OK** to confirm that the flash is correct and continue to program the flash memory.

This page is intentionally blank

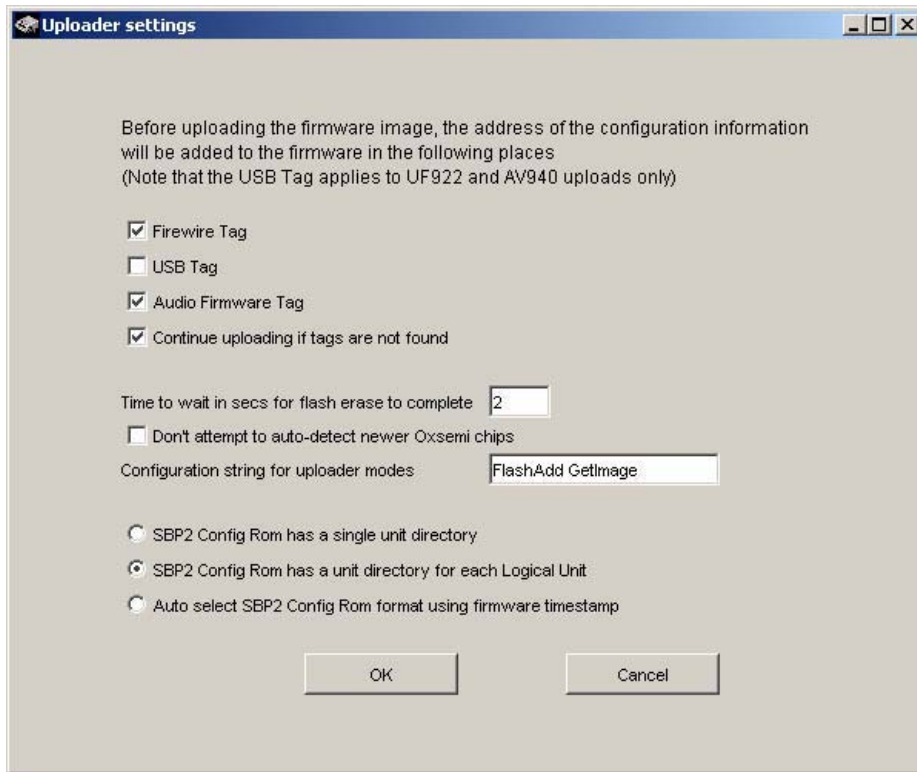
# Initializing the Uploader

The Oxford Semiconductor uploader uses the file **win\_uploader.ini** to store the uploader initialization settings. For example, it contains parameter settings that limit how the uploader can be used, assumptions about SBP-2 configuration ROM and flash-specific programming settings such as the delay between erasing the chip and commencing to reprogram the device. The uploader initialization settings can be modified as required, as explained in this chapter.

## Uploader Settings

To see the current uploader initialization settings, click the **Settings** button on the main uploader screen. [Figure 4-1](#) on page 14 shows a typical Uploader settings screen.

**Figure 4-1 Uploader Settings Screen**



## Modifying the Initialization Settings

In the Uploader settings screen, checking any of the top three boxes causes the address of the configuration information for the interface to be written to the appropriate tag in the uploaded firmware. The uploader can search for the following types of tag:

- FireWire (used for SBP-2)
- USB
- Audio

If the check box **Continue uploading if tags are not found** is set, warnings generated when the uploader cannot find a required tag in the binary image are ignored.



Tags are a device-independent, efficient and flexible means of specifying the location of configuration information in a flash memory device. Because configuration information is usually stored in the last addressable sector of flash memory, the address at which it is located depends on the flash type being used, which makes it impractical to code this address into the firmware. Instead, the uploader searches for a tag in the firmware image and sets the appropriate address in the image before uploading it to the device.

The Uploader settings window also controls the user-defined delay before commencing to program flash memory and is used to turn off the automatic detection of certain Oxford Semiconductor devices, which can be relatively time-consuming.

The parameters and their significance are summarized in [Table 4-1](#) on page 15.

<b>Table 4-1 Uploader Settings Parameter Summary</b>	
<b>Text String</b>	<b>Significance &amp; Value</b>
Time to wait in secs for flash erase to complete	Supply the time to wait after a chip-erasure before starting to program flash memory. Consult the flash memory's data sheet for details
Don't attempt to auto-detect newer Oxsemi chips	If the device is not an OXFW912, OXFW911 <sup>plus</sup> , OXFW970 or OXFW911-ATAPI, check this to speed up the detection process
Configuration string for uploader modes	Use this text box to add key words to enable certain buttons that would otherwise be grayed out, as follows: FlashAdd—allows the user to add new flash memory types GetImage—allows the user to grab an image for subsequent cloning The key words must be separated by a space. To save the setting, exit & restart the uploader
SBP2 Config Rom has a single unit directory	Check this if a single unit directory used—even if two drives are connected
SBP2 Config Rom has a unit directory for each Logical Unit <sup>(1)</sup>	Check this if two unit directories are used. If so, each one can have its own unit directory
Auto select SBP2 Config Rom format using firmware timestamp	Check this to allow the uploader to determine what type of configuration ROM can be used, to correspond to the version of firmware being uploaded

**Note:** 1 This is the usual default setting (depending on the device). If there are any doubts about the firmware and whether it is capable of using two unit directories, check **SBP2 Config Rom has a single unit directory** instead.

This page is intentionally blank

# Modifying the Configuration Settings

The configuration settings for a device specify configuration information used by the firmware (such as disk I/O modes) as well as the configuration ROM and USB descriptors, which vary according to the device type. The configuration settings reside in the last sector of the device's flash memory, which is reserved for configuration information. The uploader reads this information as part of its assessment when it detects a programmed device on the 1394 bus.



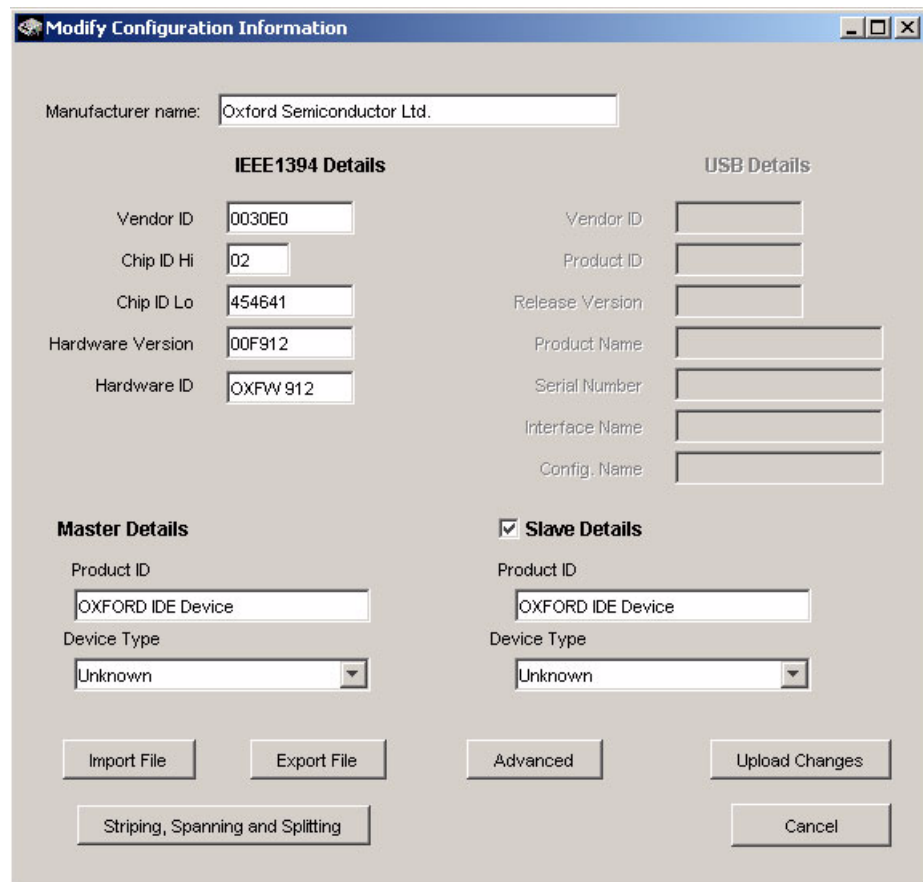
Because the uploader knows the flash memory device type, it can deduce the location of the configuration information and does not access tag information.

The uploader facilities allow the user to modify the configuration information, or replace it completely by importing a file, and then write it back to the device. Alternatively, the user can export the current configuration settings for use on another device.

## Updating the Configuration Settings

To see the current configuration information settings, click the **Modify Configuration Settings** button on the main uploader screen. [Figure 5-1](#) on page 18 shows a typical **Modify Configuration Information** screen.

**Figure 5-1 Modify Configuration Information Screen**



If the device on the 1394 bus is blank, the uploader prompts for a configuration file and allows the user to browse for it. When the configuration settings are loaded, the user can modify them for the new device.

Acceptable configuration settings for a device vary depending on the device type; the uploader grays-out fields that do not apply to the device detected on the bus, to disable them. For example, in [Figure 5-1](#) the USB details are disabled because an OXFW912 is connected to the 1394 bus—the OXFW912 has no USB.

[Table 5-1](#) on page 19 summarizes the fields and buttons on the Modify Configuration Information screen and indicates which fields are available for specific device types. If a field is not grayed-out, it can be modified.

<b>Table 5-1 Modify Configuration Information Fields</b>			
<b>Group Field</b>	<b>Device</b>	<b>Field</b>	<b>Description</b>
		Manufacturer Name	Creator of the implementation; used by third-parties to identify the device
IEEE 1394 Details	All	Vendor ID	Vendor ID; obtained from the 1394 Trade Association
		Chip ID Hi	Unique ID. Avoids potential ID conflicts when daisy-chaining several devices. Field lengths are fixed as follows: Chip ID H—2 hexadecimal digits, range 0x00 to 0xFF Chip ID Lo—6 hexadecimal digits, range 0x000000 to 0xFFFFF
		Chip ID Lo	
		Hardware Version	Part of the SBP unit directory; used by third-parties to identify the device; 6 hexadecimal digits
		Hardware ID	Device type; a string
USB Details	OXUF922 OXAV940	Vendor ID	Vendor ID; obtained from the USB -2 specification; 4 hexadecimal digits
		Product ID	From the device descriptor; consult the USB-2 specification; 4 hexadecimal digits
		Release Version	
		Product Name	From the device descriptor; consult the USB-2 specification; a string
		Serial Number	
		Interface Name	
		Config. Name	
Master Details	All	Product ID	Device attached to the chip, to be used as a master
		Device Type	The type of storage device attached to the chip; select from one of the following: Unknown Generic Win98 Drive Hard disk drive CD/DVD Drive MO Drive (magnetic optical drive) Tape Drive Selecting Unknown gives greater flexibility. It causes the device to issue an inquiry command at reset which allows the type of device to be detected and returned. If the storage device to be used is known and will not change, the type of device (CD-ROM etc.) can be specified
Slave Details	All	As for Master Details, but only to be used if a slave device will be used on the chip as well as a master	
		<i>Check box</i>	Check this to signify that the attached disk drive is a slave device
Buttons	All	Import File	Click to navigate to a previously-saved configuration file and import it
	All	Export File	Click to export these settings to disk as a text file
	All	Advanced	Click to specify the I/O modes for a drive attached to the chip; see <a href="#">“Advanced Options”</a> on page 20 for details
	OXFW912 OXFW911 <i>plus</i>	Striping, Spanning and Splitting	Click to specify ORB splitting & specialist disk handling; see <a href="#">“Striping, Spanning &amp; Splitting”</a> on page 23 for details
	All	Upload Changes	Click to write the configuration details back to the device's flash memory

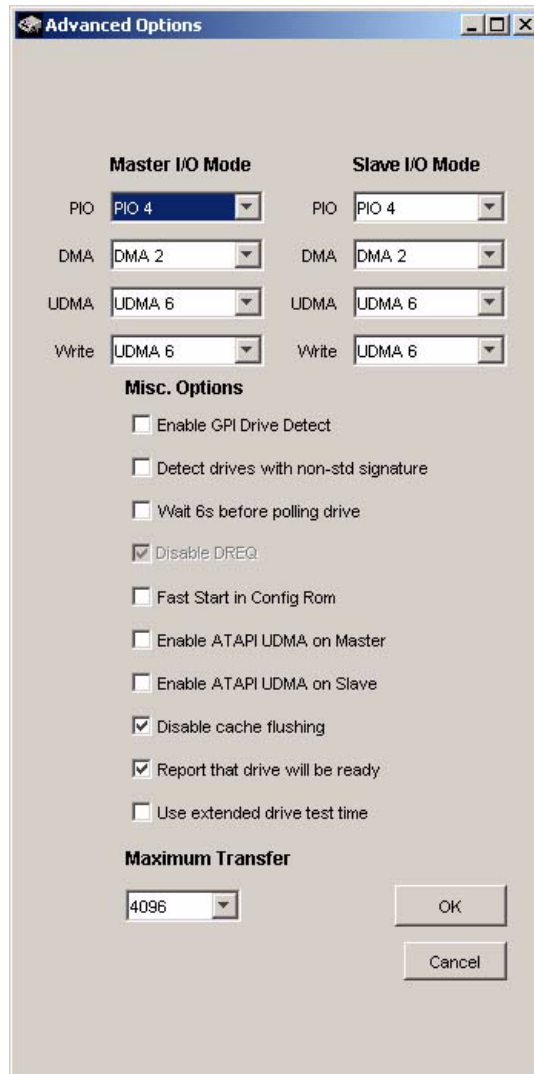
# Advanced Options

Each type of chip requires additional specific information to complete its configuration data, which the user supplies on an Advanced Options screen. For example, additional information for the OXFW912 relates to the current storage device transfer modes; and for the OXFW970, additional information covers audio configuration. For any Oxford Semiconductor chip, clicking the **Advanced** button on the Modify Configuration Options screen invokes the appropriate screen to complete its configuration process.

## Advanced Options for non-OXFW970 Devices

Figure 5-2 shows a typical Advanced Options screen for an OXFW900, OXFW911, OXFW911*plus*, OXFW911-ATAPI, OXFW912, OXUF922 or OXAV940.

**Figure 5-2 OXFW912 Advanced Options Screen**



**Table 5-1** summarizes the fields on the Advanced Options screen for non-OXFW970 devices. All fields can be modified if they are not greyed-out; clicking **OK** accepts any changes and closes the screen.

<b>Table 5-2 Advanced Options Fields (Sheet 1 of 2)</b>			
<b>Group Field</b>	<b>Device</b>	<b>Field</b>	<b>Description</b>
Master I/O Mode	All	PIO	The settings for these characteristics determine the IDE bus speed; see <a href="#">Table 5-3</a> on page 22
	All	DMA	
	All	UDMA	The UDMA mode for writing can be set to the same speed as for read mode, or can be lower
	All	Write	
Slave I/O Mode	As for Master I/O Mode, but only to be used if a slave device will be used on the chip as well as a master		
Misc. Options	OXFW911 OXUF922	Enable GPI Drive Detect	Check this to enable the facility; all OXFW911 board designs should use GPI to allow power-on drive detection.
	OXFW911 OXUF922	Detect drives with non-std. signature	Check this to enable the facility. ATA/ATAPI device detection is performed using the drive's signature, but some devices do not obey the ATA specification & are not easily detected by the firmware. This option relaxes the checking criteria used by the firmware in case of false signature, to allow certain problem devices to be used
	OXFW911 OXUF922	Wait 6s before polling drive	Check this to enable the facility. Certain ATAPI devices malfunction if they are accessed immediately after being powered. This setting causes the firmware to wait for 6 seconds before polling the drive in order to prevent this problem
	None	Disable DREQ	This is a test mode for internal use only
	OXFW911 OXUF922	Fast Start in Config ROM	Check this to enable fast start. Fast start is a feature of SBP-3 which modifies the SBP unit directory to indicate to a host that the device is capable of handling fast-start ORBs. This is a new feature which might improve performance
	All	Enable ATAPI UDMA on Master	Check this to enable an ATAPI device (CD-ROM, for example) to use UDMA data transfer mode on a master device
	All	Enable ATAPI UDMA on Slave	Check this to enable an ATAPI device (CD-ROM, for example) to use UDMA data transfer mode on a slave device
	All	Disable cache flushing	Check this to prevent cache flushing commands from being sent to the drive. Not needed if the host operating system disables flush caching. Also depends on the mode selected. This facility allows users to decide whether to synchronize cache commands.
	All	Report that drive will be ready	Must be checked
	All	Use extended drive test time	Check to allow extra time for a response to the initial <b>identify device</b> command

Group Field	Device	Field	Description
Drop-down box		Maximum Transfer	Allows users to select the maximum data payload (in bytes) of individual 1394 block read & write requests to & from the device. The maximum selectable value depends on the device as follows: OXFW900—1024 OXFW911—2048 OXUF922—2048 OXAV940—2048 OXFW912—4096 OXFW911-ATAPI—2048 OXFW911 <sup>plus</sup> —2048 OXFW970—2048

Table 5-3 lists the settings for master and slave I/O modes available for drives attached to the chip.

Mode & Protocol	Original Standard	Max Bus Speed
PIO Mode 0	ATA (ATA-1)	3.33 MBytes/s
DMA Mode 0 or Multiword DMA Mode 0	ATA	4.16 MBytes/s
PIO Mode 1	ATA	5.22 MBytes/s
PIO Mode 2	ATA	8.33 MBytes/s
PIO Mode 3	ATA-2	11.1 MBytes/s
DMA Mode 1 or Multiword DMA Mode 1	ATA-2	13.3 MBytes/s
PIO Mode 4	ATA-2	16.6 MBytes/s
DMA Mode 2 or Multiword DMA Mode 2	ATA-2	16.6 MBytes/s
Ultra DMA Mode 0	ATA/ATAPI 4	16.6 MBytes/s
Ultra DMA Mode 1	ATA/ATAPI 4	25.0 MBytes/s
Ultra DMA Mode 2, UDMA33 or ATA/33	ATA/ATAPI 4	33.3 MBytes/s
Ultra DMA Mode 3	ATA/ATAPI 5	44.4 MBytes/s
Ultra DMA Mode 4, UDMA66 or ATA/66	ATA/ATAPI 5	66.6 MBytes/s
Ultra DMA Mode 6, Ultra DMA Mode 5, UDMA100 or ATA/133	ATA/ATAPI 6	133 MBytes/s

**Note:**

- 1 There is a data integrity issue with UDMA modes in the OXFW900 which prevents them from being used with the OXFW900 devices.

## Striping, Spanning & Splitting

Disk spanning, striping and splitting can be specified for the OXFW911*plus* and OXFW912. When disk spanning is implemented, a host handles two same-sized disks as one unit for storage purposes. With disk striping, the stripes are defined on two identical drives so that data can be accessed in alternate blocks to optimize the storage access performance, and users can specify the stripe size. However, simultaneous disk striping and disk spanning is not possible.

The OXFW911*plus* and OXFW912 support both hardware and software ORB splitting. Hardware ORB-splitting can be used for ATA 5 and ATA 6 devices. Software ORB-splitting is available for all ATA devices.

Clicking the **Striping, Spanning & Splitting** button on the Modify Configuration Information screen displays the Additional FW912 Configuration screen, which allows the user to modify the disk management procedures for the device. This information comes from the device configuration settings in the flash memory. [Figure 5-3](#) shows the Additional FW912 Configuration screen.

**Figure 5-3 Additional FW912 Configuration Screen**

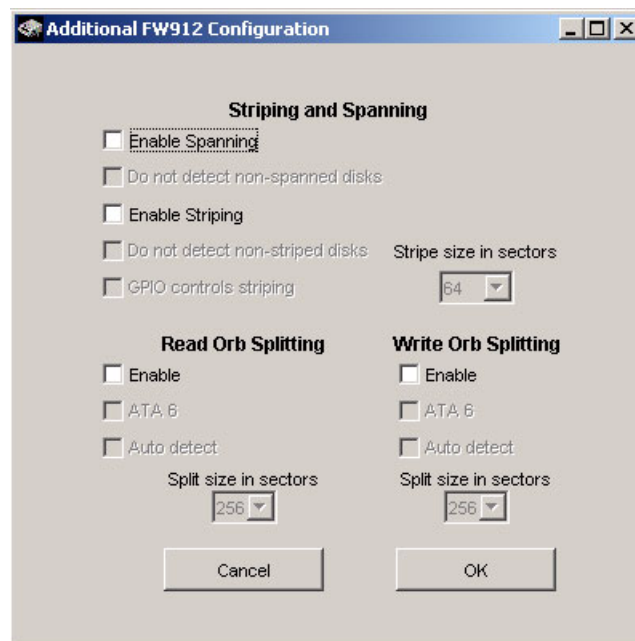


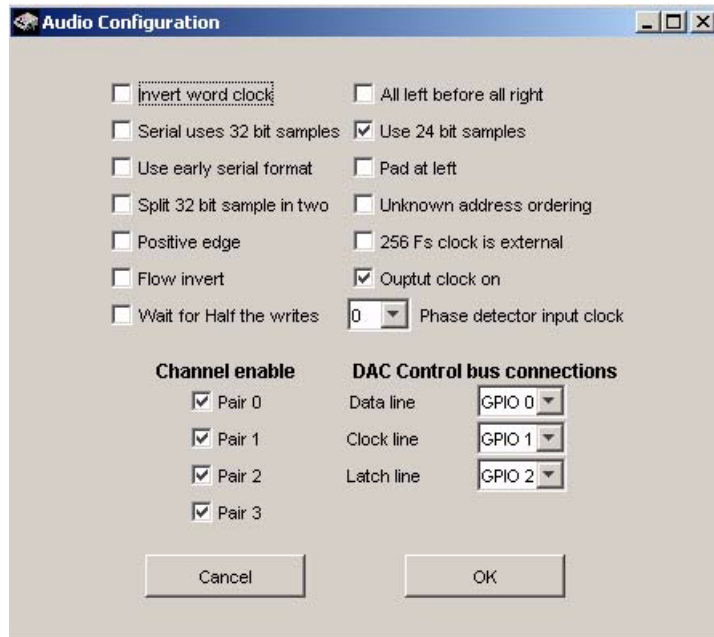
Table 5-4 summarizes the fields on the Additional FW912 Configuration screen. All fields can be modified; clicking **OK** accepts any changes and closes the screen

<b>Table 5-4 Additional FW912 Configuration Screen Fields</b>		
<b>Group Field</b>	<b>Field</b>	<b>Description</b>
Striping and Spanning	Enable Spanning	Check this to select the facility & enable the option for detecting non-spanned disks
	Do not detect non-spanned disks	Check to enable the facility & ignore drives on the bus without spanning. This prevents an individual disk from being detected—it prevents one of a pair from being detected when the other has failed. Leaving the box unchecked results in an individual drive being found, but the device being inoperable
	Enable Striping	Check this to select the facility & enable remaining options for striped disks
	Do not detect non-striped disks	Check to enable the facility and ignore drives on the bus without striping
	GPIO controls striping	Check to enable & disable striping by fitting a jumper
	Stripe size in sectors	Select a stripe size for the striped disks on the bus. For optimum performance, select 256
Read Orb Splitting	Enable	Check to enable the remaining options for read ORB splitting and allow read ORB splitting by default on ATA 5
	ATA 6	Check this to allow read ORB splitting on ATA 5 and ATA 6
	Auto detect	Check this to set the read ORB size depending on the drive manufacturer
	Split size in sectors	Select a split size for read ORBs that are to be split
Write Orb Splitting	Enable	Check to enable the remaining options for write ORB splitting and allow write ORB splitting by default on ATA 5
	ATA 6	Check this to allow write ORB splitting on ATA 5 and ATA 6
	Auto detect	Check this to set the write ORB size depending on the drive manufacturer
	Split size in sectors	Select a split size for write ORBs that are to be split. Optimal size depends on the drive

## OXFW970 Advanced Options

With an OXFW970 connected to the 1394 bus, clicking the **Advanced** button on the Modify Configuration Options screen invokes the Audio Configuration screen, as shown in [Figure 5-3](#) on page 23.

**Figure 5-4 OXFW970 Audio Configuration Screen**



[Table 5-5](#) on page 25 summarizes the fields on the Audio Configuration screen for the OXFW970. All fields can be modified; clicking **OK** accepts any changes and closes the screen.

<b>Table 5-5 OXFW970 Audio Configuration Parameters (Sheet 1 of 2)</b>		
<b>Group Field</b>	<b>Field</b>	<b>Description</b>
	Invert word clock	The word clock signal toggles to indicate that a new word is about to be transmitted when the transmitter does not know the word length of the receiving system. The receiving system accepts as many bits from each sample as it requires. If there are not enough bits for the sample, the receiver pads the least-significant bits with zeros. If too many samples are transmitted, the receiver ignores the least-significant bits
	Serial uses 32 bit samples	Check to modify the sample size
	Positive edge on	Check to use positive-edge sampling
	Use early serial format	Check to signify that the first data bit data arrives on the first rising clock edge after the word clock; otherwise the first data bit arrives on the second rising clock edge following the word clock

**Table 5-5 OXFW970 Audio Configuration Parameters (Sheet 2 of 2)**

Group Field	Field	Description
	Split 32 bit sample in two	Check to signify that 16-bit left & right samples arrive in one 32-bit register, & must be split & appropriately padded; otherwise data is assumed to be 24-bit data
	Flow invert	Check to use flow inversion
	Wait for half the writes	Check to direct hardware to wait for half the number of APB writes before proceeding
	All left before all right	Check to write all left channels followed by all right; otherwise writes left/right pairs
	Use 24 bit samples	Check to indicate that an APB write contains 24-bit data
	Pad at left	Check to pad left-most output data with zeros
	Unknown address ordering	Check to determine the RAM write address from apb_addr if the address is unknown
	256 Fs clock is external	Check to indicate that an external clock will be used instead of the audio core clock
	Output clock on	Check to output an audio clock
	Phase detector input clock	Check to indicate the presence of a phase detector input clock
	Channel enable	Check to indicate that, if a channel is disabled, no new data is presented to the data line of that channel. Software can select whether it wishes to write to this channel
Channel Enable	Pair 0	Check to enable this channel
	Pair 1	Check to enable this channel
	Pair 2	Check to enable this channel
	Pair 3	Check to enable this channel
DAC Control Bus connections		
	Data line	Select a GPIO mapping for the data line
	Clock line	Select a GPIO mapping for the clock line
	Latch line	Select a GPIO mapping for the latch line

## Uploading Changes

When all changes have been made to the configuration settings, the user can apply them to the attached device by clicking the **Upload Changes** button on the Modify Configuration Information screen. Only the configuration settings are re-written to the device at this stage.

# Uploading a Binary Image

The Oxford Semiconductor uploader copies a binary image from a specified file into the flash memory of a device attached to the host machine's 1394 bus. In conjunction with the cloner utility, the identical binary image can be uploaded to any number of devices.

Before firmware can be uploaded to the device, ensure that it is connected to the host machine, powered up and, if necessary, is in force-flash mode (see Appendix A [Force-Flash Mode](#) for details). In addition, check that the flash memory specification and the other configuration options and initialization settings are correct, because they materially affect the uploading process.

The process of uploading a binary image to the device depends on the state of the device, which is one of:

- Blank—the device needs both configuration information & binary image
- Production—devices with internal flash memory are usually supplied with the acceleration fragment pre-programmed into them during the production process
- Programmed—the device needs a binary image, but only needs configuration information if it has changed
- Programmed with non-Oxford Semiconductor firmware—the device needs both configuration information & binary image

## Program Size Limits

Table 6-1 shows the maximum sizes of program file that the uploader can handle for a device.

Device	Maximum Size
OXFW911	63.5 Kbytes
OXUF922	127 Kbytes
OXAV940	1024 Kbytes
OXFW912	62 Kbytes
OXFW911 <i>plus</i>	62 Kbytes
OXFW911-ATAPI	62 Kbytes
OXFW970	62 Kbytes

## The Uploading Process

To begin the upload process, click the **Upload/Upgrade** button on the Uploader main screen. The upload procedure differs according to the initial state of the device.

For blank devices, the sequence is as follows:

- 1 The uploader erases the flash memory contents and uploads the appropriate acceleration fragment.
- 2 The uploader resets the device, copies the acceleration fragment into RAM and executes it.
- 3 If the device is still held in force-flash by a jumper or switch and the fragment is not running, the uploader prompts you to remove the jumper and reset the device.
- 4 The uploader displays the Select Configuration File screen, which allows you to navigate to the configuration information to be uploaded. Select a file and click **OK**.
- 5 The uploader displays the Select Firmware File screen, which allows you to navigate to the binary image to be uploaded. Select a file and click **OK**.
- 6 The uploader erases the flash memory contents and uploads the firmware and configuration information to the device.
- 7 The uploader resets the device and confirms that the new firmware is running on the device.

Devices with internal flash (OXFW911, OXFW911*plus*, OXFW911-ATAPI, OXFW970 and OXFW912) usually have the acceleration fragment programmed into them as part of the production process. For these devices, the sequence is as follows:

- 1 The uploader displays the Select Configuration File screen, which allows you to navigate to the configuration information to be uploaded. Select a file and click **OK**.

- 2 The uploader displays the Select Firmware File screen, which allows you to navigate to the binary image to be uploaded. Select a file and click **OK**.
- 3 The uploader erases the flash memory contents and uploads the firmware and configuration information to the device.
- 4 The uploader resets the device and confirms that the new firmware is running on the device.

For devices that are already programmed, clicking the **Upload/Upgrade** button updates the firmware but leaves the configuration information untouched. The sequence is as follows:

- 1 The uploader displays the Select Firmware File screen, which allows you to navigate to the binary image to be uploaded. Select a file and click **OK**.
- 2 The uploader erases the firmware from flash memory and replaces it with the new firmware.
- 3 The uploader resets the device and confirms that the new firmware is running on the device.



Most firmware provided by Oxford Semiconductor has an acceleration fragment built into it. The acceleration fragment is executed when updating the firmware, which speeds the upload process. However, if the uploader detects that the firmware has no built-in acceleration fragment, it updates the firmware using the flash port, which is much slower. The update proceeds satisfactorily, but it could take longer than usual.

This page is intentionally blank

# Adding a Flash Memory Type

The facilities of the Oxford Semiconductor uploader allow users to maintain flash memory information, which ensures that the uploader is always up to date with new flash memory components and flash command programming sequences. This chapter explains how to add a new flash memory type to the selection list.

The uploader stores flash memory information in the text file **/Oxford Semiconductor/data/flashes.dat**. When a new flash memory type is added, flashes.dat is updated. The uploader cross-references flashes.dat when it detects a device on the 1394 bus, and uses it to display the details of the device's flash memory type; see [Figure 2-1](#) on page 2-4.



If flashes.dat is updated after uploading a binary image to a device, the flash memory details display is likely to be indeterminate when the device is subsequently reconnected to the 1394 bus, and the uploader issues a warning.

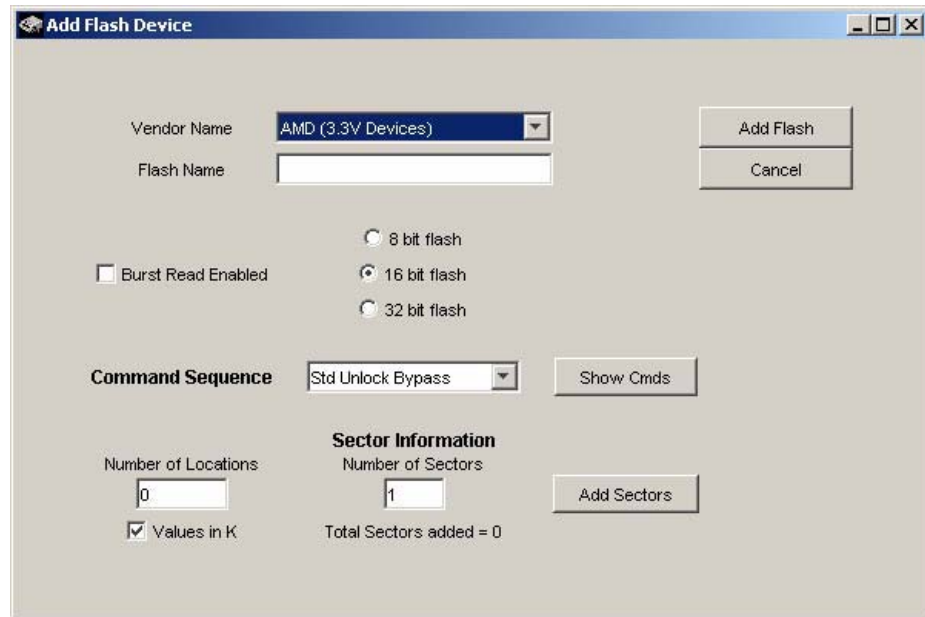
## Adding a New Flash Memory Type

To add a new flash memory type, click the **Add Flash** button on the main uploader screen to invoke the Add Flash Device screen shown in [Figure 7-1](#) on page 7-32.



If the Add Flash button is disabled, you need to modify the uploader initialization settings to enable it. For details of how to do this, see Chapter 4 [Initializing the Uploader](#).

**Figure 7-1 Add Flash Device Screen**



The process for adding a new flash memory type involves selecting a vendor from the drop-down list by **Vendor Name** and then typing the flash identifier into the **Flash Name** text box. At this point, you can supply other flash memory characteristics as detailed on the vendor’s device datasheet. The options are summarized in [Table 7-1](#).

**Table 7-1 Add Flash Device Fields (Sheet 1 of 2)**

Field Name	Description & Effect
Burst Read Enabled	Check to enable burst reads, which offers improvements in the system speed & performance by reducing sequential read access times
8 bit flash	Check the appropriate flash bus width (mutually exclusive)
16 bit flash	
32 bit flash	
Command sequence drop-down box	The command sequence/programming operation used to unlock, reset, program & chip-erase the flash memory. Click the drop-down box to select from the following types of flash programming command sequence after consulting the data sheet for the new flash memory: Std Unlock bypass Std Normal Ext Normal Ext Normal (8-bit ST) Ext Normal (16-bit ST) Unlock bypass (8 bit ST)
Show Cmds	Click to display the commands in the selected flash programming command sequence. For further details, see <a href="#">“Specifying the Flash Programming Command Sequence”</a>
Number of Locations	These define the size and number of sectors in the flash memory, which the uploader utilizes to optimize the flash programming process, especially the sector-erase operation. Group each chunk of flash memory according to its sector size (Number of Locations in <a href="#">Figure 7-1</a> ). See <a href="#">“Defining the Flash Capacity”</a> for more about this topic
Number of Sectors	
Value in K	Check this to indicate that the number of locations has been divided by 1024

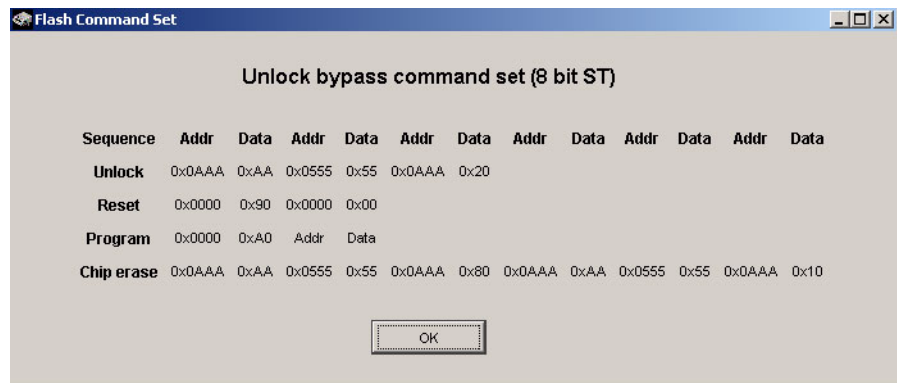
<b>Table 7-1 Add Flash Device Fields (Sheet 2 of 2)</b>	
<b>Field Name</b>	<b>Description &amp; Effect</b>
Add Sectors	Click to add this block of memory to the sector information
Total Sectors Added	This display-only field shows how many sectors have been added to the flash memory definition

Having completed the entry for the new flash memory type, clicking **Add Flash** appends it to the list of available flash types for the flash vendor.

### Specifying the Flash Programming Command Sequence

Each type of flash memory has an associated command set containing a flash programming command sequence. A flash command sequence is a rigid procedure for writing specified data to a device’s flash memory to signify a specific type of flash access; it is a safeguard to prevent inadvertent flash memory corruption. There are separate command sequences for unlocking, resetting, writing and erasing the flash memory. [Figure 7-2](#) shows an example command sequence.

**Figure 7-2 Example Flash Programming Command Sequence**



The flash programming command set for a flash memory type is usually documented in its data sheet. When a new flash memory is added, the user must match it to one of the command sets available from the uploader. To view a command set, first select an entry from the **Command Sequence** drop-down box, and then click **Show Cmds** to display the command sequences.



The Oxford Semiconductor uploader handles the major command sets for programming flash memory.

## Defining the Flash Capacity

For the uploader to work properly and efficiently, the flash memory arrangement must be defined. The memory allocation required to specify the flash capacity for a flash memory type is documented in its data sheet.

The uploader treats contiguous sectors of memory with the same number of locations as one unit of memory. For example, for the AM29LV001BB 8-bit memory, the sector information definition entered on the Add Flash Device screen would be as shown in [Table 7-2](#) on page 7-34.

<i>Table 7-2 Example AM29LV001BB 8-Bit Flash Memory Type Sector Information</i>				
Sector Size in Bytes	Add Flash Device Screen Entries			See flashes.dat lines ...
	Number of Locations (Flash Width)	Values in K	Number of Sectors	
8 K	8	Y	1	5
4 K	4	Y	2	6-7
16 K	16	Y	7	8-9

This information is stored in **flashes.dat** as follows:

```

1  0, AM29LV001BB
2  0
3  300
4  B8
5  k8
6  k4
7  k4
8  x7
9  k16
    
```

For the AM29LV200BB 16-bit memory, the sector information definition entered on the Add Flash Device screen would be as shown in [Table 7-3](#) on page 7-35.

Sector Size in Bytes	Add Flash Device Screen Entries			See flashes.dat lines ...
	Number of Locations (Flash Width)	Values in K	Number of Sectors	
16 K	8	Y	1	5
8 K	4	Y	2	6-7
32 K	16	Y	1	8
64 K	32	Y	3	9-11

This information is stored in **flashes.dat** as follows:

```

1  0,AM29LV200BB
2  0
3  300
4  B16
5  k8
6  k4
7  k4
8  k16
9  k32
10 k32
11 k32

```

## Using a Flash Memory

When you have successfully added a flash memory, it becomes available in the drop-down flash memory list box when you select the appropriate vendor for the device

To program a device that is using this flash memory, you must first select the flash type. The uploader erases the flash memory and then displays the following message:

```

The current device has been programmed with a uncontrolled flash type.
Verify that the reported flash device is correct before continuing

```

The purpose of this message is to ensure that the correct flash type has been specified. If the uploader has been installed on several PCs, it is possible that there are multiple **flashes.dat** files, and the uploader assumption could be wrong. Click **OK** to confirm that the flash is correct and continue to program the flash memory.

This page is intentionally blank

In Oxford Semiconductor devices, force-flash mode turns off a device's internal processor and enables the flash port so that a binary image can be received and stored in the flash memory. Methods of achieving force-flash mode depend on the device type. This appendix explains how to put the following Oxford Semiconductor devices into force-flash mode:

- OXFW900
- OXFW911
- OXUF922
- OXAV940
- OXFW912
- OXFW911*plus*
- OXFW911-ATAPI
- OXFW970

The steps for each device type assume that the device is initially completely disconnected and powered down.

### OXFW900

To put the OXFW900 into force-flash mode, proceed as follows:

- 1 Plug the OXFW900 board into the 1394 bus.
- 2 Power up the OXFW900.
- 3 On the OXFW900, pull pin 123 low and then release it.



While the OXFW900 is in force-flash mode, it responds BUSY to all requests except quadlet writes to offset 0xF0080000, which is the flash programming port.

### OXFW911

To put the OXFW911 into force-flash mode, proceed as follows:

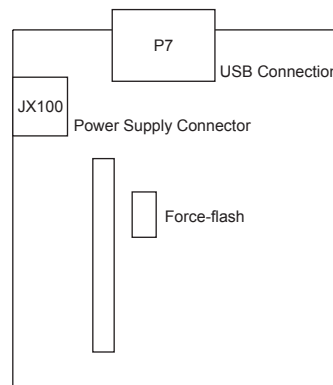
- 1 Fit jumper JP2 on the board (to short-circuit pin 57).
- 2 Power-up the OXFW911 board.
- 3 Connect the OXFW911 board to the 1394 bus.
- 4 Remove jumper JP2.

## OXUF922

The procedure for putting the OXUF922 into force-flash mode depends on the chip's test pins setting (TEST[2:0] in the pinout, refer to the *OXUF922 Hardware Reference Manual* for details of the pin locations on the package).

If the test pins have not been pulled low, the board is in test mode 111 and the firmware will not be uploaded to the board unless a jumper is placed on the force-flash pins as shown in [Figure A-1](#).

**Figure A-1 Jumper Position to Obtain Force Flash Mode on the OXUF922**



If the board is in test mode 000, the firmware can be uploaded without using the force-flash jumper.

## OXAV940

To put the OXAV940 into force-flash mode, proceed as follows:

- 1 Set switch 8 on block SW11 up.  
LED D2 illuminates to signify that the device is in force-flash mode.
- 2 Power up the OXAV940 development board.
- 3 Connect the OXAV940 development board to the 1394 bus.
- 4 Reset the device.

[Figure A-2](#) on page 39 shows the switch settings supplied with the development board, and indicates whether each switch is on or off. Switch 8 is shown as down (off), which is the setting to disable force-flash mode.

**Figure A-2 SW11 Settings**

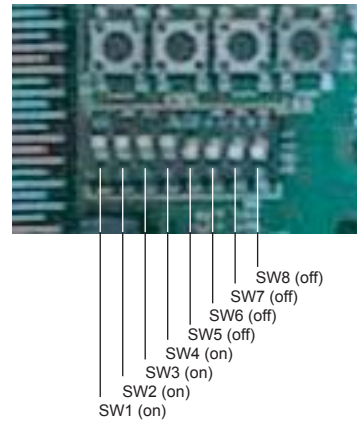
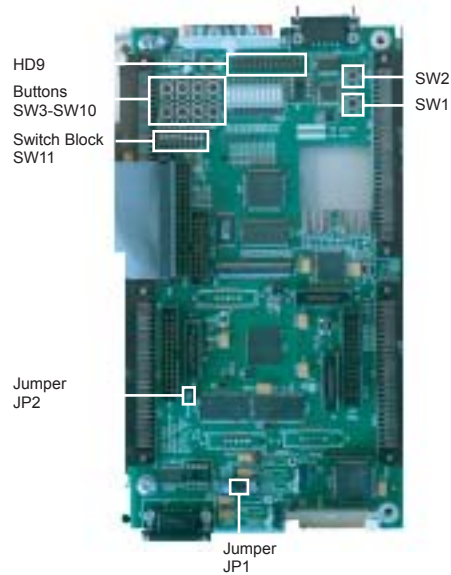


Figure A-3 locates SW11 on the OXAV940 development board.

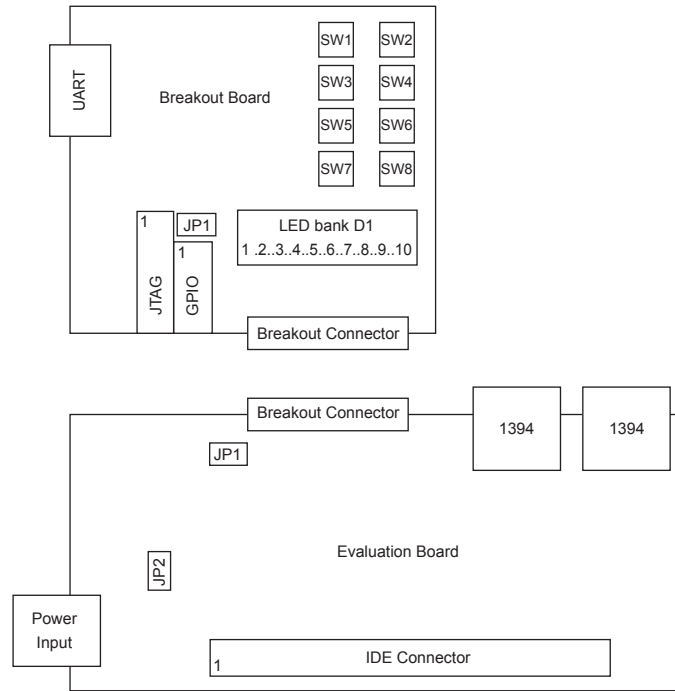
**Figure A-3 SW11 on the OXAV940 Development Board**



**OXFW912,  
OXFW911<sup>plus</sup>,  
OXFW911-  
ATAPI**

To put one of these devices into force-flash mode, fit a jumper on JP1 on the development board assembly. As shown in [Figure A-2](#) on page 39, JP1 is located on the evaluation board in the assembly.

**Figure A-4 JP2 Location on the OXFW91x Development Board Assembly**



**Note:**

- 1 In [Figure A-4](#), OXFW91x refers any of the following devices:  
OXFW912, OXFW911<sup>plus</sup>, OXFW911-ATAPI

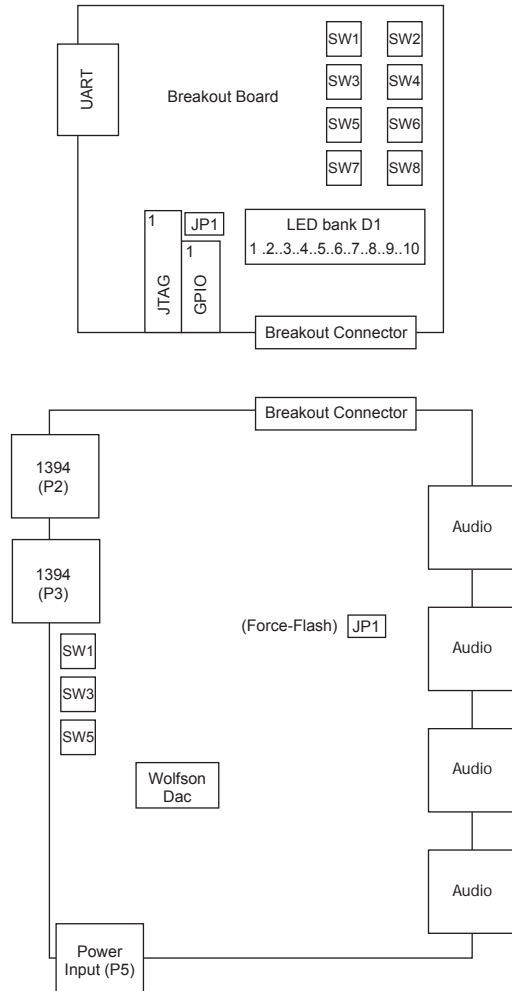


Always press SW1 to reset the device whenever you enable or disable force-flash mode.

# OXFW970

To put the OXFW970 into force-flash mode, fit a jumper on JP1 on the development board assembly. As shown in [Figure A-2](#) on page 39, JP1 is located on the evaluation board in the assembly.

**Figure A-5 JP2 Location on the OXFW970 Development Board Assembly**



Always press SW1 to reset the device whenever you enable or disable force-flash mode.

This page is intentionally blank

The Oxford Semiconductor uploader is a java program that will not run if the correct version of the Java Virtual Machine (JVM) is not installed on the host platform. The Java virtual machine starts automatically when you run the Oxsemi Uploader.

- ⚡ The JVM is located on the software development kit CD for the device. Before installing the JVM, read the licence agreement provided on the installation CD in text format.
- ⚡ The Java installation is not manufactured or supported by Oxford Semiconductor Ltd. For support information, refer to the relevant websites for this product.

### Upgrading the Uploader

Oxford Semiconductor recommends that you uninstall older versions of the uploader before loading a new version from scratch, especially if it is on the same directory path.

When the uploader is uninstalled, the configuration files are left behind in *installpath/data* because they might have been modified. They should be deleted manually if they are not required.

### Installing the Uploader

The uploader and JVM installation process differ according to which host platform is used.

- ⚡ Under Windows, the host platform must be restarted following the installation, otherwise the uploader cannot be used successfully.

## Installing on a Mac Host Platform

To install the uploader on a Macintosh platform, follow the instructions below:

1. Extract the uploader file to your preferred directory and extract the contents; we assume **~/Desktop**:

```
mkdir ~/Desktop/oxsemi-uploader
cd ~/Desktop/oxsemi-uploader
tar -xvzf /cdrom/mac/mac-uploader_n_nn_XXXXXXX.tar.gz
```

where *n\_nn* is the uploader version number and *XXXXXXX* is either Panther or Jaguar, depending on the Mac OS you are using.

## Installing on a Linux Host Platform

To install the uploader on a Linux platform, follow the instructions below:

1. Install the required version of the JVM onto your computer:

```
rpm -iv /mnt/cdrom/linux/Linux/j2re-1_4_2_0_2-linux-i586.rpm
```

2. Add the path of the JVM to the beginning of the PATH environment variable:

```
export PATH=/usr/java/j2re1.4.2_02/bin:$PATH
```

3. Extract the uploader from the CD to your required directory; we assume **/usr/local/uploader**:

```
mkdir /usr/local/uploader
cd /usr/local/uploader
tar -xvzf /cdrom/linux/Linux_Uploader_n_nn.tar.gz
```

where *n\_nn* is the uploader version number.

4. Put the uploader shared library into a standard Java directory; we assume **/usr/java/j2re1.4.2\_02/lib/i386**:

```
mv libwin_uploader_dll.so /usr/java/j2re1.4.2_02/lib/i386
```



The raw1394 module must reside in the kernel. `lsmod` lists the modules in the kernel—look for references to **ieee1394** and **ohci1394** in the module list. If they are missing, the 1394 card is not installed properly. `insmod (insmod raw1394)` adds the module to the kernel, but you need to be a root user to do it; and you need a startup script to add this module each time the machine is rebooted

5. You need read- and write-access to the raw1394 device. From root, set the owner of this device to *username*:

```
chown username /dev/raw1394
```

## Installing on a Windows Host Platform

To install the uploader on a Windows platform, follow the instructions below:

1. Install the required version of the JVM onto your computer:

Double-click **D:\Windows\j2re-1\_4\_2-windows-i586.exe**



This process installs a shortcut, **j2re**, on your desktop and in the Windows Start menu.



You must have administrator privileges to install the uploader.

2. Install the uploader on your computer:

Double-click **Win\_Uploader\_Install\_*n.nn*.exe**

where *n\_nn* is the uploader version number.

This process also installs a shortcut, **Oxsemi Uploader**, on your desktop and in the Windows Start menu.



You must restart your machine before you can use the uploader.

This page is intentionally blank

This chapter summarizes the common uploader errors.

<b>Table C-1 Basic &amp; GUI Errors</b>	
<b>Error</b>	<b>Reason</b>
Add flash button greyed out	Not enabled in the uploader initialization settings file; see Chapter 4 <a href="#">Initializing the Uploader</a> for file details.
Get image button grayed out	
No drive appears	In Windows, if the firmware is running but no drive appears, Windows might think the device is bad. Check the Device Manager to see if the device is shown (it should appear as 1394/USB disk under storage devices). If it is shown with an exclamation mark or red cross, Windows thinks it is not working (possibly due to trying to access it during programming). Remove it, then unplug the device and plug it in again. If the display reappears, the configuration might be wrong—use <b>Modify Device Configuration Settings</b> to change the Chip ID (Windows sometimes marks a particular device ID as bad if an error occurs, and ignores it). Also ensure that, for Windows 98SE operation, the device type is set to Generic Win98 device
Device on the bus not recognized	In Windows & Linux, if there are two 1394 cards in the host platform (whether a or b PHY), the uploader does not recognize devices on the bus
Two drives detected when connecting to a hard disk drive	<p>Possibly an incorrect uploader selection. The following options are available:</p> <ul style="list-style-type: none"> <li>a SBP2 Config Rom has a single unit directory—a single unit directory is used even if two drives are connected</li> <li>b SBP2 Config Rom has a unit directory for each logical unit—two unit directories are used, so if two drives are connected each one can have its own unit directory</li> <li>c Auto select SBP2 Config Rom format using firmware timestamp—the uploader determines what type of config ROM can be used, to correspond to the version of firmware being uploaded</li> </ul> <p>Two unit directories should be used in the following cases:</p> <ul style="list-style-type: none"> <li>a If the second disk ID comes up as unknown in the Windows text string for the device, e.g., in the device manager, use options b or c</li> <li>b Windows XP SP1 has a bug where on resuming from either standby or resume the second drive is not detected &amp; is unknown. Once again use option b or c</li> </ul> <p>Using option b or c with wrongly time-stamped firmware could result in detecting two disk drives where there is only one. To ensure that this is not the case, always use option a</p>
<b>Do not detect non-striped disks is selected</b> , LEDs are flashing but the disks cannot be striped. Eventually the process dies	The system is trying to detect two drives which are not striped. Uncheck <b>Do not detect non-striped disks</b>
General:	Do not use two host cards with the uploader; this can result in inconsistent programming & can also result in inconsistent detection when a 1394 hard drive is connected

<b>Table C-2 Uploader Errors</b>	
<b>Message</b>	<b>Meaning</b>
Unable to open 1394 driver!	The driver failed to open; check Windows\System32\Drivers\OXMFWLF.SYS exists. If so, try restarting Windows to reload the driver
No 1394 host card was found	Shut down the PC and restart it and then run up the Uploader again in case the host card was locked and unable to recover
Downloaded unit directory does not have the expected format /Downloaded root directory is not the expected length	This will appear when an invalid configuration ROM is being uploaded to the device. This could be due to the device being previously uploaded with an older version of the uploader and is then being looked at by a newer version
0 firewire devices and 0 USB devices were found	A device is connected & power is present on the bus, but the uploader cannot detect an Oxford Semiconductor 1394 device. Possibly a hardware problem
No device has been selected	Either nothing is plugged into the 1394 bus or there are multiple devices
No device detected	Nothing is plugged in to the bus
Failed to switch device to upload mode	The device must be uploaded in force-flash mode

<b>Table C-3 Uploader Initialization errors</b>	
<b>Message</b>	<b>Meaning</b>
Couldn't open ini file. Default settings will be used	Appears if the uploader initialization settings file has been deleted, is corrupt or is not found in C:\Program Files\Win_Java_uploader\data when the uploader is started
Couldn't open ini file to write current settings	The uploader initialization settings file has either been moved or it has been made write protected

<b>Table C-4 Configuration File Errors</b>	
<b>Message</b>	<b>Meaning</b>
Unable to download current configuration	Configuration information could not be read from the device: possibly caused by broken or incorrect firmware, or a device error. Try restarting the utility or reloading the firmware. Can also occur if you click <b>Modify Configuration</b> directly after an upload has completed. Wait for drive activity to complete before modifying the configuration
String is too large/ String was not found	The config_9xx.txt file has been read in and elements requiring quotes are not properly delimited
Configuration Information did not have a valid header quadlet	Appears if huge strings have been encountered for the configuration parameters; invalid configuration info in device.
Couldn't set the configuration address in the firmware	Appears if the wrong firmware file type is used when uploading or the uploader initialization settings are wrong
Fast buffer upload failed	Appears if the size of the binary image being downloaded is close to the limit. A file should allow for uploader configuration details etc. Reduce the binary image size or use a different type of flash memory with a smaller sector size

<b>Error</b>	<b>Meaning</b>
Firmware is too big to fit into the flash device	This occurs in the following instances: 1. If an attempt is made to upload a binary file which is bigger than 127K for the 922, 64K for the 911 and 128K for the 900. 2. This message also appears if the flash size is smaller than the file being uploaded (e.g., if the flash on the OXUF922 is 64 K instead of 128 K)
Could not erase flash - starting address was not on a sector boundary	Appears if there is incorrect data in the flashes.dat file.
Could not determine if erase has finished	Appears if there is something wrong in the upload fragment.
Buffer upload checksum does not match	This will only appear for the FW900 and is an upload problem.
File is not a suitable flash file	Appears if the flashes.dat file has been edited and incorrect data is found in it.
Could not open file to write flash image	This message will appear if the flashes.dat cannot be found on the disk.
Width of flash device has not been set	Within the flashes.dat file each flash type is listed individually. If a flash type is manually edited such that the flash width is removed for the selected flash type this message appears
Unable to load flash data file	The file <b>flashes.dat</b> in the executable directory could not be found or contained errors. Ensure this file is present and has not been modified in any way (Reinstall if necessary)

### USB Errors

On the OXUF922, firmware version 1.0 featured an older version of the USB descriptors than is present in more recent firmware releases. The uploader detects this, making it necessary to use force-flash to upload firmware.

<b>Error</b>	<b>Meaning</b>
The new firmware does not match the USB Descriptors currently in the device	A device has previously been programmed using an older version of the uploader. Put the device into force-flash mode & program it as blank
Descriptor Libraries of the firmware and uploader do not match	This message appears if the firmware descriptor library differs from the Uploader descriptor library

## MAC Uploader Debug Information

To get debug information on a MAC host, connect the device to the MAC and start a terminal program; then type the following:

```
java -jar uploadergui.jar
```

When the uploader starts, debug information appears in the terminal window which can be used to get an indication of any problems.

## Assumptions

To simplify the design of the software, Oxford Semiconductor has made assumptions about system configuration and operation in relation to OXFW900/911 devices. These are listed below:

- The device being programmed is the only Oxford Semiconductor device present on the bus.  
If two Oxford Semiconductor devices are found, there is currently no method for selecting which one to use
- The device is on system bus zero.  
The current windows driver only searches for devices on bus zero
- An external flash device must be attached to the OXFW900 on chip-select zero
- For OXFW900 devices the flash device used must have at least two sectors in the first 128 Kbytes of address space  
The OXFW900 can only address  $2^{17}$  (128 K) locations. It requires separate flash sectors for configuration information and firmware. This is irrelevant for OXFW911 devices, because the chip uses an internal flash, and for the OXUF922