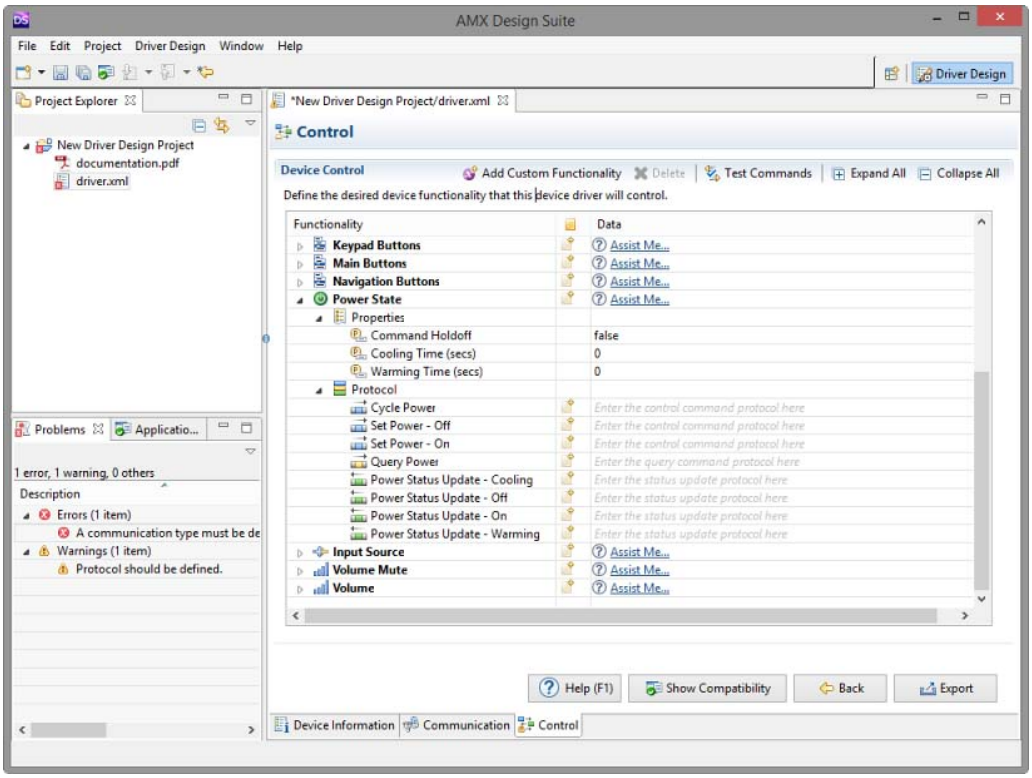
INSTRUCTION MANUAL

DRIVER DESIGN



AV FOR AN IT WORLD

COPYRIGHT NOTICE

LIABILITY NOTICE

AMX WARRANTY AND RETURN POLICY

The AMX Warranty and Return Policy and related documents can be viewed/downloaded at www.amx.com.

Table of Contents

AMX Design Suite / Driver Design

Overview

AMX Design Suite is an integrated development environment (IDE) based on the Eclipse platform that supports the *Driver Design* plug-in from AMX. Use Driver Design to create XDD files (aka Device Drivers) that can be used with NetLinx control systems.

AMX Design Suite also provides the ability to upload and download XDD's to/from AMX's online *InConcert Resource Center*.

Opening Driver Design

Driver Design is the default perspective for AMX Design Suite. When AMX Design Suite is installed, the Driver Design perspective is automatically opened when the setup is complete. The Driver Design perspective shows all the views and editors related to the Driver Design plug-in. FIG. 1 provides an example view of the Driver Design perspective:

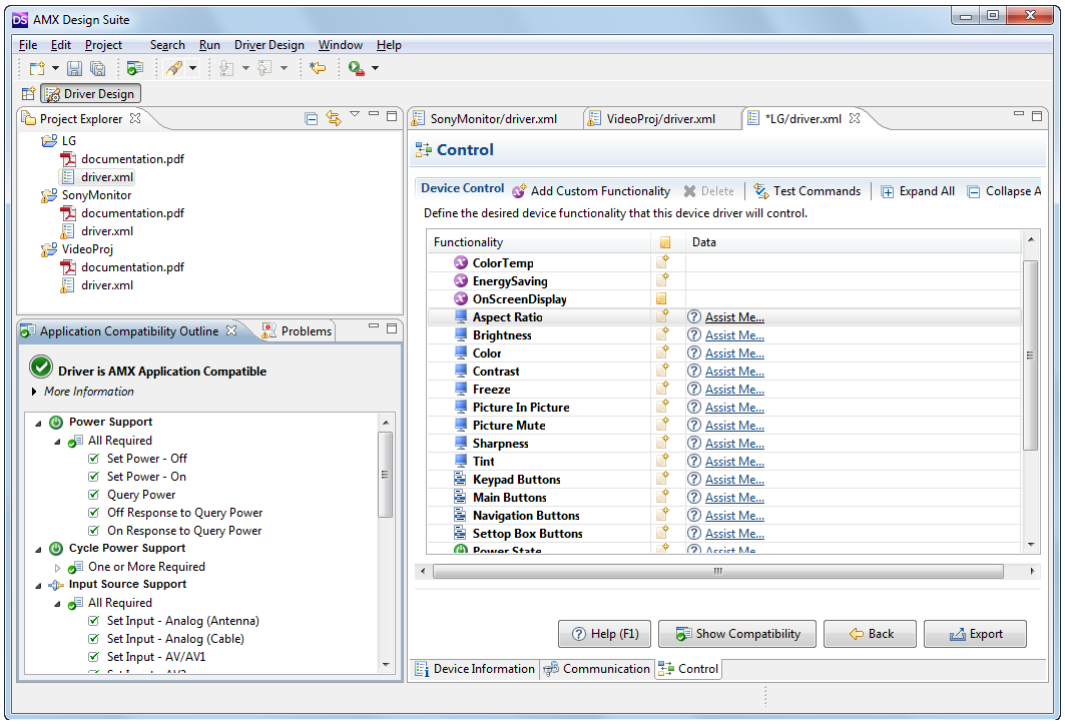


FIG. 1 Driver Design perspective (example view)

Some elements of the UI (such as the Problems view, and most menu and toolbar options) are part of the basic Eclipse UI, in which case there is Eclipse help available. Press **F1** to open a help topic specific to the active UI element.

The Driver Design UI is totally customizable - all windows and tabs can be dragged-and-dropped anywhere you like them. UI settings are saved, so your custom layout won't be lost when you close the program. The main elements of the Driver Design perspective are described in the following sections:

Project Explorer

Driver Design projects are displayed in the Project Explorer view (select Window > Show View > Project Explorer to open this view):

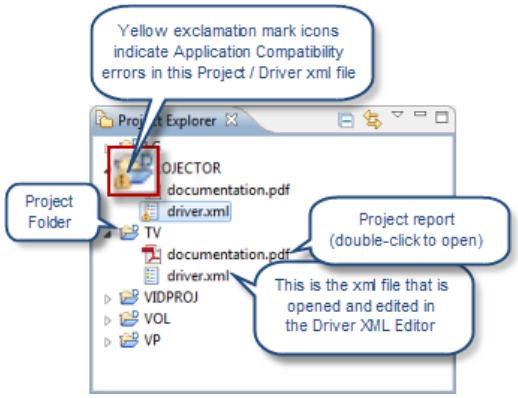


FIG. 2 Project Explorer

By default, the Project Explorer appears on the left side of the application and provides a hierarchical view of artifacts from the workspace such as Driver Design projects. The Project Explorer view shows all Driver Design projects created in the defined workspace path. If the workspace path is changed, the view will only show projects created in the new workspace path. Note that Projects are added to the Project Explorer as they are created, and they are not removed from this view unless they are deleted from disk. Note that Projects can be removed from the explorer without deleting from the disk (see *Deleting Projects* on page 21 for details).

- Projects cannot be nested inside of projects. Projects represent the top-level elements in the tree structure.
- Driver Design projects can be copied/pasted, closed/opened and deleted using the Project Folder context menu.
- Driver Design projects can be exported to the local disk or to the AMX InConcert Resource Center, and can be imported from the local disk or from InConcert.

Errors & Warnings Icons

- **Error icon**: A red icon with an "X" indicates that the project contains one or more Errors. Errors must be corrected before the project can be exported.
- **Warning icon**: A yellow icon with an "!" indicates that the project contains one or more Warnings. Warnings should be corrected, but will not prevent the project from being exported.

Project Folders and Files

Each Project folder contains a single device driver entity described by one or more resource files:

- **driver.xml** is the main resource for saving Driver Design information. Double-click on Driver.xml files to open the file in the Driver XML Editor.
- **documentation.pdf** is the project report for each Driver Design project. This file is generated and updated each time the Driver XML Editor is saved: It gets wrapped up in export to local disk (if zip option is selected, not when xpd is chosen) and when exporting to InConcert (zip only). See *Exporting Device Drivers* on page 43 for details.

Project Explorer Toolbar

The Project Explorer has it's own toolbar:

	Collapse All:	Click to collapse all Project folders
	Link With Editor:	Click to toggle whether the view selection is linked to the active editor. When this option is selected, changing the active editor will automatically update the selection to the resource being edited.
	View Menu:	Click to open the Project Explorer View Menu. This drop-down menu contains operations that apply to the entire contents of the view, but not to a specific item shown in the view. See Project Explorer View Menu for details.
	Minimize:	Click to minimize the Project Explorer view.
	Maximize:	Click to maximize the Project Explorer view.

Project Explorer View Menu

Click the *View Menu* button in the Project Explorer toolbar to open a menu of operations that apply to the entire contents of the view, but not to a specific item shown in the view (FIG. 3):

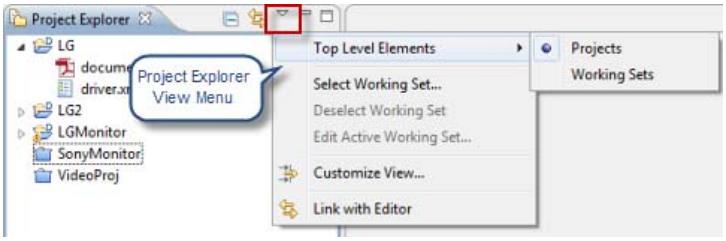


FIG. 3 Project Explorer - View Menu

Top Level Elements:	Choose whether to show working sets or projects as top level elements. Choosing working sets allows easy grouping of projects in large workspaces.
Select Working Set:	Opens the Select Working Set dialog to allow selecting a working set for the view.
Deselect Working Set:	De-selects the current working set.
Edit Active Set:	Opens the Edit Working Set dialog to allow changing the current working set.
Customize View:	This command allows customization of view filters and content modules. The previous will allow you to suppress the display of certain types of files while the later will allow entirely new types of content to be shown in the view. See Showing or hiding files in the Project Explorer view in Eclipse help for details.
Link With Editor:	Click to toggle whether the view selection is linked to the active editor. When this option is selected, changing the active editor will automatically update the selection to the resource being edited.

Project Explorer Context Menus

- Right-click in an empty area (not on a folder or file) to access the Project Explorer context menu.
- Right-click on a Project folder to access the Project Folder context menu.
- Right-click on a driver.xml file to access the Project File context menu.

Driver XML Editor

The Driver XML Editor is composed of multiple tabs at the bottom to break up the project work flow into manageable parts and a set of buttons for navigation and help. The Driver XML Editor displays the contents of the active Driver XML file in a tabbed interface (FIG. 4):

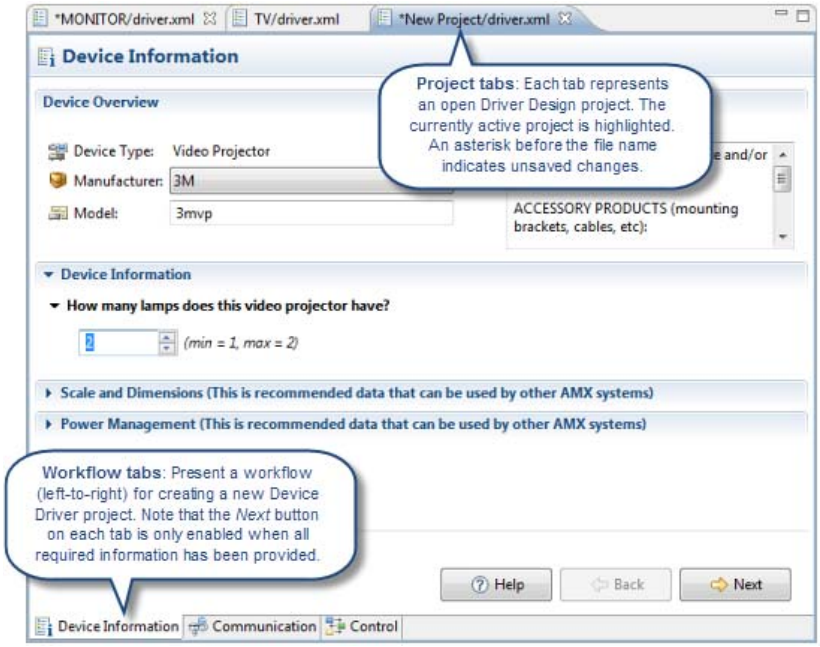


FIG. 4 Driver XML Editor

Each open Device Driver file is displayed in a tabbed window in the editor. The Workflow tabs along the bottom of the editor window present a left-to-right work flow for defining a device driver:

- **Device Information** - This is the initial view of the editor. Use the options in this tab to define basic information for the device. See *Entering Device Information* on page 22.
- **Communication** - The options in this tab allow you to define the communication settings supported by the device (Serial, TCP/IP or both). See *Defining Device Communications Settings* on page 24.
- **Control** - The options in this tab allow you to define the device functions that this driver will control. See *Defining Device Control* on page 34.

The **Next** button will take you to the next tab in the series (left-to-right) once you have provided all the required information for the current tab. If required data is missing a message will appear indicating the missing data to the left of the buttons. The **Back** button will take you to the previous tab in the series.

Application Compatibility Outline View

The Application Compatibility Outline view indicates the required implementation of Functional Protocol that is listed on the Control tab, in order for the driver to be compatible with AMX applications such as RPM (FIG. 5):

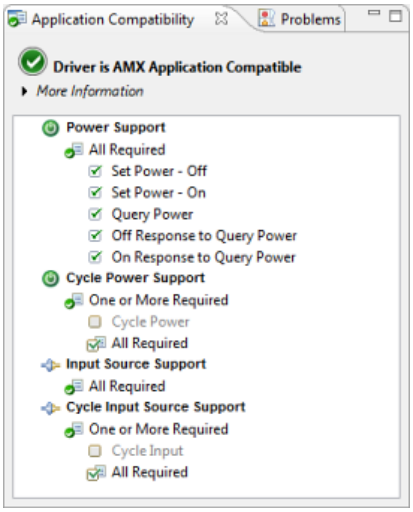
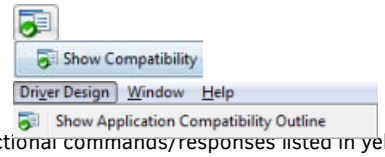


FIG. 5 Application Compatibility Outline view

- It checks for compatibility and indicates what protocol has been implemented and what is missing or optional.
- A check mark indicates implemented, orange text indicates missing and gray text indicates optional.
- The view is context sensitive to whichever editor has focus, and reloads content based on that editor.
- Logical requirements ("ALL" vs "one or more") of the API are indicated.

There are several ways to access this view:



In the toolbar, click the *Show Application Compatibility Outline* button.

Click the *Show Compatibility* button in the Control tab of the Driver XML Editor.

In the Menu bar, select *Driver Design* > *Show Application Compatibility Outline*.

Functional commands/responses listed in yellow indicate missing protocol on the Control tab and must be entered to meet compatibility requirements (FIG. 6).

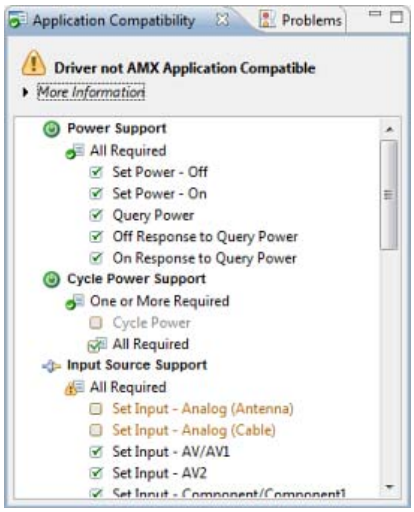


FIG. 6 Application Compatibility Outline view - Driver not compatible

- This view is updated when changes to the device driver file is saved.
- Double-clicking a yellow entry will take you to that command/response on the Control tab.

NOTE: *Only fully AMX Application Compatible Device Drivers are allowed to be exported to the AMX InConcert Resource Center. The application will alert you if the selected Project is incompatible, and therefore cannot be uploaded to InConcert. In this case, check the commands/responses listed in yellow in the Application Compatibility Outline. Correct these items by going to the Control tab and entering protocol for each one. Then save the Driver XML Editor to refresh the AMX Application Compatibility Outline to make sure the it is now compatible and export again.*

Problems View

Use the *Problems* view to display any Errors and/or Warnings detected in any open projects (this includes all projects that are open in the Project Explorer). The list is generated each time the Driver XML Editor is saved (FIG. 7):

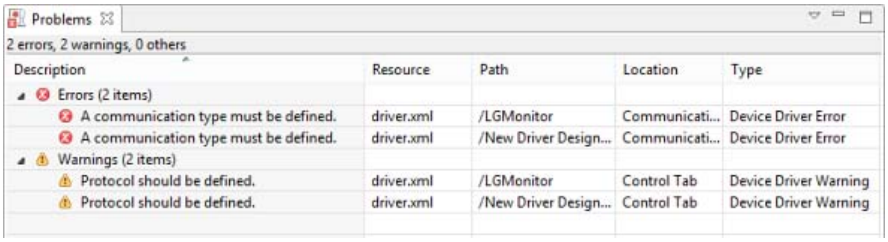


FIG. 7 Problems view

Errors

Errors must be corrected before the project is exported to InConcert or the File System. If a project has Errors, exporting is not allowed.

Warnings

While Warnings will not prevent the project from being exported, they should be corrected to ensure that the device driver functions properly.

NOTE: *These entries are different from those listed in the Application Compatibility Outline view, because these are general warnings relative to the internal logic of the xdd files (not necessarily related to compatibility with other AMX applications).*

The columns in this window provide the information required to correct each warning in the list:

- **Description**: This is a simple description of the warning.
- **Resource**: This is the name of the file in which each Warning was detected (for Driver Design xdd files, the resource name is always "driver.xml").
- **Path**: This is the path of the file in which each Warning was detected. The Path information is based on the Project Name - use this name to identify the device driver that needs to be corrected.
- **Location**: This is the tab of the Driver XML Editor that contains missing or invalid data.
- **Type**: This is the type of Warning indicated (for Driver Design files, the type is always "*Device Driver Warning*")

Errors and Warnings are removed from the list after they have been corrected and the Driver XML Editor is saved.

Getting Started

Overview

This section outlines the basic workflow of creating a new device driver:

1. Create a new Driver Design project: Select **File > New > Driver Design Project** to launch the *New Project Wizard*. The *New Project Wizard* prompts you to enter some basic device information to get started:
   a. In the *Driver Design Project* dialog, enter a unique name for the new project.
   b. In the *Device Information* dialog, select a Manufacturer, Device Type and enter a Device Model name.

   NOTE: *See the New Project Wizard* section on page 11 *for details.*

   c. Click **Finish** to close the *New Project Wizard*. The new file is opened in the Driver XML Editor.
2. Use the Workflow tabs of the Driver XML Editor to define *Device Information* (page 23), *Communications* (page 24), and *Control* settings (page 34) for the device.
3. Once the device driver has been finished, it can be exported to either the AMX InConcert Resource Center or a local directory. See *Exporting Device Drivers* on page 43 for details.
4. Once the device driver has been exported from Driver Design, the only thing left to do is to incorporate the Device Driver into the NetLinx Code that runs on the NetLinx Master that will control the device. This final step is not handled in Driver Design, since it requires editing NetLinx code. AMX provides the NetLinx Studio application (available to download and install from www.amx.com). Refer to *Loading the Device Driver in NetLinx Code* on page 53 for details.

New Project Wizard

Creating a New Device Driver

In order to create a new Device Driver, you must first create a Driver Design Project to put it in. Driver Design provides the N*ew Project Wizard* to step you through the process of creating a new project:

NOTE: *Each Driver Design project must represent a single device model (Driver Design does not support multiple model names in a single device driver).*

1. Launch the New Project Wizard. There are several ways to do this (FIG. 8):

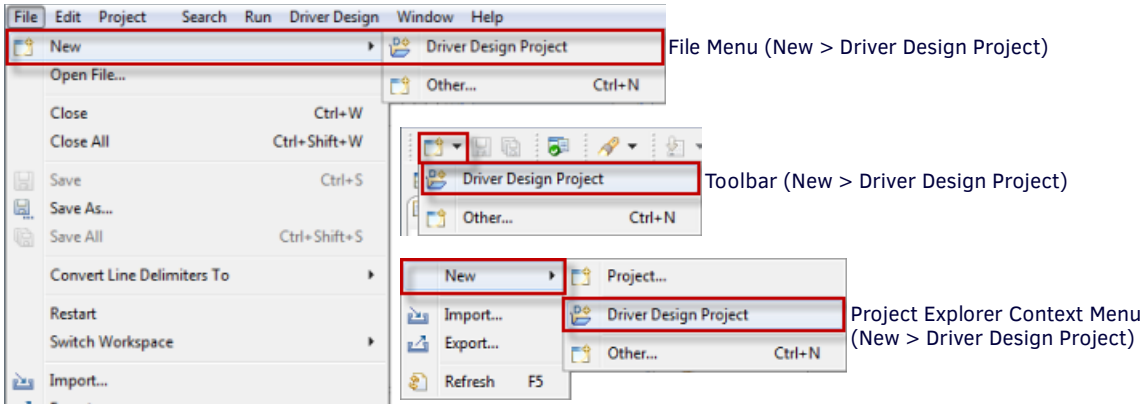


FIG. 8 File > New > Driver Design Project

- Select **File > New > Driver Design Project**
- Use the **New** button in the toolbar: Click the down arrow to access the *New* drop-down menu and select **Driver Design Project**.
- Right-click inside the Project Explorer and select **New > Driver Design Project** from the context menu.

The *New Project Wizard* consists of two dialogs: *Driver Design Project* and *Device Information*:

New Project Wizard - Driver Design Project

The first dialog in the *New Project Wizard (Driver Design Project)* prompts you to enter basic project information (FIG. 9):

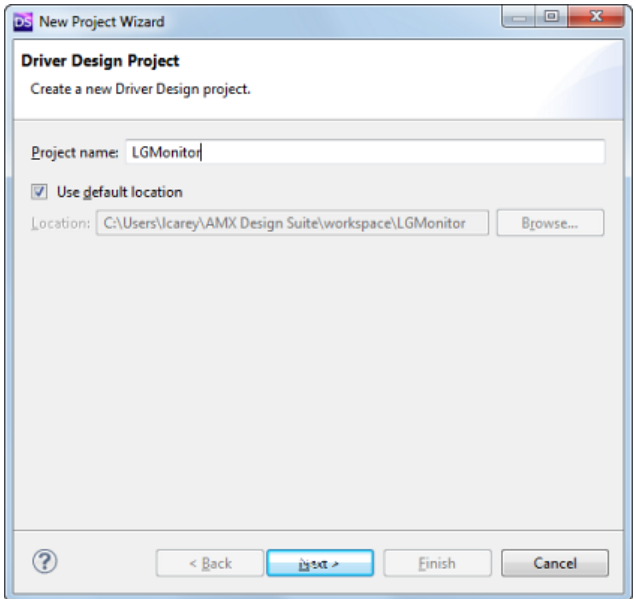


FIG. 9 New Project Wizard - Driver Design Project

- Enter a unique name for this project in the **Project Name** field.
- As indicated in the *Location* field, by default all Driver Design Projects are saved to the Workspace path, which by default points to your Windows User directory under *AMX Design Suite\workspace*.

  Each project is represented by a sub-directory with the Project Name. To save this project to a location other than the Workspace path location, de-select the **Use default location** option, and specify a target directory in the **Location** field.

Click **Next** to proceed to the second dialog in the Wizard: *Device Information*.

New Project Wizard - Device Information

The second dialog in the *New Project Wizard (Device Information)* prompts you specify the device's manufacturer, type and model (FIG. 10):

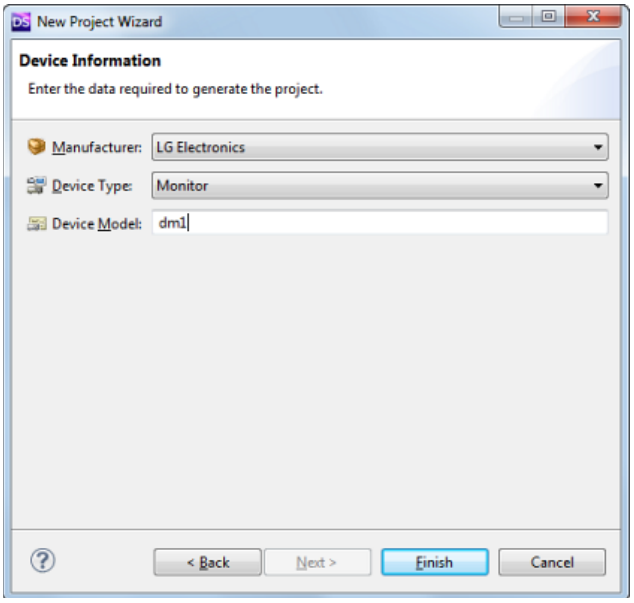
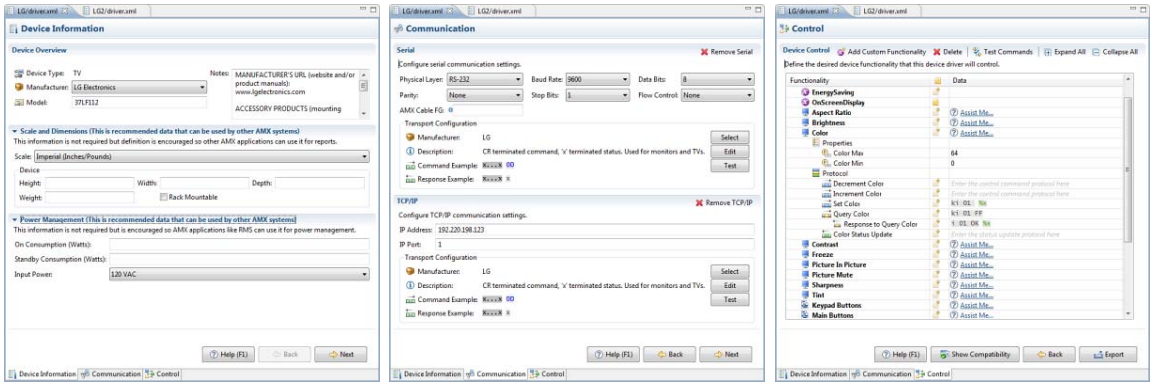


FIG. 10 New Project Wizard -Device Information

- **Manufacturer**: Select the device's manufacturer from the drop-down list.
- **Device Type**: Select the device's type from the drop-down list. Note that the default device functionality, as well as the default properties are based on this selection.
- **Device Model**: Enter the device model name in this field. Each Driver Design project must represent a single device model.

Click **Finish**. This will create the new project and open the driver.xml file in the Driver XML Editor. Note that the new project is indicated in the Project Explorer.

Use the workflow tabs in the Driver XML Editor to define your device driver (FIG. 11):



Device Information tab Communication tab Control tab

FIG. 11 Driver XML Editor

1. The initial view of the Driver XML Editor is the **Device Information** tab. This tab reflects the information entered in the New Project Wizard. Use the options in this tab to view and edit basic (static) device information. Refer to the *Entering Device Information* section on page 22 for details. Click **Next** to proceed to the *Communication* tab.
2. The options in the **Communication** tab allow you to specify how Device Driver and the control system will communicate with the device. Use the Test option in this tab to test the communication settings. Refer to the *Defining Device Communications Settings* section on page 24 for details. Click **Next** to proceed to the *Control* tab.
3. The options in the **Control** tab allow you to define the specific device functions that this driver will control. Refer to the *Defining Device Control* section on page 34 for details.
4. Click **Export** in this tab to export the Device Driver as an xdd or zip file to the local drive or to InConcert (see *Exporting Device Drivers* on page 43).

These tabs are arranged in order from left-to-right to suggest a logical workflow for creating a new device driver project. Note that each tab has a *Next* button that is only enabled when all required information has been filled in. If the information in the tab is not complete, a message is displayed along the bottom of the tab indicating what information is missing (FIG. 12):

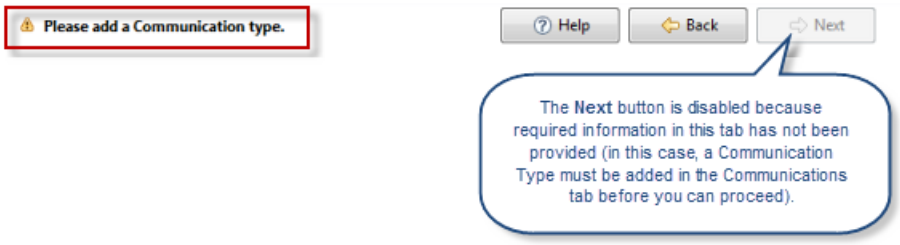


FIG. 12 Driver XML Editor - Back and Next buttons

Driver Design Projects and Project Files

NOTE: *Driver Design projects that were created and compiled in an earlier version of Driver Design (v1.1x) must be updated before they can be successfully edited and compiled with the current version of Driver Design (v1.2x). See the Updating v1.x Driver Design Projects for Use with Driver Design v1.2* section on page 16 *for details.*

Driver Design Projects

Each Device Driver Project is represented as a project folder in the Project Explorer view (FIG. 13):

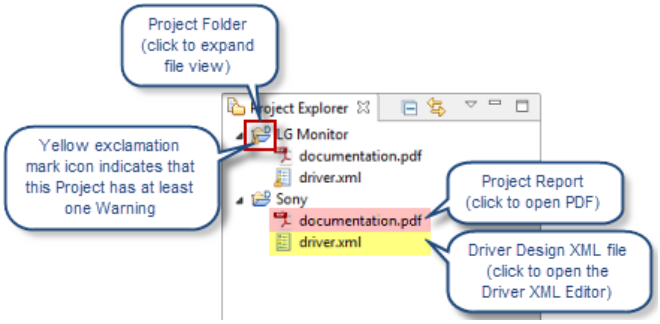


FIG. 13 Project Explorer view

While Driver Design Projects consist of several project resource files, most of them are intentionally hidden from the Driver Design user interface. In fact, the only project resource files that are visible in Driver Design are the Driver XML file ("**driver.xml**") and the Project Report file ("**documentation.pdf**").

Driver Design XML File ("driver.xml")

"*driver.xml*" is an XML-formatted data file that defines physical and operational characteristics of a device. It is this xml file that is edited via the multi-tabbed Driver XML Editor.

- driver.xml is consumed during launch of a Device Driver Engine Module instance. See *Loading the Device Driver in NetLinx Code* on page 53.
- Double-click on a driver.xml file in the Project Explorer to open the file for editing in the Driver XML Editor.

Project Report (documentation.pdf)

documentation.pdf is the project report for each Driver Design project. This PDF file is generated and added to the project folder only after the project has been exported. Double-click to open the PDF. (FIG. 14):

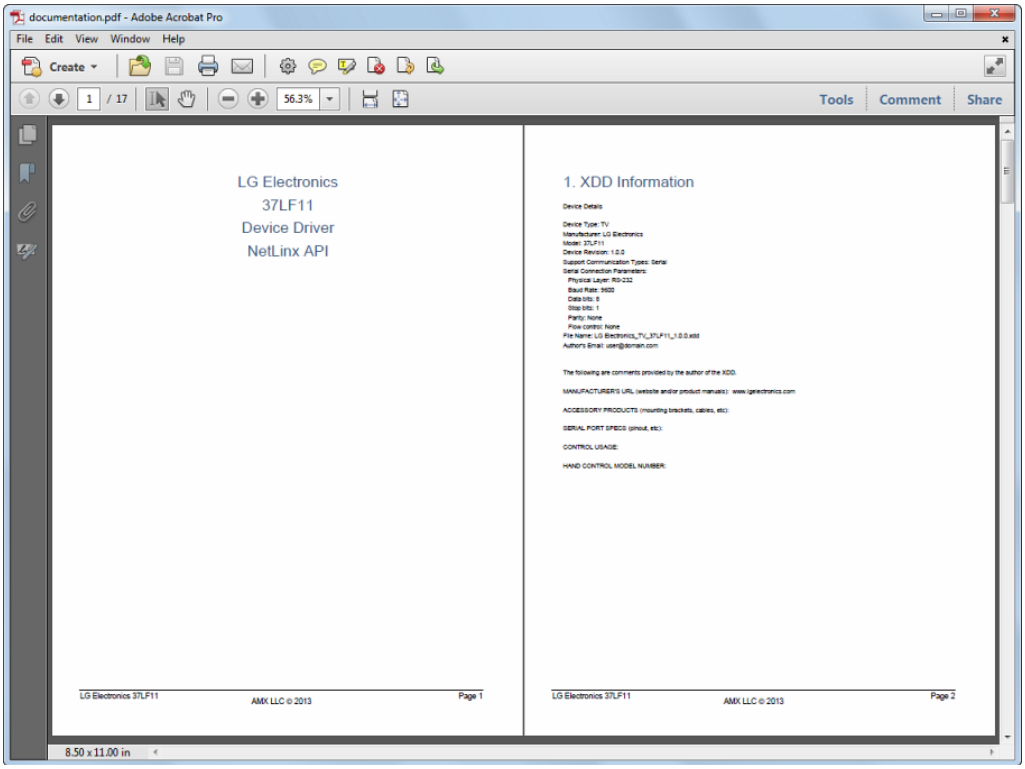


FIG. 14 Sample documentation.pdf File

This summary of the Device Driver contains the following sections:

- **XDD Information**: Provides a summary of Device Details as well as any comments added by the author.
- **Channel Information**: Provides a listing of the NetLinx Channels supported by this Device Driver (" * " denotes channels provided by Duet2 base and not protocol provided in this Device Driver).
- **Level Information**: Provides a listing of the NetLinx Levels supported by this Device Driver.
- **Command Information**: Provides a listing of the NetLinx Commands supported by this Device Driver.
- **Custom Command Information**: Provides a listing of the Custom NetLinx Commands supported by this Device Driver.
- **HAS Properties Information**: Provides a listing of the NetLinx HAS Properties supported by this Device Driver.
- **Properties Information**: Provides a listing of the NetLinx Properties supported by this Device Driver.
- **Sample NetLinx Code**: Provides a sample of NetLinx code that utilizes the information in this Device Driver.

Device Driver (.xdd)

When the Driver Design Project has been successfully built, it can be exported as a "Device Driver" (see Exporting Device Drivers), and ultimately, loaded onto the master (see). A Device Driver represents the output of a successfully built Driver Design project, in the form of a package (archival file) with a ".xdd" extension.

Opening and Closing Projects

Driver Design Projects are managed via the Project Explorer. As new Projects are created, they are added to the Project Explorer view (see *Creating a New Device Driver* on page 11). Click the arrow icon to show the file(s) contained in the Project. For Driver Design projects, this is typically the driver.xml file and all other hidden resources that make up the project (see FIG. 13 on page 14).

Opening Projects and Editing the Driver XML

Expand the project folder entry and double-click on the driver.xml file to open it in the Driver XML Editor.

- The Project Explorer indicates all Driver Design Projects that have been created, even if they are closed.
- By default, new Projects remain open until they are specifically closed.

Closing Projects

Use the *Close Project* and *Close Unrelated Projects* options in the Project Folder Context Menu to close projects.

Closed projects will not be included in project tasks such as building, compiling, searching, etc... Additionally, closed projects do not allow opening resources via the CTRL+SHIFT+R shortcut key, or being expanded to see the resources.

- With a project folder selected in the Project Explorer, right-click and select **Close Project** to close the selected project.
- With a project folder selected in the Project Explorer, right-click and select **Close Unrelated Projects** to close all projects other than the selected project.
- Closed Projects are represented with a closed folder icon (FIG. 15):

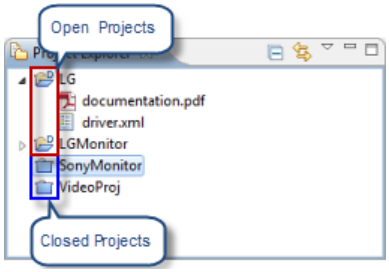


FIG. 15 Project Explorer - open and closed projects

The only way to remove a Project from the list in the Project Explorer is to delete the Project. See *Deleting Projects* on page 21 for details.

Opening Closed Projects

To open a closed Project, right-click on a closed Project folder in the Project Explorer and select **Open Project** from the Project Folder Context Menu. You can also open a file via the F**ile > Open File** menu option.

NOTE: *The Close Project and Open Project options in the Project Explorer context menu open and close projects in the Project Explorer only (not in the Editor window).*

Updating v1.x Driver Design Projects for Use with Driver Design v1.2

Driver Design project files that were created and compiled in an earlier (v1.1x) version of Driver Design must be updated before they can be successfully edited and compiled with the current (v1.2x) version of Driver Design. Follow the instructions below to update v1.1x projects to v1.2x:

1. In Driver Design, click **File > Import** to import the v1.1x Device Driver (XDD) file via the *Import* dialog (FIG. 16):

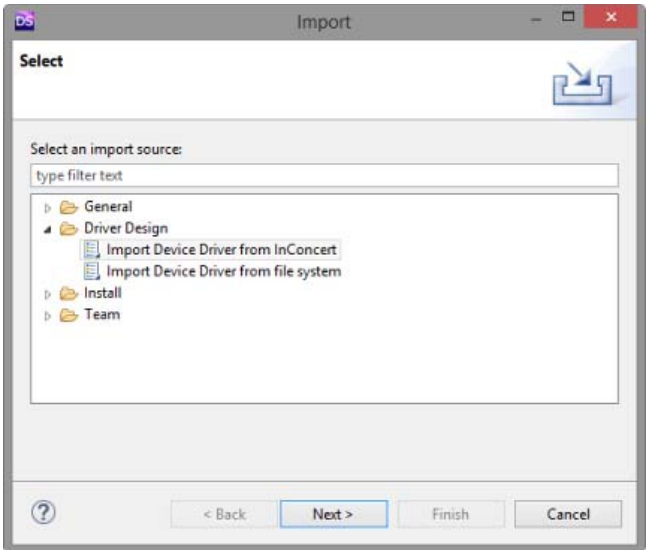


FIG. 16 Import dialog

2. In the *Select an import source* window, click on **Driver Design**, then select either *Import Device Driver From InConcert* or *Import Device Driver From file system*.
   - If *Import Device Driver from InConcert* was selected, click **Next** to open the *Import Device Driver from InConcert* dialog. Use the options in this dialog to download and import a device driver from the online AMX InConcert Resource Center. See the *Importing a Device Driver from InConcert* section on page 18 for details.
   - If *Import Device Driver from file system* was selected, click **Next** to open the *Import Device Driver from File System* dialog. Use the options in this dialog to download and import a device driver from the online AMX InConcert Resource Center. See the *Importing a Device Driver from a Local File System* section on page 19 for details.
3. When the imported XDD is opened in the Driver Design workspace, increment the *Bundle-Version* number in the *Device Overview* tab (FIG. 17):

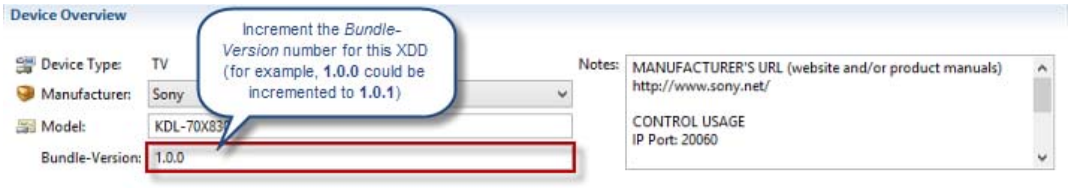


FIG. 17 Driver Design - Device Overview tab

4. In the *Notes* window, edit the REVISION HISTORY text to indicate this update to the XDD:
   "REVISION HISTORY (When you modify a driver, increment the version and note the changes here)
   1.0.0: Initial Revision" (indicate the new bundle version number in place of "1.0.0".
   This change identifies this XDD file as a different version of the file.
5. Click **Next** to open the *Communication* tab, make changes as desired.
6. Click **Next** to open the *Control* tab, make changes as desired.
7. Click **Export** to export the XDD file as a v1.2x-compatible Device Driver.

NOTE: *Once exported, the XDD is automatically complied using the current (v1.2x) compiler. Note that once the file has been compiled with the current (v1.2x) compiler, it is no longer compatible with previous versions of Driver Design.*

Copying and Pasting Projects

A convenient way to create a new Driver Design Project is to copy and paste an existing project, and make modifications as necessary:

1. Right-click on a Project folder in the Project Explorer.
2. Select **Copy** from the Project Folder context menu (FIG. 18):

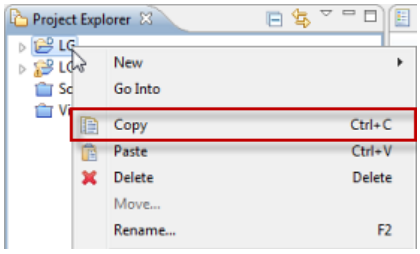


FIG. 18 Project Folder Context Menu

3. Right-click in the Project Explorer and select **Paste** from the context menu. This action invokes the *Copy Project* dialog (FIG. 19):

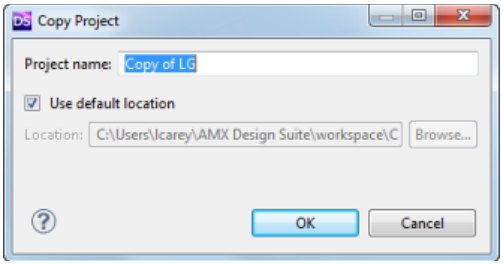


FIG. 19 Copy Project dialog

- Note that by default, the new Project name is the same as the selected Project, with "*Copy of*" prepended to the Project name.
- To save the copied project to a location other than the default location indicated in the Location field, de-select the *Use default location* option, and specify a target directory:

4. Click **OK** to paste a copy of the selected Project to the Project Explorer (FIG. 20):

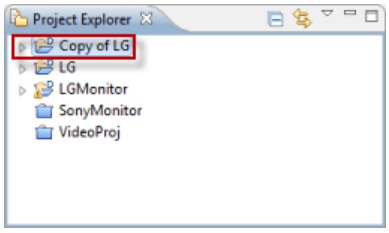


FIG. 20 Project Explorer indicating a copied Project

5. (*optional*) - To rename the pasted Project, right-click on it and select **Rename** from the context menu. This action invokes the *Rename Resource* dialog (FIG. 21). Enter a unique Project Name in the *New Name* field (the program will alert you if there is already a file or folder using the new Project name), and press **OK** to save changes and close this dialog. The new name is reflected in the Project Explorer.

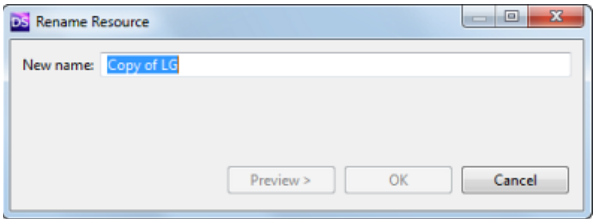


FIG. 21 Rename Resource dialog

Renaming Projects

1. With a project folder selected in the Project Explorer, right-click and select **Rename** from the context menu. This opens the *Rename Resource* dialog.
2. Click **OK** to rename the selected project immediately, or click *Preview* to preview and verify the changes before they are made (FIG. 22):

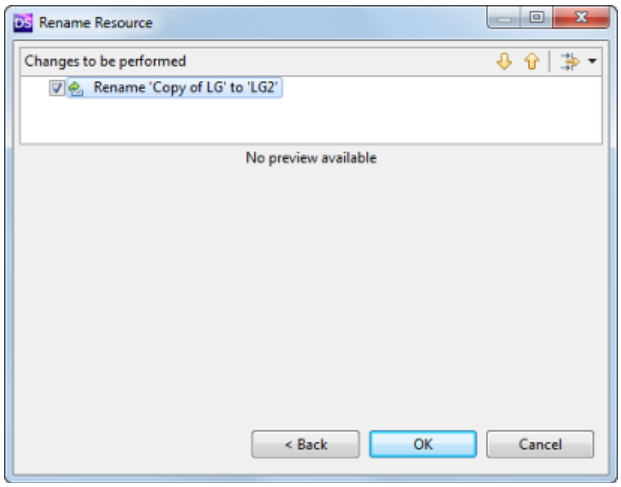


FIG. 22 Rename Resource (Preview) dialog

3. The new Project name is indicated in the Project Explorer (FIG. 23):

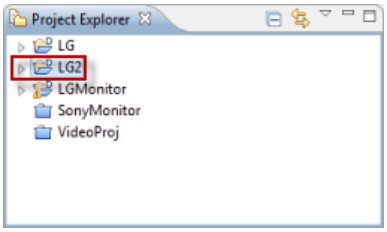


FIG. 23 Project Explorer indicating new Project Name

Importing Device Drivers Into Driver Design

Use the options in the *Import Wizard* to import a device driver into a new Driver Design Project. The Import Wizard contains options for importing different types of files and project resources into an existing project. The majority of these (including General, Install, Run/Debug and Team options) are offered as part of the Eclipse platform, and are fully documented in Eclipse help. However, the Import Device Driver from InConcert and Import Device Driver from file system options are specific to the Driver Design perspective. These options allow you to import Device Drivers (*.xdd) into Driver Design as new Projects.

Importing a Device Driver from InConcert

1. To open the Import Wizard, select **File > Import** (or select **Import** from the Project Folder Context Menu. This opens the first dialog in the Import Wizard: *Select*. The options in this dialog allow you to specify the source of the file to import.
2. In the *Select an import source* window, select **Driver Design > Import Device Driver From InConcert** (FIG. 24):

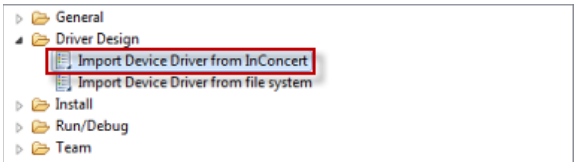


FIG. 24 Import Wizard: Select dialog - Import Device Driver From InConcert

3. Click **Next** to proceed to the *Search InConcert For Device Driver* dialog. Use the options in this dialog to find and download the desired Device Driver on the AMX InConcert Resource Center:
   a. Use the drop-down menus at the top of the dialog to select a *Manufacturer* and *Device Type*, and enter the *Device Model* and the email address of the author in the text fields provided. Enter as much information as possible to ensure concise search results.
   b. Click **Search** to search for Device Drivers based on the search criteria entered. The search results are listed in the main window of this dialog (click *Clear* to clear the results of the previous search). Hover over the column headers to view a brief description of each column:

Search InConcert For Device Driver dialog - Results Icons	
	A checkmark in this column indicates that this device driver is AMX Certified. AMX Certified drivers have been tested and verified by AMX.
	A checkmark in this column indicates that this device driver meets compatibility requirements for AMX applications (such as RMS).
	This column indicates the number of APIs implemented in this driver.
	An icon in this column indicates that this driver specifies serial transport.
	An icon in this column indicates that this driver specifies TCP/IP transport.

c. Select a Device Driver from the results list.

4. Click **Next** to proceed to the *Project For Device Driver* dialog. The options in this dialog allow you to create a Project for the Device Driver to be imported into.
   a. Enter a unique name for the new Project in the *Project Name* text field.
   b. The default target directory for Projects is indicated in the *Location* field. To change the target directory for this Project, de-select the *Use default location* option and click **Browse** to locate and select the desired target directory.
5. Click **Finish**. The new Project is indicated in the Project Explorer and in the Driver Design Editor.

Importing a Device Driver from a Local File System

1. Select **File > Import** (or select **Import** from the Project Folder Context Menu. This opens the first dialog in the Import Wizard: *Select*.
2. In the *Select an import source* window, select **Driver Design > Import Device Driver From File System** (FIG. 25):

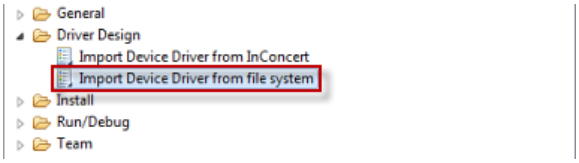


FIG. 25 Import Wizard: Select dialog - Import Device Driver From file system

3. Click **Next** to proceed to the *Search Device Driver from your computer* dialog.
4. Click **Browse** to locate and select the desired Device Driver (.xdd) in the *Choose a Device Driver file To import* dialog.
5. Click **Next** to proceed to the *Project For Device Driver* dialog. The options in this dialog allow you to create a Project for the Device Driver to be imported into.
   a. Enter a unique name for the new Project in the *Project Name* text field.
   b. The default target directory for Projects is indicated in the *Location* field. To change the target directory for this Project, de-select the Use default location option and click **Browse** to locate and select the desired target directory.
6. Click **Finish**. The new Project is indicated in the Project Explorer and in the Driver Design Editor.

Exporting Device Drivers

A *Device Driver* represents the output of an exported Driver Design project, in the form of a package (archival file) with an "**.xdd**" extension. Device Drivers can be exported to either the AMX InConcert Resource Center or a local directory. Refer to the *Exporting Device Drivers* section on page 43 for details.

Saving Projects

Click the Save toolbar button, press Ctrl+S, or select **File > Save** to save the currently active Driver XML Editor (FIG. 26).

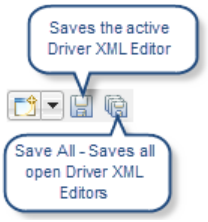


FIG. 26 Save and Save All toolbar buttons

These options are only enabled if there are unsaved changes in the active Driver XML Editor.

By default, Driver Design builds all open Projects automatically, every time a save operation is performed. This featured can be turned off via the **Project > Build Automatically** option. See *Building Projects* on page 20 for details.

Note that while changes are not saved automatically, the application will prompt you to save any unsaved changes before closing a Driver XML or closing Driver Design.

Saving the Active Driver XML File to a New Project

1. Select **File > Save As** to open the *Save As* dialog.
2. Enter a name for the new Driver Design Project in the **Project Name** text field.
3. The default location for saved Driver XML files is indicated in the **Location** field. To change the target directory, de-select the Use default location option and click Browse to locate and select a different location in the *Browse For Folder* dialog.
4. Click **Finish** to close the dialog

The new Project containing the saved Driver XML file is indicated in the Project Explorer.

Saving All Open Driver XML Files

Select **File > Save All** to save all open Driver XML files.

Building Projects

By default, Driver Design builds all open Projects automatically, every time a file save operation is performed. However, there are options (available via the Project menu) for manually building Projects either individually or as a group:

- **Build Automatically**: By default, Driver Design builds all open Projects automatically, every time a file save operation is performed. Note that this option is selected by default.
- **Build All**: Click this option to build all open Projects.
- **Build Project**: Click this option to build only the active Project. This option is only enabled if Build Automatically is de-selected.

  NOTE: *Changing the build setting requires a restart.*

- **Clean**: Click this option to open the *Clean* dialog. Use the options in this dialog to discard all build problems and built states, and rebuild Projects from scratch. See *Cleaning Projects* (below) for details.

Cleaning Projects

A *Clean* operation discards all build problems and build states. Once a Clean has been performed, all projects will rebuilt from scratch. Select **Project > Clean** to open the *Clean* dialog (FIG. 27):

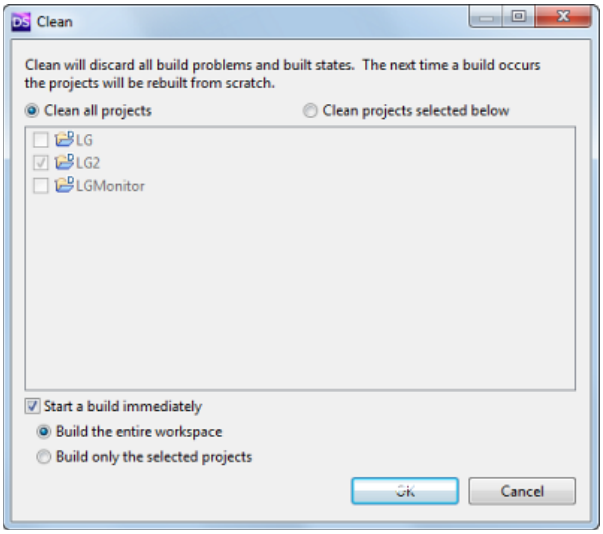


FIG. 27 Clean dialog

The options in this dialog allow you to clean all open projects, or only selected projects:

- **Clean All Projects** - This is the default selection.
- **Clean Projects Selected Below** - Select this option, then select the project(s) that you want to clean.

Click **OK** to close the *Clean* dialog, clean and rebuild the selected projects.

Deleting Projects

1. With a Driver Design project folder selected in the Project Explorer, right-click and select **Delete** from the context menu. This opens the *Delete Resources* dialog (FIG. 28):

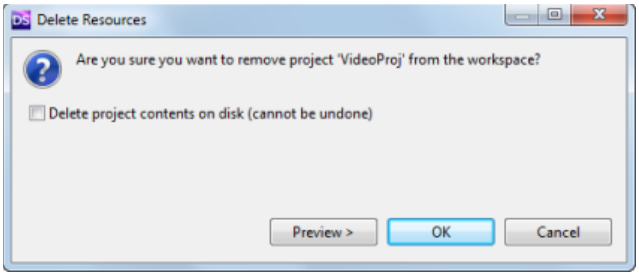


FIG. 28 Delete Resources dialog

By default, this action removes the selected project from the Workspace, but the project contents are not deleted from the disc. Select the *Delete project contents on disk (cannot be undone)* option to permanently delete all of the files included in the selected project. As indicated in this dialog, this action cannot be undone (and is de-selected by default). If a project is deleted without this option selected, then the application will not allow a new project with the same name to be created (FIG. 29):

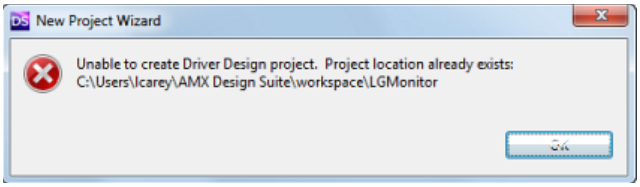


FIG. 29 Error dialog - Project Name already exists

2. Click **OK** to delete the selected project immediately, or click **Preview** to preview and verify the changes before they are made (FIG. 30):

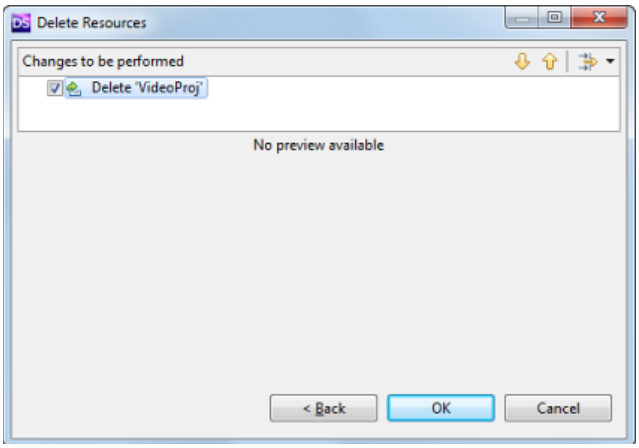


FIG. 30 Delete Resources Preview dialog

Entering Device Information

Overview

The options in the *Device Information* tab of the Driver XML Editor provide the starting point for new Driver Design projects. The options in this tab allow you to specify the make and model of the target device for your project, as well as physical properties such as packaging dimensions and power consumption (FIG. 31):

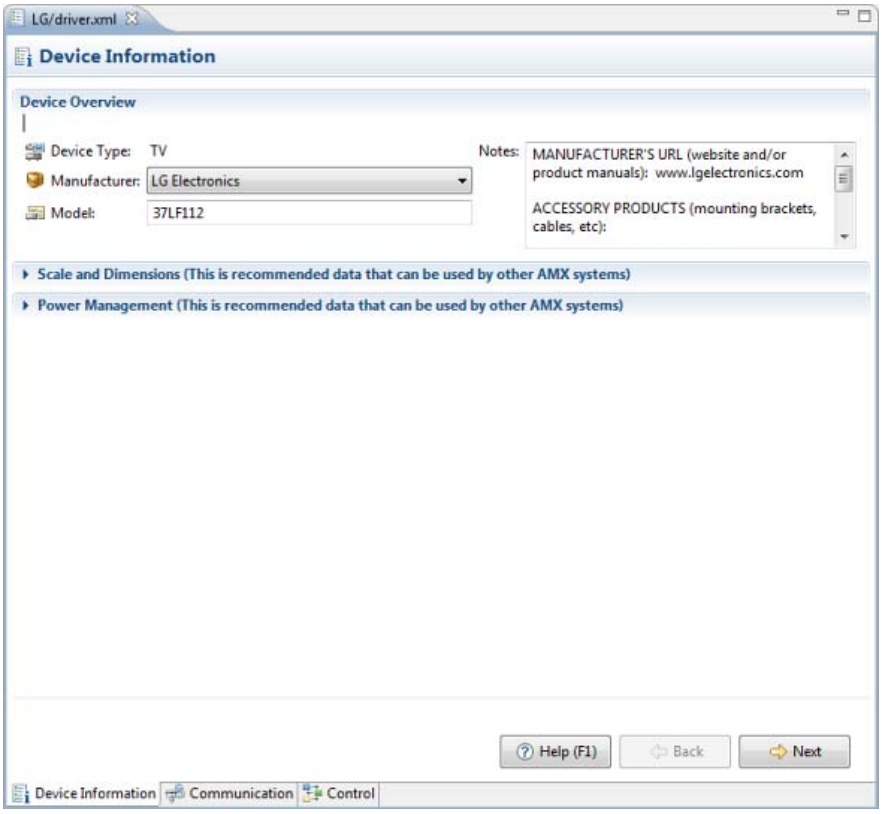


FIG. 31 Device Information tab of the Driver XML Editor

Device Overview

The information in this section reflects the selections made in the *New Project Wizard: Device Information* dialog (see FIG. 10 on page 12). You can change basic device information in this section, as well as provide notes for your project:

- **Device Type**: This read-only field indicates the Device Type selected in the New Project Wizard.
- **Manufacturer**: Select the device Manufacturer's name from the drop-down list. This list is provided by the InConcert database. Note: If a connection to the InConcert database has never been made, a default list is shown. If you have connected to the database, the last known Manufacturer list is displayed.
- **Model**: Type the Model Name of this device (32 characters max).
- **Notes**: Use this text field to enter additional information that would be useful to save with this device. The following headings are provided to categorize this information:

  MANUFACTURER'S URL (website and/or product manuals)

  ACCESSORY PRODUCTS (mounting brackets, cables, etc)

  SERIAL PORT SPECS (pinout, etc)

  CONTROL USAGE

  HAND CONTROL MODEL NUMBER

  To enter additional notes outside the context of these headings, click below the HAND CONTROL MODEL NUMBER heading and type directly into the multi-line text field (1500 characters max).

Device Information

Some device types support additional *Device Information* options. For example, if *Video Projector* is selected as the Device Type, the following Device Information option (number of lamps) is presented (FIG. 32):

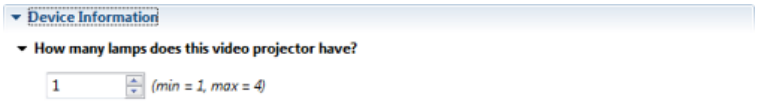


FIG. 32 Device Information tab - Device Information option

The specific options presented in this section will depend n the device type selected - use the fields in this section to change the default values as desired.

Scale and Dimensions and Power Management

The *Scale and Dimensions* and *Power Management* options allow you to specify optional physical properties for the target device. Click the arrow icon to expand these sections:

NOTE: *Although not required for a Driver Design project, providing information in these sections is strongly encouraged, as other AMX applications (such as RPM) can utilize this data.*

Scale and Dimensions

- **Scale**: Select either Imperial (Inches/Pounds) or Metric (Centimeters/Kilograms) from the drop-down menu.
- **Device**: Enter Height, Width and Depth dimensions as well as Weight for the device itself. If the device mounts into a standard 19" equipment rack, select the Rack Mountable option.
- **Packaging**: Enter Height, Width and Depth dimensions as well as Weight for the device in it's packaging.

Power Management

- **On Consumption (Watts)**: Enter the devices power consumption in it's normal On state.
- **Standby Consumption (Watts)**: Enter the devices power consumption in it's Standby state.
- **Input Power**: Select the appropriate input power required for this device from the drop-down menu (120 VAC, 240 VAC, 120/240 VAC or DC). The default setting is 120 VAC.

  DC Voltage: If DC is selected (for Input Power), then the DC Voltage field is provided - enter a decimal value for the DC Voltage required by this device.

  Refer to the device manufacturer's documentation for accurate power specifications.

The **Next** button will take you to the next tab in the Driver XML Editor (left-to-right) once you have provided all the required information for the current tab. Since this is the first tab, the Back button is disabled.

Defining Device Communications Settings

Overview

Device communication settings are managed via options in the *Communication* tab of the Driver XML Editor (FIG. 33):

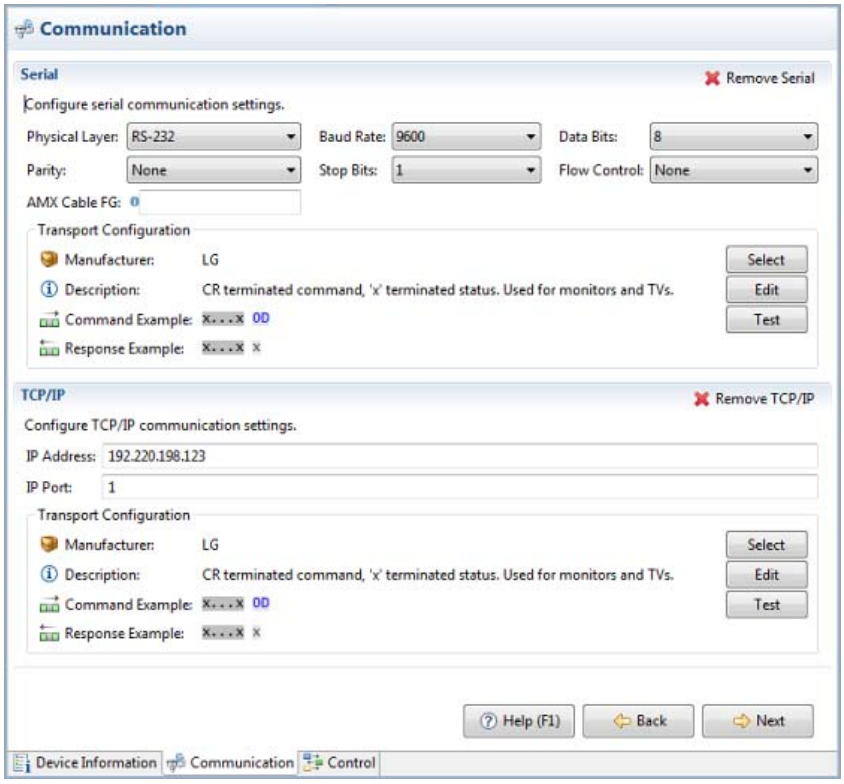


FIG. 33 Communication tab of the Driver XML Editor

Driver Design provides Transport Configurations (pre-configured device communication settings based on device manufacturer and type) that can be used with many devices. In most cases, communication settings can be set by simply selecting a Transport Configuration. However, if you have a device that is not represented in the list provided, you can select a configuration that is similar to your transport and make the required modifications (see *Editing the Current Transport Configuration* on page 29 for details). You can also create a new Custom configuration if necessary (see *Creating a Custom Transport Configuration* on page 27).

Transport Configurations

Transport configurations represent pre-configured device communication protocol settings. More specifically, transport configurations contain pre-configured formatting of the transmit and receive messages (such as header, message, and footer formats, checksum etc). Transport Configurations are selected based on the device's communication requirements.

In many cases, communication settings can be set by simply selecting a transport configuration that suits your device. However, if you have a device that is not represented in the list provided, you can select a configuration that is similar to your transport and make the required modifications (see Editing the Current Transport Configuration). You can also create a new Custom configuration if necessary (see Creating a Custom Transport Configuration).

To select a transport configuration, (in the Communication tab) select **Add Serial Control** to select a serial transport configuration, or select **Add TCP/IP Control** for network devices. Consult your device manufacturer's product documentation to decide which type of transport is required.

Defining a Serial Transport Configuration

To define a transport configuration, (in the Communication tab) select *Add Serial Control* to select a serial transport configuration, or select *Add TCP/IP Control* for network devices. Consult your device manufacturer's product documentation to decide which type of transport is required.

1. In the *Communication* tab, click **Add Serial Control** (FIG. 34):

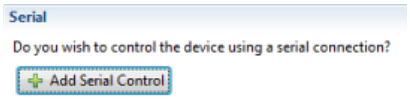


FIG. 34 Communication tab - Add Serial Control

2. This opens the *Serial Transport Configuration* dialog. Select a built-in transport from the list of available serial transport configurations, based on the Device Type and Manufacturer selected in the *Device Information* tab. Note that the Transport Description column provides details on each transport configuration (FIG. 35).

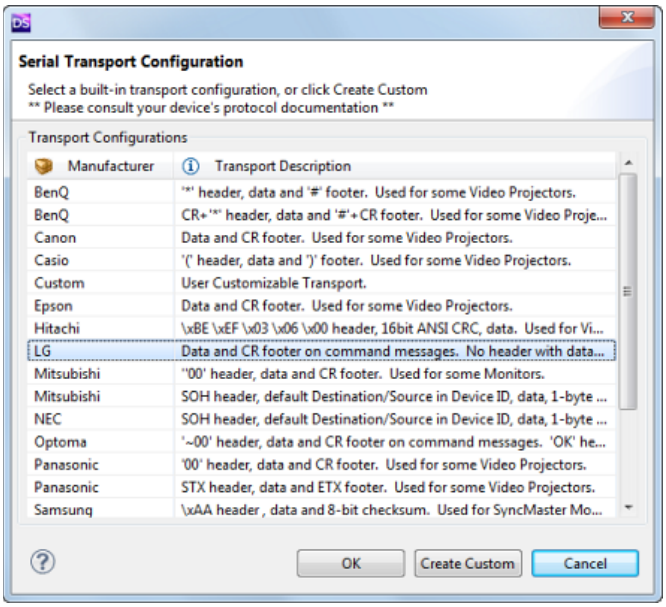


FIG. 35 Serial Transport Configuration dialog

3. Click **OK** to close the *Serial Transport Configuration* dialog.
4. Once a transport configuration has been selected, a set of communication fields are provided (as well as *Select/Edit/Test* command buttons, and a *Remove Serial* button). Use these fields to modify communication settings as required by the device (FIG. 36):

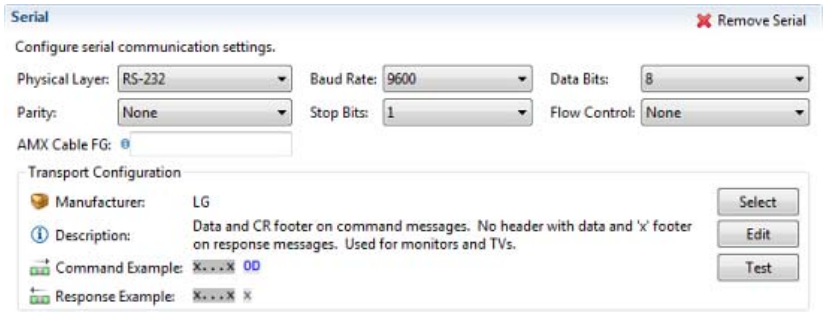


FIG. 36 Serial Configuration options

Serial Communication Settings	
Physical Layer:	Use the drop-down menu to specify the Physical Layer setting required by this device.
Baud Rate:	Use the drop-down menu to specify the Baud Rate setting required by this device.
Data Bits:	Use the drop-down menu to specify the Data Bits setting required by this device.
Parity:	Use the drop-down menu to specify the Parity setting required by this device.
Stop Bits:	Use the drop-down menu to specify the Stop Bits setting required by this device.
Flow Control:	Use the drop-down menu to specify the Flow Control setting required by this device.
AMX Cable FG#:	Enter the AMX FG number associated with the serial communication cable used with this device (optional).

Serial Communication Settings (Cont.)	
Transport Configuration: The (read-only) fields in the Transport Configuration section provide a summary of the currently selected transport configuration. Note that these options are the same for Serial or TCP/IP transports, and are available only after a transport configuration has been selected.	
• Manufacturer:	This field displays the configuration's manufacturer, based on the selected predefined configuration. *Note: An asterisk ("*") prefix denotes that the configuration has been edited.*
• Description:	This field provides a basic description of the configuration, based on the selected predefined configuration. The Description can be edited via the *Serial Transport Configuration - Command Format* dialog (click the **Edit** command button to access this dialog).
• Command Example:	This field provides an example of the command messages sent to the device, according to the current message format defined in the *Serial Transport Configuration - Command Format* dialog (click the **Edit** command button to access this dialog). Note that the example is colorized to indicate hex, ASCII, and regex values.
• Response Example:	This field provides an example of the response messages sent from the device, according to the current transport configuration.
The *Select*, *Edit* and *Test* command buttons in this section apply to the Serial Transport Configuration settings:	
• Click **Select** to open the *Serial Transport Configuration* dialog, to select a different serial transport configuration.	
• Click **Edit** to open the first dialog of the *New Transport Configuration Wizard* - the *Serial Transport Configuration - Command Format* dialog. Use the options in this dialog to define the Command Format. The Command Format specifies the format of command messages that are sent to the device (see *Defining the Command Format* on page 29 for details). The second dialog in the Wizard is the *Serial Transport Configuration - Receive Format* dialog. The Receive Format specifies the format of messages that are received from the device (see *Defining the Response Format* on page 32 for details).	
• Click **Test** to open the *Test Serial Communication* dialog. This feature allows you to test the current transport configuration. See *Testing a Transport Configuration* on page 32 for details.	

Defining a TCP/IP Transport Configuration

To add a *TCP/IP* transport configuration to your project:

1. In the *Communication* tab, click **Add TCP/IP Control** (FIG. 37):

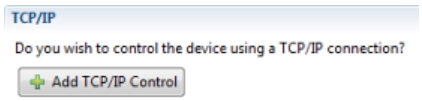


FIG. 37 Communication tab - Add TCP/IP Control

2. This opens the *TCP/IP Transport Configuration* dialog. Select a built-in transport from the list of available TCP/IP transport configurations, based on the Device Type and Manufacturer selected in the *Device Information* tab. Note that the Transport Description column provides details on each transport configuration (FIG. 38):

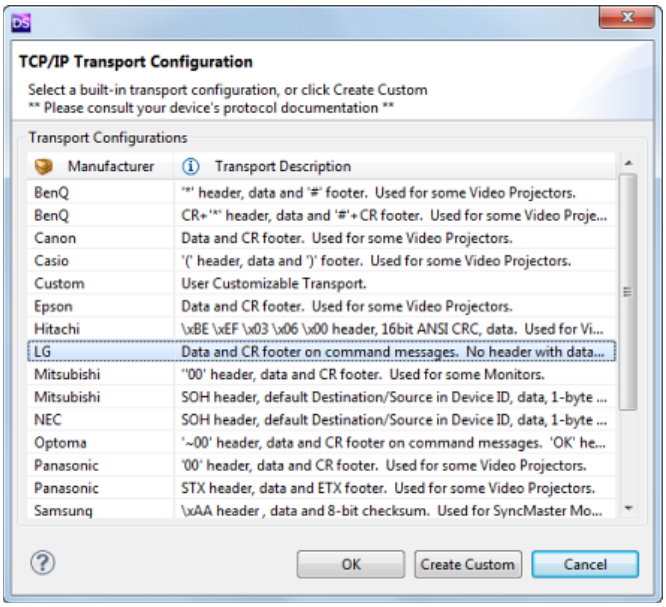


FIG. 38 TCP/IP Transport Configuration dialog

3. Click **OK** to close the *TCP/IP Transport Configuration* dialog.

4. Once a transport configuration has been selected, a set of communication fields are provided (as well as *Select/Edit/Test* command buttons, and a *Remove TCP/IP* button). Use these fields to modify communication settings as required by the device (FIG. 39):

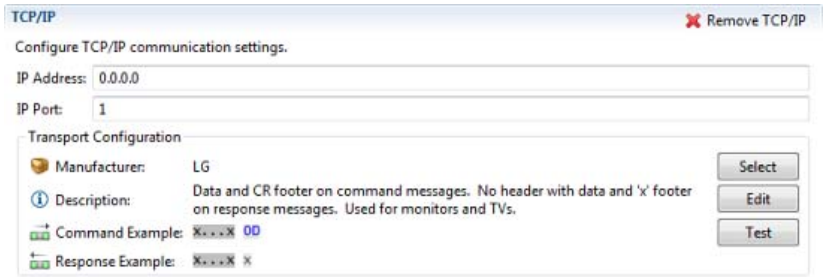


FIG. 39 TCP/IP Configuration options

TCP/IP Communication Settings	
IP Address:	Enter the default IP address of the device, if it has one. Otherwise, enter "0.0.0.0". At runtime, the NetLinx application may provide the device's ip address.
IP Port:	Enter the IP Port used by the device.
Transport Configuration: The (read-only) fields in the Transport Configuration section provide a summary of the currently selected transport configuration. Note that these options are the same for Serial or TCP/IP transports, and are available only after a transport configuration has been selected.	
• Manufacturer:	This field displays the configuration's manufacturer, based on the selected predefined configuration. *Note: An asterisk ("*") prefix denotes that the configuration has been edited.*
• Description:	This field provides a basic description of the configuration, based on the selected predefined configuration. The Description can be edited via the *TCP/IP Transport Configuration - Command Format* dialog (click the **Edit** command button to access this dialog).
• Command Example:	This field provides an example of the command messages sent to the device, according to the current message format defined in the *TCP/IP Transport Configuration - Command Format* dialog (click the **Edit** command button to access this dialog).
• Response Example:	This field provides an example of the response messages sent from the device, according to the current transport configuration.
The *Select*, *Edit* and *Test* command buttons in this section apply to the TCP/IP Transport Configuration settings:	
• Click **Select** to open the *TCP/IP Transport Configuration* dialog, to select a different transport configuration.	
• Click **Edit** to open the first dialog of the *New Transport Configuration Wizard* - the *TCP/IP Transport Configuration - Command Format* dialog. Use the options in this dialog to define the Command Format. The Command Format specifies the format of command messages that are sent to the device (see *Defining the Command Format* on page 29 for details). The second dialog in the Wizard is the *TCP/IP Transport Configuration - Receive Format* dialog. The Receive Format specifies the format of messages that are received from the device (see *Defining the Response Format* on page 32 for details).	
• Click **Test** to open the *Test TCP/IP Communication* dialog. This feature allows you to test the current transport configuration. See *Testing a Transport Configuration* on page 32 for details.	

Creating a Custom Transport Configuration

If there is not an appropriate Transport Configuration available for your device, you can create a custom transport to accommodate the communication requirements of your device. There are two ways to approach creating a custom transport configuration:

- Select an existing transport configuration that most closely matches the requirements of your device and edit it as required, via the options in the *Communication* tab. See *Editing the Current Transport Configuration* on page 29.
- Create a Custom transport configuration from scratch via the **Create Custom** option in the *Serial Transport Configuration* and *TCP/IP Transport Configuration* dialogs - as described below.

The procedure for creating a custom transport configuration is identical for Serial and TCP/IP transports.

1. In the *Communication* tab:
   - Click **Add Serial Control** to open the *Serial Transport Configuration* dialog.
   - Click **Add TCP/IP Control** to open the *TCP/IP Transport Configuration* dialog.
2. In the *<Serial or TCP/IP> Transport Configuration* dialog, click **Create Custom** to open the *New Transport Configuration* wizard (FIG. 40).

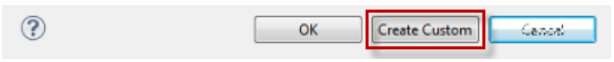


FIG. 40 <Serial or TCP/IP> Transport Configuration dialog - Create Custom button

The dialogs in this wizard allow you to define the Command and Receive message formats to be used with the new transport configuration. Note that the options in these dialogs are identical for serial and TCP/IP transports.

- The first dialog in the wizard is the *<Serial or TCP/IP> Transport Configuration - Command Format* dialog.

- The second dialog is the *<Serial or TCP/IP> Transport Configuration - Response Format* dialog.

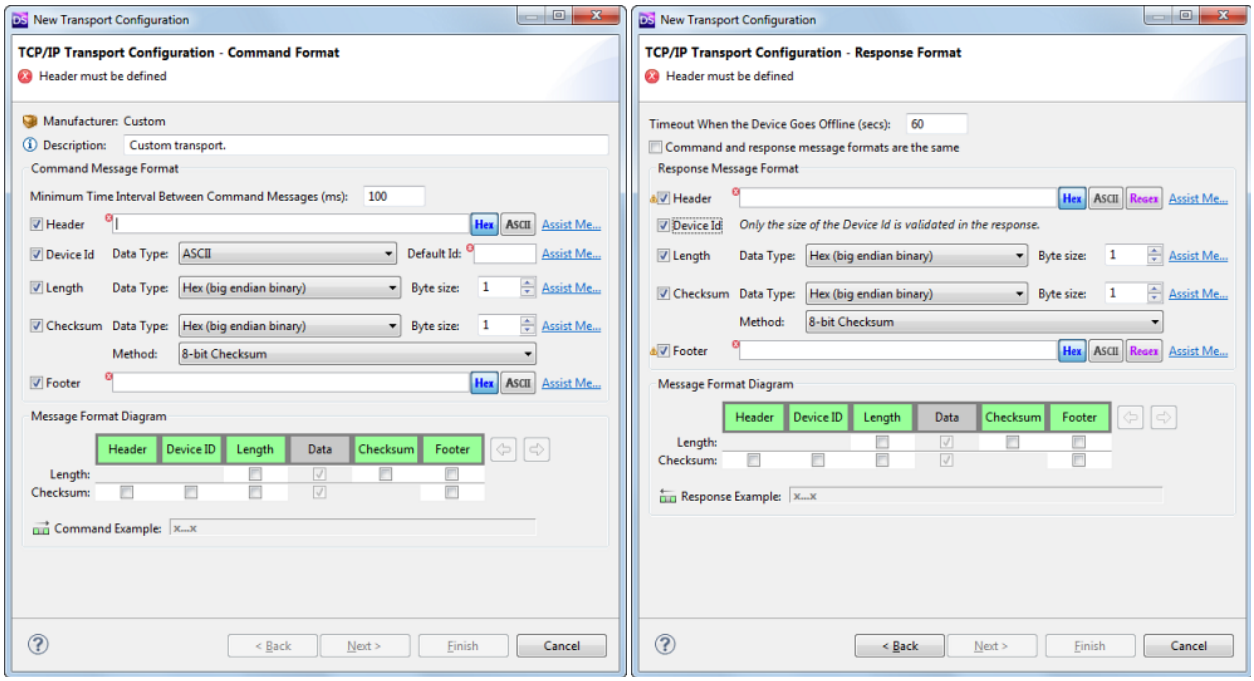


FIG. 41 New Transport Configuration Wizard dialogs

3. In the *<Serial or TCP/IP> Transport Configuration - Command Format* dialog, fill in the fields to define the Command Format to be used. See *Defining the Command Format* on page 29 for details.
4. Click **Next** to proceed.
5. In the *<Serial or TCP/IP> Transport Configuration - Response Format* dialog, fill in the fields to define the Response Format to be used. See *Defining the Response Format* on page 32.
6. Click **Finish** to save the message format settings and close the *New Transport Configuration* wizard.

Creating an "Empty" Custom Transport

To create an empty custom transport (named "*Custom transport.*") that uses default transport settings, and has no message formats defined:

1. In the *Communication* tab:
   - Click **Add Serial Control** to open the *Serial Transport Configuration* dialog.
   - Click **Add TCP/IP Control** to open the *TCP/IP Transport Configuration* dialog.
2. In the *<Serial or TCP/IP> Transport Configuration* dialog, select **Custom** in the *Transport Configurations* list (FIG. 42):

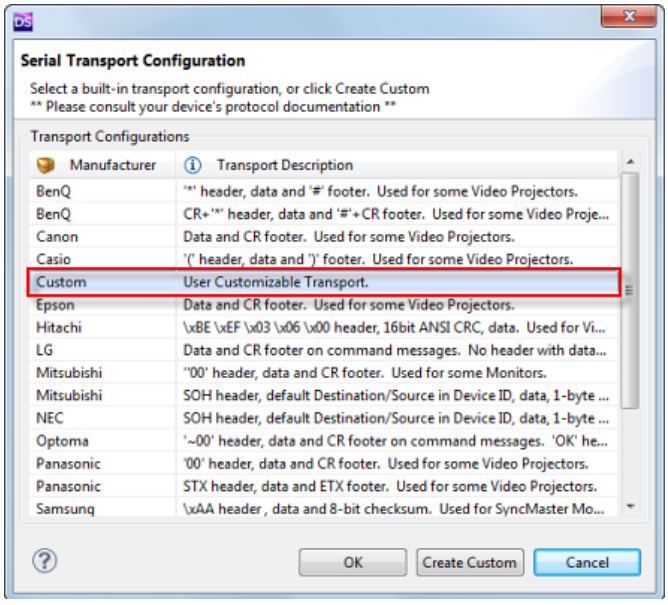


FIG. 42 <Serial or TCP/IP> Transport Configuration dialog - "Custom" selected

3. Click **OK** to close this dialog.

The *Communication Settings* and *Transport Configuration* settings indicate an "empty" transport with default values assigned. FIG. 43 shows an empty Serial Transport:

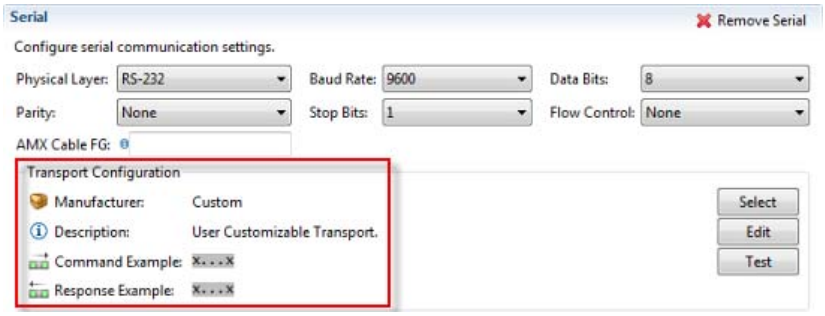


FIG. 43 Communication tab indicating an "empty" Custom (Serial) Transport

Editing the Current Transport Configuration

Use the options in the Communications tab to edit the currently selected Transport Configuration:

1. Use the *Communications Settings* fields to edit basic (Serial or TCP/IP) communications settings.
2. Use the **Edit** command button to edit the Command and Receive Message formats.
   - See *Defining the Command Format* on page 29
   - See *Defining the Response Format* on page 32
3. Click **Next** to apply your changes, and save your project.

Defining the Command Format

Use the options in the *<Serial or TCP/IP> Transport Configuration - Command Format* dialog (FIG. 44) to define the Command Format (the message format for commands sent to the device):

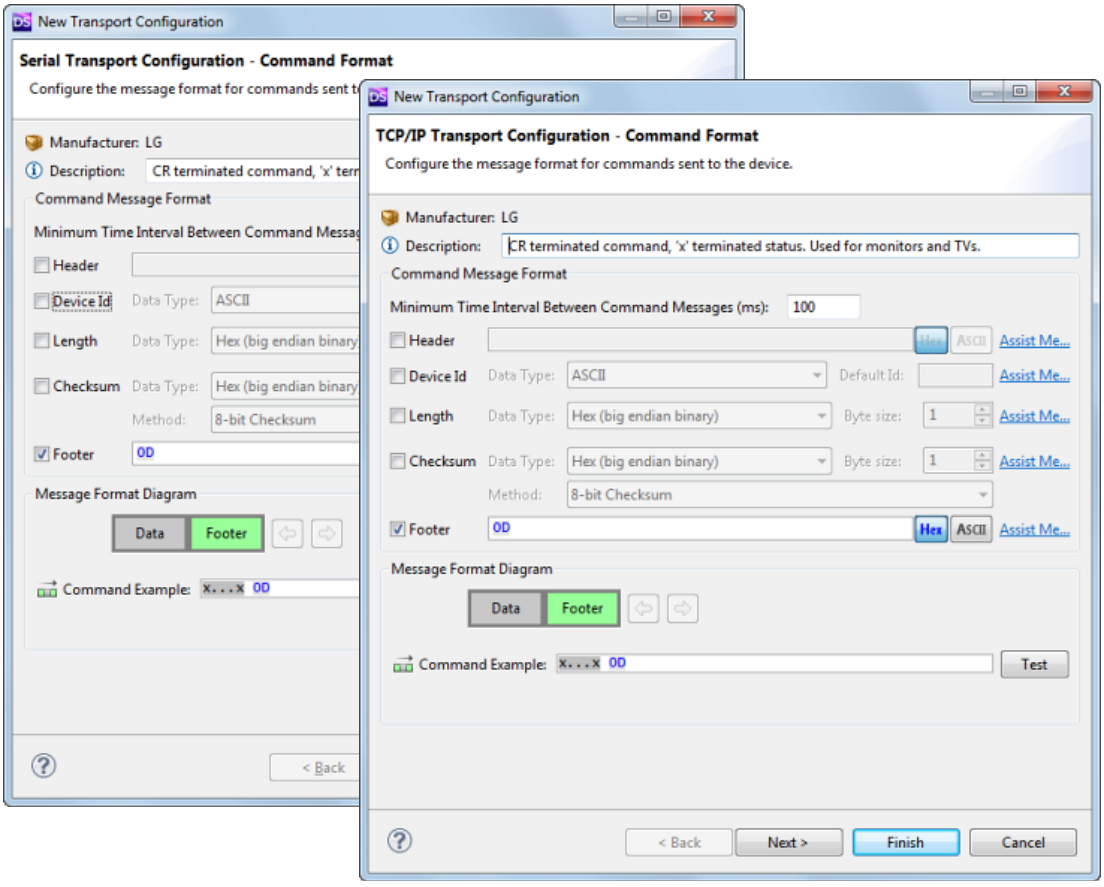


FIG. 44 <Serial / TCP/IP> Transport Configuration - Command Format dialogs

NOTE: *The options in these dialogs (Serial and TCP/IP) are the same.*

The fields in this dialog are described below:

- **Manufacturer**: This read-only field indicates the Manufacturer selected in the Device Information tab.
- **Description**: Enter a descriptive name for this transport configuration (default = "*Custom transport*")

Command Message Format options

- **Minimum Time Interval Between Command Messages**: Set in milliseconds,
  - The default interval for predefined transports is either **100ms** or **500ms**
  - The default interval for custom transports the default is **100ms**.
- **Header**: If the device requires a header (such as a STX code) to mark the beginning of a message, select this option and enter the command header format here.

  Use the **Hex/ASCII** buttons to enter hexadecimal or ASCII formatted characters: by default, Hex input is selected (FIG. 45):

FIG. 45 Hex / ASCII input mode buttons

  With *Hex* selected, only hexadecimal input is allowed. To input ASCII characters, select the *ASCII* button. Note that the command examples are colorized to indicate hex (blue), ASCII (gray), and regex (purple).

NOTE: *These character types can be used interchangeably.*

  Click the **Assist Me** link to open a dialog that provides some basic information regarding this field. Note that if the Header option is selected, the Header element is added to the Message Format Diagram on this dialog (see below).

- **Device Id**: If the device requires a device ID (for device addressing), select this option and enter the default device ID here.

  Use the *Data Type* drop-down menu to specify the type of data used to store the Device ID value:
  - ASCII (default selection)
  - Hex (big endian binary)
  - Hex (little endian binary)

  Click the *Assist Me* link to open a dialog that provides some basic information regarding this field. Note that if the Device Id option is selected, the Device Id element is added to the Message Format Diagram on this dialog (see below).
- **Length**: If the device requires a length segment to identify the next n-bytes of data in the message, select this option and set the default *Byte Size* here (1-4). The default setting is 1.

  Use the *Data Type* drop-down menu to specify the type of data used to store the Length value:
  - ASCII Decimal
  - ASCII Decimal (space-padded)
  - ASCII Decimal (zero-padded)
  - ASCII Hex Lowercase
  - ASCII Hex Lowercase (space-padded)
  - ASCII Hex Lowercase (zero-padded)
  - ASCII Hex Uppercase
  - ASCII Hex Uppercase (space-padded)
  - ASCII Hex Uppercase (zero-padded)
  - Hex (big endian binary)
  - Hex (little endian binary)

  By default, *Hex* (big endian binary) is selected.

  Click the **Assist Me** link to open a dialog that provides some basic information regarding this field. Note that if the *Length* option is selected, the Length element is added to the Message Format Diagram on this dialog (see below).
- **Checksum**: If the device requires a checksum, select this option and set the Data Type and Method to be used:

  Use the *Data Type* drop-down menu to specify the type of data used to store the Checksum value:
  - ASCII Decimal
  - ASCII Decimal (space-padded)
  - ASCII Decimal (zero-padded)
  - ASCII Hex Lowercase
  - ASCII Hex Lowercase (space-padded)
  - ASCII Hex Lowercase (zero-padded)
  - ASCII Decimal Number
  - ASCII Decimal Number (space-padded)
  - ASCII Decimal Number (zero-padded)
  - Hexadecimal Binary Value (big endian)
  - Hexadecimal Binary Value (little endian)

  By default, *Hex* (big endian binary) is selected.

  Use the **Byte Size** field to specify the number of bytes that make up the checksum (1-4); default setting = 1.

  Use the **Method** drop-down to select the checksum method to be used.

Click the **Assist Me** link to open a dialog that provides some basic information regarding this field. Note that if the Checksum option is selected, the Checksum element is added to the Message Format Diagram on this dialog (see below).

- **Footer**: If the device requires a footer (such as a ETX code) to mark the end of a message, select this option and enter the command footer format here.

  Use the **Hex/ASCII** buttons to enter hexadecimal or ASCII formatted characters: by default, Hex input is selected. With *Hex* selected, only hexadecimal input is allowed. To input ASCII characters, select the ASCII button. These character types can be used interchangeably.

  Click the **Assist Me** link to open a dialog that provides some basic information regarding this field. Note that if the Footer option is selected, the Footer element is added to the Message Format Diagram on this dialog (see below).

- **Message Format Diagram**: As command message format elements are enabled (via the checkboxes in this dialog), enabled elements are added to the Message Format Diagram. The Message Format Diagram (FIG. 46) indicates the order in which these elements are arranged. For example, if you have selected all of the command message format elements, the Message Format Diagram indicates the default order of the elements (*Header* first, followed by *Device ID*, *Length*, *Data*, *Checksum* and then *Footer*):

FIG. 46 Message Format Diagram

  - **Length**: The checkboxes in the *Length* row indicate which element(s) are to be included in the Length calculation. Note that by default only *Data* is selected for inclusion. You can manually select other elements to include by selecting other checkboxes. The application will enforce basic logical restrictions on these selection.
  - **Checksum**: The checkboxes in the *Checksum* row indicate which element(s) are to be included in the Checksum calculation. Note that by default only Data is selected for inclusion. You can manually select other elements to include by selecting other checkboxes. The application will enforce basic logical restrictions on these selection.

- **Command Example**: This read-only field indicates the structure of the command message format, based on the information entered in this dialog.
- **Test**: Click to test the command message format. This action opens the *Test <Serial or TCP-IP> communication* dialog. Use this dialog to connect the PC's COM port to the device, and enter the device protocol to test (and click *Test*). See *Testing a Transport Configuration* on page 32 for details.
- **Back**: Click to go to the previous dialog in the wizard (disabled in this dialog)
- **Next**: Click to proceed to the next dialog in the wizard (the *<Serial or TCP/IP> Transport Configuration - Command Format* dialog). Note that this button is only enabled when all required information has been provided in this dialog.
- **Finish**: Click to save your changes and close the New Transport Configuration Wizard.
- **Cancel**: Click to close the wizard without saving changes.

Re-Ordering Message Format Elements

In the Message Format Diagram, select a message format element to move, then use the left and right arrow buttons to move the selected element to it's proper position in the command message format. Note that there are some restrictions regarding the placement of each element (for example, the Header must come first). The arrow buttons indicate when the selected element cannot be moved in either direction (FIG. 47):

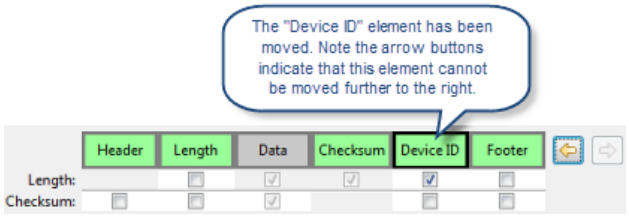


FIG. 47 Message Format Diagram - Re-ordering messaging elements

Defining the Response Format

Use the options In the *<Serial or TCP/IP> Transport Configuration - Response Format* dialog (FIG. 48) to define the Response Format (the message format for commands sent from the device to the control system).

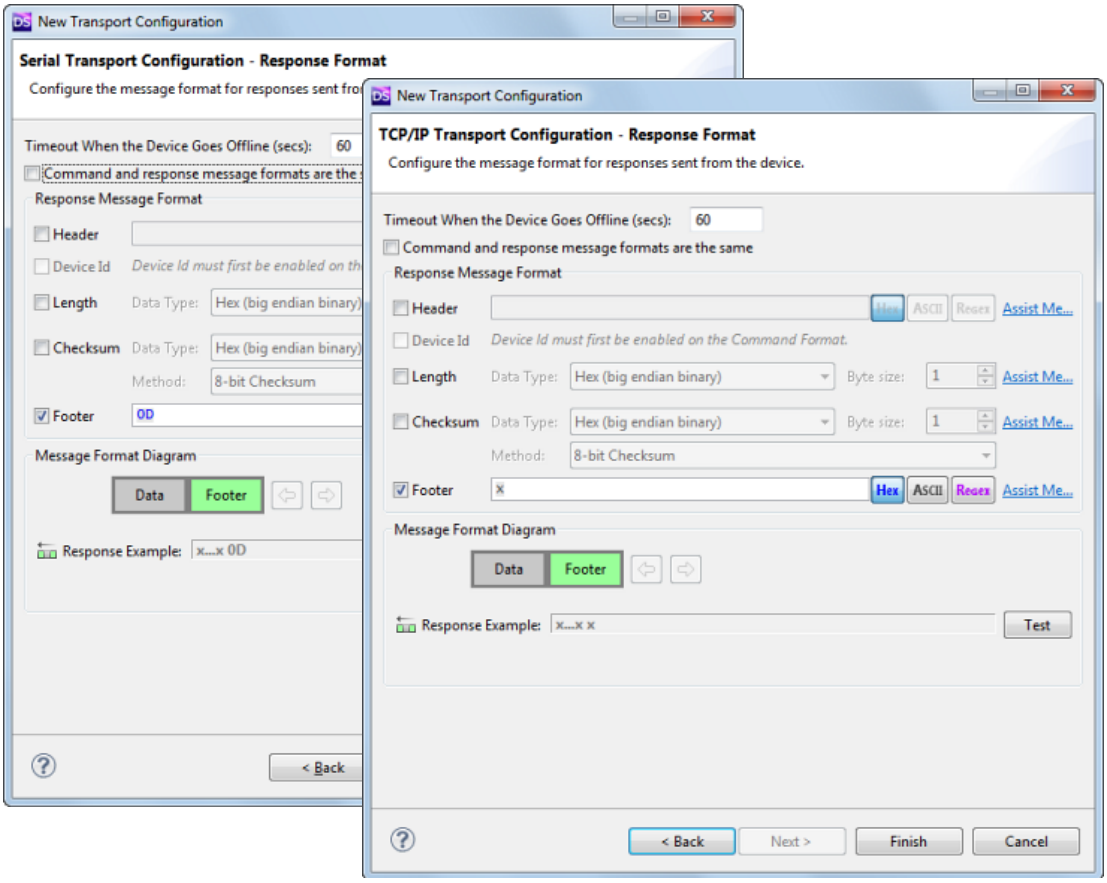


FIG. 48 <Serial / TCP/IP> Transport Configuration - Response Format dialogs

NOTE: *Note that fields in this dialog are the same as those in the previous (Command format) dialog (see the Command and response message formats are the same option below). Also note that command and response options are identical for Serial and TCP/IP transports.*

Command and response message formats are the same

Click this option if the device uses the same message format for command and response messages. In this case there is no need to define the response message format. Note that with this option selected, the Response Message Format options (below) are disabled. By default, this option is de-selected.

NOTE: *Device ID is ignored in the response message format.*

Testing a Transport Configuration

Once a Transport Configuration has been selected (in the Communication tab), you can test the current Serial or TCP/IP configuration (FIG. 49):

FIG. 49 Communication tab - Test button

1. Click **Test** to open the *Test Serial Communication* or *Test TCP/IP Communication* dialog (depending on which type of transport you are testing).
2. Enter the "x...x" portion of an actual command in the *Device Protocol* text field, and the transport will take care of packaging the command with all other defined fields (header, footer...) before sending it out to the device. This enables the **Test** button in this dialog (FIG. 50):

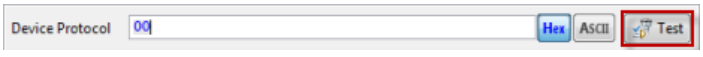


FIG. 50 Test Serial Communication dialog

For Serial transport, the default serial COM port connection uses the COM1 port. To change the Serial COM port connection, select from the COM port drop-down menu (FIG. 51):

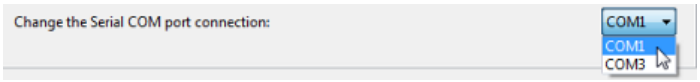


FIG. 51 Test Serial Communication dialog - COM menu

3. Click **Test** to start. The results of the test are displayed in the *Test Results* window.

Removing a Transport Configuration

Once a Serial or TCP/IP Transport Configuration has been defined (in the Communication tab), the *Remove Serial* and/or *Remove TCP/IP* buttons are enabled (FIG. 52):

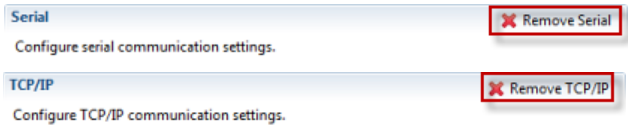


FIG. 52 Remove Serial and Remove TCP/IP buttons

- Click **Remove Serial** to remove Serial Control from the device. Note that this action enables the *Add Serial Control* option.
- Click **Remove TCP/IP** to remove TCP/IP Control from the device. Note that this action enables the *Add TCP/IP Control* option.

Defining Device Control

Overview

Use the options in the *Control* tab of the Driver XML Editor to define device control. Device communication settings are managed via options in the *Communication* tab of the Driver XML Editor (FIG. 53):

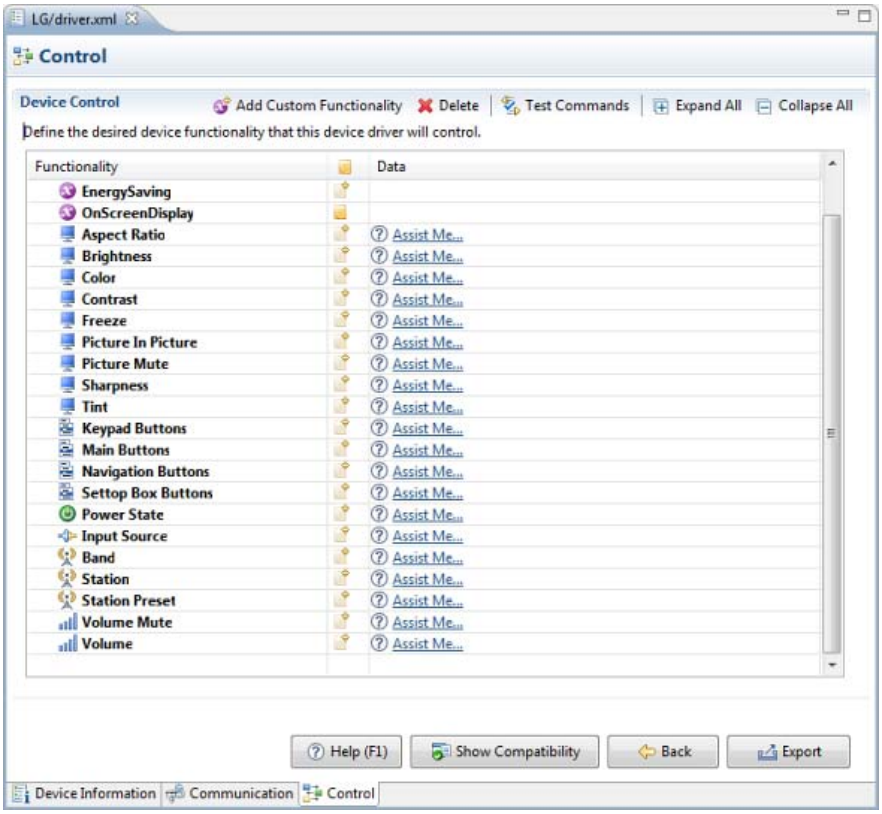


FIG. 53 Control tab of the Driver XML Editor

This entails several steps:

1. Defining Device Properties (see page 34)
2. Defining Device Protocols (see page 36)
3. Testing Commands (see page 42)

Once the device control has been fully defined and tested, the driver design file can be exported as a driver.XML file. See *Exporting Device Drivers* on page 43 for details.

Defining Device Properties

Many device functions utilize device properties, as indicated in the Device Control table (in the Control tab). For example, the Brightness Functional Group uses two properties: *Brightness Max* and *Brightness Min* (FIG. 54):

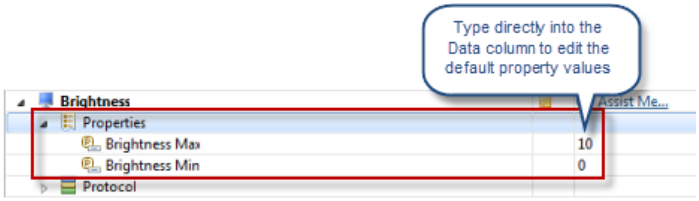


FIG. 54 Device Properties - Brightness Max and Brightness Min

NOTE: *Click the arrow icon next to the Functional Group to expand the view to show Properties and Protocols. Click Expand All in the toolbar to expand all Functional Groups.*

Note that device properties typically have default values pre-assigned. To change these values, click on a device property (in the Value column), and type directly in the text field.

Function Icons

In the *Control* tab of the Driver XML Editor, each device function uses a specific icon to identify it as either a specific type of device property or a command: There are three types of Property functions: *Device Property*, *Property Constant* and *Input Source* (FIG. 55):

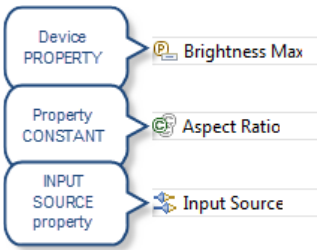


FIG. 55 Control tab - Property Icons

Property Icons	
Device Property:	These are basic properties of the device that require a value input.
Property Constant:	These are properties that require selection from a device-specific set of options (these pre-defined options are called constants).
Input Source:	This property is used to add an input source to devices that support input devices.

Using the "Assist Me" Feature

The *Assist Me* links that are provided in the Control tab provide guidance for defining standard device functions. Note that there is an Assist Me link for each top-level device function in the Control tab (FIG. 56):

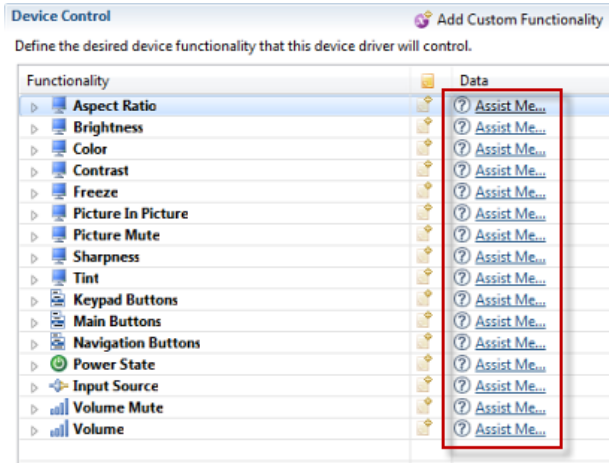


FIG. 56 Control tab - "Assist Me" links

Note that Assist Me links are not provided for Custom Functions.

To use this feature, click on **Assist Me** on any device function for which you would like assistance in defining. This opens a *<Function> Assistance* dialog that guides you through the process of defining the selected function. The contents of this dialog depends on the function selected, but generally these dialogs provide additional information on each property and protocol required by the function (FIG. 57):

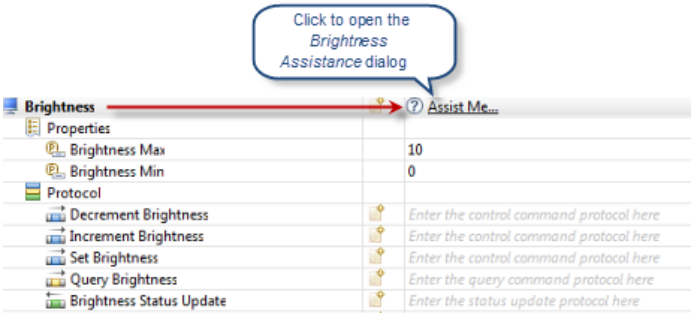


FIG. 57 Control tab - "Assist Me" link for Brightness

For example, clicking the *Assist Me* link for the Brightness function opens the first of a series of *Brightness Assistance* dialogs. These dialogs will guide you through entering all of the Data indicated in the Control tab (FIG. 58):

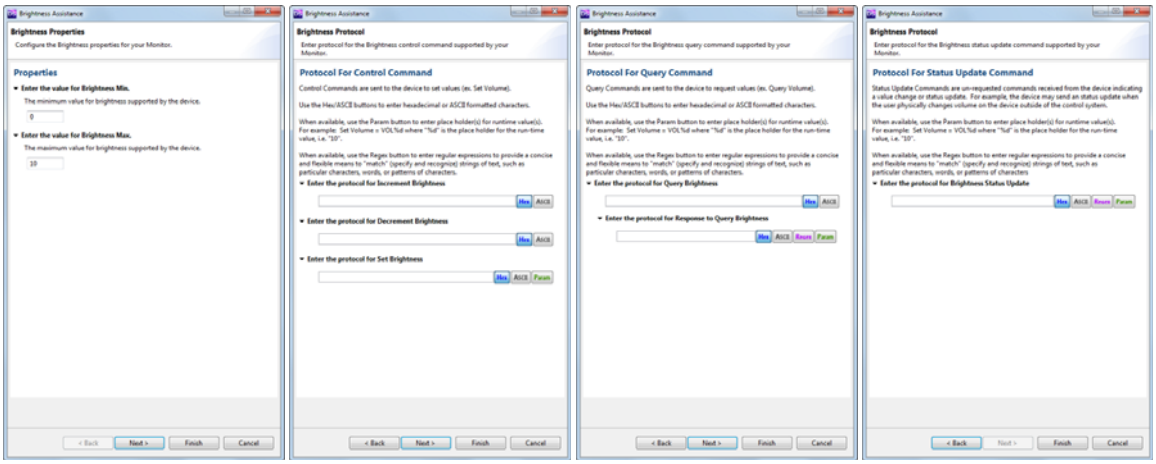


FIG. 58 Brightness Assistance dialogs

- The first Assistance dialog (**Properties**) prompts you to enter properties (in this example, *Brightness Max* and *Brightness Min*). Click *Next* to proceed to the next dialog.
- The next Assistance dialog (**Protocol For Control Commands)** prompts you to enter protocol for the standard control commands, and provides basic information on entering control command protocol. Click *Next* to proceed to the next dialog.
- The next Assistance dialog (**Protocol For Query Commands**) prompts you to enter protocol for the standard queries, and provides basic information on entering query command protocol. Click *Next* to proceed.
- The next Assistance dialog (**Protocol For Status Update Command**) prompts you to enter protocol for the standard Status Update command, and provides basic information on entering this protocol.

Click **Finish** to close the last Assistance dialog. Note that the data entered in these dialogs is reflected in the Control tab.

Defining Device Protocols

Each device command must be defined in the protocol that is expected by the device in order to control each function. Consult your device manufacturer's product documentation to determine the correct protocol required for each function's command.

Use the *Protocol* entries in the Device Control table (Control tab) to define the protocol for each device command. All device functions utilize command protocols, as indicated in the Device Control table. For example, the Brightness Functional Group uses several Protocols: *Decrement/Increment/Set/Query Brightness* and *Brightness Status Update* (FIG. 59):

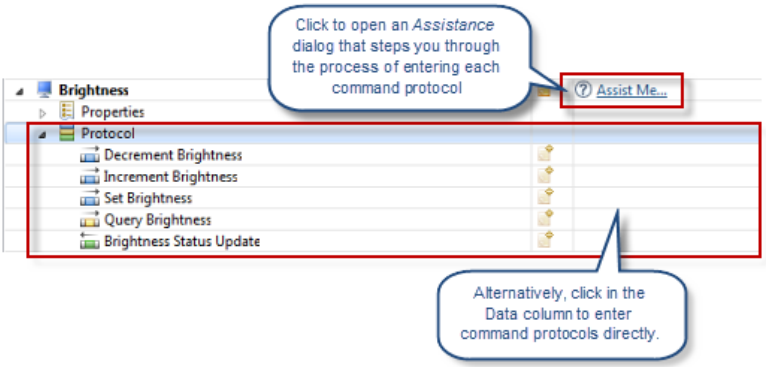


FIG. 59 Device Protocols

NOTE: *Click the arrow icon next to the Functional Group to expand the view to show Properties and Protocols. Click Expand All in the toolbar to expand all Functional Groups.*

There are two ways to approach entering command protocols:

- Click the **Assist Me** link to open an Assistance dialog that steps you through the process of entering the protocol of each command. See *Using the "Assist Me" Feature* on page 35 for details.
- Click in the **Data** column to enter the protocol directly, if you don't require assistance. Note that when you click in a Data cell, a set of *Input Mode Options* buttons are presented.

Protocol Icons

There are four types of Protocol functions: *Control Commands*, *Status Updates*, *Query Commands* and *Query Responses*:

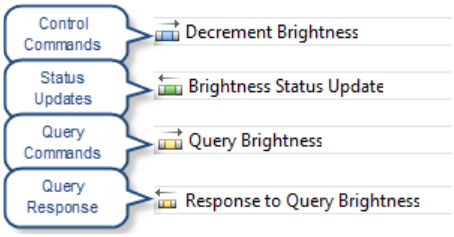


FIG. 60 Control tab - Protocol Icons

Protocol Icons	
Control Commands:	These are commands that are sent from the control system to the device.
Status Updates:	These are asynchronous (unsolicited) commands that are sent from the device to the control system (no response is required).
Query Commands:	These are queries that are sent from the control system to the device. For query commands, a response is expected from the device.
Query Response:	These are the responses associated with each Query Command. Note that Query Responses are only indicated in the Device Control table after the associated Query Command has been defined.

Input Mode Options

The *Input Mode Option* buttons (displayed only when the cursor is placed within the *Data* cell for a Protocol entry) allow you to specify the mode of input to match the device's protocol requirements. Note that depending on the protocol, some of these mode options will not apply and therefore are not presented (FIG. 61):

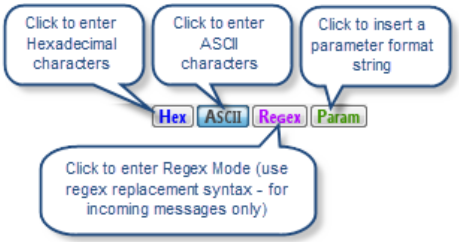


FIG. 61 Input Mode Option buttons

Input Mode Options	
Hex:	Click to limit entry to pairs of hexadecimal characters, with each pair representing 1 byte of data.
ASCII:	Click to limit entry to ASCII characters, with each character representing 1 byte of data.
Regex:	Click to enter Regex Mode, in which regex replacement syntax is used - for incoming messages only. See *AMX Character Class Syntax Redefinition from perl 5.6* on page 63 and *AMX Macro Syntax* on page 63 for details on regex conventions used by Driver Design.
Param:	Click to insert a parameter format string. Clicking on this option opens the *Protocol Parameter Definition* dialog, where you can define a placeholder format string for the parameter. See *Inserting a Parameter String* on page 38 for details.

Adding Notes

The *Notes* column provides a way to add user-defined protocol notes to commands, and to the top-level folder for each functional group. To add a note, click on the **Add Note** icon (FIG. 62) to open the *Edit Notes* dialog (FIG. 62):

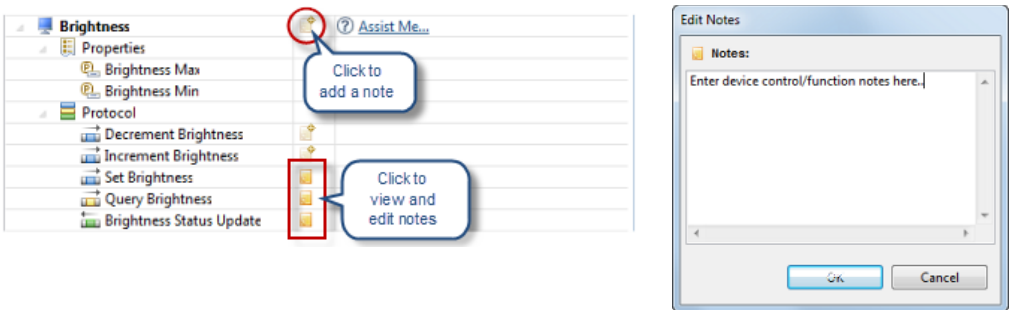


FIG. 62 Add / Edit Note Icons and Edit Notes dialog

Enter the note text and click **OK** to close the dialog. Once a note has been added, the icon changes to indicate that a Note exists for this command. Click the Note icon to view or edit the note text in the *Edit Notes* dialog.

Inserting a Parameter String

Some command protocols require one or more parameters. For example, the protocol for setting brightness (Set Brightness) requires a parameter value to set the brightness level. In order to utilize a parameter string, you must first define the format required by the device for the parameter. The parameter's format definition is called the Placeholder Format String. Consult your device manufacturer's product documentation to determine the required format for command parameters.

A yellow exclamation mark icon is displayed in the Device Control table for commands that require one or more parameters (FIG. 63):

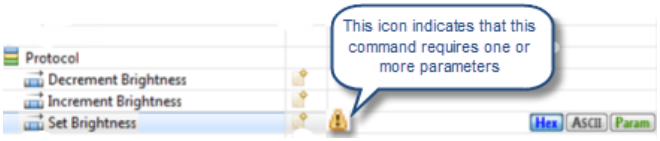


FIG. 63 Parameter Icon

NOTE: *Hover the cursor over this icon to display a brief summary of the parameter(s) required by this command.*

The options in the *Protocol Parameter Definition* dialog can assist you in defining a placeholder format string for the parameters:

1. With a command protocol selected in the Device Control table, click in the *Data* column to enable the *Input Mode Option* buttons (see page 37):

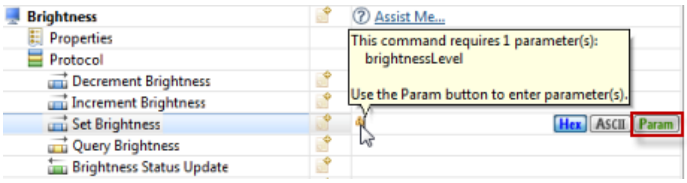


FIG. 64 Protocol Parameter Definition dialog

2. Click the **Param** *Input Mode Option* button to open the *Protocol Parameter Definition* dialog (FIG. 65). This input mode option is only displayed for command protocols that require one or more parameters.

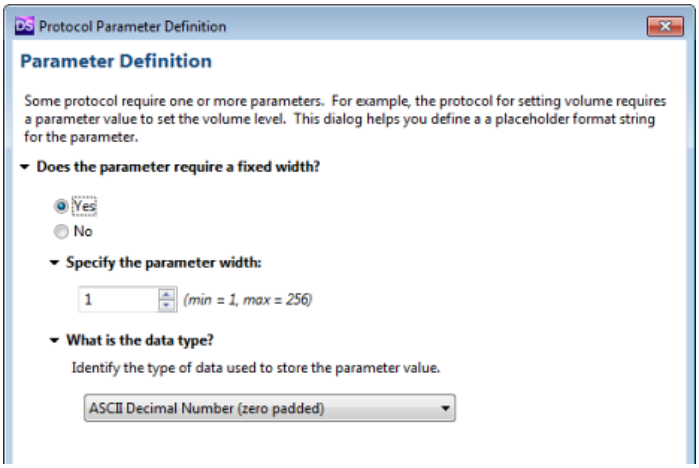


FIG. 65 Protocol Parameter Definition dialog

3. Under **Does the parameter require a fixed width?**, select either *Yes* or *No* according to the device's requirements (default = *No*). If **Yes** is selected, then the **Specify the parameter width** field is presented (with a default setting of "1"). Either enter the value directly in the text field (range - 1-256), or use the arrow buttons to adjust the value as required.
4. Under **What is the data type?**, select the type of data required by the device to store the parameter value. Note that the options in this drop-down list depend on the fixed width selection (above).

   If **No** is selected (the default setting), then the data type options are:

Date Type	Format
ASCII Characters	**"%s"**
ASCII Hexadecimal Number (lowercase)	**"%x"**
ASCII Hexadecimal Number (uppercase)	**"%X"**
ASCII Decimal Integer	**"%d"**
ASCII Decimal Integer (signed with +/-)	**"%+d"**
Hexadecimal Binary Value (big-endian - most significant byte first)	**"%B"**
Hexadecimal Binary Value (little-endian - least significant byte first)	**"%b"**

If **Yes** is selected, then the data type options are:

Date Type	Format
ASCII Hexadecimal Number (lowercase, space padded)	"%<parameter width value> x"
ASCII Hexadecimal Number (lowercase, zero padded)	"%<parameter width value>0x"
ASCII Hexadecimal Number (uppercase, space padded)	"%<parameter width value>X"
ASCII Hexadecimal Number (uppercase, zero padded)	"%<parameter width value>0X"
ASCII Decimal Number (space padded)	"%<parameter width value> d"
ASCII Decimal Number (space padded and signed with +/-)	"%+<parameter width value> d"
ASCII Decimal Number (zero padded)	"%<parameter width value>d"
ASCII Decimal Number (zero padded and signed with +/-)	"%+<parameter width value>0d"
Hexadecimal Binary Value (big-endian - most significant byte first)	"%B"
Hexadecimal Binary Value (little-endian - least significant byte first)	"%b"

These settings will be used when the command protocol is tested (see *Testing Commands* on page 42: when a command that uses a parameter is tested, Driver Design will prompt you to enter a value that conforms to the parameter's placeholder format string defined in this dialog.

Adding Input Sources

Many device functions utilize device properties, as indicated in the Device Control table (in the Control tab). Refer to *Defining Device Properties* on page 34 for details. The **Input Source** property is used to add an input source to devices that support input devices. To add an input source:

1. In the Device Control table, click inside the *Data* cell, for **Input Source > Properties > Input Source** to invoke the Browse icon (FIG. 66):

FIG. 66 Device Control table - Data cell (Input Source > Properties > Input Source): Browse icon

2. Click the Browse (**...**) icon to open the *Edit Input Sources* dialog. Use the options in this dialog to define one or more input sources, and specify the Signal Types that each input source provides to connect to your device.
3. Click inside the first row in the *Input Sources* column, and type a descriptive name for an input source (for example, "DVD"):

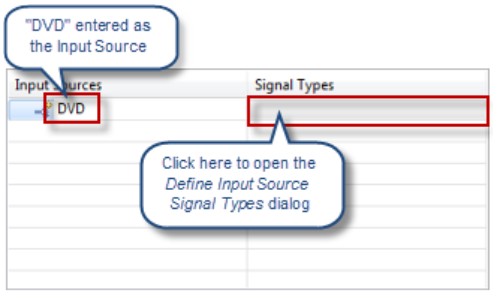


FIG. 67 Device Control table - Data cell (Input Source > Properties > Input Source): Browse icon

4. Click inside the Signal Types cell to open the *Define Input Source Signal Types* dialog. Use the options in this dialog to define the signal types supported by this input source:
   a. Click the checkboxes to select the supported signal types.
   b. Click **OK** to close the *Define Input Source Signal Types* dialog. The selected signal types are indicated in the Signal Types cell for the new input source (FIG. 68):

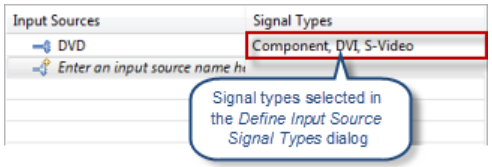


FIG. 68 Device Control table - Data cell (Input Source > Properties > Input Source): Browse icon

NOTE: *Once you have defined a source input, select it to enable the Delete and Rename command buttons:*

- Click **Delete** to delete the selected input source.
- Click **Rename** to edit the name of the selected input source.

5. To add more input sources, click on the next row in the *Edit Input Sources* dialog and repeat steps 3 and 4 above for each input source.

   NOTE: *Once you have more than one source input defined, select one to enable the Up and Down buttons. Use these buttons to re-order the input sources.*

6. Click **OK** to close the *Edit Input Sources* dialog. The input sources are now indicated in the Data row for the Input Source function (FIG. 69):

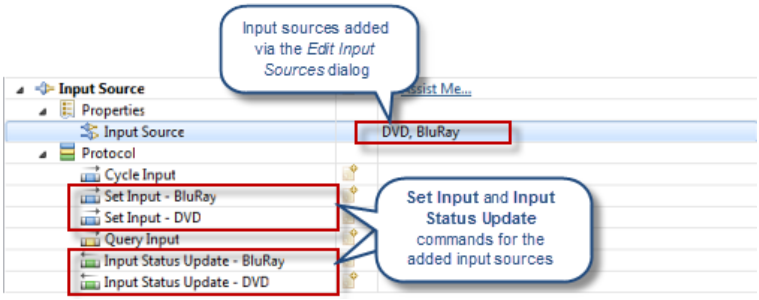


FIG. 69 Added Input Sources

7. Enter protocol for the **Set Input** and **Input Status Update** commands (these commands are added for each input source defined).

Editing Input Sources

Once one or more input sources have been defined, they can be edited via the *Edit Input Sources* dialog:

1. Click on an input source (in the Device Control table) to enable the *Edit (...)* button (FIG. 70):

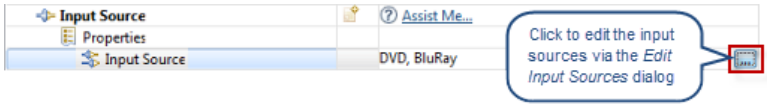


FIG. 70 Editing an Input Source

2. Click the **Edit (...)** button to open the *Edit Input Sources* dialog.
3. Edit the input sources as desired, and click **OK** to save changes and close the dialog.

Adding Custom Functions

The *Functionality* column of the Device Control table lists default device functions, based on AMX application compatibility requirements for the Device Type selected in the *New Project Wizard* (see page 11). This initial list of device functions will provide a starting point for defining the functionality that this device driver will control.

However, if a device function is needed that is not in the default listing, you can add custom functions via the *Add Custom Functionality* toolbar button (FIG. 71):

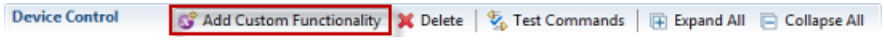


FIG. 71 Device Control toolbar - Add Custom Functionality

To add a custom function:

1. In the Control tab toolbar, click **Add Custom Functionality** to open the *Add Custom Functionality* dialog (FIG. 72):

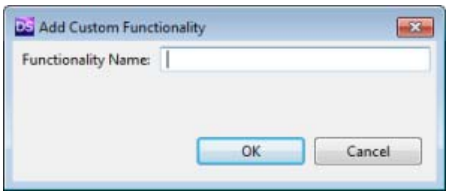


FIG. 72 Add Custom Functionality dialog

2. Enter a descriptive name for the new function in the **Functionality Name** text field and click **OK** to close the dialog.
3. The new function is added to the Device Control table, with placeholder entries for three commands: *Control* command, *Query* command and a *Status Update* command (FIG. 73):

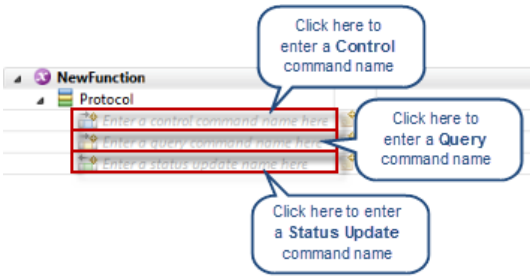


FIG. 73 New Function - command placeholders

4. To a add a note to describe the custom function, click on the *Notes:* icon to open the *Edit Notes* dialog. The note entered here will be added to the generated documentation (FIG. 74):

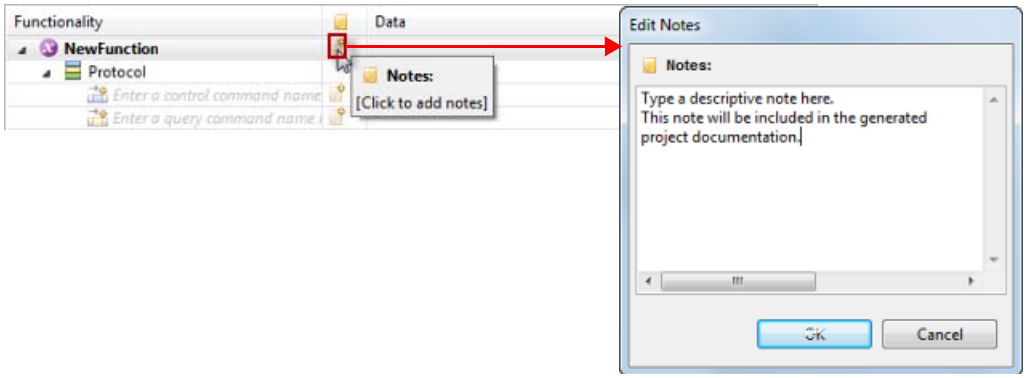


FIG. 74 New Function - Entering a descriptive Note

5. Enter command names for the *Control*, *Query* and *Status Update* commands as required by the new function:
   a. Click in the first cell in the *Functionality* column (labeled "**Enter a control command name here**") and type a descriptive name for the new Control command. Note that each time you enter a new command name, a row is added to the table directly beneath the cursor position. Enter the command protocol in the Data column (see *Defining Device Protocols* on page 36 for details). Repeat to add more Control commands as necessary (FIG. 75):

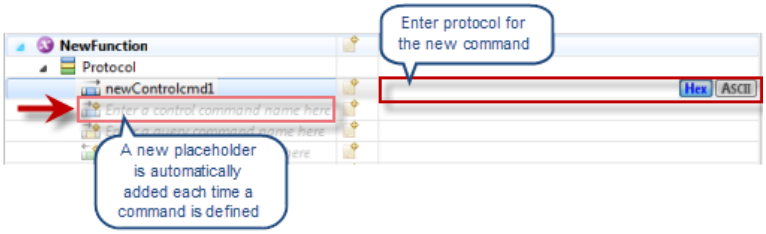


FIG. 75 Entering protocol for new commands

   b. Click in the cell (labeled "**Enter a query command name here**") and type a descriptive name for the new Query command. Enter the command protocol in the Data column (see *Defining Device Protocols* on page 36 for details).
      - Once a query command and its associated protocol are entered, the row immediately below the query command becomes a row for entering the query response and the protocol format for the response message. Define the query response and the response message format.
      - Repeat to add more Query commands as necessary.
   c. Click in the cell (labeled "**Enter a status update command name here**") and type a descriptive name for the new Status Update command. Enter the command protocol in the Data column (see *Defining Device Protocols* on page 36 for details). Repeat to add more Status Update commands as necessary.
6. Enter notes as desired (see *Adding Notes* on page 37).
7. Test the new function (see *Testing Commands* on page 42).

Deleting Custom Functions

Note that unlike default Functions, custom functions (and commands) can be deleted:

1. In the Device Control table, select the custom function (or a protocol entry within a custom function) that you want to delete from the project.
2. Click the **Delete** toolbar button.

NOTE: *If you select the Function itself, the selected function and all of its protocol will be deleted. Select a protocol entry to delete only the selected protocol, leaving the function in place. The program will prompt you to verify this action before the selection is deleted.*

Testing Commands

Once you have defined the device's command protocol, you should test the commands against the target device, via the *Test Commands* toolbar button (FIG. 76):

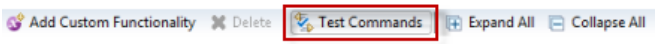


FIG. 76 Device Control toolbar -Test Commands

1. Click **Test Commands** to add a new column to the Device Control table: **Test** (FIG. 77):

FIG. 77 Device Control table -Test links

2. Click a *Test* link to open the *Configure Test Connection* dialog (FIG. 78). Use the options in this dialog to define the configuration for establishing a test connection to the device. By default, this dialog is set to use the *Transport Configuration* specified in the Communication tab (see page 24).

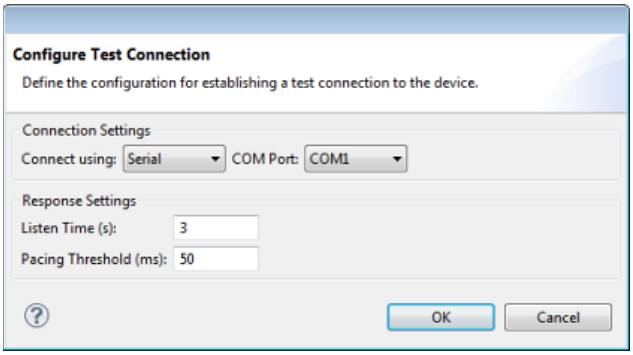


FIG. 78 Configure Test Connection dialog (Serial connection shown)

   a. Edit the communication settings if necessary to connect to the device for testing.
   b. Click **OK** to attempt a connection based on the current transport configuration.

   Note that when Driver Design attempts to connect to the device, the *Test* window displays test activity and feedback (FIG. 79). The results of the connection attempt are indicated here. If the connection fails, review the current transport configuration and try again.

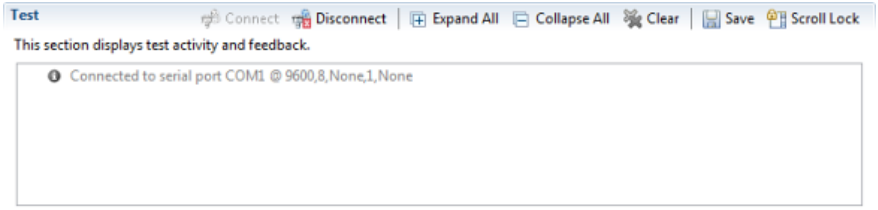


FIG. 79 Test Window

3. With a successful connection, the message data associated with the selected command will be transmitted to the device, and the test engine will begin listening for responses.

   NOTE: *No further commands can be tested until the listening process has been completed.*

The *Test* window displays the following:

- The name of the command being tested
- The raw bytes sent to the device
- Any data received from the device
- Any Events that are triggered as a result of the received data

Exporting Device Drivers

Overview

A Device Driver represents the output of an exported Driver Design project, in the form of a package (archival file) with a "**.xdd**" extension. Device Drivers can be exported to either the AMX InConcert Resource Center or a local directory. The *Device Driver Export* dialog can be accessed via the Control tab of the Driver XML Editor (to export the current project only), or via the File menu (select **File > Export** to export multiple projects). Note that when a project is exported, a project report file (*documentation.pdf*) is generated and added to the project folder in the Project Explorer.

AMX InConcert Resource Center

The AMX InConcert Resource Center is an online repository of AMX certified device modules, accessible to dealers via www.amx.com.

- Driver Design provides the ability to export Device Drivers to the AMX InConcert Resource Center. See *Exporting Device Drivers* (below) for details.
- You can also import Device Drivers into Driver Design from the AMX InConcert Resource Center. See *Importing Device Drivers Into Driver Design* on page 18 for details.

Exporting the Current Driver Design Project

When your Driver Design project is complete, you can export the project as an Device Driver (.xdd file) to either a local directory, or to the online AMX InConcert Resource Center:

1. In the *Control* tab, click **Export** (FIG. 80):

FIG. 80 Control tab - Export button

This opens the *Device Driver Export* dialog (FIG. 81).

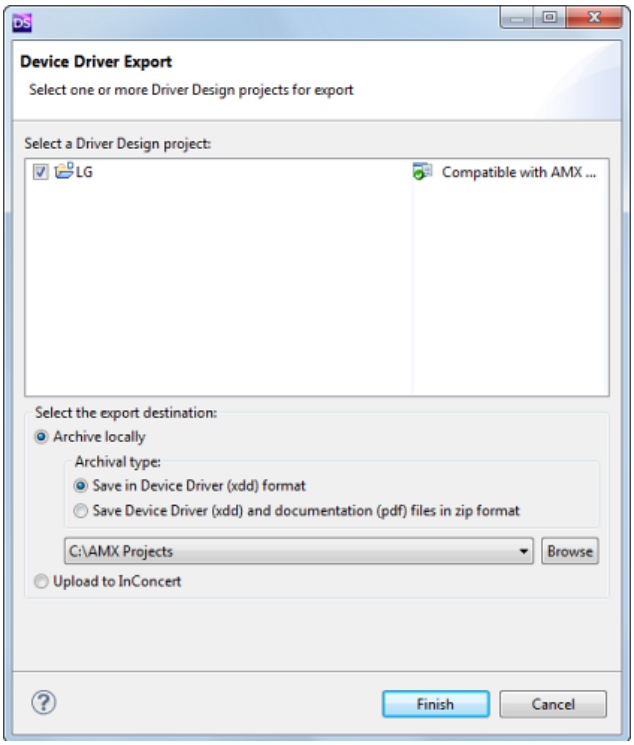


FIG. 81 Device Driver Export dialog

Note that the current project is indicated and pre-selected for export. Note that this dialog indicates whether the Driver is Application Compatible - only AMX Application Compatible drivers can be uploaded to the AMX InConcert Resource Center (see below):

2. Under *Select the export destination*, select to export the Device Driver to a local directory, or to InConcert:

   Select **Archive locally** to export the project file to the specified directory (FIG. 82):

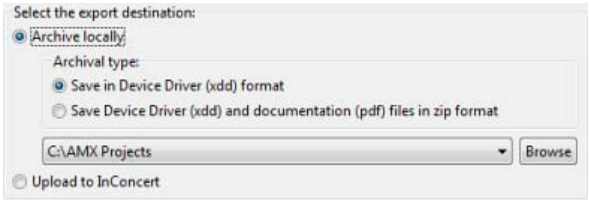


FIG. 82 Export Destination - Archive Locally options

This selection enables the following additional options:

Archival Type: These options allow you to specify how to export the Device Driver:

- **Save in Device Driver (xdd) format**: With this option selected, the Device Driver only will be exported as an xdd file to the specified directory. Click the Browse button to locate and select a different target directory (in the Export Directory dialog); use the drop-down menu to select from previously used directories.
- **Save Device Driver (xdd) and documentation (pdf) in zip format**: With this option selected, the Device Driver xdd file, and the documentation.pdf (project report) file will both be exported as a ZIP file to the specified directory.

Select **Upload To InConcert** to export the project file to InConcert (FIG. 83):

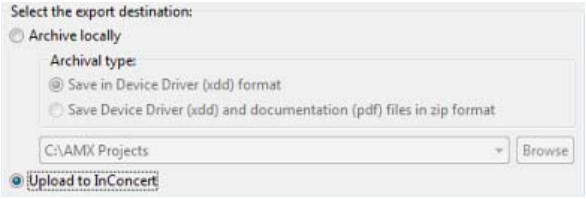


FIG. 83 Export Destination - Upload to InConcert

Only fully AMX Application Compatible Device Driver files are allowed to be uploaded to the AMX InConcert Resource Center. The application will alert you if the selected project is incompatible, and therefore cannot be uploaded to InConcert. In this case, check the commands/responses indicated in yellow in the *Application Compatibility Outline* (see page 9). Correct these items by going to the control tab and entering protocol for each one. Then save the Driver XML Editor to refresh the AMX Application Compatibility Outline to make sure the it is now compatible and export again.

3. Click **Finish** to export the file and close this dialog.
4. Depending on the export destination option selected, either the *Export To File System* dialog or the *Upload to InConcert* dialog indicates the results of the export (click **OK** to proceed).

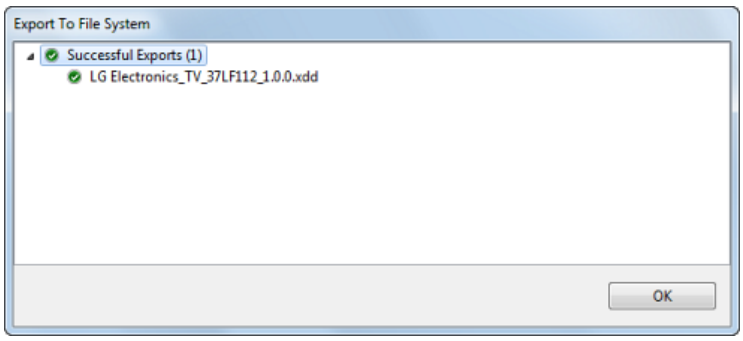


FIG. 84 Export To File System dialog

Note that when the project is exported, the project report file (*documentation.pdf*) is generated and added to the project folder in the Project Explorer.

Exporting Multiple Driver Design Projects

Driver Design supports exporting multiple Driver Design Projects via the Export Wizard:

1. Select **File > Export**, or select **Export** from the Project File Context Menu to open the first dialog in the *Export Wizard* (**Select**):

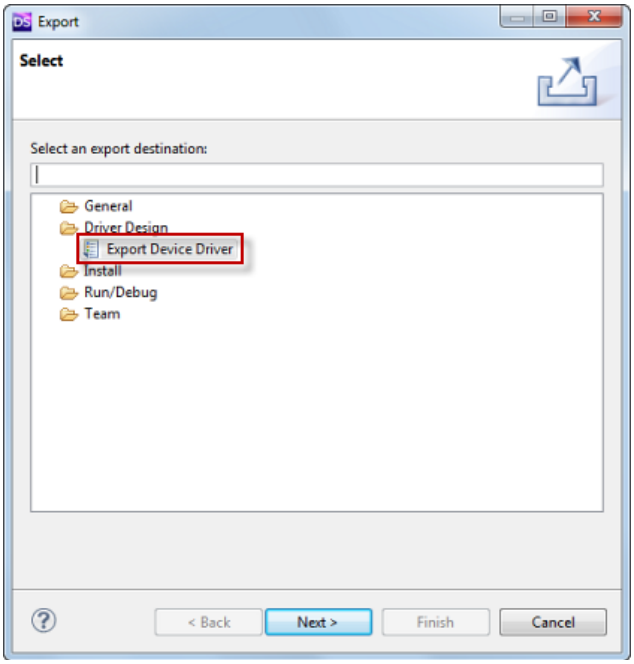


FIG. 85 Export To File System dialog

2. Select **Driver Design > Export Device Driver**.
3. Click **Next** to proceed to the *Export Device Drivers* dialog (FIG. 86):

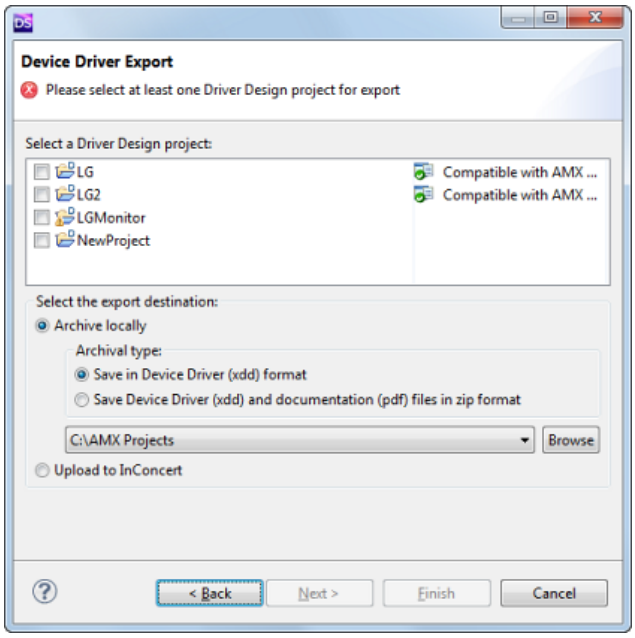


FIG. 86 Export Device Drivers dialog

4. Under *Select a Driver Design project:*, select the projects that you want to export (FIG. 87):

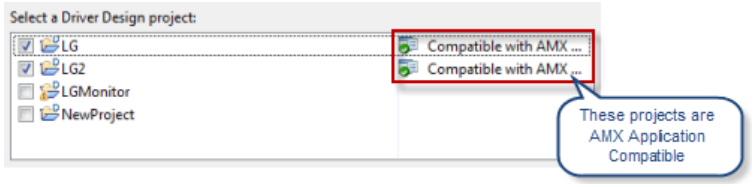


FIG. 87 Export Device Drivers dialog - Two Projects Selected for Export

5. Under *Select the export destination*, select to export the selected projects to a local directory, or to InConcert.
   Select **Archive locally** to export the project file to the specified directory (FIG. 82):

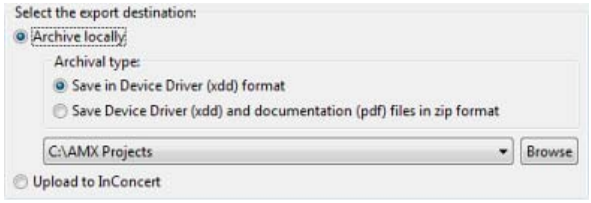


FIG. 88 Export Destination - Archive Locally options

This selection enables the following additional options:

Archival Type: These options allow you to specify how to export the Device Driver:

- **Save in Device Driver (xdd) format**: With this option selected, the Device Driver only will be exported as an xdd file to the specified directory. Click the Browse button to locate and select a different target directory (in the Export Directory dialog); use the drop-down menu to select from previously used directories.
- **Save Device Driver (xdd) and documentation (pdf) in zip format**: With this option selected, the Device Driver xdd file, and the documentation.pdf (project report) file will both be exported as a ZIP file to the specified directory.

Select **Upload To InConcert** to export the project file to InConcert (FIG. 83):

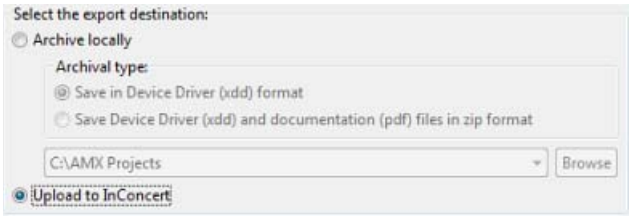


FIG. 89 Export Destination - Upload to InConcert

Only fully AMX Application Compatible xdd files are allowed to be uploaded to the AMX InConcert Resource Center. The application will alert you if the selected Project is incompatible, and therefore cannot be uploaded to InConcert. In this case, check the errors listed in the Application Compatibility Outline. Once the errors have been corrected and the file is indicated to be AMX Application Compatible, try to export again.

6. Click **Finish** to export the file and close this dialog.
7. Depending on the export destination option selected, either the *Export To File System* dialog or the *Upload to InConcert* dialog indicates the results of the export (click **OK** to proceed).

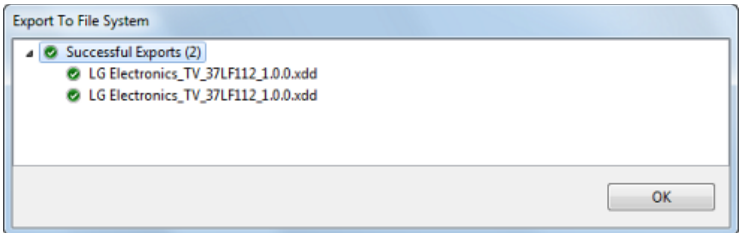


FIG. 90 Export To File System dialog (indicating two exported files)

Note that when multiple projects are exported, a project report file (documentation.pdf) is generated for each exported project and added to each project folder in the Project Explorer.

Uploading a Device Driver to the Master

1. In the *Device Personality* page of the Master's Web Console, click **Browse** to locate and select the Device Driver file to be uploaded for binding.
2. Once the file has been selected, the **Select a personality file to upload** text field will indicate something similar to the following:

```
C:\fakepath\<Device Driver filename>.xdd
```

an example is shown below (FIG. 91):

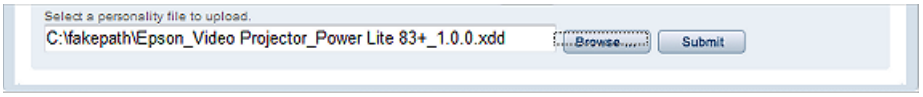


FIG. 91 NetLinx Master's WebConsole - Device Personality Page: Select a personality file to upload

3. Click **Submit** to upload the file to the master, and refresh the Device Personality page. The uploaded file will then appear in the list of personality files (under *Manage Personality Binding*):

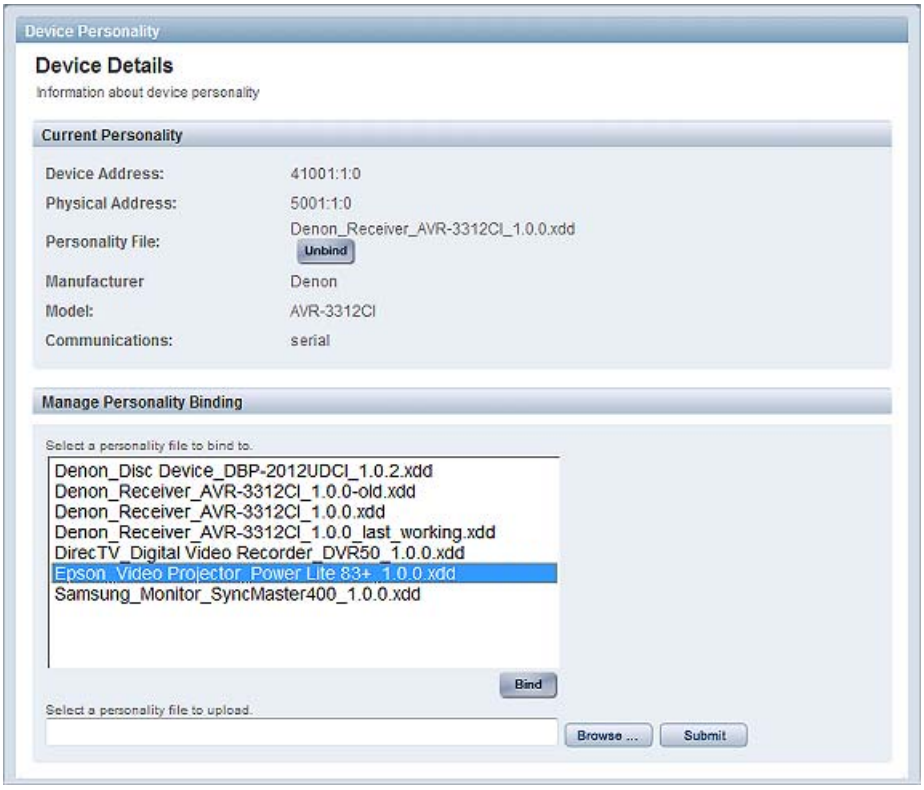


FIG. 92 NetLinx Master's WebConsole - Device Personality Page indicating uploaded personality file

Integrating a Device Driver in NetLinx Code

Overview

DeviceDriverEngine is a generic Duet module that allows you to integrate one or more Device Driver (*.xdd) files into working NetLinx code. This document describes the process of loading the Device Driver in NetLinx code (see page 53), and uploading a Device Driver to the NetLinx Master (see page 58).

Minimum System Requirements

Minimum Master Firmware Version

In order to utilize Device Drivers in your NetLinx Code, the target Master needs to be running NetLinx Master Firmware version **4.xx**.

- The latest version of Master Firmware can be downloaded from www.amx.com.
- To locate the correct file for your Master, search by product name or FG#, and click the appropriate link in the *Firmware Files* section (FIG. 93):

FIG. 93 www.amx.com - Sample product page with links to Firmware files for download.

Use the NetLinx Studio application (also available to download from www.amx.com) to determine the current firmware version on the Master, as well as to transfer firmware files to the Master. Refer to the NetLinx Studio online help for details.

NOTE: *Click here to go to the NetLinx Studio download page at www.amx.com.*

Minimum Duet Platform Runtime Version

In order to utilize Device Drivers in your NetLinx Code, the PC that runs the NetLinx Studio application must have the correct *Duet Platform Runtime* version installed.

The minimum version required is version **2.03**.

The Duet Platform Runtime files are installed with NetLinx Studio, and can be updated at any time via the *WebUpdate* application (also available to download from www.amx.com).

NOTE: *Click here to go to the WebUpdate download page at www.amx.com.*

Updating Duet Platform Runtime

1. Launch the WebUpdate application (**Programs > AMX Control Disc > WebUpdate**).
2. Click **Update Selections** (FIG. 94):

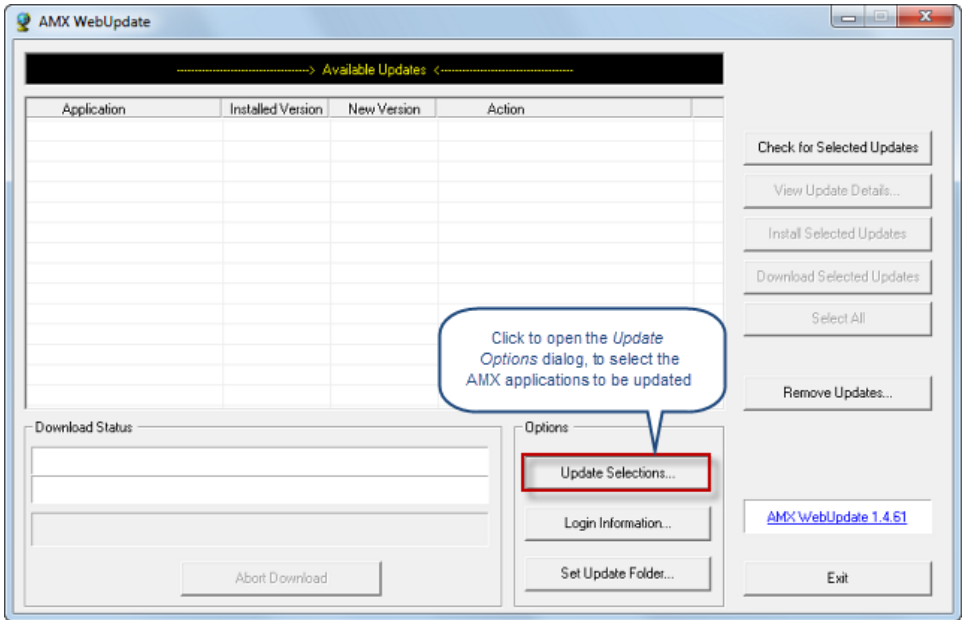


FIG. 94 WebUpdate

3. In the *Update Options* dialog, select **Duet Platform Runtime** (FIG. 95):

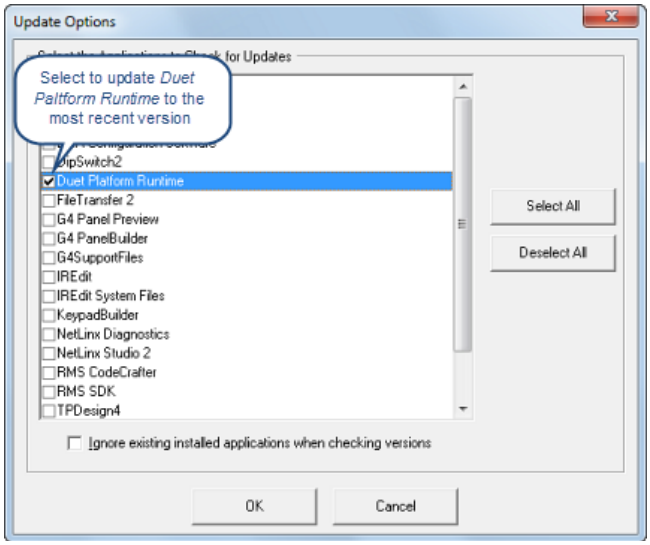


FIG. 95 WebUpdate - Update Options dialog

4. Click **OK** to close this dialog and return to the main work area. Note that *Duet Platform Runtime* is indicated in the update list (FIG. 96):

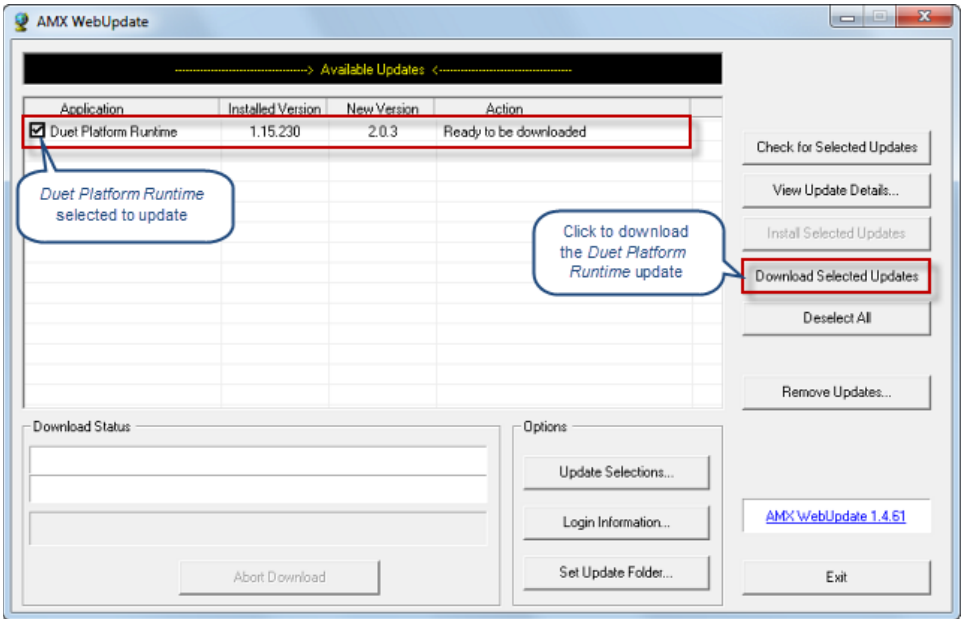


FIG. 96 WebUpdate - Duet Platform Runtime selected for update

5. Click **Download Selected Updates** to download the *Duet Platform Runtime* update. The download is indicated in the *Download Status* progress bars.
6. When the download is complete, click **Install Selected Updates** to install the *Duet Platform Runtime* update (FIG. 97):

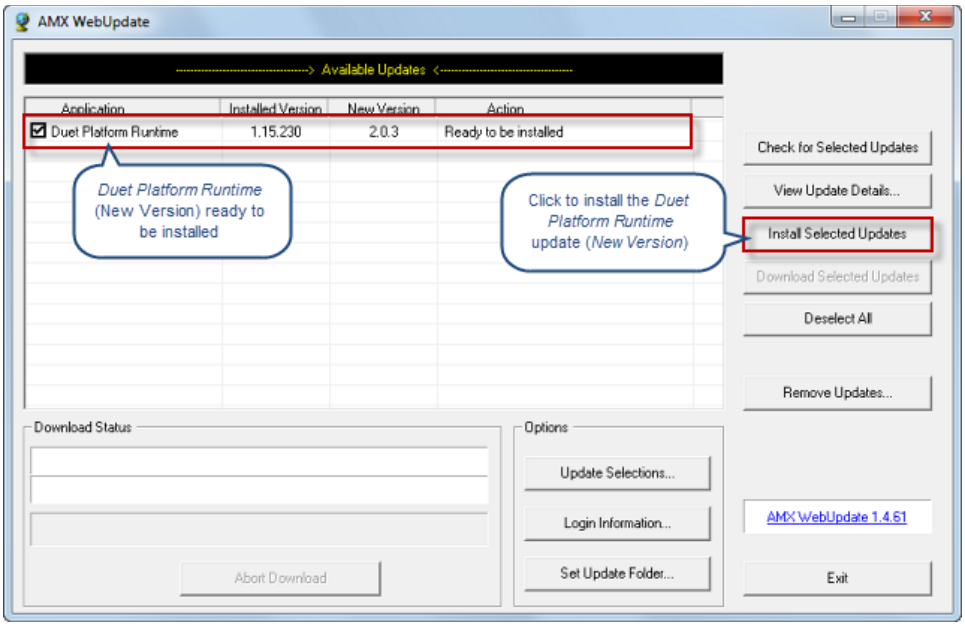


FIG. 97 WebUpdate - Duet Platform Runtime selected for installation

7. Follow the on-screen instructions to complete the installation. Note that the installation dialogs refer to "*Cafe Duet Runtime*" - this is just another name for the *Duet Platform Runtime*.
8. When prompted, click **Finish**.

Duet Memory Allocation

In order to utilize Device Drivers in your NetLinx Code, the target Master needs to have at least **15MB** of Duet Memory allocated. The Duet memory allocation setting can be viewed and adjusted via the telnet terminal commands described below:

DUET MEMORY Telnet Commands	
GET DUET MEMORY	Display the amount of memory allocated for Duet Java pool. This is the current Java memory heap size as measured in Megabytes (for example, "12" = 12MB).
SET DUET MEMORY	Set the amount of memory allocated for Duet Java pool. This is the current Java memory heap size as measured in Megabytes. This feature is used so that if a NetLinx program requires a certain size of memory be allotted for its currently used Duet Modules, it can be reserved on the Central Controller. The default Duet memory allocation value for NI Central Controllers with 64MB of SDRAM (as well as the DVX-2100HD) is **12MB** *Note: This setting does not take effect until the next reboot.*

Setting the Duet Memory Allocation Value

NOTE: *Terminal commands can be sent directly to the NI Controller or DVX-2100HD via either a Program Port or a Telnet terminal session. In your terminal program, type "Help" or a question mark ("?") and <Enter> to access the Help Menu, and display the supported Program port commands. Refer to the "NetLinx Integrated Controllers WebConsole and Programming Guide" for a full listing of supported telnet terminal commands.*

1. Telnet into the Controller (refer to the relevant *Operation/Reference Guide* for details).
2. Type `SET DUET MEMORY`. You will be presented with the current duet memory allocation value and a prompt for the new setting.
3. Enter the new setting (such as 15 to set the Duet memory allocation to 15MB), then press ENTER.
4. Reboot the master and test your code.

Repeat if necessary.

NOTE: *For additional information on Duet Memory Allocation, refer to the NetLinx Integrated Controllers WebConsole & Programming Guide.*

Downloading Driver Modules from InConcert

The InConcert Database is an online repository of AMX and Manufacturer device modules, drivers and IR files. Use the InConcert Database at www.amx.com to search and download Driver Modules.

NOTE: *Click here go to the InConcert Database page at www.amx.com.*

1. Log in to www.amx.com and click **Partners** in the navigation bar (FIG. 98):

FIG. 98 www.amx.com - Navigation Bar

2. In the *AMX Device Discovery Partners* page, click **Search Devices**, under *InConcert Resource Center* (FIG. 99):

FIG. 99 www.amx.com - AMX Device Discovery Partners page (InConcert Resource Center > Search Devices)

3. In the *Search InConcert Database* page, fill in the fields to perform a search for a device (FIG. 100):

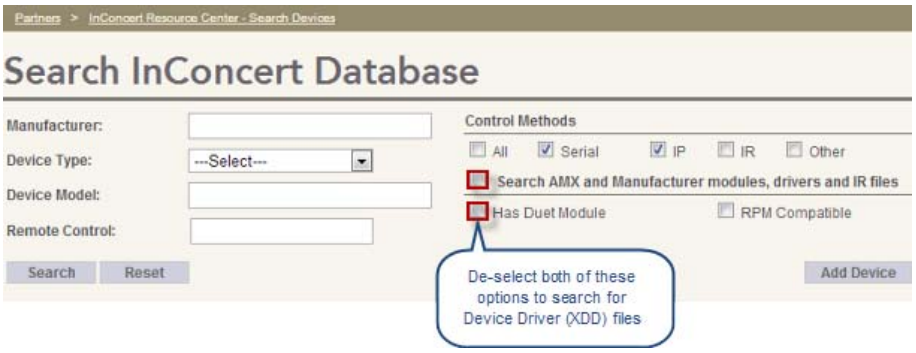


FIG. 100 www.amx.com - Search InConcert Database page

4. De-select the **Search AMX and Manufacturer modules, drivers and IR files** and the **Has Duet Module** option to filter the results to return only Device Driver (XDD) files.
5. Click **Search**. The search results are displayed on the page (FIG. 101):

FIG. 101 Search InConcert Database page - Search results

6. Click the link in the **Model** column to open the *Device Model Details* window for the selected device model. This window presents the files that available to download (FIG. 102):

FIG. 102 Device Model Details window - File(s) Available To Download

7. Click the File icon to download the XDD file to view a listing of all XDD versions available for the device, in the *Available Device Drivers* section (FIG. 103):

FIG. 103 Device Model Details window - Available Device Drivers

8. Click on the **DD** icon to download the XDD file.

Loading the Device Driver in NetLinx Code

Overview

The DeviceDriverEngine module specifies the Device Driver(s) that will be loaded in the NetLinx code. The method of loading your Device Driver(s) in NetLinx code depends on whether the NetLinx code utilizes *static* device binding or *dynamic* device binding.

The basic difference between the two device binding methods is that static device binding requires that the .xdd file is specifically named in the DEFINE_VARIABLE and DEFINE_START portions of the NetLinx code. Conversely, dynamic device binding does not reference the Device Driver by filename in the NetLinx code. Instead, use the Master's WebConsole to bind the Device Driver.

NOTE: *Either way, the steps described in this document require that your Device Driver (*.xdd) has been successfully exported to a local directory that you can access (in order to transfer the Device Driver to the Master). The Device Driver will reside in the directory specified in the Export Device Drivers dialog, when the file was exported to a local directory.*

Static Device Binding

These instructions assume that your NetLinx code utilizes static device binding. Use the NetLinx Studio application to modify, compile and transfer NetLinx code to the NetLinx Master. Refer to the NetLinx Studio on-line help for details.

1. Transfer the Device Driver (*.xdd) file to the NetLinx Master ("drivers/" directory), via FTP:
   a. In Windows Explorer, locate and select the Device Driver (*.xdd) file that you want to transfer.
   b. Select **Edit > Copy**.
   c. In the address bar, enter "**FTP://<IP Address of the Master>**" and press *Enter*.
   d. The Master will require you to login - enter a valid **Username** (default = *administrator*) and **Password** (default = *password*).
   e. Paste the Device Driver file into the "**drivers/**" directory.

   NOTE: *See the Transferring the Device Driver to the Master via NetLinx Studio* section on page 58 *for additional details.*
2. In NetLinx Studio, modify the DEFINE_VARIABLE and DEFINE_START sections of the main NetLinx code:
   a. Under DEFINE_VARIABLE, enter the name of the Device Driver file (*.xdd) using the CHAR[] parameter to specify the name of the Device Driver (*.xdd) file. The syntax is shown below:

   ```
   DEFINE_VARIABLE
   CHAR <variableName>[] = 'name of the Device Driver xdd file'
   ```

   b. Under DEFINE_START, use DEFINE_MODULE to load the 'DeviceDriverEngine' module. The parameters for the DeviceDriverEngine module will define the device and the device driver to use. The syntax is shown below:

   ```
   DEFINE_START
   DEFINE_MODULE 'DeviceDriverEngine' InstanceName(<parameter list>)
   ```

   Where:

   - **InstanceName**: the name to assign to the instance of the module.
   - **<parameter list>**: the list of parameters that is available to the DeviceDriverEngine module. For statically bound devices, the parameters include device information for the device that will use this Device Driver (*vdvDevice*, *dvDevice*), as well as the name of the Device Driver file to be used.

     The example below shows a Device Driver named "**Hitachi_Video_Projector_CPWX3014WN_1.0.0.xdd**" loaded:

   ```
   PROGRAM_NAME='DeviceDriverTest'

   DEFINE_DEVICE

   dvSerial1 = 5001:1:0
   dvSerial2 = 5001:2:0
   vdvDevice1 = 41001:1:0
   vdvDevice2 = 41002:1:0

   DEFINE_VARIABLE
   CHAR myDriver1[] = 'Hitachi_Video_Projector_CPWX3014WN_1.0.0.xdd'
   CHAR myDriver2[] = ''

   DEFINE_START

   // This is an example of loading a static device driver. If using NetLinx
   // Studio version 3.3.525 (or earlier), the device driver file has to be
   // manually transferred to the NetLinx system.
   DEFINE_MODULE 'DeviceDriverEngine' staticDev(vdvDevice1, dvSerial1, myDriver1)

   DEFINE_PROGRAM
   ```

3. Save and compile the NetLinx code.
4. Transfer the updated code to the NetLinx Master.
5. Open the Master's built-in WebConsole to the **System Page/Manage Devices** tab (FIG. 104):

FIG. 104 WebConsole - System Page/Manage Devices tab

Under **Device Configuration Pages**, the Device Driver file that was uploaded to the Master's "drivers" directory (via FTP - see above) is indicated. This entry includes relevant information for the statically bound device (manufacturer and model number | virtual device address | real device address | bound XDD file).

6. Click on the **Driver Design** entry to open the *Device Personality* page of the Master's WebConsole. The information under *Current Personality* reflects the properties of the bound Device Driver. (FIG. 105):

FIG. 105 WebConsole - Device Personality page

Dynamic Device Binding

These instructions assume that your NetLinx code utilizes dynamic device binding. Use the NetLinx Studio application to modify, compile and transfer NetLinx code to the NetLinx Master. Refer to the NetLinx Studio on-line help for details.

1. In NetLinx Studio, modify the `DEFINE_VARIABLE` and `DEFINE_START` sections of the main NetLinx code:
   a. Under `DEFINE_VARIABLE`, enter an empty `CHAR[]` parameter:

```
DEFINE_VARIABLE
CHAR <variableName>[] = ''
```

   b. Under `DEFINE_START`, use `DEFINE_MODULE` to load the 'DeviceDriverEngine' module. The parameters for the DeviceDriverEngine module will define the device and the device driver to use. The syntax is shown below:

```
DEFINE_START
DEFINE_MODULE 'DeviceDriverEngine' InstanceName(<parameter list>)
```

   Where:

   - **InstanceName:** the name to assign to the instance of the module.
   - **<parameter list>**: the list of parameters that is available to the DeviceDriverEngine module.

For dynamically bound devices, the parameters include device information for the device that will use this Device Driver (*vdvDevice*, *dvDevice*), as well as the name of the Device Driver file to be used.

Example:

```
PROGRAM_NAME='DeviceDriverTest'

DEFINE_DEVICE

dvSerial1 = 5001:1:0
dvSerial2 = 5001:2:0
vdvDevice1 = 41001:1:0
vdvDevice2 = 41002:1:0

DEFINE_VARIABLE
CHAR myDriver1[] = 'Hitachi_Video_Projector_CPWX3014WN_1.0.0.xdd'
CHAR myDriver2[] = ''

DEFINE_START

// This is an example of loading a dynamic device driver. Use the NetLinx
// master's web pages to transfer and bind the device driver to the
// DeviceDriverEngine module.
DEFINE_MODULE 'DeviceDriverEngine' dynamicDev(vdvDevice2, dvSerial2, myDriver2)

DEFINE_PROGRAM
```

2. Save and compile the NetLinx code.
3. Transfer the updated code to the NetLinx Master.
4. Open the Master's built-in WebConsole to the **System Page/Manage Devices** tab (FIG. 106):

FIG. 106 WebConsole - System Page/Manage Devices tab

Under *Device Configuration Pages*, see the entry named "**Unknown**". For dynamically bound devices, the entry under *Device Configuration Pages* does not indicate any information for a bound device (because initially there is no device bound). However, the entry does indicate the virtual device address and real device address that were defined in the NetLinx code.

5. Click on the Driver Design entry that is to be dynamically bound. This opens the *Device Personality* page of the Master's WebConsole (FIG. 107):

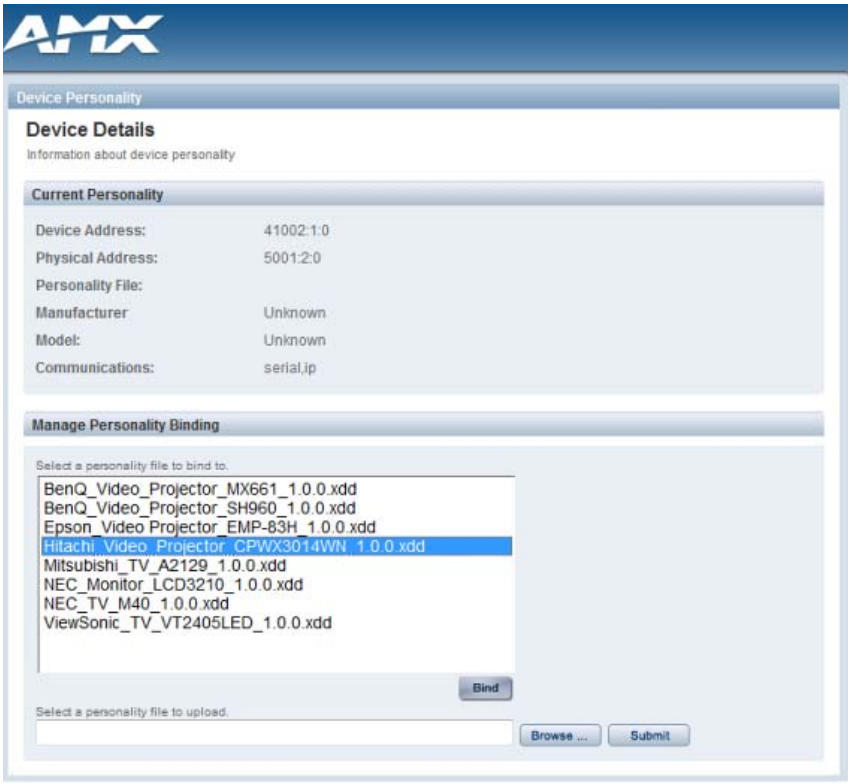


FIG. 107 WebConsole - Device Details page (no file bound)

Note that on this page:

- Because no file is yet bound, Personality File is blank, and the Manufacturer and Model fields indicate "*Unknown*".
- Because a file is not yet bound, the *Communications* information indicates "**serial,ip**", because both options are available to the user.
- The *Manage Personality Binding* section contains a listing of all personality files (including Device Driver *xdd files) that are currently stored in the "**drivers/**" directory of the NetLinx Master. Any of these files can be selected to be bound to the device.

6. Under *Select a personality file to bind to*, specify the Device Driver (*.xdd) file that you want to upload to the NetLinx Master (in the **/user/drivers/ directory**):
   - If the desired *.xdd file is present in the file list, select it and press **Bind**.
   - If the desired *.xdd file is not in the list, upload the file to the Master, and then select it from the list and click Bind. See the *Uploading a Device Driver to the Master* section on page 58 for details.
7. After the selected file is bound, the *Device Details* page is refreshed to display the information for the bound file, in the *Current Personality* section (FIG. 108):

FIG. 108 WebConsole - Device Personality page (indicated bound file)

Device Driver Module Servlet Pages

Servlet Link

Each running Device Driver Engine module will register a link to its servlet page with the master's web server. These links will be placed on the *System* > *Manage Devices* > *Device Options* page (on the master's WebConsole), grouped together with other Device Driver Engine modules.

The link for each Device Driver Engine module will display the device manufacturer and model, the virtual NetLinx address (D:P:S) used to communicate with the module, the physical NetLinx address used to communicate with the actual device, and the personality file currently being used (if applicable). Clicking the device link will open a new browser window/tab with the servlet page for the device.

Static-Bound Devices

For Device Driver Engine modules that have their device driver file specified in NetLinx code via the third parameter of the DEFINE_MODULE statement, the servlet page is informational only, presenting a table with the virtual and physical NetLinx addresses of the device, the name of the device driver file, the manufacturer and model, and the communications type (serial, IP, etc.).

Dynamically Bound Devices

For Device Driver Engine modules that do not have their device driver file specified in NetLinx code (that is, an empty third parameter of the DEFINE_MODULE statement) the servlet page allows for various binding operations in additional to displaying the device information mentioned above. The page provides the ability to browse for a device driver file on a local drive (or LAN), and upload it to the master.

The files will be uploaded into the **/user/drivers/** folder.

- If the upload overwrites a personality file currently in use by the device, a reboot is required. The system will prompt you to continue - the master will reboot automatically upon completion of the transfer. Upon completion of the reboot, the page will be refreshed.
- If the upload is a new personality file, the page will be automatically refreshed upon completion of the transfer.

The page presents a list of device driver files currently loaded on the master (in the /user/drivers/ folder), and provides the ability to select a personality file and bind it to the current device.

- If you select the personality file currently bound to the device, no action will occur.
- If you select a new device driver file when the device was previously not bound to another device driver file, the binding will occur and the page will be automatically refreshed upon completion of the action.
- If you select a new device driver file when the device was previously bound to another device driver file, the system will prompt you to reboot. In this case the master will reboot upon completion of the action. Upon completion of the reboot, the page will be automatically refreshed.

For devices which have been bound to a device driver file, the page will provide the ability to unbind the device, returning it to a state where it is not bound to any personality file. The page will be automatically refreshed upon completion of the action.

Uploading a Device Driver to the Master

Overview

The process of uploading a Device Driver (*.xdd) file to a NetLinx Master entails two steps: first, the *.xdd file must be copied into the "*drivers*" directory on the target Master, via the File Transfer functionality in NetLinx Studio (v3.4 or higher). This makes the file available for selection via the Master's WebConsole to upload, as described below.

Transferring the Device Driver to the Master via NetLinx Studio

NOTE: *XDD files are supported in NetLinx Studio v3.4 or higher. Download the latest version of NetLinx Studio, at http://www.amx.com/products/NetLinxStudio.asp.*

If the XDD File is Present in the Current Workspace

1. Launch NetLinx Studio (**All Programs > AMX Control Disc > NetLinx Studio**).
2. Select **Tools > File Transfer** to open the *File Transfer* dialog.
3. Click **Quick Load** to open the *Quick Load* dialog, and select **XDD Files** in the *Selection Options* (FIG. 109):

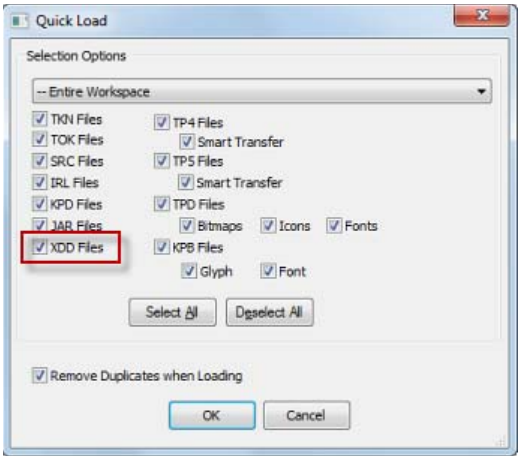


FIG. 109 Quick Load dialog with XDD Files selected

4. Click **OK** to return to the *File Transfer* dialog - note that any XDD files that are in the Workspace are included in the files list in the *Send* tab.
5. Verify that the **Reboot** option is selected (FIG. 110):

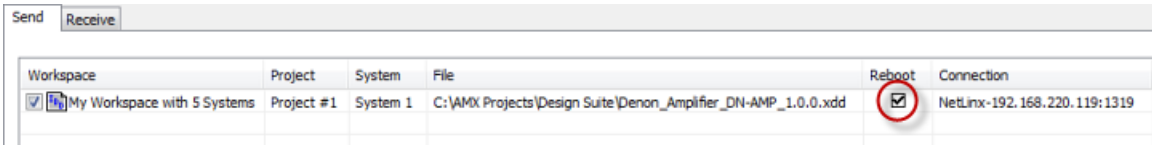


FIG. 110 File Transfer dialog (send tab) - Reboot option selected

6. Click **Send** to transfer the files listed in the *Send* tab.
7. The progress and final status of the file transfer is indicated in the *Output* bar (FIG. 111):

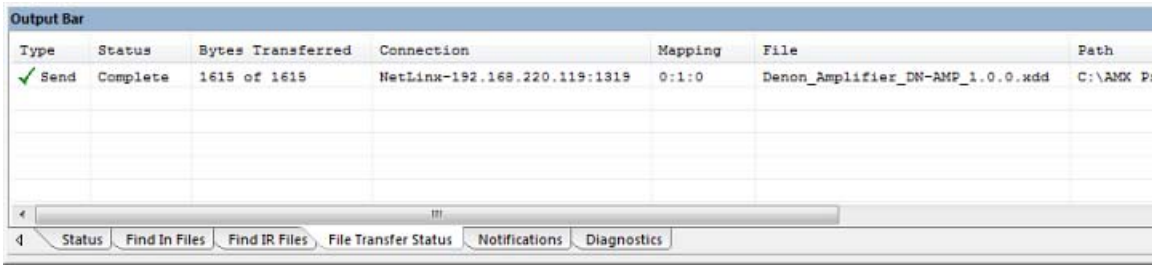


FIG. 111 Output Bar (File Transfer Status tab) - XDD File Transfer complete

If the XDD File is not Included in the Current Workspace

1. Launch NetLinx Studio (**All Programs > AMX Control Disc > NetLinx Studio**).
2. Select **Tools > File Transfer** to open the *File Transfer* dialog.
3. In the *File Transfer* dialog, click **Add** to open the *Select Files For File Transfer* dialog, and open the **Other** tab (FIG. 112):

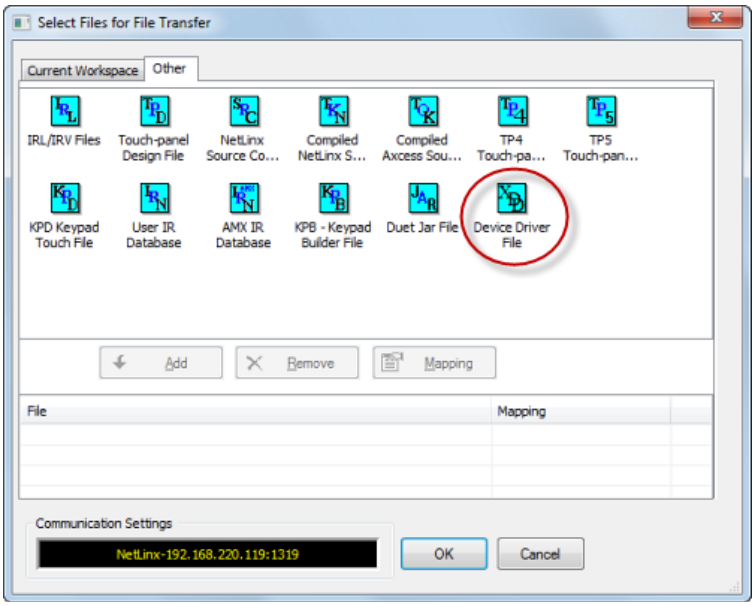


FIG. 112 Select Files for File Transfer dialog - Other tab

4. Double-click on **Device Driver File** to locate and select the XDD file in the *Open* dialog. Note that "**Device Driver Files (*.xdd)**" is pre-selected as the file type.
5. Select the Driver Design (*.XDD) file to transfer, and click **Open** to invoke the *Enter Device-Mapping Information* dialog (FIG. 113):

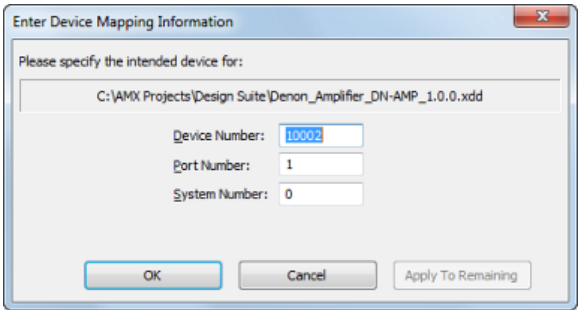


FIG. 113 Enter Device Mapping Information dialog

6. Enter the Device Mapping information for the XDD file and click **OK** to return to the *Select Files for File Transfer* dialog (*Other* tab). The selected XDD file is now indicated in the *Files to transfer* list (FIG. 114):

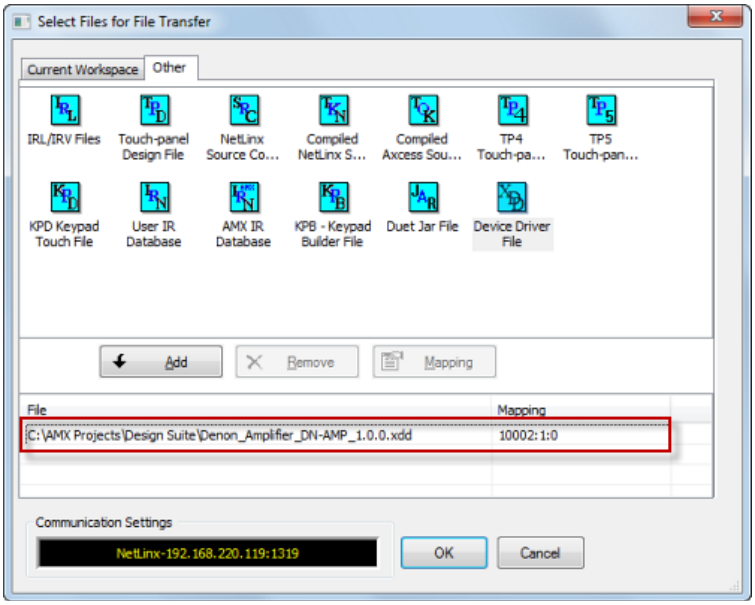


FIG. 114 Select Files for File Transfer dialog (Other tab) - XDD file indicated in the Files to Transfer list

7. Click **OK** to close the *Select Files for File Transfer* dialog and return to the *File Transfer dialog (Send tab)*. The selected XDD file is now indicated in the *Files to transfer* list (FIG. 115):

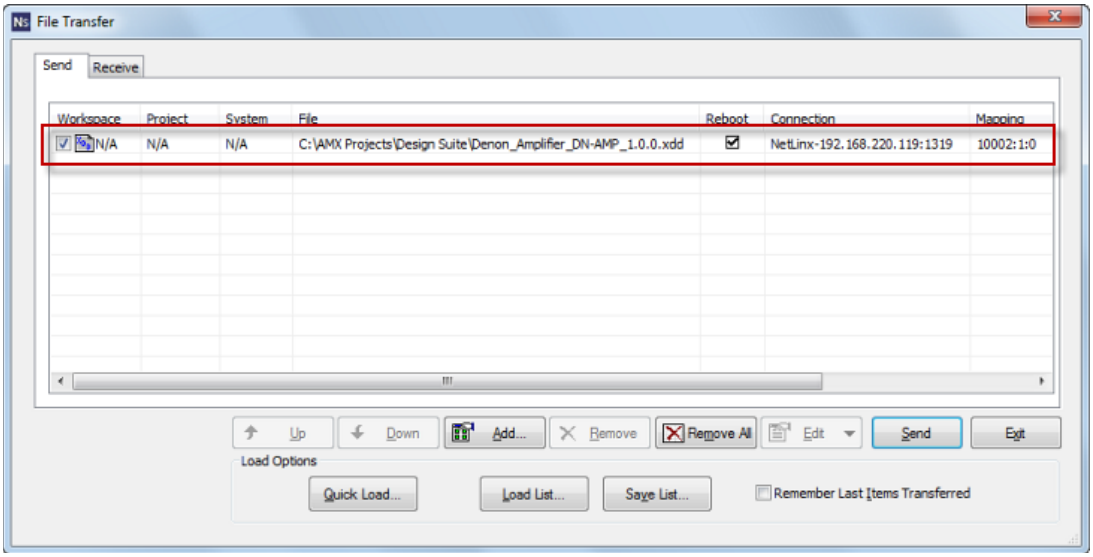


FIG. 115 File Transfer dialog (Send tab) - XDD file indicated in the Files to Transfer list

8. Click **Send** to begin the file transfer.
9. The progress and final status of the file transfer is indicated in the *Output* bar (FIG. 116):

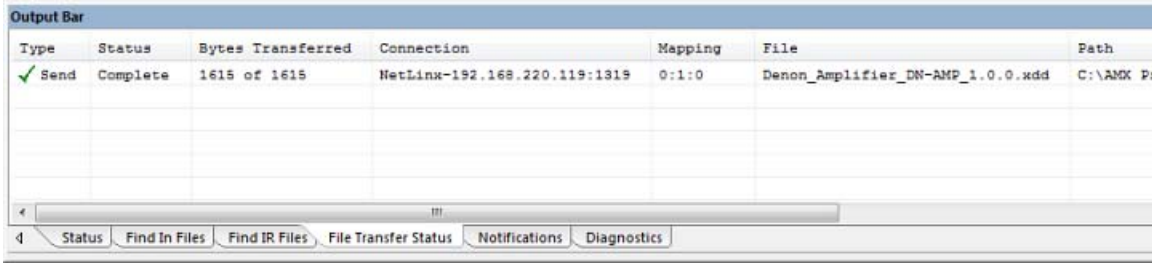


FIG. 116 Output Bar (File Transfer Status tab) - XDD File Transfer complete

Uploading a Device Driver via the Master's WebConsole

1. In the *Device Personality (Device Details)* page of the Master's WebConsole, click **Browse** to locate and select the Device Driver file to be uploaded for binding in the *Choose File To Upload* dialog (FIG. 117):

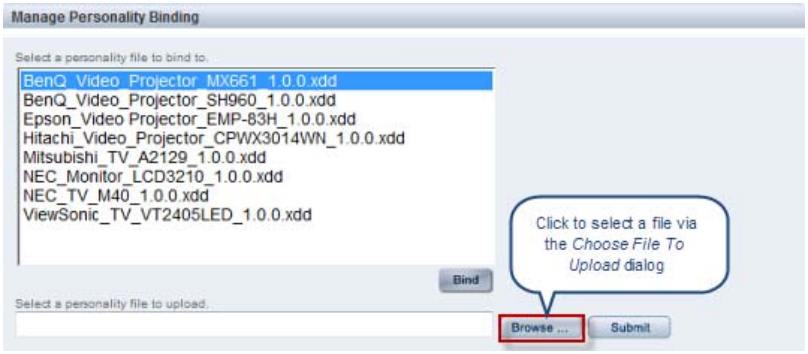


FIG. 117 Device Personality Page - Device Details (Manage Personality Binding section)

2. Select an *.xdd file and click **Open** to close the *Choose File To Upload* dialog. The selected file is indicated in the *Select a personality file to upload* field (FIG. 118):

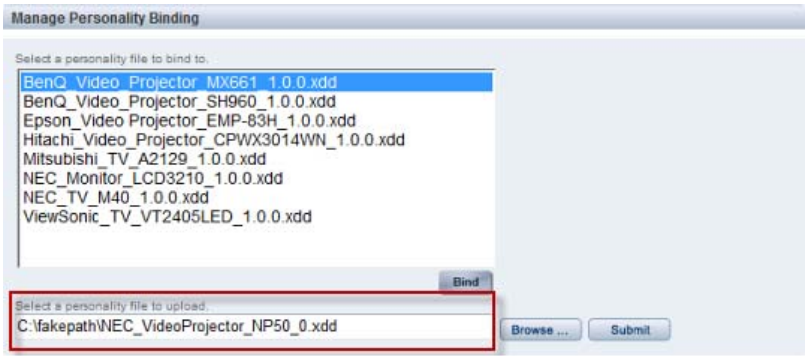


FIG. 118 Device Personality Page - Device Details (fakepath)

This field will indicate something similar to the following:

```
C:\fakepath\<Device Driver filename>.xdd
```

3. Click **Submit** to upload the file to the master, and refresh the *Device Personality* page. The uploaded file will then appear in the list of personality files, under Manage Personality Binding (FIG. 119):

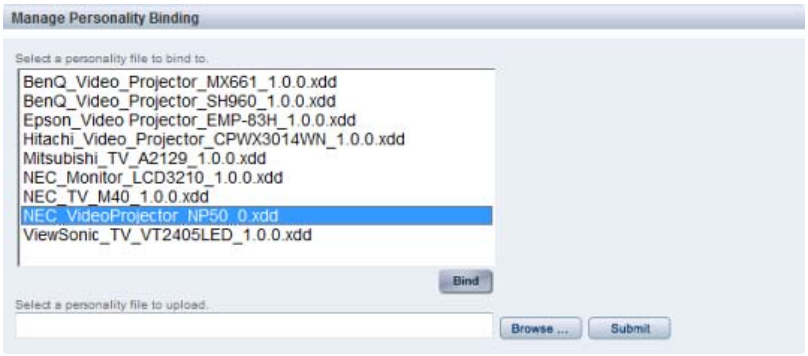


FIG. 119 Dynamic Binding - Device Details (xdd file submitted)

Appendix

Updating Driver Design

Updating Driver Design via "Check For Updates"

1. Select **Help > Check For Updates**.
2. In the *Available Updates* dialog, any available software updates are listed in the top window, indicating basic information (Name, Version and ID):
   - Click on an item in the list to populate the *Details* window with more detailed information.
   - Click *More* to open the *Properties* dialog to view additional details including Copyright information and the AMX License Agreement information for the selected item.
3. Check the updates that you wish to install in the list of available updates.
4. Click **Next** to proceed to the *Update Details* dialog, to review and confirm the selected update(s).
5. Click **Next** to proceed to the *Review Licenses* dialog. Review and accept the license, and click **Finish** to begin installing update(s).
6. The program will prompt you to restart AMX Design Suite (select **Restart Now**).

NOTE: *AMX Design Suite can be restarted at any time via the File > Restart option.*

Updating Plug-ins via Automatic Update

The *Automatic Update* feature in AMX Design Suite automatically alerts you when new updates for installed plug-ins are available to install.

By default, when this prompt appears the available updates have already been downloaded, and only need to be installed. The default actions for automatic updates can be changed via the Preferences dialog (see the AMX Design Suite User Guide help for details).

1. Click inside the popup to review the available updates, in the *Available Updates* dialog.
   - By default, all available updates are selected for installation.
   - Click on any update in the list to view a brief description of the update in the *Details* window.
2. Click **Next** to proceed to the *Update Details* dialog. This dialog indicates the updates that are selected for installation.
3. Review this list and click **Next** to proceed to the *Review Licenses* dialog.
4. Click *I accept the terms of the license agreement* to enable the *Finish* button.
5. Click **Finish** to install the selected updates.
6. The progress of the installation is indicated in the *Updating Software* dialog.
7. When the installation is complete, the program will prompt you to restart AMX Design Suite.
8. Click **Restart Now** to restart AMX Design Suite with the updated software installed.

Restarting AMX Design Suite

There may be times when you will need to restart AMX Design Suite:

- If you install new plug-ins when AMX Design Suite is active/open, the new plug in functionality is not activated until you restart AMX Design Suite.
- If you run AMX Design Suite for several days without a restart, it can become sluggish. A quick look at the memory occupied by AMX Design Suite can serve as an indication to restart AMX Design Suite.

To restart AMX Design Suite: select **File > Restart**. This action closes the application and restarts it with the same Perspective, Projects and Files that were open when the application closed.

AMX Character Class Syntax Redefinition from perl 5.6

The following table lists the AMX-specific redefinitions of perl 5.6 character class definitions.

AMX Character Class Syntax Redefinition from perl 5.6		
Character Class	**perl 5.6 Equivalent**	**Description**
\a	[A-Z,a-z]	Match any letter
\A	[^A-Z,a-z]	Match any non-letter
\c	[\x00-\x1f\x7f]	Match any control character
\C	[^\x00-\x1f\x7f]	Match any non-control character
\l	[a-z]	Match any lower case letter
\p	[^A-Za-z0-9_]	Match any punctuation
\P	[A-Za-z0-9_]	Match any non-punctuation
\s	[\t\n\r\\x0b\\x0c]	Match any whitespace character
\S	[^\t\n\r\\x0b\\x0c]	Match any non-whitespace character
\u	[A-Z]	Match any upper case character
\x	[0-9A-Fa-f]	Match any hex digit *Note: hexadecimal constants as defined in the perl 5.6 regex syntax (e.g. \xaf) are also supported*
\X	[^0-9A-Fa-f]	Match any non-hex digit

AMX Macro Syntax

The following table lists the AMX macro definitions and the Perl 5.6 equivalent for each:

AMX Macro Syntax		
Macro	**perl 5.6 Equivalent**	**Description**
%b	(.)	A 1-byte little-endian binary value
%B	(.)	A 1-byte big-endian binary value
%d	(\d+)	A variable length ASCII decimal value
%D	([\x00-\x09]+)	A variable length big-endian BCD value
%x	((?:[0-9a-f])+)	Variable length lowercase ASCII hexadecimal value
%X	((?:[0-9A-F])+)	Variable length uppercase ASCII hexadecimal value
%nb	(.{n})	An n-byte little-endian binary value
%nB	(.{n})	An n-byte big-endian binary value
%nd	(\d{n})	A fixed length ASCII decimal value
%nD	([\x00-\x09]{n})	A fixed length big-endian BCD value
%nx	((?:[0-9a-f]){n})	Fixed length lowercase ASCII hexadecimal value (*n is an ASCII decimal*)
%nX	((?:[0-9A-F]){n})	Fixed length uppercase ASCII hexadecimal value (*n is an ASCII decimal*)

Supported Components

The following Device Components are supported by Driver Design:

- Custom
- Device
- Display
- Lens
- Menu
- Module
- Power
- Preset
- SourceSelect
- Tuner
- Volume

Supported Device Types & Device Composition

The following device types are supported by Driver Design:

Supported Device Types	
Device Type	**Standard Components**
Monitor	• Display • Menu • Power • SourceSelect • Volume
TV	• Display • Menu • Power • SourceSelect • Tuner • Volume
Video Projector	• Display • Lens • Menu • Power • Preset • SourceSelect • Volume

Base Implementation of Device Components

The following table describes the base implementation that is provided with specific device components. When a method from the *Base Method* column is implemented, the corresponding methods from the *Provides* column are automatically available for use:

Base Implementation of Device Components				
Component	**Base Method**		**Provides**	
	Java API	**Driver Design Syntax**	**Java API**	**Driver Design Syntax**
DisplayComponentInstance	setBrightness (int level)	Display.setBrightness	incrementBrightness() decrementBrightness()	Display.incrementBrightness Display.decrementBrightness
	setColor (int level)	Display.setColor	incrementColor() decrementColor()	Display.incrementColor Display.decrementColor
	setContrast (int level)	Display.setContrast	incrementContrast() decrementContrast()	Display.incrementContrast Display.decrementContrast
	setSharpness (int level)	Display.setSharpness	incrementSharpness() decrementSharpness()	Display.incrementSharpness Display.decrementSharpness
	setTint (int level)	Display.setTint	incrementTint() decrementTint()	Display.incrementTint Display.decrementTint
	setAspectRatio (String aspectRatio)	Display.setAspectRatio	cycleAspectRatio()	Display.cycleAspectRatio
	setFreezeOn (boolean state)	Display.setFreeze	cycleFreeze()	Display.cycleFreeze
	setPictureMuteOn (boolean b)	Display.setPictureMute	cyclePictureMute()	Display.cyclePictureMute
	setPIPOn (boolean state)	Display.setPIP	cyclePIP()	Display.cyclePIP
PowerComponentInstance	setPower (String powerState)	Power.setPower	cyclePower (String powerState)	Power.cyclePower
SourceSelectComponentInstance	setInputSelect (int index)	SourceSelect.setInput	cycleInputSelect()	SourceSelect.cycleInput
TunerComponentInstance	setBand (String band)	Tuner.setTunerBand	cycleBand()	Tuner.cycleTunerBand
VolumeComponentInstance	setVolumeMuteOn (boolean b)	Volume.setVolumeMute	cycleVolumeMute()	Volume.cycleVolumeMute

3000 RESEARCH DRIVE, RICHARDSON, TX 75082
AMX.com | 800.222.0193 | 469.624.8000 | +1.469.624.7400 | fax 469.624.7153

Last Revised:
8/19/2015