Freescale Semiconductor
Application Note

Document Number:
Rev. C, 1/2012

i.MX IOMUX Design Aid

(User's Guide for IOMux.exe)

by David DiCarlo
Multimedia Applications Division

Freescale Semiconductor, Inc.
Austin, TX

Package pin count limitations require each device in the i.MX family to have an I/O multiplexer (IOMUX) in order to facilitate the connection of internal peripheral signals to the external system. The IOMUX module allows up to 8 internal signals to be connected to a single I/O pad on a device.

In order to ease the assignment of internal signals to external device balls/pins, the IOMux design aid tool was developed. IOMux.exe can be used to make signal assignments for supported i.MX devices. The tool identifies and assists in the resolution conflicts as they arise in real-time.

The tool allows notes or comments for each signal to be added to the list of assignments. Design configurations can be saved for future use and the exported for use in schematics.

The tool can also be used to generate C code to configure the IOMUXC registers according to the users design. This alleviates software developers from the tedious task of comparing schematics to the i.MX Reference Manual to establish connections to peripherals and also serves as supplementary documentation of a system.

This document is intended to provide the user a guide to use of this system design aid.

Table of Contents

1 Introduction

Each device in the i.MX family has a wealth of on-chip peripherals to use in an application with many more signals available to connect to the outside world. Not all the signals of each on-chip peripheral needs to be connected in every application. Some peripherals are not even used in many applications.

The IOMUX module was designed to minimize a device's pin count to keep the package cost and complexity down by providing a configurable switch matrix between on-chip signals and the external system. Each ball (or pin) of an i.MX device may be connected to one of up to eight internal signals by the IOMUX.

The IOMux design aid was developed to address several challenges: One of the challenges in designing an i.MX device into an application is in the assignment of internal signals to external signals. Another challenge is verifying connectivity of devices on a newly designed and assembled printed circuit board. Lastly, conveying the hardware configuration to software developers such that the internal signals may be connected by the initialization code to the proper external traces without them having to reverse engineer from the schematic may pose a challenge as well.

In order to use the IOMux.exe application, the user is required to be using Microsoft Windows XP or newer with Microsoft's .NET Framework, version 4.0 or newer installed. The IOMux application currently supports the following devices in all the available package variations: i.MX6Dual/Quad.

The IOMux application allows the user to iteratively assign signals and resolve conflicts with other signals in real time. A system design may be saved for later use or modification, saved to plain or formatted text, or printed out onto paper.

The IOMux application can also be used to programmatically generate C code to configure the IOMUXC registers according to the board designer's mux selections. Armed with the board designer's IOMux design file, software developers can automatically generate the source code and header files necessary to route internal signals to the proper external peripheral connection.

2 Application Overview

Figure 1 shows a screen shot of the IOMux application window. In the figure, various areas are labeled. The following subsections describe each area and its features.

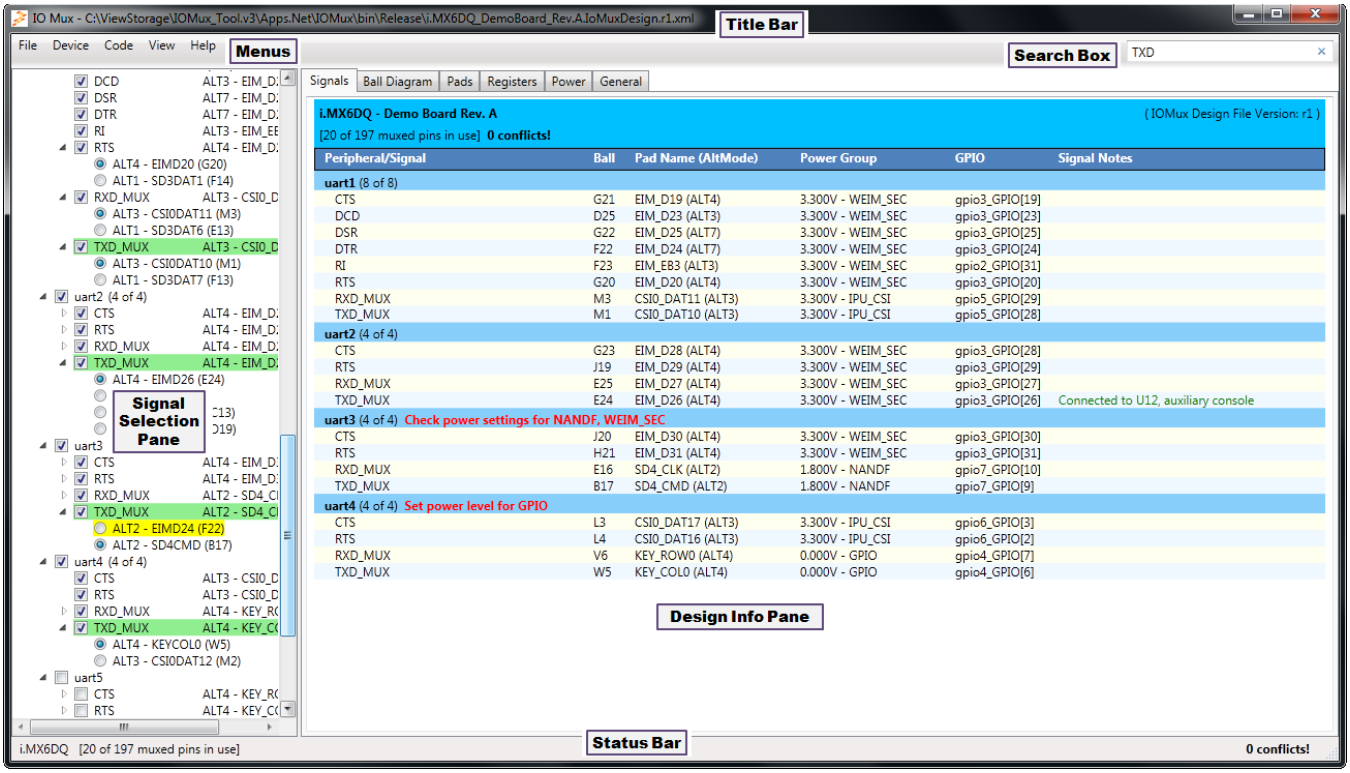

Figure 1. IOMux.exe application window overview

2.1 Menus

As with any normal graphical application, the IOMux application has a menu bar in the upper left-hand region of the application window. The menus are used to select the i.MX device for a design, open/save design files, and to change view settings.

2.2 Signal Selection Pane

The Signal Selection pane is where most of the action in the IOMux tool takes place. Once an i.MX device is selected, the Signal Selection pane is populated with all of the IOMUX options for that device and package combination.

By default, all signals are initially unassigned. Signals are assigned by clicking in the check box next to a peripheral group or, when expanded, an individual signal.

When a signal exits the i.MX device in more than one place, the ALT-mode/Ball combination desired may be selected by expanding (clicking the "+" next to the signal name). By default, the first ALT-

mode/Ball combination is the one selected. Choose a different assignment by clicking the radio button next to the ALT-mode/Ball combination.

Normally, the background color in is white. When a conflicting assignment is selected, all conflicting signals, ALT-mode/Ball assignments and corresponding lines in the Assigned Signals tab are highlighted in orange. Unassigned ALT-mode/Ball combinations for a signal that will conflict with other existing assignments are highlighted in yellow.

When the mouse is hovered over a signal name or an ALT-mode/Ball combination, a pop-up tooltip containing the list of all other internal signals that may be assigned to that ALT-mode/Ball combination appears. Any conflicting assignments are bolded in the list.

Right-clicking on a peripheral group or a signal brings up a contextual menu to toggle whether that peripheral or signal is excluded from the design. This may be used to signify that these signals have been considered for the design but for whatever reason have been used. This feature may also be used in order to prevent signals within a peripheral group from being assigned when the checkbox at the peripheral level is selected.

2.3 Search Box

The Search Box is available to perform a simple textual search for full or partial signal names or package ball/pin IDs. It is useful for quickly finding an item of interest. Note that the text search is not case sensitive nor is it a very intelligent search: it's just a basic string search. Search hits show up in the Signal Selection pane highlighted in green. Figure 1 shows the partial results of a search for the text "TXD".

2.4 Assigned Signals Tab

The Assigned Signals tab contains the list of all currently assigned signals in the current design. This tab is selected by default but it may be selected by clicking in the "Signals" tab if the alternate "Ball Diagram" tab is displayed. The columns descriptions are listed below:

Peripheral/Signal: This column contains the peripheral and signal names. The peripheral/module names and the related signals come from the IOMUXC chapter of the device's Reference Manual.

Ball: This column contains the package ball/pin assigned to the each signal.

Pad Name (AltMode): This column contains the package ball/pin name and ALT-mode assigned to the each signal.

Power Group: The Power Group column is provided to view the power rail associated with selected pad. The voltage level and power group are both listed separated by a hyphen. The power groups should be assigned using the Power tab of the Design Info Pane. The IOMux application checks for two power conditions for each peripheral. The warning, "Set power level for <PowerGroupName>" is displayed if a signal's power group voltage level is zero. The warning, "Check power settings for <PowerGroup1Name>, <PowerGroup2Name>" is displayed if the voltage level for all the signals for the peripheral are not equal. The warning message is shown in red on the peripheral line in the Design Info Pane. If there is no warning message on

the peripheral line, all power groups for the peripheral have been assigned equal, non-zero voltage levels.

GPIO: The GPIO column is provided as a handy reference for the hardware bring-up team for testing that basic connectivity where possible and without having to set up the peripherals intended for use in the application.

Signal Notes: The Signal Notes column is provided to the user as a means of annotating the specific signal assignments. Signal notes are added by right-clicking on an assignment row. Information about application usage, connection options on the board, or anything else the user feels relevant may be inserted in the text box that pops up. The user should consider placing info here that applies to others that review the schematics as well as the software developers. The output of this tool can be pasted into both the schematic files as well as the application source code.

Shared Signals: The Shared Signals column shows all of the other signals available to be muxed out on the assigned ball/pin. This information is the same as is available by hovering the mouse over an assignment row in the same pane. The Shared Signals column is shown by selecting View/Shared Pins from the application menu. In this mode, the Signal Notes are shown in the tooltip instead of the Shared Signals. The GPIO column is not visible since these signals are listed along with the other shared signals.

It should be noted that the columns GPIO + Signal Notes and Shared Signals are not simultaneously visible in the application. Switching between them is done in the View menu of the IOMux application.

2.5 Ball Diagram Tab

The Ball Diagram tab is provided to give a visual indication of where assigned signals enter/exit the device package. Each ball/pin is color coded according to a legend at the bottom of the pane (see Figure 19).

Hovering over a ball/pin brings a pop-up that contains a list of available internal signals that may be assigned to that ball/pin, for those package interconnects that the IOMUX controls. Signal(s) already routed to the ball/pin will be bolded in the tooltip list of possible signals.

2.6 Pads Tab

The Pads Diagram tab (shown below) is provided to give spreadsheet-like view of all the balls/pins on the device package. Each row represents a ball/pin and the column details include Ball Number, Pad Name, Power Group, Default Mode, and all of the implemented ALT-modes. Ascending/Descending sorting can be done on each column by clicking the column header. Column order can be changed by dragging the column header to another position.

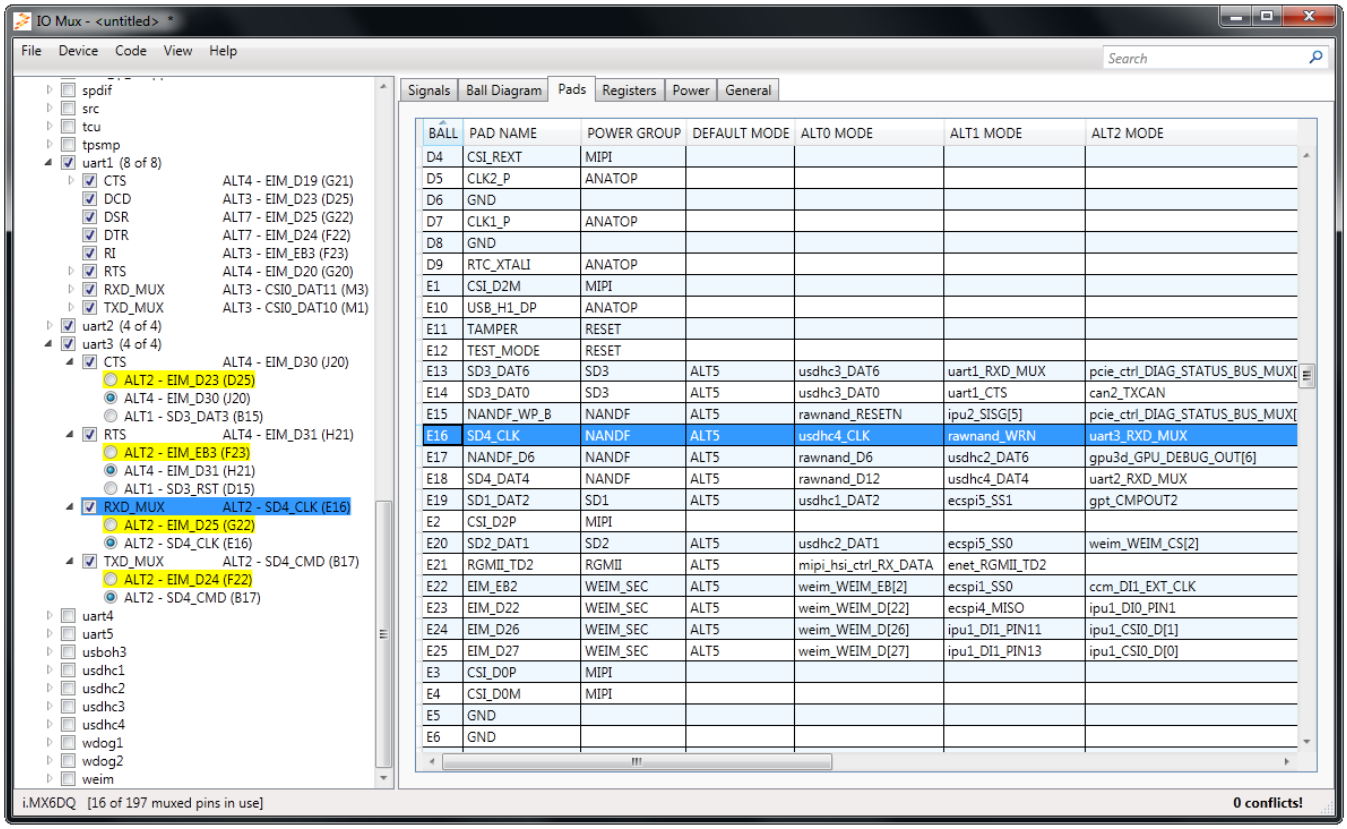

Figure 2. Pads Tab

2.7 Registers Tab

The Registers tab allows access to all of the IOMUXC registers associated with the selected signal in the Signal Selection Pane (see Figure 20).

2.8 Power Tab

The Power tab (see Figure 21) lists the voltage levels for each Power Group for the device. Setting the voltage levels for the power groups allows the IOMux application to ensure consistency across all of the signals for a given peripheral. Inconsistencies or zero-valued settings are reflected as warnings on the respective peripheral line in the Signals tab of the Design Info Pane.

Please refer to the device's Data Sheet for appropriate voltage settings for each of the listed power groups.

2.9 General Tab

The General tab allows the board designer to enter information describing the board. The *Board*, *Board revision*, and *Version* fields are used to construct a default file name for saving the design and are displayed on the Signals tab of the Design Info Pane. The *Version* field is intended to represent the version of the IOMux application design file (see Figure 23).

2.10 Title Bar

As with any Windows application, the Title bar contains the application name in the upper left corner. Additionally, the tool will also indicate the name of the design to the left of the application name, if it has been previously saved. If it has not yet been saved, this will be indicated by "<untitled>". An asterisk will be placed to the right of the design name to indicate that the design has pending changes that have not yet been saved.

2.11 Status Bar

The Status bar is at the very bottom of the application window. On the left, the device selected for the design is indicated. Next to it, a count of the assigned and total muxable signals available to the IOMUX for that design's device are given. On the right side of the bar, a count of the conflicting assignments is given.

3 Menus

This section provides details about each menu in the application.

3.1 File Menu

A partial screen shot of the File menu is shown in Figure 3.

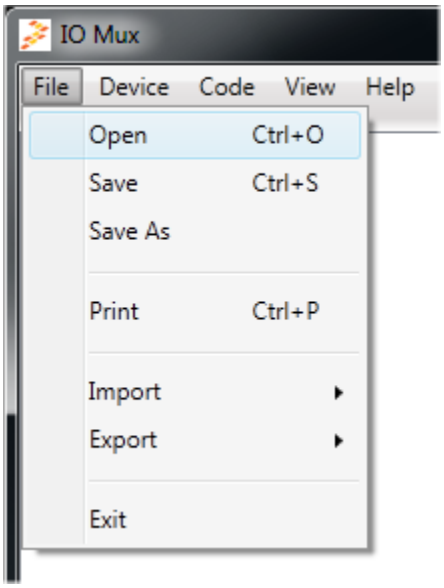

Figure 3. A partial screen shot of the File menu.

3.1.1 Open

The Open menu item is used to open a previously saved design file. The file format is a custom XML text file. The user should not edit or modify saved design files by hand because the application does not perform any error checking on input files.

3.1.2 Save

The Save menu item is used to save a design to the native XML text format.

If the design is new and has never been saved before, Windows save dialog box will pop up. A default filename is inserted into the name field of the dialog box, which the user may change. A warning will be given if the desired filename already exists and will get overwritten.

3.1.3 Save As

The Save As menu item is used to save the design to a plain text file or to a Rich Text Format (RTF) file. The plain text file is intended to be used for copying and pasting into an application's schematic or source code archive, or both. The RTF format file contains some formatting that makes the design more readable.

3.1.4 Print

The Print menu item will send the list of assigned signals in the Signal Selection tab, Ball Diagram tab, Power tab, and General tab to an attached Windows printer. If sent to a color printer, the colorization information in the tabs is retained. If a virtual printer that generates PDF files is used, such as Adobe's Distiller or PDF995, the design's assignments and ball diagram can be printed to a PDF file.

Note that the best results are obtained by setting the printer preferences to Landscape mode.

3.1.5 Import ->Import IOMux Tool v3.0 design file

This function is to facilitate moving a design from the initial IOMux Tool v3.0 release. The v3.0 release saved and restored designs correctly as long as signals were not moved or renamed. If signals we moved or renamed, v3.0 would not properly save the system design.

The design information that is saved in the current v3.1.2 is dramatically changed to accommodate moving and renaming signals. All signals are now saved to the design file, thus the designs are all in the 650 KB range regardless of how many signals are used in the design.

The v3.0 implementation would generate proper code for renamed/moved modules and signals unless there were conflicts in the system. A conflict is when 2 signals are routed to the same physical pad and are indicated in orange in the tool. IOMux Tool v3.1.2 produces correct code for conflicted signals.

For migration to the latest v3.1.2 tool please consider the following cases:

1. V3.0 system design had no conflicts and all of the native module and signal names were used.
    a. Simply import the v3.0 design file and use the File -> Save menu to save the design in the new format. Use the File -> SaveAs menu if you want do not want to overwrite the v3.0 design file.
2. V3.0 system design used all of the native module and signal names but contained some conflicts.
    a. Follow the same procedure listed in 1.a above, but verify that the register settings are correct for each of the conflicted signals and resave the design.

3. V3.0 system design employed signal regrouping and/or renaming.
    a. Generate code using the v3.0 tool and design.
    b. Import the v3.0 design file and use the File -> Save menu to save the design in the new format. Use the File -> SaveAs menu if you want do not want to overwrite the v3.0 design file.
    c. Generate code with the new v3.1.2 tool.
    d. Compare generated codebases and make changes in the v3.1.2 tool until the codebases align.
    e. Save the design again.

3.1.6 Export -> Export to .csv

The Export to .csv menu item produces a comma delimited file with the following columns:

Ball, Name, PowerGroup, Instance.Signal, Instance-Alias.Signal-Alias, Comment

The file can be manipulated in Microsoft Excel and is intended to provide a simple mapping between the internal i.MX signals and the external i.MX pins/balls and the function of those physical connections.

Please note that a design exported to this format cannot be imported or otherwise opened by the application. To save the design in a format that can be reopened, you must use the Save or SaveAs menu items.

3.1.7 Export -> Export to .rtf / Export to .txt

The Export to .txt and .rtf menu items are used to save the design to a plain text file or a Rich Text Format (RTF) file respectively. The plain text file is intended to be used for copying and pasting into an application's schematic or source code archive, or both. The RTF format file contains some formatting that makes the design more readable.

Please note that a design exported to either of these formats cannot be imported or otherwise opened by the application. To save the design in a format that can be reopened, you must use the Save or SaveAs menu items.

3.1.8 Exit

The Exit menu item is used to quit the application. If a design has been modified and has not yet been saved, a warning of such will be issued to the user.

3.2 Device Menu

A partial screen shot of the Device menu is shown in Figure 4.

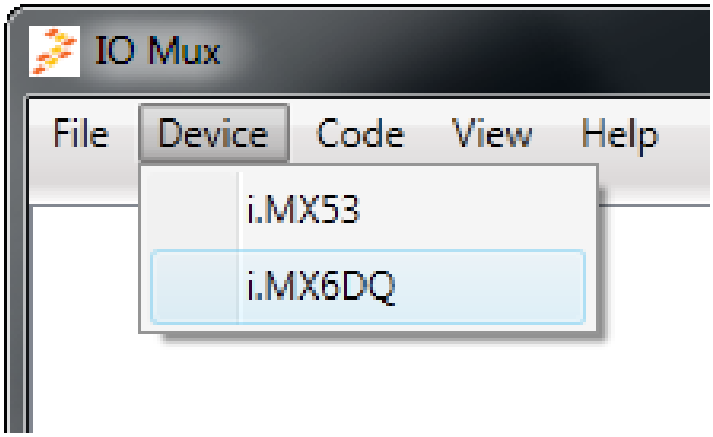

Figure 4. Partial screen shot of the Device menu.

The device to be used in a design is chosen through the Device menu. Only supported device and package options appear in the Device menu.

If the current design has unsaved changes and any item in the Device menu is selected, a warning will be issued to the user that there are unsaved changes.

Currently the only devices supported with version 3.1 of the IOMux application is the i.MX6DQ and the i.MX53.

3.3 Code Menu

The Code menu is used to select the coding style of the generated IOMUXC configuration code for the board design. By default, Full Comments is selected. Note that the Signal Notes are included in all coding styles.

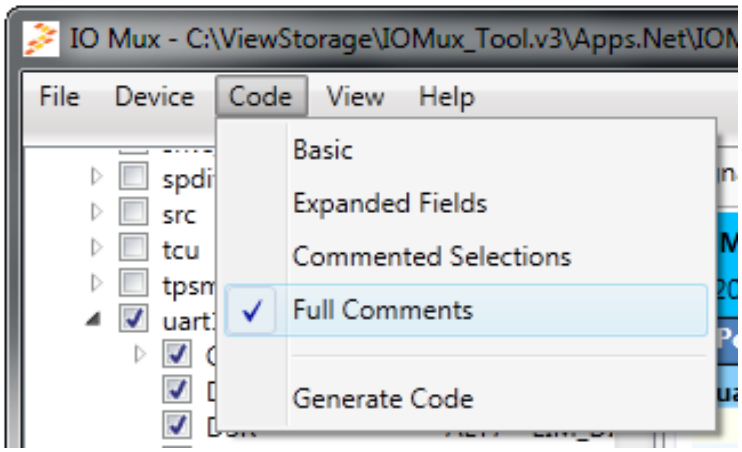

Figure 5. Partial screen shot of the View menu.

3.3.1 Basic

The most compact form of the <peripheral_name>_iomux_config() functions. This style is displayed on the Signals tab when the mouse is hovered over a peripheral line (see **Error! Reference source not found.**). The tooltip code can be copied to the clipboard by using the peripheral line context menu "Copy <peripheral_name> configuration code to clipboard."

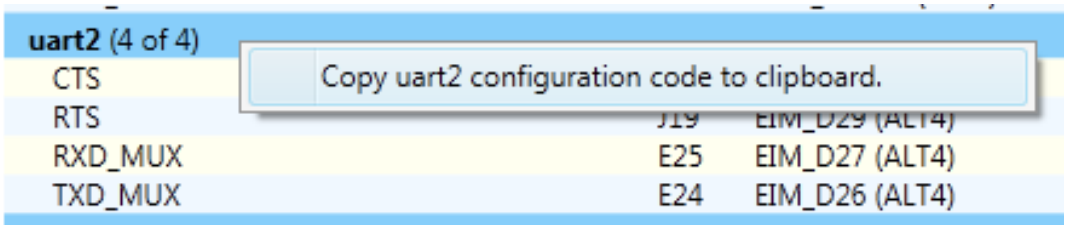

Figure 6. Copy code to clipboard context menu.

Below is an example of the Basic code style.

```
// Function to config iomux for instance uart2.
void uart2_iomux_config(void)
{
    // Config uart2_CTS to pad EIM_D28(G23)
    writel(0x00000004, IOMUXC_SW_MUX_CTL_PAD_EIM_D28); // SION_DISABLED, ALT4
    writel(0x00000000, IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT); // SEL_EIM_D28_ALT4
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D28);

    // Config uart2_RTS to pad EIM_D29(J19)
    writel(0x00000004, IOMUXC_SW_MUX_CTL_PAD_EIM_D29); // SION_DISABLED, ALT4
    writel(0x00000000, IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT); // SEL_EIM_D28_ALT4
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D29);

    // Config uart2_RXD_MUX to pad EIM_D27(E25)
    writel(0x00000004, IOMUXC_SW_MUX_CTL_PAD_EIM_D27); // SION_DISABLED, ALT4
    writel(0x00000000, IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT); // SEL_EIM_D26_ALT4
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D27);

    // Config uart2_TXD_MUX to pad EIM_D26(E24)
    // Connected to U12, auxiliary console
    writel(0x00000004, IOMUXC_SW_MUX_CTL_PAD_EIM_D26); // SION_DISABLED, ALT4
    writel(0x00000000, IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT); // SEL_EIM_D26_ALT4
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D26);
}
```

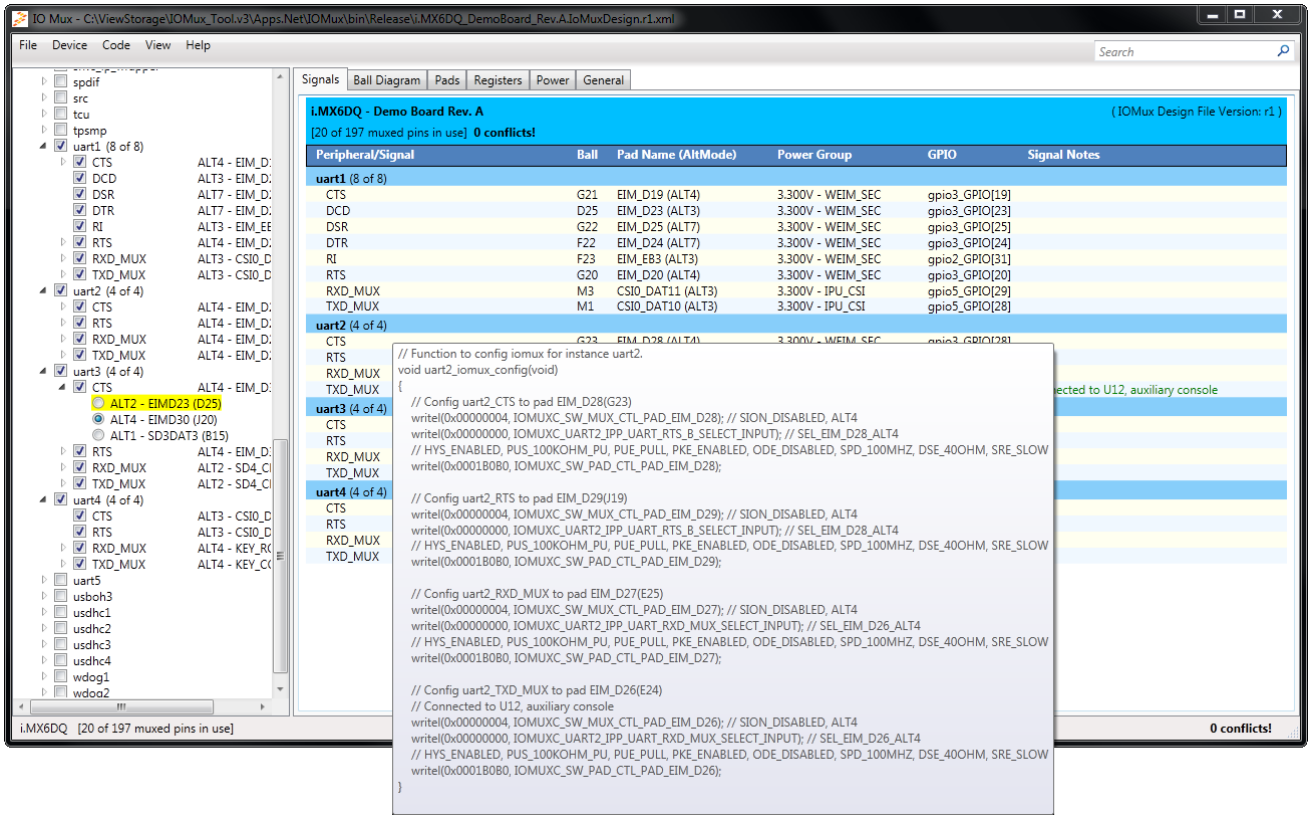

Figure 7. Basic code style as peripheral tooltip.

3.3.2 Expanded Fields

The register fields are displayed symbolically. Below is an example of the Expanded Fields code style.

```
// Function to config iomux for instance uart2.
void uart2_iomux_config(void)
{
    // Config uart2_CTS to pad EIM_D28(G23)
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D28);
    writel((SEL_EIM_D28_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT);
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D28);

    // Config uart2_RTS to pad EIM_D29(J19)
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D29);
    writel((SEL_EIM_D28_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT);
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D29);

    // Config uart2_RXD_MUX to pad EIM_D27(E25)
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D27);
    writel((SEL_EIM_D26_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT);
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D27);

    // Config uart2_TXD_MUX to pad EIM_D26(E24)
    // Connected to U12, auxiliary console
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D26);
    writel((SEL_EIM_D26_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT);
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D26);
}
```

3.3.3 Commented Selections

The descriptions of the selected field values are included in the comments. Below is an example of the Commented Selections code style.

```
// Function to config iomux for instance uart2.
void uart2_iomux_config(void)
{
    // Config uart2_CTS to pad EIM_D28(G23)
    // Mux Register:
    // IOMUXC_SW_MUX_CTL_PAD_EIM_D28(0x020E00C4)
    //   SION (4) - SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
    //   MUX_MODE (2-0) - ALT4 (4) - Select mux mode: ALT4 mux port: CTS of instance: uart2.
    //                              NOTE: - Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4.
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D28);
    // Pad EIM_D28 is involved in Daisy Chain.
    // Input Select Register:
    // IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT(0x020E0924)
    //   DAISY (2-0) - SEL_EIM_D28_ALT4 (0) - Selecting Pad: EIM_D28 for Mode: ALT4.
    writel((SEL_EIM_D28_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT);
    // Pad Control Register:
    // IOMUXC_SW_PAD_CTL_PAD_EIM_D28(0x020E03D8)
    //   HYS (16) - HYS_ENABLED (1) - Hysteresis Enabled
    //   PUS (15-14) - PUS_100KOHM_PU (2) - 100K Ohm Pull Up
    //   PUE (13) - PUE_PULL (1) - Pull
    //   PKE (12) - PKE_ENABLED (1) - Pull/Keeper Enabled
    //   ODE (11) - ODE_DISABLED (0) - Open Drain Disabled
    //   SPEED (7-6) - SPD_100MHZ (2) - Medium(100 MHz)
    //   DSE (5-3) - DSE_40OHM (6) - 40 Ohm
    //   SRE (0) - SRE_SLOW (0) - Slow Slew Rate
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D28);

    // Config uart2_RTS to pad EIM_D29(J19)
    // Mux Register:
    // IOMUXC_SW_MUX_CTL_PAD_EIM_D29(0x020E00C8)
    //   SION (4) - SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
    //   MUX_MODE (2-0) - ALT4 (4) - Select mux mode: ALT4 mux port: RTS of instance: uart2.
    //                              NOTE: - Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4.
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D29);
    // Pad EIM_D29 is involved in Daisy Chain.
    // Input Select Register:
    // IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT(0x020E0924)
    //   DAISY (2-0) - SEL_EIM_D28_ALT4 (0) - Selecting Pad: EIM_D28 for Mode: ALT4.
    writel((SEL_EIM_D28_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT);
    // Pad Control Register:
    // IOMUXC_SW_PAD_CTL_PAD_EIM_D29(0x020E03DC)
    //   HYS (16) - HYS_ENABLED (1) - Hysteresis Enabled
    //   PUS (15-14) - PUS_100KOHM_PU (2) - 100K Ohm Pull Up
    //   PUE (13) - PUE_PULL (1) - Pull
    //   PKE (12) - PKE_ENABLED (1) - Pull/Keeper Enabled
    //   ODE (11) - ODE_DISABLED (0) - Open Drain Disabled
    //   SPEED (7-6) - SPD_100MHZ (2) - Medium(100 MHz)
    //   DSE (5-3) - DSE_40OHM (6) - 40 Ohm
    //   SRE (0) - SRE_SLOW (0) - Slow Slew Rate
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D29);

    // Config uart2_RXD_MUX to pad EIM_D27(E25)
    // Mux Register:
    // IOMUXC_SW_MUX_CTL_PAD_EIM_D27(0x020E00C0)
    //   SION (4) - SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
    //   MUX_MODE (2-0) - ALT4 (4) - Select mux mode: ALT4 mux port: RXD_MUX of instance: uart2.
    //                              NOTE: - Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT4.
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D27);
    // Pad EIM_D27 is involved in Daisy Chain.
    // Input Select Register:
    // IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT(0x020E0928)
```

```
    //   DAISY (2-0) - SEL_EIM_D26_ALT4 (0) - Selecting Pad: EIM_D26 for Mode: ALT4.
    writel((SEL_EIM_D26_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT);
    // Pad Control Register:
    // IOMUXC_SW_PAD_CTL_PAD_EIM_D27(0x020E03D4)
    //   HYS (16) - HYS_ENABLED (1) - Hysteresis Enabled
    //   PUS (15-14) - PUS_100KOHM_PU (2) - 100K Ohm Pull Up
    //   PUE (13) - PUE_PULL (1) - Pull
    //   PKE (12) - PKE_ENABLED (1) - Pull/Keeper Enabled
    //   ODE (11) - ODE_DISABLED (0) - Open Drain Disabled
    //   SPEED (7-6) - SPD_100MHZ (2) - Medium(100 MHz)
    //   DSE (5-3) - DSE_40OHM (6) - 40 Ohm
    //   SRE (0) - SRE_SLOW (0) - Slow Slew Rate
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D27);

    // Config uart2_TXD_MUX to pad EIM_D26(E24)
    // Connected to U12, auxiliary console
    // Mux Register:
    // IOMUXC_SW_MUX_CTL_PAD_EIM_D26(0x020E00BC)
    //   SION (4) - SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
    //   MUX_MODE (2-0) - ALT4 (4) - Select mux mode: ALT4 mux port: TXD_MUX of instance: uart2.
    //                               NOTE: - Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT4.
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D26);
    // Pad EIM_D26 is involved in Daisy Chain.
    // Input Select Register:
    // IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT(0x020E0928)
    //   DAISY (2-0) - SEL_EIM_D26_ALT4 (0) - Selecting Pad: EIM_D26 for Mode: ALT4.
    writel((SEL_EIM_D26_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT);
    // Pad Control Register:
    // IOMUXC_SW_PAD_CTL_PAD_EIM_D26(0x020E03D0)
    //   HYS (16) - HYS_ENABLED (1) - Hysteresis Enabled
    //   PUS (15-14) - PUS_100KOHM_PU (2) - 100K Ohm Pull Up
    //   PUE (13) - PUE_PULL (1) - Pull
    //   PKE (12) - PKE_ENABLED (1) - Pull/Keeper Enabled
    //   ODE (11) - ODE_DISABLED (0) - Open Drain Disabled
    //   SPEED (7-6) - SPD_100MHZ (2) - Medium(100 MHz)
    //   DSE (5-3) - DSE_40OHM (6) - 40 Ohm
    //   SRE (0) - SRE_SLOW (0) - Slow Slew Rate
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D26);
}
```

3.3.4 Full Comments

The descriptions of all possible field values are included in the comments. Below is an example of the Full Comments code style.

```
// Function to config iomux for instance uart2.
void uart2_iomux_config(void)
{
    // Config uart2_CTS to pad EIM_D28(G23)
    // Mux Register:
    // IOMUXC_SW_MUX_CTL_PAD_EIM_D28(0x020E00C4)
    //   SION (4) - Software Input On Field Reset: SION_DISABLED
    //              Force the selected mux mode Input path no matter of MUX_MODE functionality.
    //     SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
    //     SION_ENABLED (1) - Force input path of pad EIM_D28.
    //   MUX_MODE (2-0) - MUX Mode Select Field Reset: ALT5
    //                    Select 1 of 8 iomux modes to be used for pad: EIM_D28.
    //     ALT0 (0) - Select mux mode: ALT0 mux port: WEIM_D[28] of instance: weim.
    //     ALT1 (1) - Select mux mode: ALT1 mux port: SDA of instance: i2c1.
    //                NOTE: - Config Register IOMUXC_I2C1_IPP_SDA_IN_SELECT_INPUT for mode ALT1.
    //     ALT2 (2) - Select mux mode: ALT2 mux port: MOSI of instance: ecspi4.
    //     ALT3 (3) - Select mux mode: ALT3 mux port: CSI1_D[12] of instance: ipu2.
    //                NOTE: - Config Register IOMUXC_IPU2_IPP_IND_SENS1_DATA_12_SELECT_INPUT for mode ALT3.
    //     ALT4 (4) - Select mux mode: ALT4 mux port: CTS of instance: uart2.
    //                NOTE: - Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4.
    //     ALT5 (5) - Select mux mode: ALT5 mux port: GPIO[28] of instance: gpio3.
    //     ALT6 (6) - Select mux mode: ALT6 mux port: EXT_TRIG of instance: ipu1.
    //     ALT7 (7) - Select mux mode: ALT7 mux port: DI0_PIN13 of instance: ipu1.
    writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D28);
    // Pad EIM_D28 is involved in Daisy Chain.
    // Input Select Register:
    // IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT(0x020E0924)
    //   DAISY (2-0) Reset: SEL_EIM_D28_ALT4
    //                Selecting Pads Involved in Daisy Chain.
    //                NOTE: Instance: uart2, In Pin: ipp_uart_rts_b
    //     SEL_EIM_D28_ALT4 (0) - Selecting Pad: EIM_D28 for Mode: ALT4.
    //     SEL_EIM_D29_ALT4 (1) - Selecting Pad: EIM_D29 for Mode: ALT4.
    //     SEL_SD3_CMD_ALT1 (2) - Selecting Pad: SD3_CMD for Mode: ALT1.
    //     SEL_SD3_CLK_ALT1 (3) - Selecting Pad: SD3_CLK for Mode: ALT1.
    //     SEL_SD4_DAT5_ALT2 (4) - Selecting Pad: SD4_DAT5 for Mode: ALT2.
    //     SEL_SD4_DAT6_ALT2 (5) - Selecting Pad: SD4_DAT6 for Mode: ALT2.
    writel((SEL_EIM_D28_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT);
    // Pad Control Register:
    // IOMUXC_SW_PAD_CTL_PAD_EIM_D28(0x020E03D8)
    //   HYS (16) - Hysteresis Enable Field Reset: HYS_ENABLED
    //              Select one out of next values for pad: EIM_D28.
    //     HYS_DISABLED (0) - Hysteresis Disabled
    //     HYS_ENABLED (1) - Hysteresis Enabled
    //   PUS (15-14) - Pull Up / Down Config. Field Reset: PUS_100KOHM_PU
    //                 Select one out of next values for pad: EIM_D28.
    //     PUS_100KOHM_PD (0) - 100K Ohm Pull Down
    //     PUS_47KOHM_PU (1) - 47K Ohm Pull Up
    //     PUS_100KOHM_PU (2) - 100K Ohm Pull Up
    //     PUS_22KOHM_PU (3) - 22K Ohm Pull Up
    //   PUE (13) - Pull / Keep Select Field Reset: PUE_PULL
    //              Select one out of next values for pad: EIM_D28.
    //     PUE_KEEP (0) - Keeper
    //     PUE_PULL (1) - Pull
    //   PKE (12) - Pull / Keep Enable Field Reset: PKE_ENABLED
    //              Select one out of next values for pad: EIM_D28.
    //     PKE_DISABLED (0) - Pull/Keeper Disabled
    //     PKE_ENABLED (1) - Pull/Keeper Enabled
    //   ODE (11) - Open Drain Enable Field Reset: ODE_DISABLED
    //              Select one out of next values for pad: EIM_D28.
    //     ODE_DISABLED (0) - Open Drain Disabled
    //     ODE_ENABLED (1) - Open Drain Enabled
    //   SPEED (7-6) - Speed Field Reset: SPD_100MHZ
    //                 Select one out of next values for pad: EIM_D28.
    //     SPD_TBD (0) - TBD
    //     SPD_50MHZ (1) - Low(50 MHz)
```

```
//     SPD_100MHZ (2) - Medium(100 MHz)
//     SPD_200MHZ (3) - Maximum(200 MHz)
//   DSE (5-3) - Drive Strength Field Reset: DSE_40OHM
//               Select one out of next values for pad: EIM_D28.
//     DSE_DISABLED (0) - Output driver disabled.
//     DSE_240OHM (1) - 240 Ohm
//     DSE_120OHM (2) - 120 Ohm
//     DSE_80OHM (3) - 80 Ohm
//     DSE_60OHM (4) - 60 Ohm
//     DSE_48OHM (5) - 48 Ohm
//     DSE_40OHM (6) - 40 Ohm
//     DSE_34OHM (7) - 34 Ohm
//   SRE (0) - Slew Rate Field Reset: SRE_SLOW
//             Select one out of next values for pad: EIM_D28.
//     SRE_SLOW (0) - Slow Slew Rate
//     SRE_FAST (1) - Fast Slew Rate
writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
       (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
       (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D28);

// Config uart2_RTS to pad EIM_D29(J19)
// Mux Register:
// IOMUXC_SW_MUX_CTL_PAD_EIM_D29(0x020E00C8)
//   SION (4) - Software Input On Field Reset: SION_DISABLED
//              Force the selected mux mode Input path no matter of MUX_MODE functionality.
//     SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
//     SION_ENABLED (1) - Force input path of pad EIM_D29.
//   MUX_MODE (2-0) - MUX Mode Select Field Reset: ALT5
//                    Select 1 of 7 iomux modes to be used for pad: EIM_D29.
//     ALT0 (0) - Select mux mode: ALT0 mux port: WEIM_D[29] of instance: weim.
//     ALT1 (1) - Select mux mode: ALT1 mux port: DI1_PIN15 of instance: ipu1.
//     ALT2 (2) - Select mux mode: ALT2 mux port: SS0 of instance: ecspi4.
//                NOTE: - Config Register IOMUXC_ECSPI4_IPP_IND_SS_B_0_SELECT_INPUT for mode ALT2.
//     ALT4 (4) - Select mux mode: ALT4 mux port: RTS of instance: uart2.
//                NOTE: - Config Register IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT for mode ALT4.
//     ALT5 (5) - Select mux mode: ALT5 mux port: GPIO[29] of instance: gpio3.
//     ALT6 (6) - Select mux mode: ALT6 mux port: CSI1_VSYNC of instance: ipu2.
//                NOTE: - Config Register IOMUXC_IPU2_IPP_IND_SENS1_VSYNC_SELECT_INPUT for mode ALT6.
//     ALT7 (7) - Select mux mode: ALT7 mux port: DI0_PIN14 of instance: ipu1.
writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D29);
// Pad EIM_D29 is involved in Daisy Chain.
// Input Select Register:
// IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT(0x020E0924)
//   DAISY (2-0) Reset: SEL_EIM_D28_ALT4
//                 Selecting Pads Involved in Daisy Chain.
//                 NOTE: Instance: uart2, In Pin: ipp_uart_rts_b
//     SEL_EIM_D28_ALT4 (0) - Selecting Pad: EIM_D28 for Mode: ALT4.
//     SEL_EIM_D29_ALT4 (1) - Selecting Pad: EIM_D29 for Mode: ALT4.
//     SEL_SD3_CMD_ALT1 (2) - Selecting Pad: SD3_CMD for Mode: ALT1.
//     SEL_SD3_CLK_ALT1 (3) - Selecting Pad: SD3_CLK for Mode: ALT1.
//     SEL_SD4_DAT5_ALT2 (4) - Selecting Pad: SD4_DAT5 for Mode: ALT2.
//     SEL_SD4_DAT6_ALT2 (5) - Selecting Pad: SD4_DAT6 for Mode: ALT2.
writel((SEL_EIM_D28_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RTS_B_SELECT_INPUT);
// Pad Control Register:
// IOMUXC_SW_PAD_CTL_PAD_EIM_D29(0x020E03DC)
//   HYS (16) - Hysteresis Enable Field Reset: HYS_ENABLED
//              Select one out of next values for pad: EIM_D29.
//     HYS_DISABLED (0) - Hysteresis Disabled
//     HYS_ENABLED (1) - Hysteresis Enabled
//   PUS (15-14) - Pull Up / Down Config. Field Reset: PUS_100KOHM_PU
//                 Select one out of next values for pad: EIM_D29.
//     PUS_100KOHM_PD (0) - 100K Ohm Pull Down
//     PUS_47KOHM_PU (1) - 47K Ohm Pull Up
//     PUS_100KOHM_PU (2) - 100K Ohm Pull Up
//     PUS_22KOHM_PU (3) - 22K Ohm Pull Up
//   PUE (13) - Pull / Keep Select Field Reset: PUE_PULL
//              Select one out of next values for pad: EIM_D29.
//     PUE_KEEP (0) - Keeper
//     PUE_PULL (1) - Pull
//   PKE (12) - Pull / Keep Enable Field Reset: PKE_ENABLED
//              Select one out of next values for pad: EIM_D29.
//     PKE_DISABLED (0) - Pull/Keeper Disabled
```

```
//     PKE_ENABLED (1) - Pull/Keeper Enabled
//   ODE (11) - Open Drain Enable Field Reset: ODE_DISABLED
//              Select one out of next values for pad: EIM_D29.
//     ODE_DISABLED (0) - Open Drain Disabled
//     ODE_ENABLED (1) - Open Drain Enabled
//   SPEED (7-6) - Speed Field Reset: SPD_100MHZ
//                 Select one out of next values for pad: EIM_D29.
//     SPD_TBD (0) - TBD
//     SPD_50MHZ (1) - Low(50 MHz)
//     SPD_100MHZ (2) - Medium(100 MHz)
//     SPD_200MHZ (3) - Maximum(200 MHz)
//   DSE (5-3) - Drive Strength Field Reset: DSE_40OHM
//               Select one out of next values for pad: EIM_D29.
//     DSE_DISABLED (0) - Output driver disabled.
//     DSE_240OHM (1) - 240 Ohm
//     DSE_120OHM (2) - 120 Ohm
//     DSE_80OHM (3) - 80 Ohm
//     DSE_60OHM (4) - 60 Ohm
//     DSE_48OHM (5) - 48 Ohm
//     DSE_40OHM (6) - 40 Ohm
//     DSE_34OHM (7) - 34 Ohm
//   SRE (0) - Slew Rate Field Reset: SRE_SLOW
//             Select one out of next values for pad: EIM_D29.
//     SRE_SLOW (0) - Slow Slew Rate
//     SRE_FAST (1) - Fast Slew Rate
writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
       (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
       (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D29);

// Config uart2_RXD_MUX to pad EIM_D27(E25)
// Mux Register:
// IOMUXC_SW_MUX_CTL_PAD_EIM_D27(0x020E00C0)
//   SION (4) - Software Input On Field Reset: SION_DISABLED
//              Force the selected mux mode Input path no matter of MUX_MODE functionality.
//     SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
//     SION_ENABLED (1) - Force input path of pad EIM_D27.
//   MUX_MODE (2-0) - MUX Mode Select Field Reset: ALT5
//                    Select 1 of 8 iomux modes to be used for pad: EIM_D27.
//     ALT0 (0) - Select mux mode: ALT0 mux port: WEIM_D[27] of instance: weim.
//     ALT1 (1) - Select mux mode: ALT1 mux port: DI1_PIN13 of instance: ipu1.
//     ALT2 (2) - Select mux mode: ALT2 mux port: CSI0_D[0] of instance: ipu1.
//     ALT3 (3) - Select mux mode: ALT3 mux port: CSI1_D[13] of instance: ipu2.
//                NOTE: - Config Register IOMUXC_IPU2_IPP_IND_SENS1_DATA_13_SELECT_INPUT for mode ALT3.
//     ALT4 (4) - Select mux mode: ALT4 mux port: RXD_MUX of instance: uart2.
//                NOTE: - Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT4.
//     ALT5 (5) - Select mux mode: ALT5 mux port: GPIO[27] of instance: gpio3.
//     ALT6 (6) - Select mux mode: ALT6 mux port: SISG[3] of instance: ipu1.
//     ALT7 (7) - Select mux mode: ALT7 mux port: DISP1_DAT[23] of instance: ipu1.
writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D27);
// Pad EIM_D27 is involved in Daisy Chain.
// Input Select Register:
// IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT(0x020E0928)
//   DAISY (2-0) Reset: SEL_EIM_D26_ALT4
//                 Selecting Pads Involved in Daisy Chain.
//                 NOTE: Instance: uart2, In Pin: ipp_uart_rxd_mux
//     SEL_EIM_D26_ALT4 (0) - Selecting Pad: EIM_D26 for Mode: ALT4.
//     SEL_EIM_D27_ALT4 (1) - Selecting Pad: EIM_D27 for Mode: ALT4.
//     SEL_GPIO_7_ALT4 (2) - Selecting Pad: GPIO_7 for Mode: ALT4.
//     SEL_GPIO_8_ALT4 (3) - Selecting Pad: GPIO_8 for Mode: ALT4.
//     SEL_SD3_DAT5_ALT1 (4) - Selecting Pad: SD3_DAT5 for Mode: ALT1.
//     SEL_SD3_DAT4_ALT1 (5) - Selecting Pad: SD3_DAT4 for Mode: ALT1.
//     SEL_SD4_DAT4_ALT2 (6) - Selecting Pad: SD4_DAT4 for Mode: ALT2.
//     SEL_SD4_DAT7_ALT2 (7) - Selecting Pad: SD4_DAT7 for Mode: ALT2.
writel((SEL_EIM_D26_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT);
// Pad Control Register:
// IOMUXC_SW_PAD_CTL_PAD_EIM_D27(0x020E03D4)
//   HYS (16) - Hysteresis Enable Field Reset: HYS_ENABLED
//              Select one out of next values for pad: EIM_D27.
//     HYS_DISABLED (0) - Hysteresis Disabled
//     HYS_ENABLED (1) - Hysteresis Enabled
//   PUS (15-14) - Pull Up / Down Config. Field Reset: PUS_100KOHM_PU
//                 Select one out of next values for pad: EIM_D27.
```

```
//     PUS_100KOHM_PD (0) - 100K Ohm Pull Down
//     PUS_47KOHM_PU (1) - 47K Ohm Pull Up
//     PUS_100KOHM_PU (2) - 100K Ohm Pull Up
//     PUS_22KOHM_PU (3) - 22K Ohm Pull Up
//   PUE (13) - Pull / Keep Select Field Reset: PUE_PULL
//              Select one out of next values for pad: EIM_D27.
//     PUE_KEEP (0) - Keeper
//     PUE_PULL (1) - Pull
//   PKE (12) - Pull / Keep Enable Field Reset: PKE_ENABLED
//              Select one out of next values for pad: EIM_D27.
//     PKE_DISABLED (0) - Pull/Keeper Disabled
//     PKE_ENABLED (1) - Pull/Keeper Enabled
//   ODE (11) - Open Drain Enable Field Reset: ODE_DISABLED
//              Select one out of next values for pad: EIM_D27.
//     ODE_DISABLED (0) - Open Drain Disabled
//     ODE_ENABLED (1) - Open Drain Enabled
//   SPEED (7-6) - Speed Field Reset: SPD_100MHZ
//                 Select one out of next values for pad: EIM_D27.
//     SPD_TBD (0) - TBD
//     SPD_50MHZ (1) - Low(50 MHz)
//     SPD_100MHZ (2) - Medium(100 MHz)
//     SPD_200MHZ (3) - Maximum(200 MHz)
//   DSE (5-3) - Drive Strength Field Reset: DSE_40OHM
//               Select one out of next values for pad: EIM_D27.
//     DSE_DISABLED (0) - Output driver disabled.
//     DSE_240OHM (1) - 240 Ohm
//     DSE_120OHM (2) - 120 Ohm
//     DSE_80OHM (3) - 80 Ohm
//     DSE_60OHM (4) - 60 Ohm
//     DSE_48OHM (5) - 48 Ohm
//     DSE_40OHM (6) - 40 Ohm
//     DSE_34OHM (7) - 34 Ohm
//   SRE (0) - Slew Rate Field Reset: SRE_SLOW
//             Select one out of next values for pad: EIM_D27.
//     SRE_SLOW (0) - Slow Slew Rate
//     SRE_FAST (1) - Fast Slew Rate
writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
       (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
       (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D27);

// Config uart2_TXD_MUX to pad EIM_D26(E24)
// Connected to U12, auxiliary console
// Mux Register:
// IOMUXC_SW_MUX_CTL_PAD_EIM_D26(0x020E00BC)
//   SION (4) - Software Input On Field Reset: SION_DISABLED
//              Force the selected mux mode Input path no matter of MUX_MODE functionality.
//     SION_DISABLED (0) - Input Path is determined by functionality of the selected mux mode (regular).
//     SION_ENABLED (1) - Force input path of pad EIM_D26.
//   MUX_MODE (2-0) - MUX Mode Select Field Reset: ALT5
//                    Select 1 of 8 iomux modes to be used for pad: EIM_D26.
//     ALT0 (0) - Select mux mode: ALT0 mux port: WEIM_D[26] of instance: weim.
//     ALT1 (1) - Select mux mode: ALT1 mux port: DI1_PIN11 of instance: ipu1.
//     ALT2 (2) - Select mux mode: ALT2 mux port: CSI0_D[1] of instance: ipu1.
//     ALT3 (3) - Select mux mode: ALT3 mux port: CSI1_D[14] of instance: ipu2.
//                NOTE: - Config Register IOMUXC_IPU2_IPP_IND_SENS1_DATA_14_SELECT_INPUT for mode ALT3.
//     ALT4 (4) - Select mux mode: ALT4 mux port: TXD_MUX of instance: uart2.
//                NOTE: - Config Register IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT for mode ALT4.
//     ALT5 (5) - Select mux mode: ALT5 mux port: GPIO[26] of instance: gpio3.
//     ALT6 (6) - Select mux mode: ALT6 mux port: SISG[2] of instance: ipu1.
//     ALT7 (7) - Select mux mode: ALT7 mux port: DISP1_DAT[22] of instance: ipu1.
writel((SION_DISABLED & 0x1) << 4 | (ALT4 & 0x7), IOMUXC_SW_MUX_CTL_PAD_EIM_D26);
// Pad EIM_D26 is involved in Daisy Chain.
// Input Select Register:
// IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT(0x020E0928)
//   DAISY (2-0) Reset: SEL_EIM_D26_ALT4
//                Selecting Pads Involved in Daisy Chain.
//                NOTE: Instance: uart2, In Pin: ipp_uart_rxd_mux
//     SEL_EIM_D26_ALT4 (0) - Selecting Pad: EIM_D26 for Mode: ALT4.
//     SEL_EIM_D27_ALT4 (1) - Selecting Pad: EIM_D27 for Mode: ALT4.
//     SEL_GPIO_7_ALT4 (2) - Selecting Pad: GPIO_7 for Mode: ALT4.
//     SEL_GPIO_8_ALT4 (3) - Selecting Pad: GPIO_8 for Mode: ALT4.
//     SEL_SD3_DAT5_ALT1 (4) - Selecting Pad: SD3_DAT5 for Mode: ALT1.
```

```
    //     SEL_SD3_DAT4_ALT1 (5) - Selecting Pad: SD3_DAT4 for Mode: ALT1.
    //     SEL_SD4_DAT4_ALT2 (6) - Selecting Pad: SD4_DAT4 for Mode: ALT2.
    //     SEL_SD4_DAT7_ALT2 (7) - Selecting Pad: SD4_DAT7 for Mode: ALT2.
    writel((SEL_EIM_D26_ALT4 & 0x7), IOMUXC_UART2_IPP_UART_RXD_MUX_SELECT_INPUT);
    // Pad Control Register:
    // IOMUXC_SW_PAD_CTL_PAD_EIM_D26(0x020E03D0)
    //   HYS (16) - Hysteresis Enable Field Reset: HYS_ENABLED
    //              Select one out of next values for pad: EIM_D26.
    //     HYS_DISABLED (0) - Hysteresis Disabled
    //     HYS_ENABLED (1) - Hysteresis Enabled
    //   PUS (15-14) - Pull Up / Down Config. Field Reset: PUS_100KOHM_PU
    //                 Select one out of next values for pad: EIM_D26.
    //     PUS_100KOHM_PD (0) - 100K Ohm Pull Down
    //     PUS_47KOHM_PU (1) - 47K Ohm Pull Up
    //     PUS_100KOHM_PU (2) - 100K Ohm Pull Up
    //     PUS_22KOHM_PU (3) - 22K Ohm Pull Up
    //   PUE (13) - Pull / Keep Select Field Reset: PUE_PULL
    //              Select one out of next values for pad: EIM_D26.
    //     PUE_KEEP (0) - Keeper
    //     PUE_PULL (1) - Pull
    //   PKE (12) - Pull / Keep Enable Field Reset: PKE_ENABLED
    //              Select one out of next values for pad: EIM_D26.
    //     PKE_DISABLED (0) - Pull/Keeper Disabled
    //     PKE_ENABLED (1) - Pull/Keeper Enabled
    //   ODE (11) - Open Drain Enable Field Reset: ODE_DISABLED
    //              Select one out of next values for pad: EIM_D26.
    //     ODE_DISABLED (0) - Open Drain Disabled
    //     ODE_ENABLED (1) - Open Drain Enabled
    //   SPEED (7-6) - Speed Field Reset: SPD_100MHZ
    //                 Select one out of next values for pad: EIM_D26.
    //     SPD_TBD (0) - TBD
    //     SPD_50MHZ (1) - Low(50 MHz)
    //     SPD_100MHZ (2) - Medium(100 MHz)
    //     SPD_200MHZ (3) - Maximum(200 MHz)
    //   DSE (5-3) - Drive Strength Field Reset: DSE_40OHM
    //               Select one out of next values for pad: EIM_D26.
    //     DSE_DISABLED (0) - Output driver disabled.
    //     DSE_240OHM (1) - 240 Ohm
    //     DSE_120OHM (2) - 120 Ohm
    //     DSE_80OHM (3) - 80 Ohm
    //     DSE_60OHM (4) - 60 Ohm
    //     DSE_48OHM (5) - 48 Ohm
    //     DSE_40OHM (6) - 40 Ohm
    //     DSE_34OHM (7) - 34 Ohm
    //   SRE (0) - Slew Rate Field Reset: SRE_SLOW
    //             Select one out of next values for pad: EIM_D26.
    //     SRE_SLOW (0) - Slow Slew Rate
    //     SRE_FAST (1) - Fast Slew Rate
    writel((HYS_ENABLED & 0x1) << 16 | (PUS_100KOHM_PU & 0x3) << 14 | (PUE_PULL & 0x1) << 13 |
           (PKE_ENABLED & 0x1) << 12 | (ODE_DISABLED & 0x1) << 11 | (SPD_100MHZ & 0x3) << 6 |
           (DSE_40OHM & 0x7) << 3 | (SRE_SLOW & 0x1), IOMUXC_SW_PAD_CTL_PAD_EIM_D26);
}
```

3.3.5 Generate Code

Clicking the Generate Code menu item will compose the IOMUXC configuration code in the currently selected code style for all peripherals in the design. A C file will be generated for each peripheral as well as a board level file that calls all of the peripheral functions. The code will be written to the src directory where the IOMux application is currently running. If the design contains changes that have not been saved, the user will be prompted to save the design before generating code. i.MX device specific header files are placed in the src/include/<device> folder (Figure 8). Board specific source and header files are placed in the src/<device>/iomux/<board> folder (Figure 9). A bell will sound after the code files have been generated indicating the process has finished.

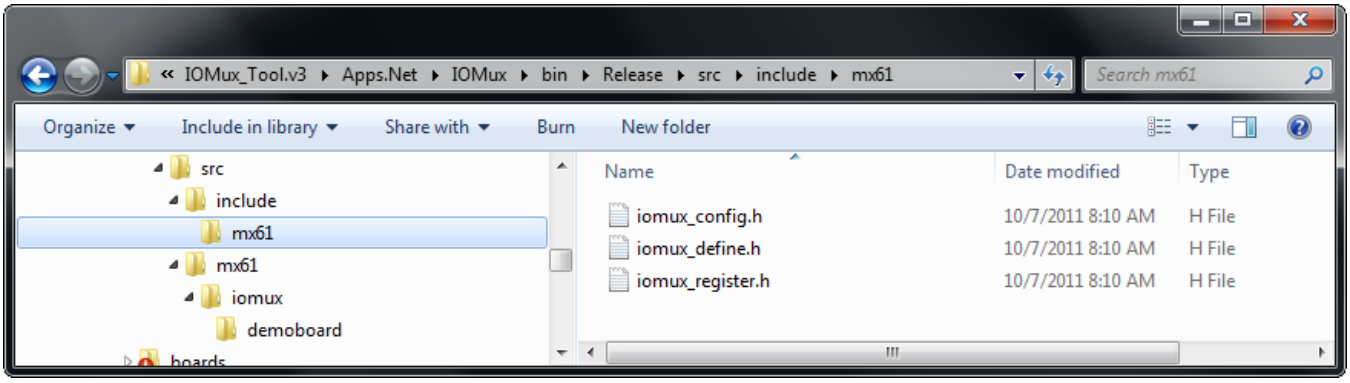

Figure 8. Chip specific files.

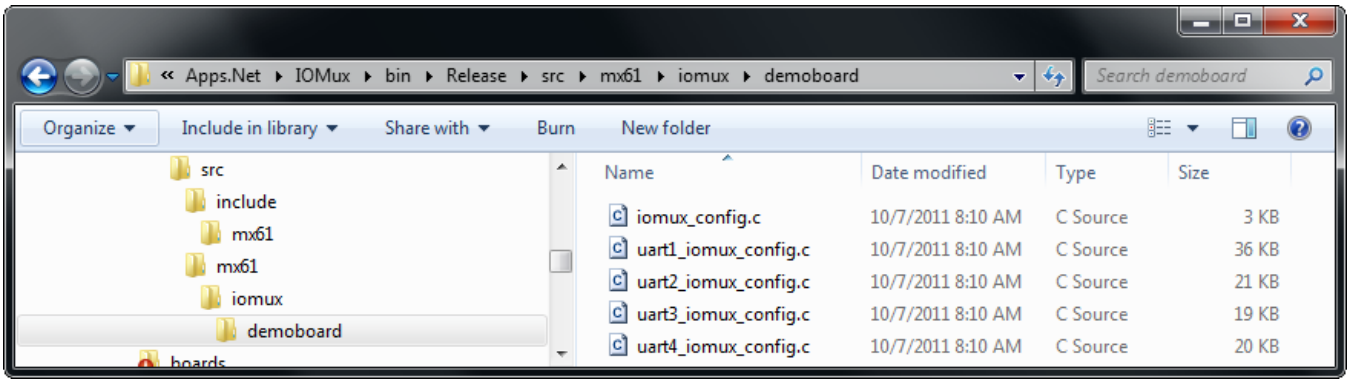

Figure 9. Board specific files.

3.4 View Menu

Figure 10 shows a partial screen shot of the View menu.

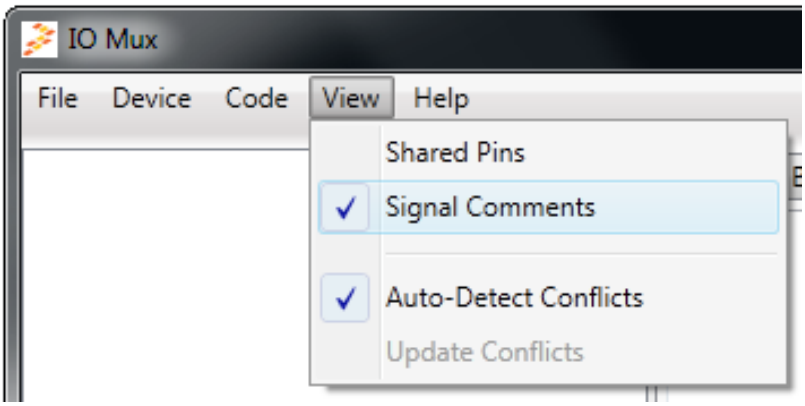

Figure 10. Partial screen shot of the View menu.

The View menu is used to control the behavior and presentation of the design to the user.

3.4.1 Shared Pins and Signal Comments

The Shared Pins and Signal Comments menu items of the View menu are mutually exclusive. These items select which column appears in the Assigned Signals Tab. By default, the Signal Comments item is selected and the GPIO and Signal Notes columns are displayed on the Signals tab in the Design Info Pane. In the Signal Comments view mode, the tooltip shown when hovering over a signal line will display a list of the other signals (Shared Signals) that could be routed to the selected pad. Selecting the Shared Pins menu item will remove the GPIO and Signal Notes columns and show the Shared Signals column. In this view mode, the Signal Notes are shown in the tooltip when hovering over a signal line.

3.4.2 Auto-Detect Conflicts

The Auto-Detect Conflicts menu item toggles how the application checks a design for conflicts. If this item is selected, any time a single signal assignment updated, the whole design will be checked for conflicts. When unselected, a design will not be checked for conflicts until the Update Conflicts menu item is selected, forcing an update to be performed.

3.4.3 Update Conflicts

The Update Conflicts item is not enabled unless the Auto-Detect option is unselected. Selecting the Update Conflicts menu item will manually force the design's assignments to be checked. Usually this option is used for very large designs with many signals to speed up the selection.

3.5 Help Menu

The Help menu, shown in Figure 11, contains a single item. Selecting it will display some information about the IOMux application including the version of the tool.

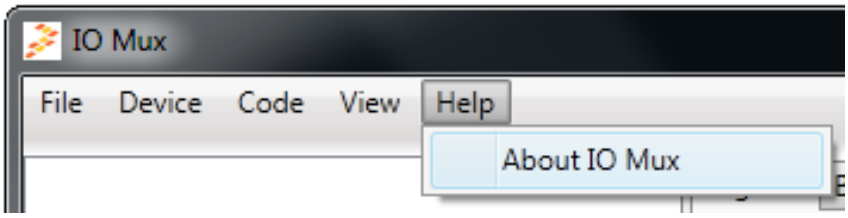

Figure 11. A partial screen shot of the Help menu.

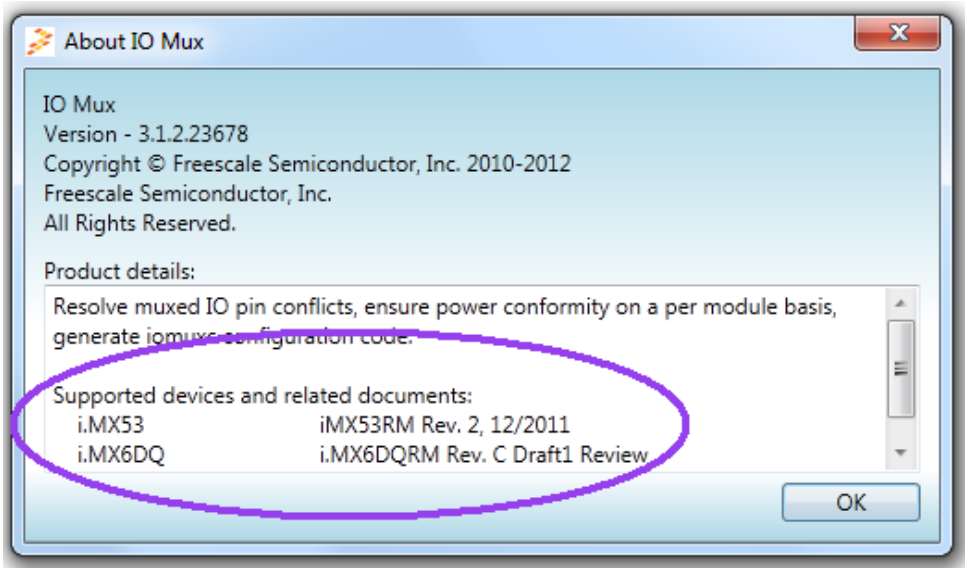

Figure 12. Example About Box Dialog

Information about supported devices and the source of the IOMUXC register information supporting each device can be found in the About Box.

4 Basic Usage Example with Explanation

In this section, a brief walk through of the IOMux tool will be presented.

4.1 Select Peripherals and Signals for the Board Design

Figure 13 shows the application window after the following sequence:

1. Launch IOMux.exe application.
2. Select Device > “i.MX6DQ.
3. Check the UARTS: UART1, UART2 and UART3.
4. Expand all signals under UART3.

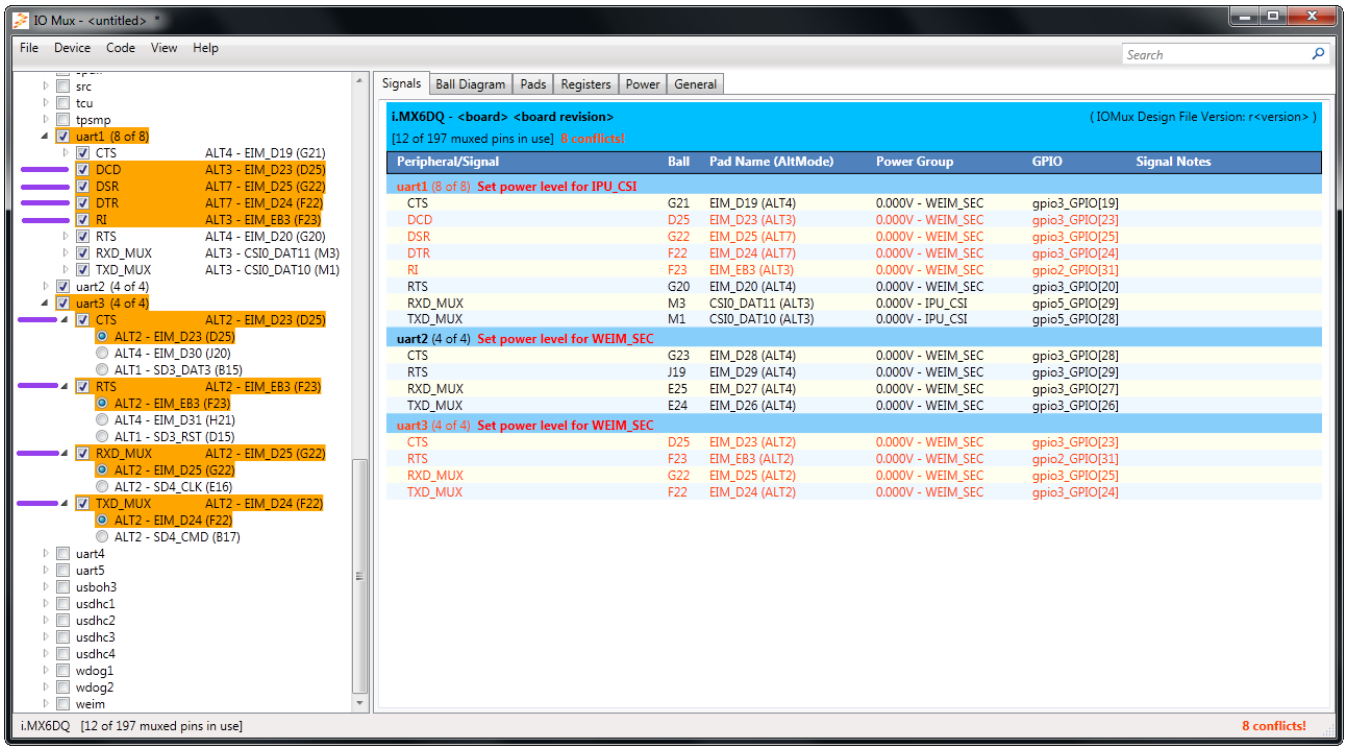

Figure 13. Application window after selecting UARTs 1, 2 and 3 of i.MX6DQ.

As can be seen in Figure 13, conflicts between eight signals are highlighted in orange. They are marked in purple in the figure. The conflicts were detected immediately after selecting the third UART because Auto-Detect Conflicts is enabled by default when the application starts up. Peripheral groups are expanded to show the conflicts at the signal level once they are detected. The number of conflicting signals is indicated in the status bar in the lower right corner of the application window.

To see what other signals share the assigned ball for a signal, hover the mouse over a signal, UART3/TXD_MUX is shown in Figure 14. Notice that UART1_DTR is bolded in the list that pops up. This is the signal that is conflicting with UART3/TXD_MUX. All conflicts on a ball will be bolded so that the user knows what the other conflicts are.

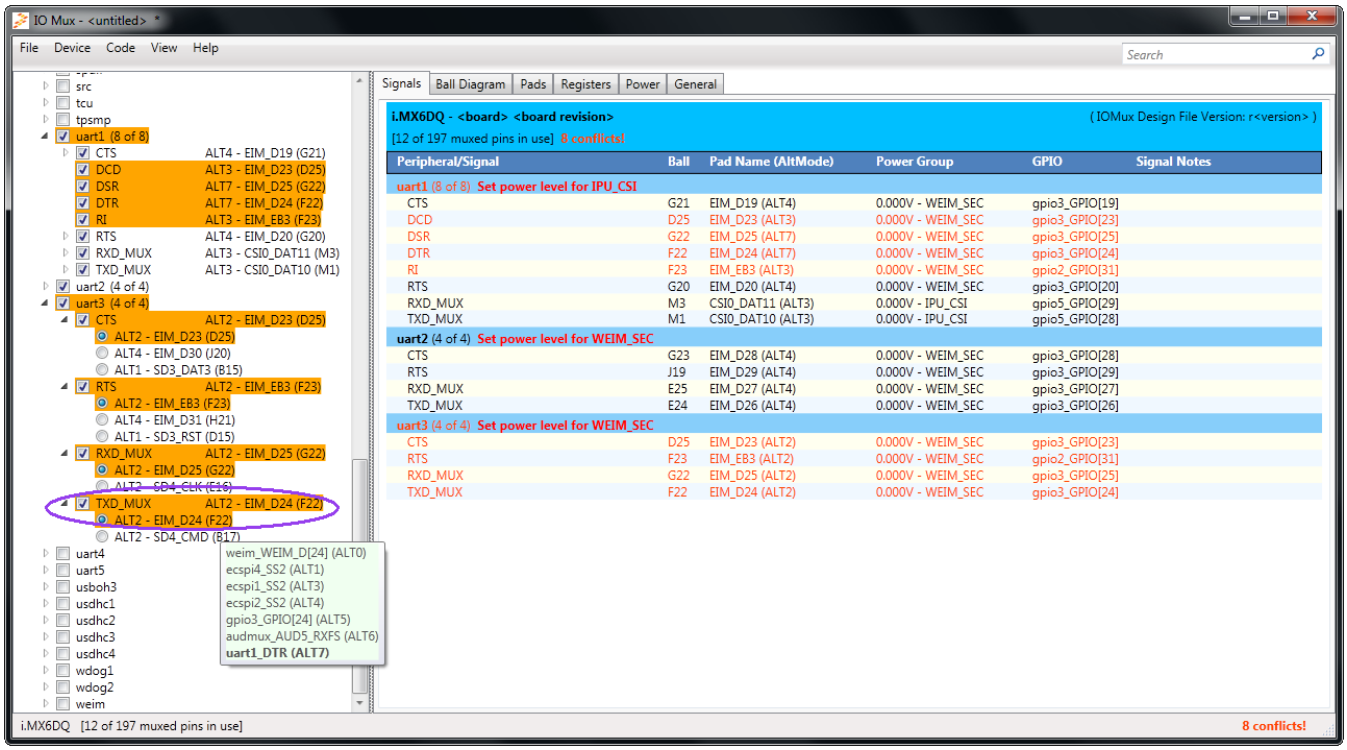

Figure 14. The tooltip for a signal in the Signal Selection Pane shows shared signals for that ball/pin.

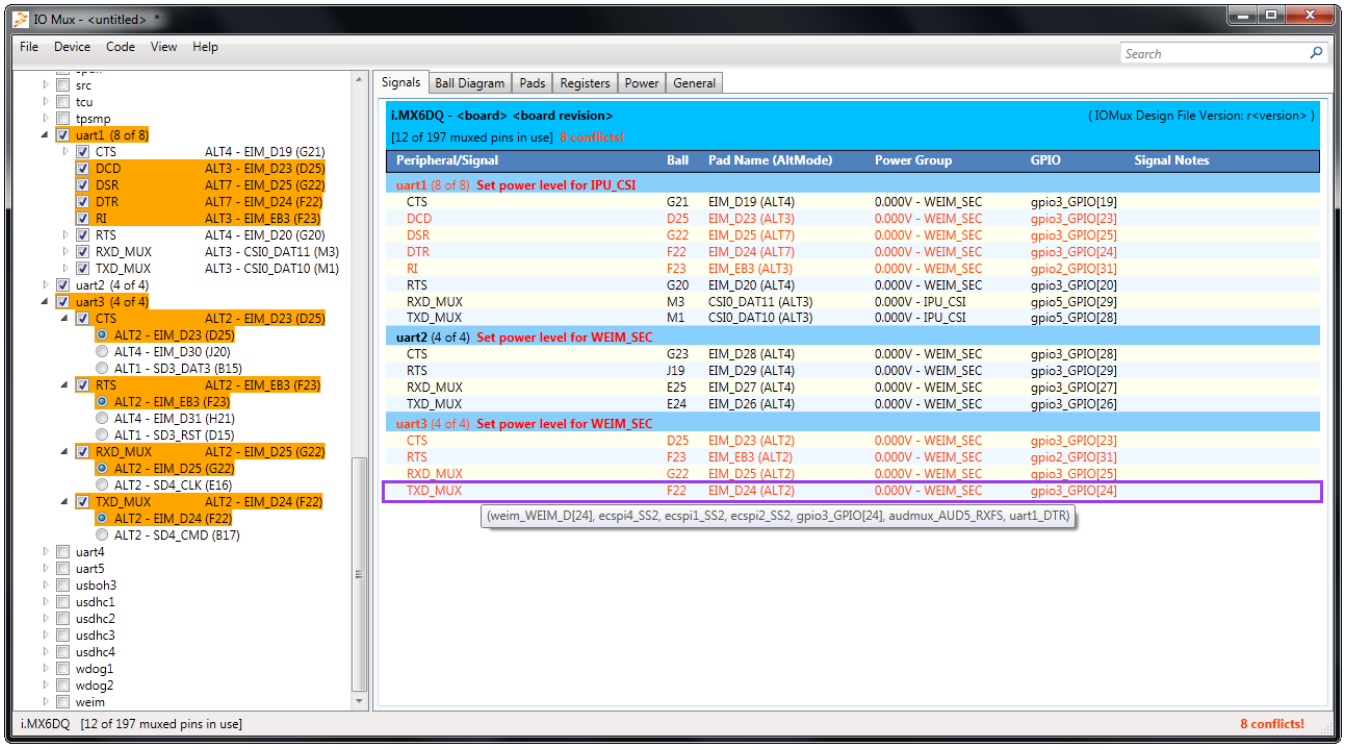

Figure 15. The tooltip for a signal in the Signals tab shows shared signals.

Hovering the mouse over the same signal, UART3/TXD_MUX in the Signals tab, as seen in Figure 15 also shows a list of signals shared on a given ball/pin. This pop up list does not indicate any conflicts.

4.2 Resolve Conflicting Signals by Selecting Alternate Pads

Figure 16 shows the application window after the following sequence:

1. Select ALT4 – EIM_D30(J20) for UART3/CTS.
2. Select ALT4 – EIM_D31(H21) for UART3/RTS.
3. Select ALT2 – SD4_CLK(E16) for UART3/RXD_MUX.
4. Select ALT2 – SD4_CMD(B17) for UART3/TXD_MUX.

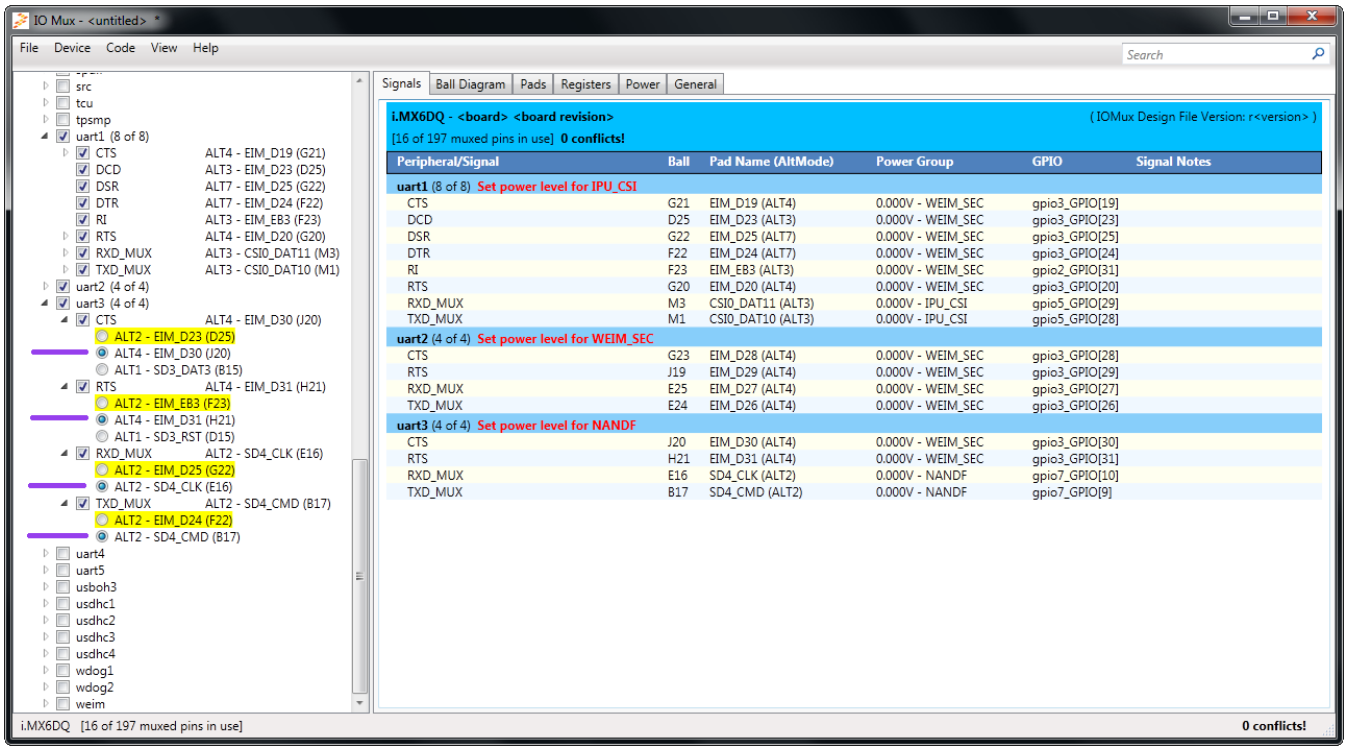

Figure 16. The conflicts in the example are resolved by changing ball assignments.

The conflicts in the example so far may be resolved by assigning J20 to UART3/CTS, H21 to UART3/RTS, E16 to UART3/RXD_MUX, and B17 to UART3/TXD_MUX, as shown in Figure 16. Notice that the original ball assignments for these four signals are now highlighted in yellow indicating that potential conflicts exist if those balls are chosen for those signals. All the conflicts in our example have now been resolved.

In an actual design, several iterations of re-assigning balls for signals may be required to resolve all the conflicts for chosen signals. In this example, all the signals for each peripheral group were selected by checking the box next to the peripheral. In actual usage, only a subset of all the signals for a peripheral are selected.

4.3 Add Comments to Signals for Reference

Now that conflicts have been resolved, let's move on to another neat feature of the IOMux application whereby the user may add information to the pin assignments, giving the opportunity to explain why selections are made, what they connect to, or how they are intended to be used.

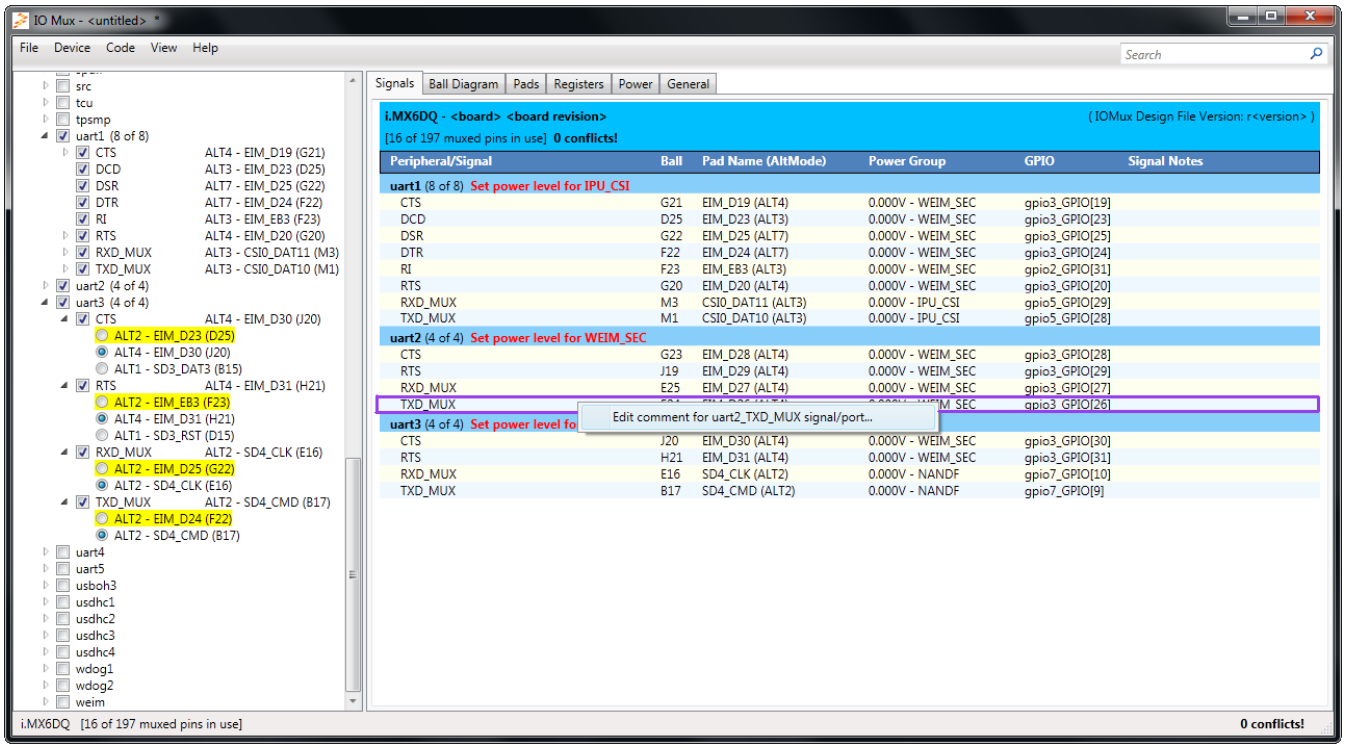

Figure 17. Right-clicking on a signal assignment brings up the comment entry menu.

In Figure 17, a right-click was performed on the Signals tab on the UART2/TXD_MUX row to bring up the context menu for entering a comment. Clicking on the menu will bring up a text entry field where the user may enter text.

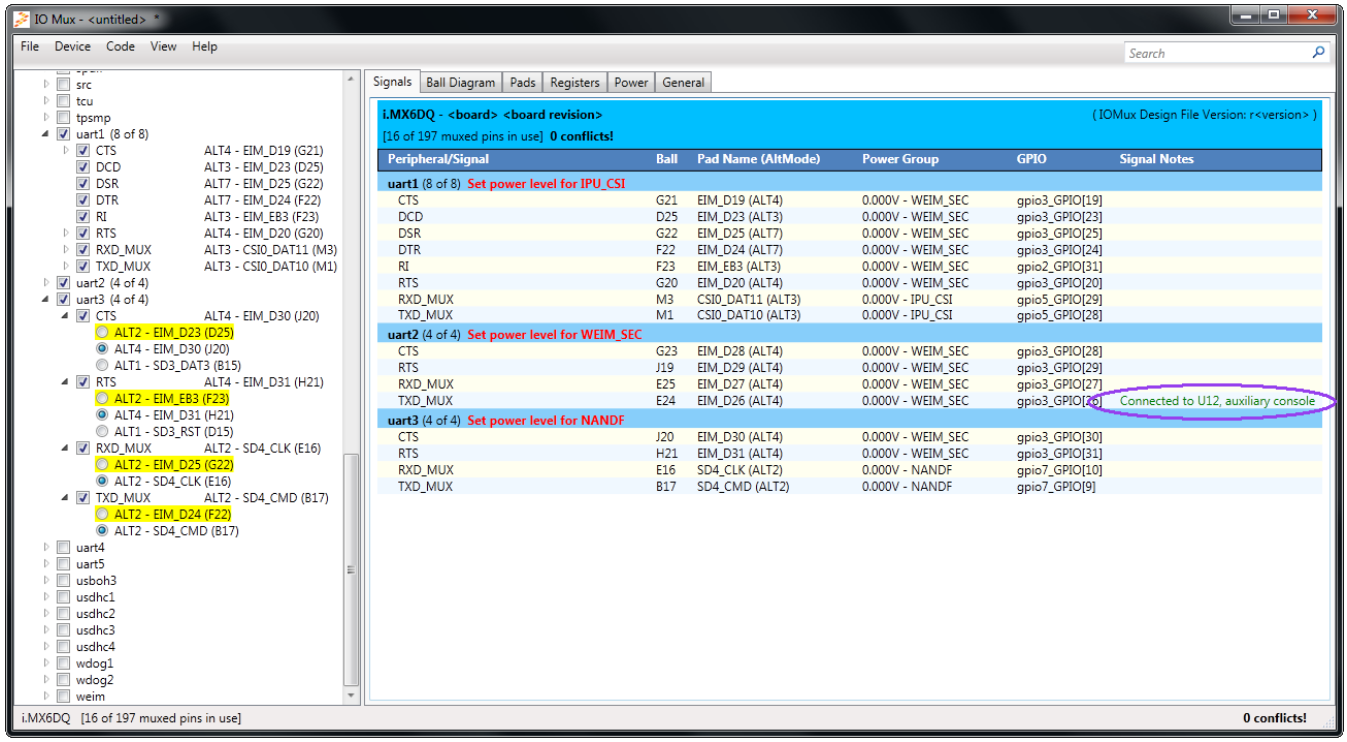

Figure 18. A note has been entered for UART2/TXD_MUX.

After entering the text and clicking OK, the comment is added to the Signals tab under the Signal Notes column as shown in Figure 18.

4.4 Ball Diagram View

For some signals, proximity to other devices may be an issue because of critical routing. In order to see where assigned signals are located on the actual package, click on the Ball Diagram tab. Once chosen, the Ball Diagram will be seen as in Figure 19.

By hovering the mouse over a ball, a pop up list showing all the signals that may be connected to that ball, with the assigned one bolded.

Changes on the left hand Signal Selection Pane are immediately shown in the Ball Diagram tab, but changes cannot be made directly from the Ball Diagram tab.

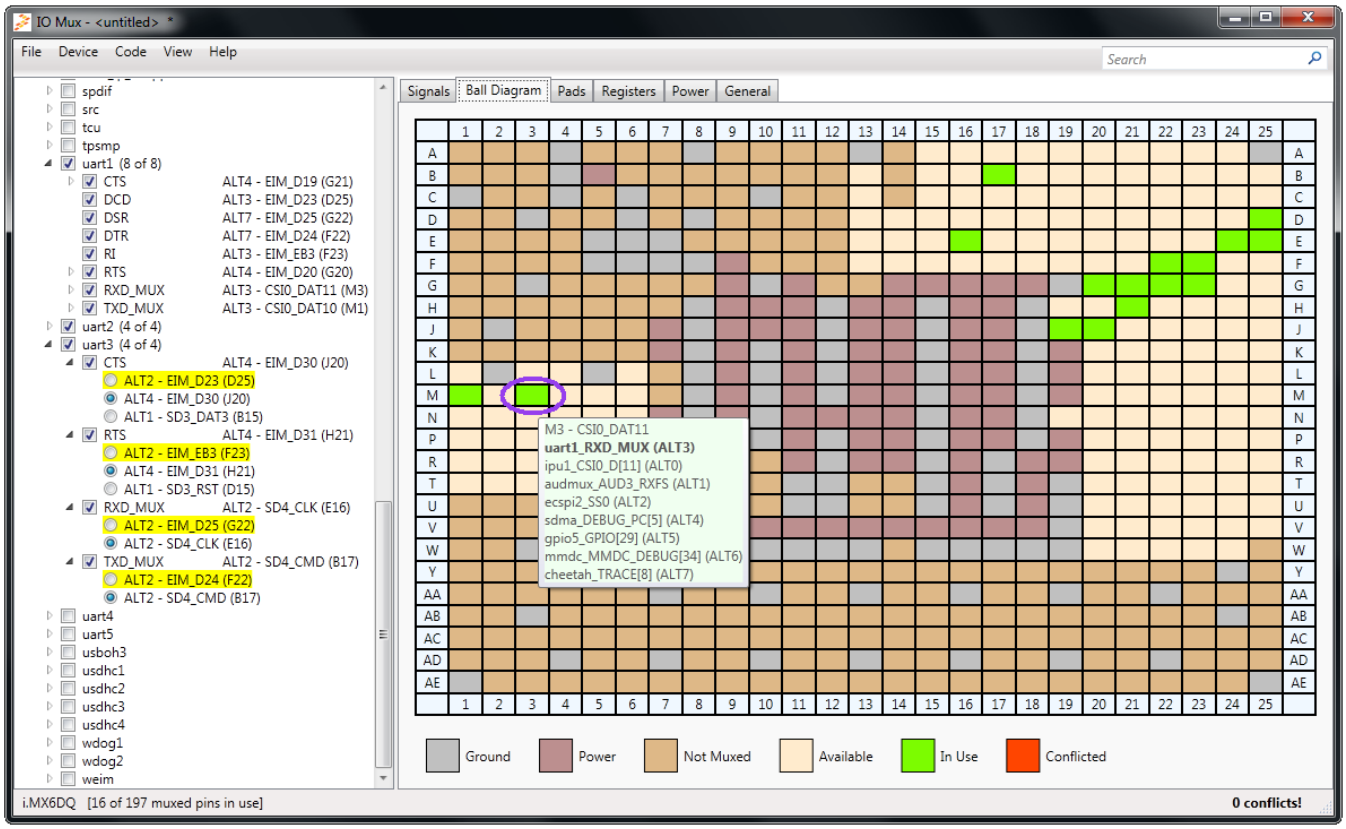

Figure 19. Choosing the Ball Diagram tab displays the device's ball map.

4.5 Setup IOMUXC Registers

Next we need to make sure that all of the IOMUXC registers are set up to configure the routing and pad settings as the signals are intend to be used. Select a signal in the left hand pane and click the Registers tab in the right hand pane to see the IOMUXC registers associated with the selected signal.

Figure 20 shows the application window after the following sequence:

1. Select UART3/RXD_MUX in the left hand pane.
2. Click the Register tab in the right hand pane.

3. Change the value for the DAISY field of the IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT register to SEL_SD4_CLK_ALT2.

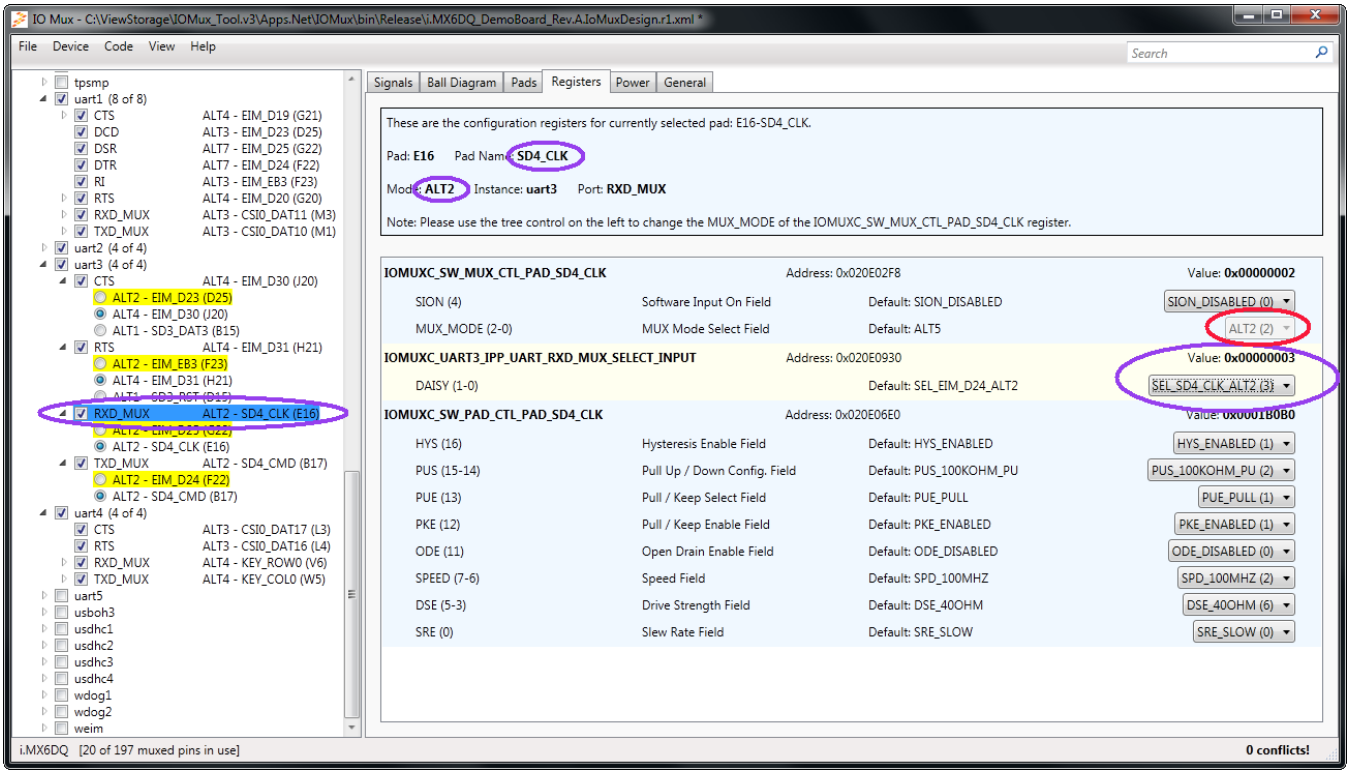

Figure 20. Setting the signals IOMUXC registers.

When we select the UART3/RXD_MUX signal in the left hand pane (indicated with blue highlighting) for this demonstration, there is an INPUT_SELECT register in addition to the MUX_CTL and PAD_CTL registers. Since the SD4_CLK(E16) pad is associated with an SELECT_INPUT register and UART3/RXD_MUX is an input, we need to select the DAISY field value that will reflect the Pad Name and Mode setting for the currently selected ALT-mode.

The values of the registers will immediately reflect the changes in the fields for that register. The code displayed in the peripheral tooltip on the Signals tab will immediately show the new register settings. The values for the registers will be saved and restored in the board design file.

Note that the value of the MUX_MODE field of the MUX_CTL register cannot be changed on the Registers tab. This is because changing the ALT-mode will change the pad and the registers displayed are associated with the pad/ball/pin currently selected in the left hand pane. ALT-mode can only be changed in the Signal Selection Pane on the left side of the application.

4.6 Setup Power Groups

The IOMux application tries to ensure consistency of voltage levels for all signals in a given peripheral. In order to make these checks possible, the voltage settings need to be set for each power group listed on the Power tab in the Design Info Pane.

Figure 21 shows the application window after the following sequence:

1. Select Signals tab in the right hand pane.
2. Note that the Power Groups being used by all signals are WEIM_SEC, IPU_CSI, and NANDF.
3. Note that both voltage levels are 0.000 and there is a warning in the UART1 peripheral line stating “Set power level for IPU_CSI”.
4. Select the Power tab in the right hand pane.
5. Set the voltage levels for IPU_CSI and WEIM_SEC to 3.3.
6. Set the voltage level for NANDF to 1.8.

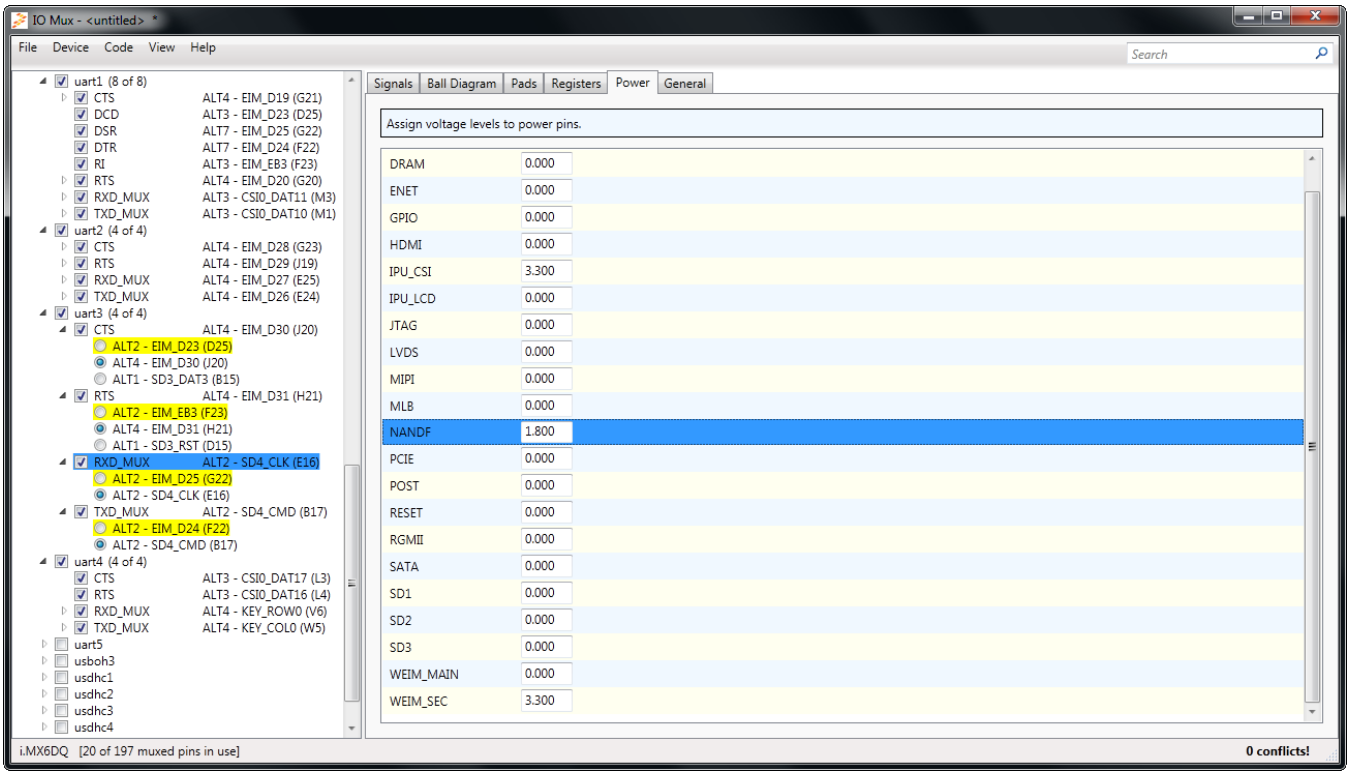

Figure 21. Enter the voltage levels for the power groups.

7. Click back to the Signals tab.

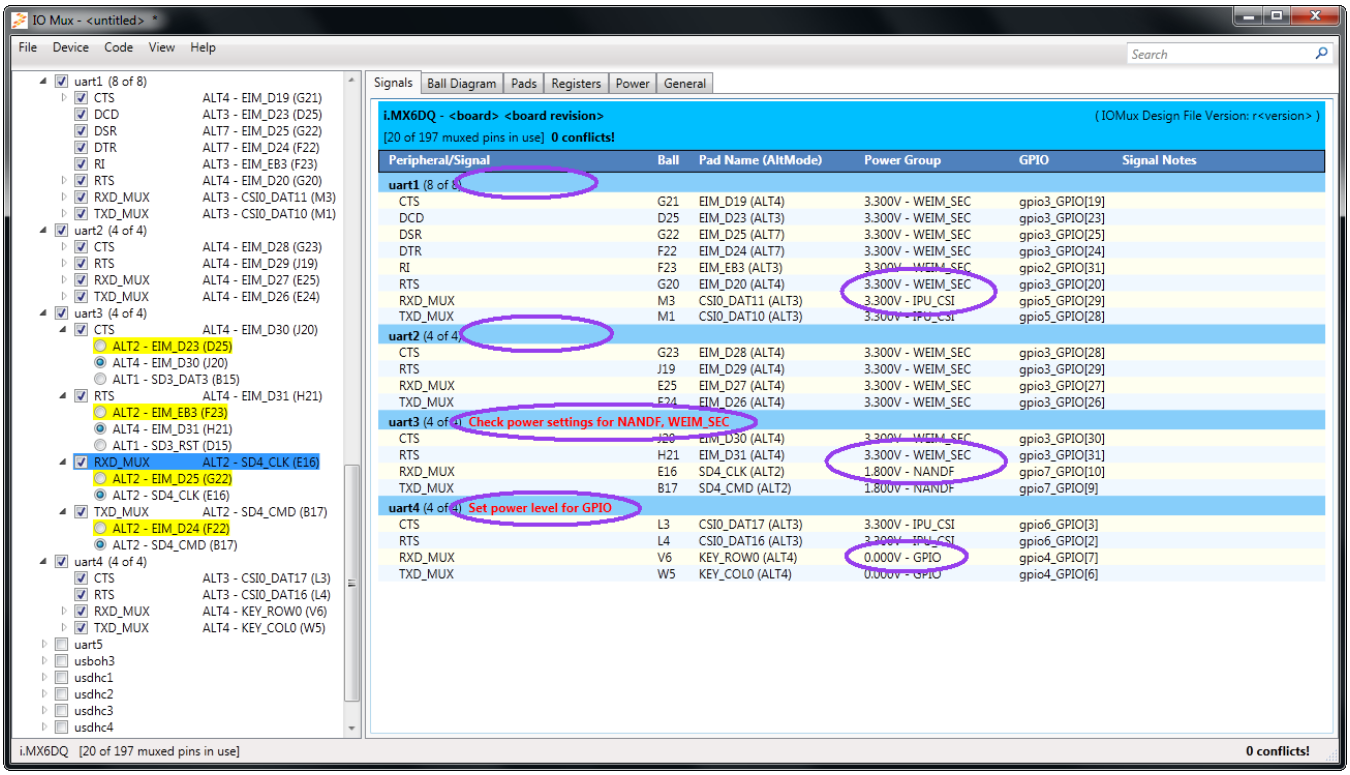

Figure 22. The Signals tab after setting some of the Power Group voltages.

Figure 22 shows that the “Set power level for IPU_CSI” for UART1 is no longer present. This warning was cleared because both the WEIM_SEC and IPU_CSI power groups were set to 3.3 volts.

There is no warning for UART2 because all of the signals are on power group WEIM_SEC which is set to a non-zero value.

There is a warning for UART3, “Check power settings for NADF, WEIM_SEC,” because the NANDF voltage level is not the same as the WEIM_SEC voltage level.

There is a warning for UART4, “Set power level for GPIO,” because the GPIO voltage level is still at the default zero voltage level.

To clear the remaining warnings:

8. Click back to the Power tab.
9. Set the voltage level for NANDF and GPIO to 3.3.
10. Click back to the Signals tab to check all power warnings are clear.

4.7 Enter Board Design Info on the General Tab

General information about the board design and IOMux application design file are tracked on the General tab in Device Info Pane.

Figure 23 shows the application window after the following sequence:

1. Select General tab in the right hand pane.
2. Enter board information.

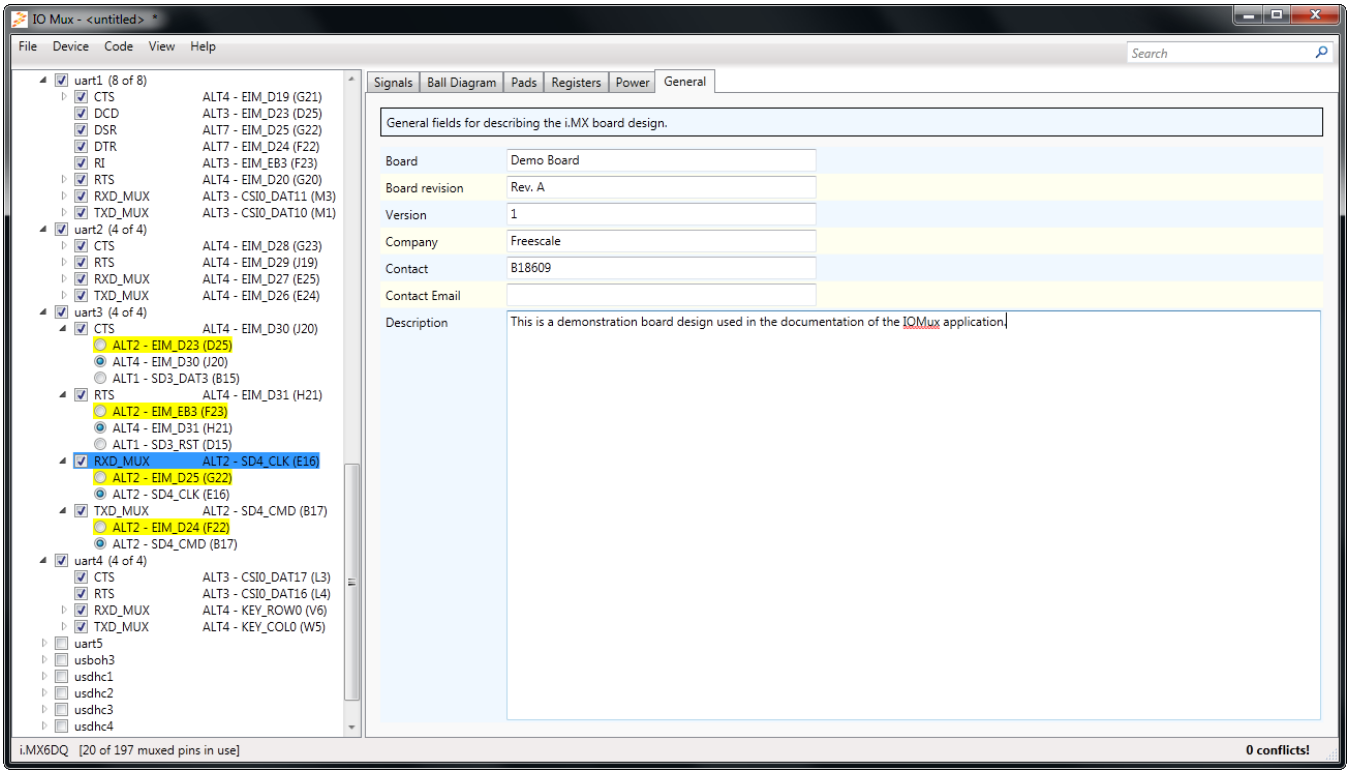

Figure 23. Enter Board Design information.

3. Switch back to the Signals tab.

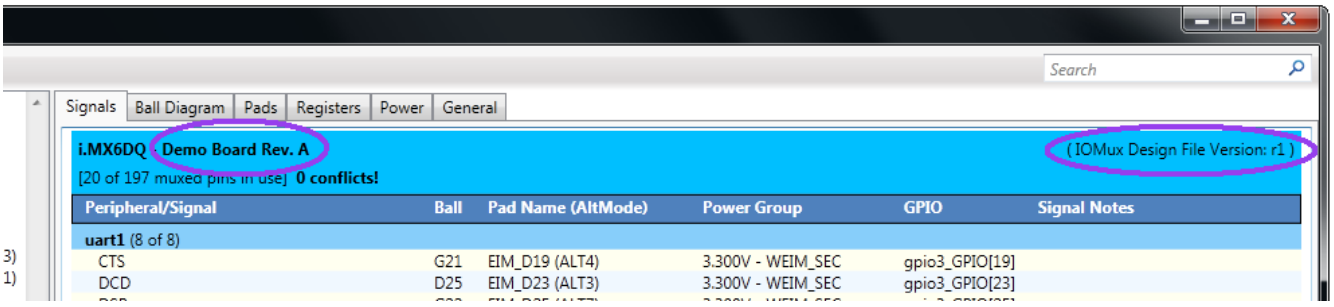

Figure 24. Info from General tab appears on Signals tab.

Figure 24 shows the Board, Board revision and Version information from the General tab appears in the Signals tab. The remaining information is only stored in the IOMux application design file.

The Version field is intended to denote the version of the IOMux design file itself and not directly related to the board information.

Entering contact information on the General tab could be useful if passing the design file between internal or external hardware and software departments and questions arise about pad settings, input routing, etc.

4.8 Save the Design

Once completed, the design should be saved for future use or reference. Using the Save item in the File menu, a dialog box is brought up as shown in Figure 25.

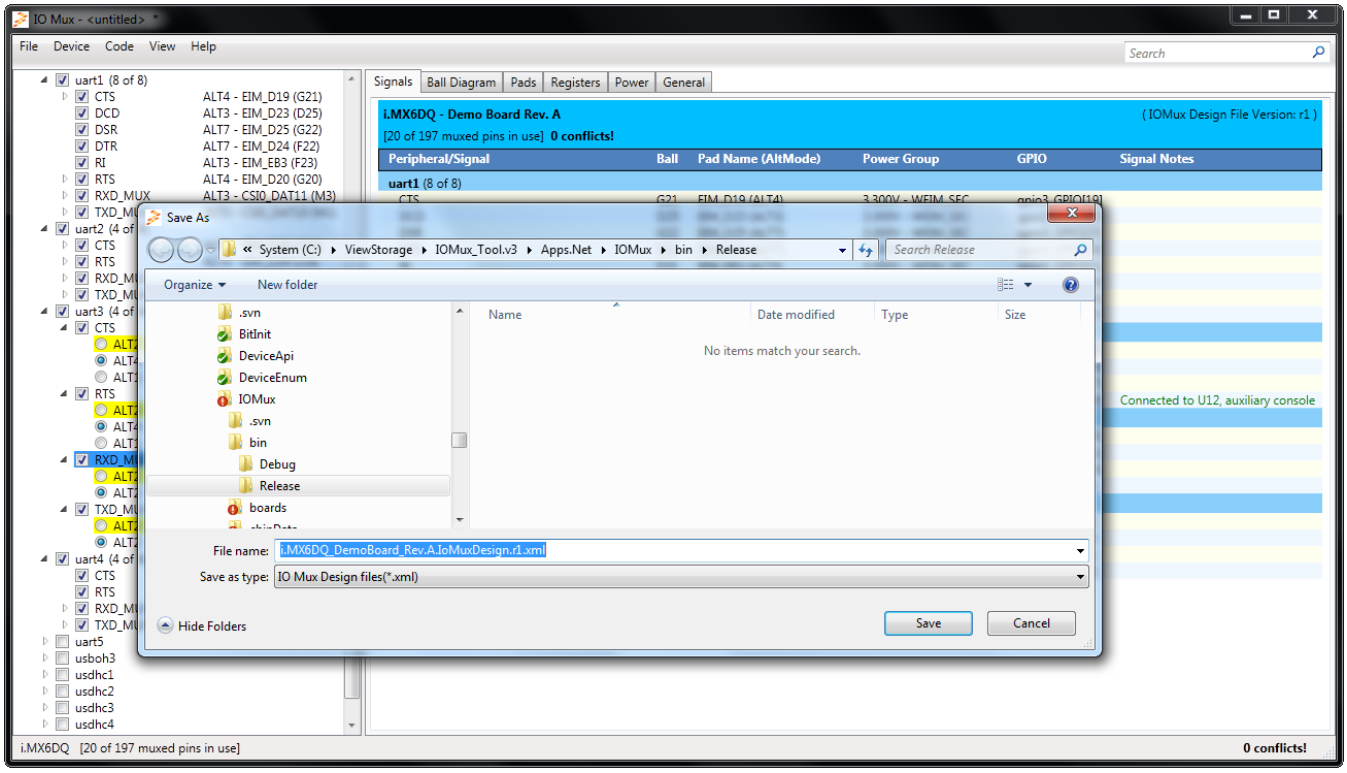

Figure 25. The File > Save dialog box showing default filename for saving design info.

Information from the General tab is used to construct a default file name for the design.

4.9 Generating IOMUXC Configuration Code

Generating code to configure the IOMUXC registers to route signals and configure pads according to the currently loaded board design is as easy as selecting the code style from the Code menu and then selecting Generate Code from the same menu. A bell will be heard after the code files have been written.

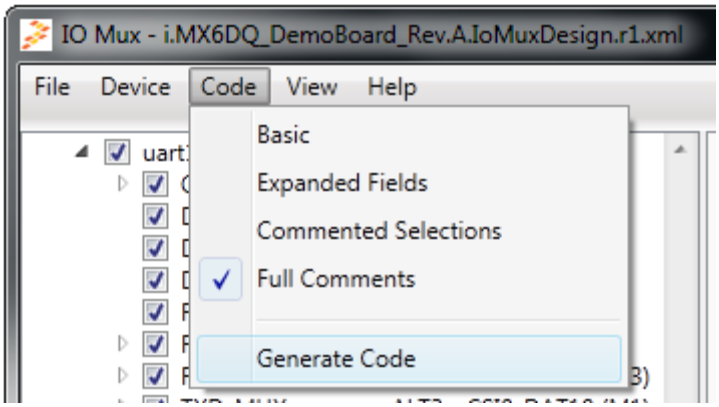

Figure 26. Generating code.

Examples of the various code styles and the type and location of the generated files are detailed in the Code Menu section of this document.

5 Advanced Usage Example with Explanation

In this section, we will cover “best practices” concepts to use the tool to create the cleanest, clearest code possible. The IOMux Tool application allows the designer to group signals logically, and even rename them, to better reflect their function and/or name on the schematics.

For this example, an excerpt of the i.MX6DQ Saber Tablet schematics is included for reference (Figure 27), and we will be working with the i.MX6DQ_Sabre_Tablet.Rev.A.IoMuxDesign.xml included in the IOMux Tool release package in the boards directory.

5.1 GPS Signals on the i.MX6DQ Sabre Tablet Reference Board

The user has the ability to drag signals from one module to another and/or create entirely new modules. This allows the signals to be grouped together so that a single <module_name>_iomux_config() call can be used to configure all of the relevant IOMUXC registers. For example, the i.MX6DQ Saber Tablet board supports a GPS interface.

Table 1. i.MX6DQ Pad to Signal Mapping

Ball	Ball Name	Original Function	Current Function
K21	EIM_EB0	GPIO2.GPIO[28]	GPS_RESET_B
D24	EIM_D18	GPIO3.GPIO[18]	GPS_PPS
L20	EIM_DA0	GPIO3.GPIO[0]	GPS_PWREN
F22	EIM_D24	UART3.TXD_MUX	GPS_TXD
G22	EIM_D25	UART3.RXD_MUX	GPS_RXD

Without grouping the signals together, configuration software would have to call uart3_iomux_config(), gpio2_iomux_config(), and gpio3_iomux_config() in order to route all the GPS signals to their proper places.

In this exercise, we will create a new module, gps, and associate all the above signals with it. Then, initialization software will only have to call gps_iomux_config() to configure the GPS IOMUXC registers, and no other modules will be affected due to routing signals that are not associated with the GPS module.

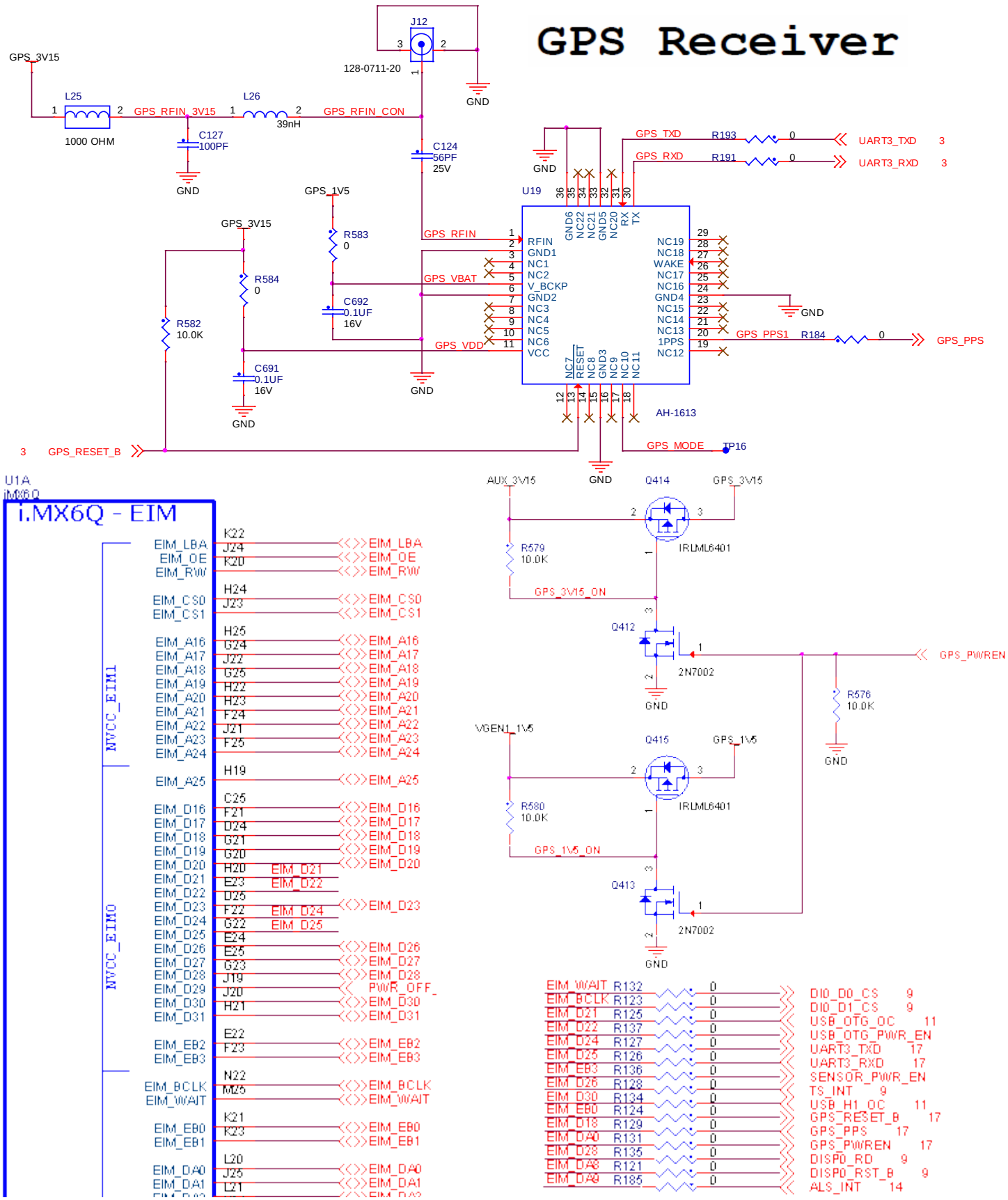

Figure 27. i.MX6DQ GPS Module Schematics

5.2 Create a New Module

Figure 28 shows the application window after the following sequence:

1. Open the i.MX6DQ_Sabre_Tablet.Rev.A.IoMuxDesign.xml design file.
2. Right-click on a module name or the device name in the tree in the left pane and select ‘New’.
3. Rename the ‘NewModule’ to gps.
4. Expand gpio2 and gpio3.

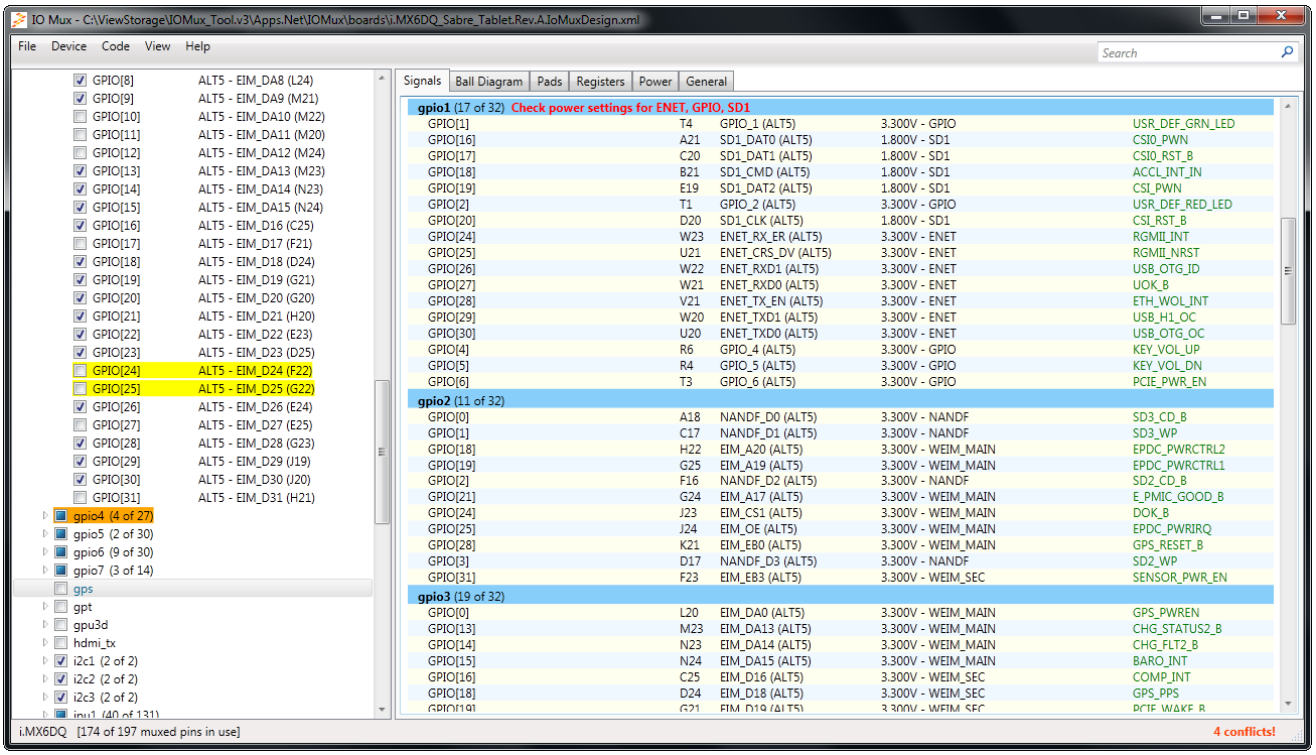

Figure 28. Application window after opening design file and creating the ‘gps’ module.

From Table 1, Figure 27, and Figure 28, we know what signals we need to add to our new gps module and where they are in the tool.

5.3 Drag GPS Signals to New ‘gps’ Module

Add the appropriate signals to the new module as follows:

5. Left-click on the gpio3.GPIO[0] and drag it to the gps module.
6. Left-click on the gpio3.GPIO[18] and drag it to the gps module.
7. Left-click on the gpio2.GPIO[28] and drag it to the gps module.

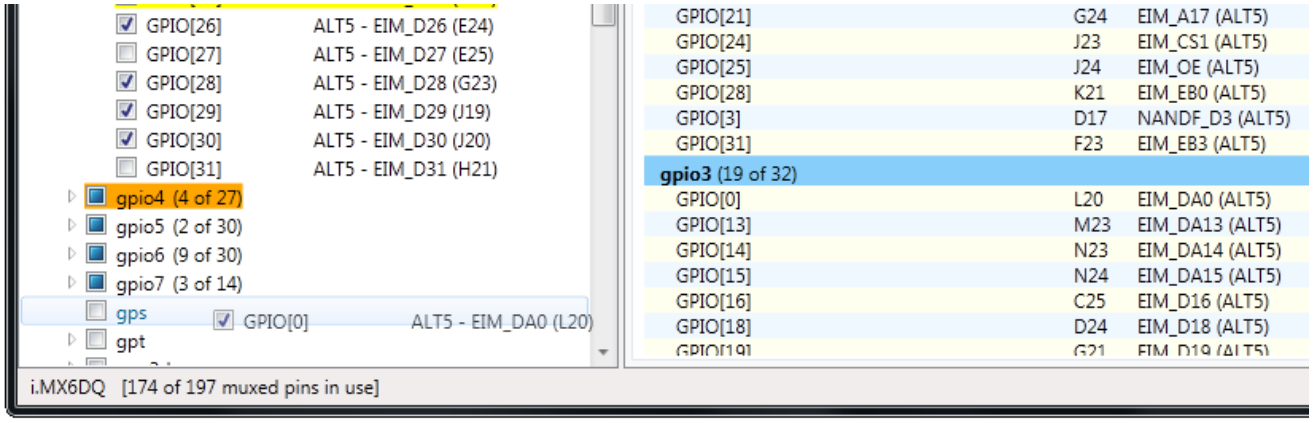

Figure 29. Dragging gpio3.GPIO[0] to the new ‘gps’ module.

*** EASE OF USE ***

8. Right-click->Rename the new ‘gps’ module to ‘uart-gps’ to make it easier to drag the UART3 signals into the ‘gps’ module (see Figure 30).

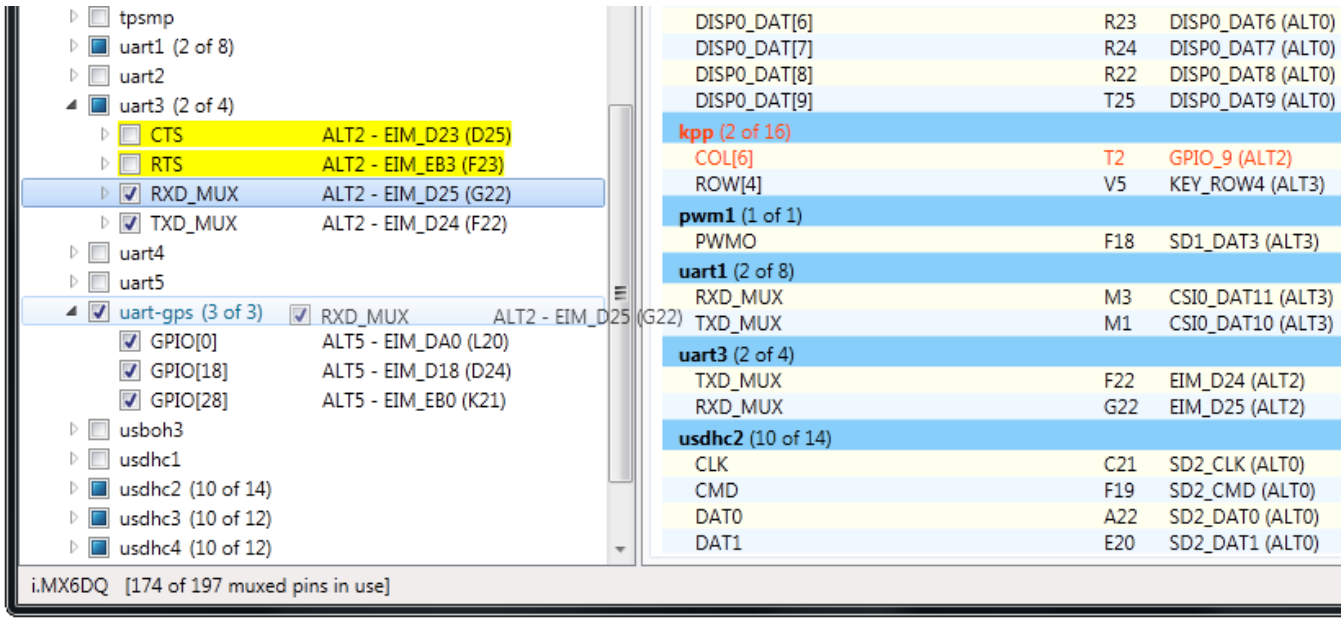

Figure 30. Temporarily renaming the destination module to facilitate signal relocation.

9. Left-click on the uart3.RXD_MUX and drag it to the uart-gps module.
10. Left-click on the uart3.TXD_MUX and drag it to the uart-gps module.
11. Rename the ‘uart-gps’ module back to ‘gps’.

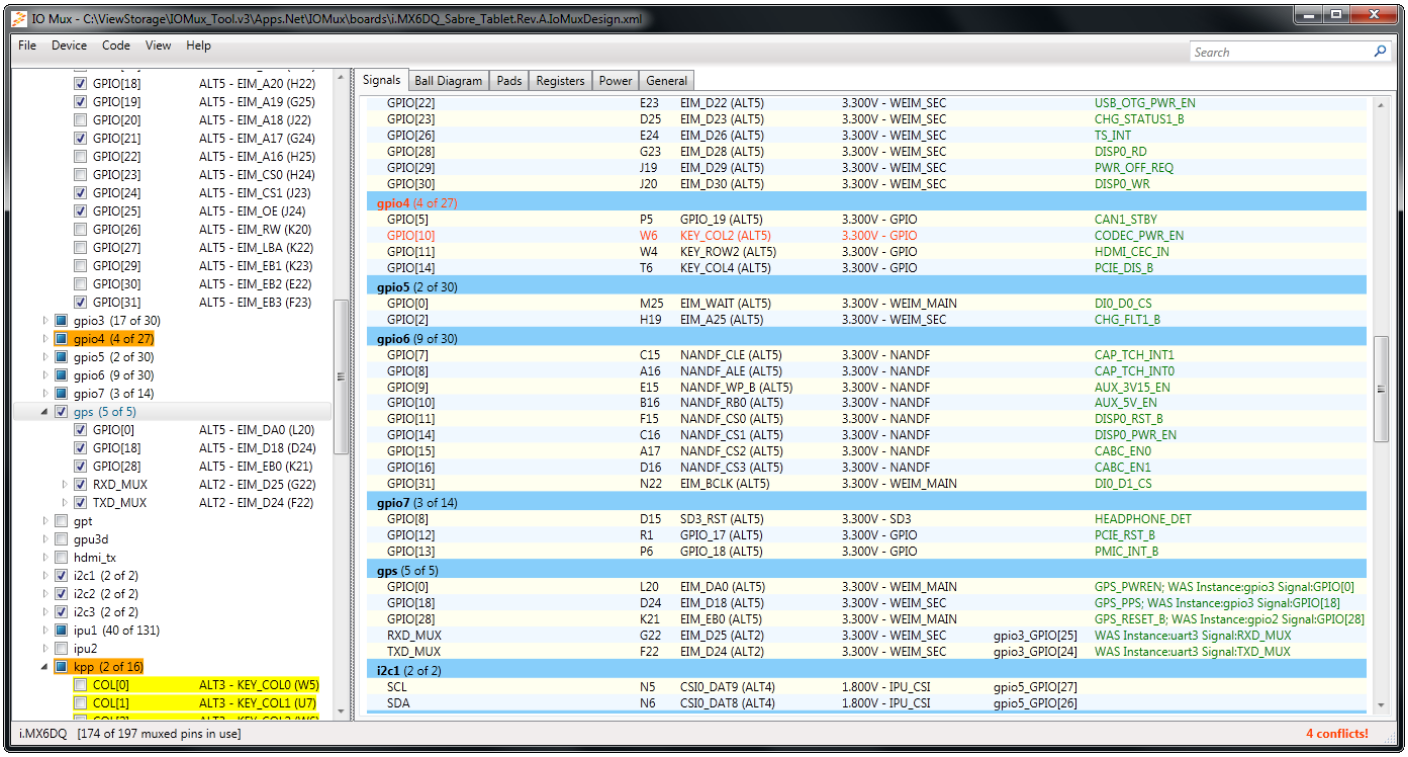

Figure 31. All the GPS signals under the new 'gps' module.

Note that the original location is maintained in the comments for the moved or renamed signals. Now let’s rename the signals to reflect their actual function from Table 1.

12. Left-click on gps.GPIO[0] and rename it to GPS_PWREN.
13. Left-click on gps.GPIO[18] and rename it to GPS_PPS.
14. Left-click on gps.GPIO[28] and rename it to GPS_RESET_B.
15. Left-click on gps.RXD_MUX and rename it to GPS_RXD.
16. Left-click on gps.TXD_MUX and rename it to GPS_TXD.
17. Clean up the comments since some of the information is extraneous now that the signals are logically grouped.

The completed ‘gps’ module and the corresponding IOMUXC configuration code are shown below.

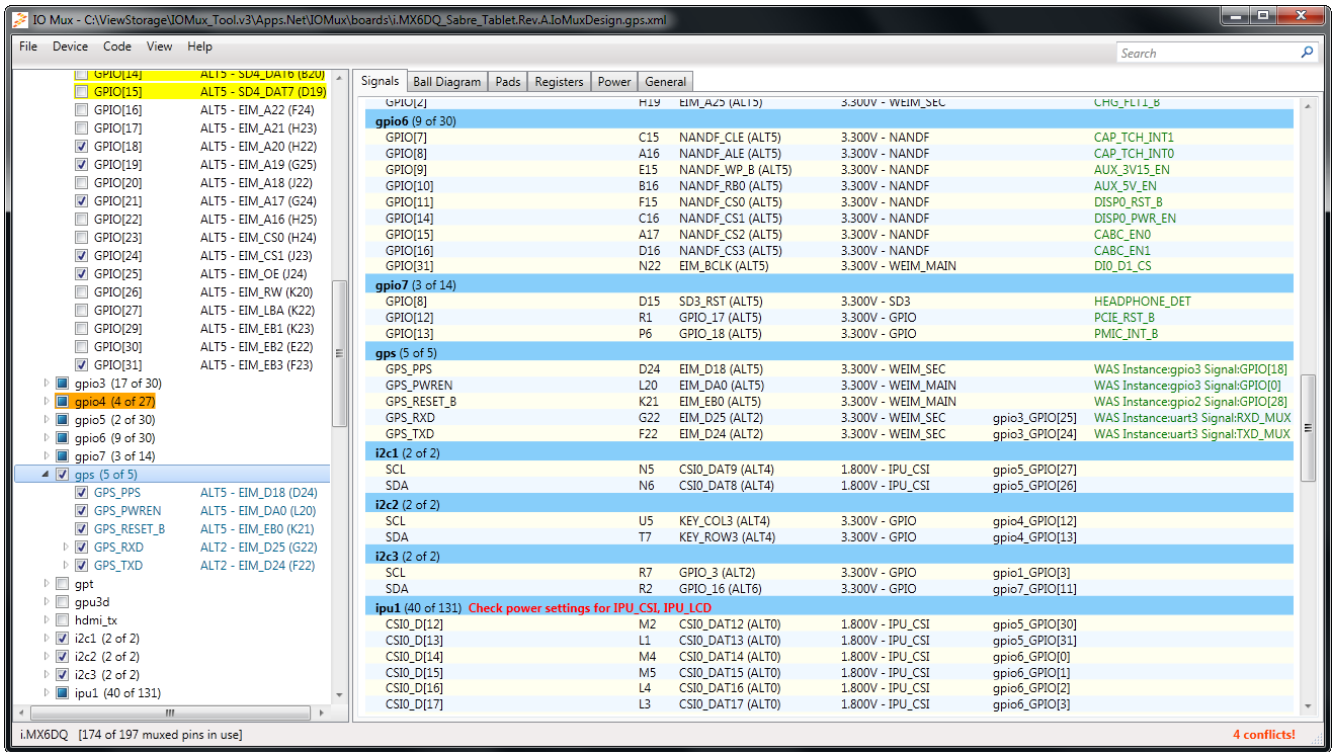

Figure 32. The completed 'gps' module.

```
// Function to config iomux for instance gps.
void gps_iomux_config(void)
{
    // Config gps.GPS_PWREN to pad EIM_DA0(L20)
    // WAS Instance:gpio3 Signal:GPIO[0]
    writel(0x00000005, IOMUXC_SW_MUX_CTL_PAD_EIM_DA0); // SION_DISABLED, ALT5
    // HYS_DISABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_FAST
    writel(0x0000B0B1, IOMUXC_SW_PAD_CTL_PAD_EIM_DA0);

    // Config gps.GPS_PPS to pad EIM_D18(D24)
    // WAS Instance:gpio3 Signal:GPIO[18]
    writel(0x00000005, IOMUXC_SW_MUX_CTL_PAD_EIM_D18); // SION_DISABLED, ALT5
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D18);

    // Config gps.GPS_RESET_B to pad EIM_EB0(K21)
    // WAS Instance:gpio2 Signal:GPIO[28]
    writel(0x00000005, IOMUXC_SW_MUX_CTL_PAD_EIM_EB0); // SION_DISABLED, ALT5
    // HYS_DISABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_FAST
    writel(0x0000B0B1, IOMUXC_SW_PAD_CTL_PAD_EIM_EB0);

    // Config gps.GPS_TXD to pad EIM_D24(F22)
    // WAS Instance:uart3 Signal:TXD_MUX
    writel(0x00000002, IOMUXC_SW_MUX_CTL_PAD_EIM_D24); // SION_DISABLED, ALT2
    writel(0x00000001, IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT); // SEL_EIM_D25_ALT2
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D24);

    // Config gps.GPS_RXD to pad EIM_D25(G22)
    // WAS Instance:uart3 Signal:RXD_MUX
    writel(0x00000002, IOMUXC_SW_MUX_CTL_PAD_EIM_D25); // SION_DISABLED, ALT2
    writel(0x00000001, IOMUXC_UART3_IPP_UART_RXD_MUX_SELECT_INPUT); // SEL_EIM_D25_ALT2
    // HYS_ENABLED, PUS_100KOHM_PU, PUE_PULL, PKE_ENABLED, ODE_DISABLED, SPD_100MHZ, DSE_40OHM, SRE_SLOW
    writel(0x0001B0B0, IOMUXC_SW_PAD_CTL_PAD_EIM_D25);
}
```

Revision History

Table 2. Revision History

Rev. Number	Date	Substantive Change
A	9/24/2009	Initial creation
B	10/7/2011	Document additional features including: Code Generation, Default ALT-modes, Power consistency checking, non-muxed signal inclusion, addition of design tracking information, and general bug fixes.
C	1/27/2012	Added Import/Export menu items and edited SaveAs. Documented Advanced Usage with custom signal grouping.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number:
Rev:B
Date: 10/2011