AM57xx Sitara™ IO Configuration Requirements

Texas Instruments Sitara™ Family of Products

Application Report

Literature Number: SPRAC44A
August 2016–Revised August 2017

Contents

Copyright © 2016–2017, Texas Instruments Incorporated

List of Figures

List of Tables

Copyright © 2016–2017, Texas Instruments Incorporated

Revision History

Changes from September 1, 2016 to August 30, 2017 (from * Revision (September 2016) to A Revision) **Page**

- Removed dos2unix usage note. The converter script now works with windows line endings. 14
- Removed user edits of boardPadDelayTune.h file. The most recent tool now automatically uncomments the proper #defines based on GUI configuration from the mode selection menus. 15
- Updated IO Delay Output Conversion Script (am57xx_generate_pin_config_data.pl) and migrated to git.ti.com for direct download. 20

Copyright © 2016–2017, Texas Instruments Incorporated

AM57xx Sitara™ IO Configuration Requirements

This application report provides an overview of the software requirements for configuring the AM57xx IOs.

To ensure the IO timings in the AM57xx data manual over the lifetime of the device, the AM57xx software must implement the proper pad configuration requirements. This application report provides an overview of the required software configurations, along with a tutorial on using the Pinmux Tool (PMT) to select and export the configurations for use in Linux and TI-RTOS platforms.

Copyright © 2016–2017, Texas Instruments Incorporated

1.1 Introduction

To ensure the IO timing values published in the *Timing Requirements* and *Switching Characteristics* tables of the AM57xx data manual over the lifetime of the device, the AM57xx software must implement the proper pad configuration requirements. The impact of not following these requirements may not be observed immediately. However, in the long term, failure to adhere to this procedure may cause system issues.

It is recommended to read the *Pad Configuration* section of the *Control Module* in the AM57xx TRM to become familiar with the detailed requirements including the Pad Configuration registers, IOSETs, virtual IO timing modes, manual IO timing modes, isolation requirements, and IO delay recalibration. All of these requirements are automatically configured when the PinMux Tool is used in conjunction with the Board Library within Processor SDK RTOS or in conjunction with U-Boot when running Linux.

To meet these requirements, the system designer should:

1. Utilize the PinMux Tool (PMT) to select the desired peripherals and peripheral operating modes.
2. Use the PinMux Tool to generate the desired files to use with RTOS or Linux. These files include the pad configuration register and IODELAY register values.
3. When using TI-RTOS replace the pinmux configuration files within the Platform Development Kit (PDK) Board Library with those generated by the PinMux Tool. When using Linux a script is provided to convert the generic file formats to a format understood by U-Boot. Software details are in Section 1.6.

This application report summarizes the AM57xx pad configuration requirements (primarily abstracted for the user by the PMT), and also outlines how to use the PMT to select and export the pad configurations.

1.2 Hardware IO Configurations

The required IO configuration operation that software is responsible for implementing includes the following:

- **Pad Multiplexing** (CTRL_CORE_PAD_*[3:0] MUXMODE) – Selection must be compliant with the IOSETs defined in the data manual. An IOSet is a specific combination of muxing options. These IOSets must be selected to ensure the peripheral timing published in the data manual.
  - See the *IOSETs* section in the AM57xx TRM and the IOSET tables in the *Timing Requirements* and *Switching Characteristics* section of the AM57xx data manual.
  - The PinMux Tool will not allow pin selections that do not match a supported IOSET.
- **Slew Control** (CTRL_CORE_PAD_*[19] SLEWCONTROL) – Slew settings must be left at their default values with one exception: the vout*_* signals require (in the AM571x) or recommend (in the AM572x) SLOW slew, instead of the default FAST slew.
  - See the *1.8-V and 3.3-V Signal Transition Rates* section of the AM57xx data manual.
  - The PinMux Tool is configured to generate the correct register values to satisfy this requirement.
- **Virtual and Manual IO Timing Modes** – IO timing modes must be configured as required for the desired IP mode of operation.
  - See the *Virtual IO Timing Modes* and *Manual IO Timing Modes* sections in the AM57xx TRM.
  - See Section 1.3 for examples of IO timing mode selection.
  - The PMT should be used to abstract the virtual and manual IO timing mode selection, as described in Section 1.5.
- **IO Delay Recalibration** – Performed after adjusting the AVS voltage for VDD_CORE_L voltage domain.
  - See the *IO Delay Recalibration* section in the AM57xx TRM.
  - The PDK Secondary Boot Loader (SBL) or U-boot are configured to handle this requirement.

The pin multiplexing and virtual and manual IO timing mode selection is abstracted by the PMT. Thus, TI recommends using this tool to accurately select and configure the device IOs. The tool generates output configuration files used by software to auto-configure the IOs. More details are available in Section 1.6.1.

Copyright © 2016–2017, Texas Instruments Incorporated

There are two scenarios in which software can configure the IOs: static and run-time. The run-time configuration is only supported for MMC peripherals. All other IOs must be done using the static configuration at boot time. Both scenarios are explained briefly in following sections.

1.2.1 Static IO Configurations (Boot Time)

Any changes to the Pad Configuration registers or IODELAYCONFIG registers can potentially result in an undesirable state (such as output state changes or output enable changes) on the associated IOs. To ensure an IO state, the device pins must be placed in isolation mode when making any changes to the Pad Configuration registers or IODELAYCONFIG module registers. See the *Isolation Requirements* section in the AM57xx TRM for details. Note that while the IOs are isolated, code must execute only from internal SRAM, and the isolated IO interfaces should not be actively used. This requirement is most easily accomplished at boot time, as the secondary bootloaders (following the ROM bootloader) and Linux MLO execute from OCMC RAM.

1.2.2 Run-Time IO Configurations (MMC Interface Signals Only)

With MMC, the IOs must be configured or reconfigured during run-time, without isolating the IOs. Because reconfiguring the IOs is a serial process, the undesirable state described in Section 1.2.1 does not occur simultaneously on two or more pads. This mitigates the risk for MMC, and allows for run-time IO configuration, because the interface requires concurrent toggling of two or more signals (for example Clock, Command, or Chip Select) to be registered as a valid operation on the bus.

1.2.3 MMC1 Special Considerations

The MMC1 IOs and PBIAS have an additional Control Module register, CTRL_CORE_CONTROL_PBIAS, that is used to power-up and power-down the IOs and select their voltage (1.8 V vs 3.3 V). This register must be programmed as part of the isolation sequence, and when configuring these IOs for use and power-down. See the *Isolation Requirements* and *Pad Multiplexing* sections in the AM57xx TRM for more details. The MMC driver included with the Processor SDK supports automatic SD/MMC speed and voltage negotiation.

Copyright © 2016–2017, Texas Instruments Incorporated

1.3 Manual IO Timing Mode Example Configuration

Manual IO timing modes are IO timing settings that must be calculated and programmed by system software, based on seed values in the data manual. The following example illustrates a possible configuration for the AM572x video input port VIN4A.

1.3.1 Pin Multiplexing

The pin multiplexing selection must be compliant with the IOSETs defined in the data manual. Therefore, in this example, select the balls to be used for the VIN4A interface from within a single IOSET (or column) of the VIN4 IOSETs table in the AM572x data manual, as shown in Table 1-1.

Table 1-1. VIN4 IOSETs

Signals	IOSET1		IOSET2		IOSET3	
	BALL	MUX	BALL	MUX	BALL	MUX
vin4a						
vin4a_d0	R6	4	B7	3	B14	8
vin4a_d1	T9	4	B8	3	J14	8
vin4a_d2	T6	4	A7	3	G13	8
vin4a_d3	T7	4	A8	3	J11	8
vin4a_d4	P6	4	C9	3	E12	8
vin4a_d5	R9	4	A9	3	F13	8
vin4a_d6	R5	4	B9	3	C12	8
vin4a_d7	P5	4	A10	3	D12	8
vin4a_d8	U2	4	E8	3	E15	8
vin4a_d9	U1	4	D9	3	A20	8
vin4a_d10	P3	4	D7	3	B15	8
vin4a_d11	R2	4	D8	3	A15	8
vin4a_d12	K7	4	A5	3	D15	8
vin4a_d13	M7	4	C6	3	B16	8
vin4a_d14	J5	4	C8	3	B17	8
vin4a_d15	K6	4	C7	3	A17	8
vin4a_d16	-	-	F11	3	C18	8
vin4a_d17	-	-	G10	3	A21	8
vin4a_d18	-	-	F10	3	G16	8
vin4a_d19	-	-	G11	3	D17	8
vin4a_d20	-	-	E9	3	AA3	8
vin4a_d21	-	-	F9	3	AB9	8
vin4a_d22	-	-	F8	3	AB3	8
vin4a_d23	-	-	E7	3	AA4	8
vin4a_hsync0	R3/ P7	4 / 4	C11	3	E21	8
vin4a_vsync0	T2/ N1	4 / 4	E11	3	F20	8

IOSET2 is chosen for this example. Therefore, the CTRL_CORE_PAD_*[3:0] MUXMODE for the balls listed in IOSET2 should be programmed to select the vin4a signals.

The pin multiplexing configurations are recommended to be done as part of the IO delay recalibration sequence described in the *IO Delay Recalibration* section in the AM57xx TRM.

Copyright © 2016–2017, Texas Instruments Incorporated

1.3.2 Virtual versus Manual IO Timing Mode Determination

To determine if a virtual or manual IO timing mode is required for VIN4A IOSET2, the system designer should find the desired mode of operation in *Modes Summary* table found in the *Timing Requirements and Switching Characteristics* section of the AM57xx data manual. A sample excerpt of the VIP section of this table is shown in Table 1-2.

Table 1-2. Modes Summary

Virtual or Manual IO Mode Name	Data Manual Timing Mode
VIP	
VIP1_MANUAL1	VIN1A/1B/2A Rise-Edge Capture Mode Timings
VIP1_2B_MANUAL1	VIN2B Rise-Edge Capture Mode Timings
VIP1_MANUAL2	VIN1A/1B/2A Fall-Edge Capture Mode Timings
VIP1_2B_MANUAL2	VIN2B Fall-Edge Capture Mode Timings
VIP2_MANUAL1	VIN3A and VIN3B IOSET1 Rise-Edge Capture Mode Timings
VIP2_4A_MANUAL1	VIN4A IOSET1/2 Rise-Edge Capture Mode Timings
VIP2_4A_IOSET3_MANUAL1	VIN4A IOSET3 Rise-Edge Capture Mode Timings
VIP2_4B_MANUAL1	VIN4B Rise-Edge Capture Mode Timings
VIP2_3B_IOSET2_MANUAL1	VIN3B IOSET2 Rise-Edge Capture Mode Timings
…	…
…	…

Rise-edge capture timings are chosen for this example, which corresponds to manual IO timing mode VIP2_4A_MANUAL1.

1.3.3 Manual Mode Example

As indicated in the example in Section 1.3.2, the required manual IO timing mode values for VIP2_4A_MANUAL1 can be found in the manual functions mapping for VIP2 4A IOSET3 table in the AM572x data manual, as shown in Table 1-3.

Table 1-3. Manual Functions Mapping for VIP2 4A

BALL	BALL NAME	VIP2_4A_MANUAL1			MUXMODE
		A_DELAY (ps)[(1)]	G_DELAY (ps)[(1)]	CFG REGISTER	3
…	…	…	…	…	…
D11	vout1_clk	2388	0	CFG_VOUT1_CLK_IN	vin4a_fld0
F11	vout1_d0	2747	236	CFG_VOUT1_D0_IN	vin4a_d16
G10	vout1_d1	2593	208	CFG_VOUT1_D1_IN	vin4a_d17
D7	vout1_d10	2559	27	CFG_VOUT1_D10_IN	vin4a_d10
D8	vout1_d11	2754	12	CFG_VOUT1_D11_IN	vin4a_d11
…	…	…	…	…	…

(1) The A_Delay and G_Delay values are valid for AM572x SR1.1.

The values in the A_DELAY and G_DELAY column for each ball are seed values for the equations provided in the *Manual IO Timing Modes* section of the TRM. These A_DELAY and G_DELAY seed values must be used to calculate the correct values to be written to the associated CFG_x_IN, CFG_x_OEN, CFG_x_OUT registers listed in the CFG REGISTER column. TI distributions of Linux u-boot and RTOS natively have IO delay support for the formally mentioned equations and calculations.

The manual IO timing mode configurations are required as part of the IO delay recalibration sequence described in the *IO Delay Recalibration* section in the AM57xx TRM.

Copyright © 2016–2017, Texas Instruments Incorporated

1.4 Virtual IO Timing Mode Example Configuration

Virtual IO timing modes are predefined IO timing settings that are coded in the device ROM. The following example illustrates a possible configuration for the AM572x SR1.1 QSPI.

1.4.1 Pin Multiplexing

There is only one pin multiplexing option for QSPI, and thus no IOSET requirements. The CTRL_CORE_PAD_*[3:0] MUXMODE for the associated gpmc_* balls should be programmed to the selected qspi1_* signals. The pin multiplexing configurations are recommended to be done as part of the IO delay recalibration sequence described in the *IO Delay Recalibration* section of the TRM.

1.4.2 Virtual versus Manual IO Timing Mode Determination

To determine if a virtual or manual IO timing mode is required for QSPI, find the desired mode of operation in the Modes Summary table found in the data manual. An excerpt of the QSPI section of this table is shown in Table 1-4.

Table 1-4. Modes Summary

Virtual or Manual IO Mode Name	Datasheet Timing Mode
QSPI	
No Virtual or Manual IO Timing Mode Required	QSPI Mode 3 Default Timing Mode
QSPI1_VIRTUAL1	QSPI Mode 3 Alternate Timing Mode 1
QSPI1_VIRTUAL2	QSPI Mode 3 Alternate Timing Mode 2
QSPI_MODE0_MANUAL1	QSPI Mode 0 Default Timing Mode
QSPI_MODE0_MANUAL2	QSPI Mode 0 Alternate Timing Mode 1
QSPI_MODE0_MANUAL3	QSPI Mode 0 Alternate Timing Mode 2
…	…

Mode 3 alternate timing mode 1 is chosen for this example, which corresponds to virtual IO timing mode QSPI1_VIRTUAL1.

1.4.3 Virtual IO Timing Mode

The required virtual IO timing mode values for QSPI1_VIRTUAL1 can be found in Table 1-5.

Table 1-5. Virtual Functions Mapping for QSPI

BALL NUMBER	BALL NAME	Delay Mode Value		MUXMODE[15:0]
		QSPI1_VIRTUAL1	QSPI1_VIRTUAL2	1
T7	gpmc_a3	9	8	qspi1_cs2
P6	gpmc_a4	9	8	qspi1_cs3
R3	gpmc_a13	11	10	qspi1_rtclk
T2	gpmc_a14	11	10	qspi1_d3
U2	gpmc_a15	11	10	qspi1_d2
U1	gpmc_a16	11	10	qspi1_d0
P3	gpmc_a17	11	10	qspi1_d1
R2	gpmc_a18	11	10	qspi1_sclk
P2	gpmc_cs2	11	10	qspi1_cs0
P1	gpmc_cs3	9	8	qspi1_cs1

The values in the QSPI1_VIRTUAL1 column for each ball should be written to into the associated CTRL_CORE_PAD_* registers as described in the *Virtual IO Timing Modes* section of the TRM.

The virtual IO timing mode configurations are recommended to be done as part of the IO delay recalibration sequence described in the *IO Delay Recalibration* section of the TRM.

Copyright © 2016–2017, Texas Instruments Incorporated

1.5 PinMux Tool

NOTE: The below information provides a detailed example for configuring the AM57xx device IO delays. For the general quick start guide, refer to http://processors.wiki.ti.com/index.php/TI_PinMux_Tool_v4.

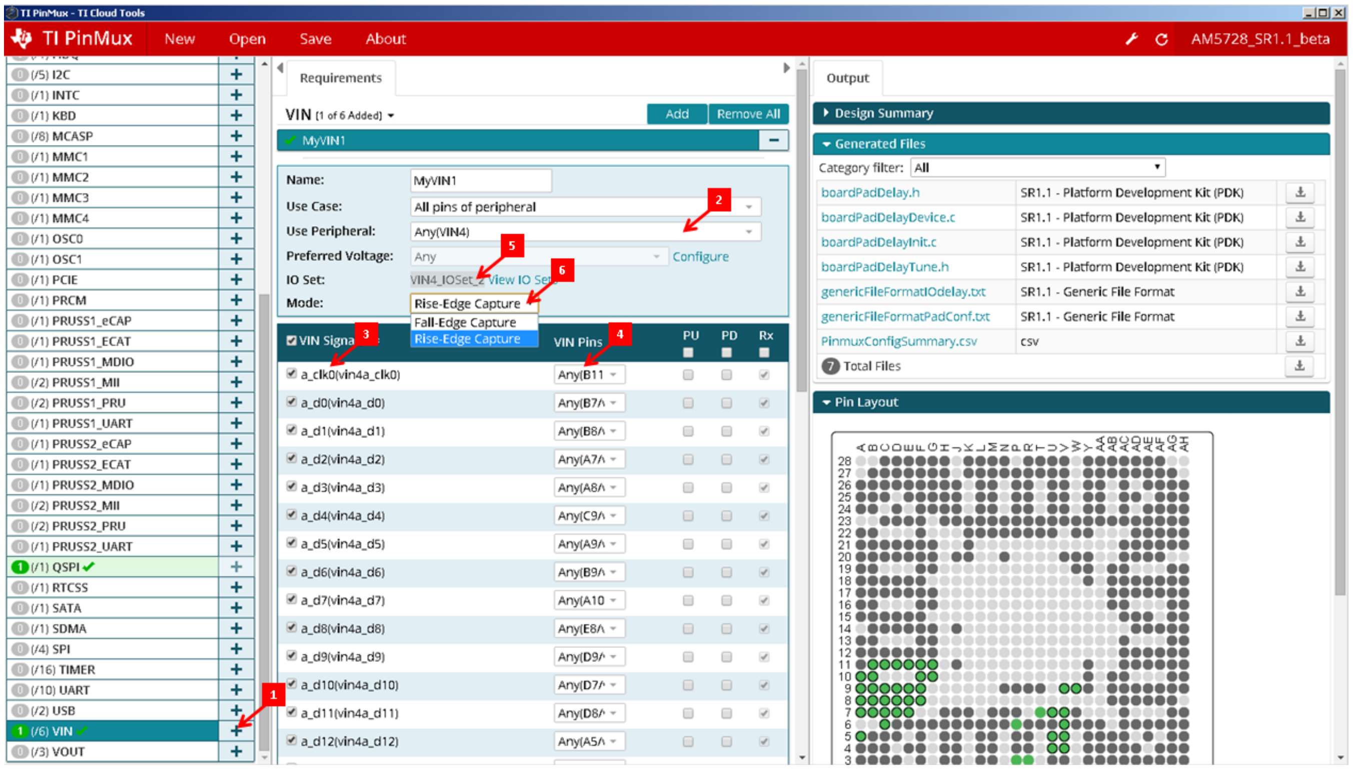
The PinMux tool can be used to configure and output the pin multiplexing and virtual and manual IO timing mode selections, as shown in Figure 1-1. If using the VIN4A configuration described earlier in this document, the GUI can be used to configure vin4a_d0 as follows:

1. VIN can be added as a required peripheral.
2. vin4a can be selected in the Use Peripheral drop-down menu.
3. vin4a_x signals can be selected in the Signals menu.
4. vout1_x pins can be selected in the Pins menu.
5. PinMux tool automatically solves for an existing IOSET.
6. VIN4A rise-edge capture mode timings can be selected under the Mode drop-down menus.

CAUTION

The selection from the Use Case, Peripheral, IOset, and Mode menus must match the required configuration to output the correct IO delay values.

Figure 1-1. PinMux Tool



(/5) I2C
(/1) INTC
(/1) KBD
(/8) MCASP
(/1) MMC1
(/1) MMC2
(/1) MMC3
(/1) MMC4
(/1) OSC0
(/1) OSC1
(/1) PCIE
(/1) PRCM
(/1) PRUSS1_eCAP
(/1) PRUSS1_ECAT
(/1) PRUSS1_MDIO
(/2) PRUSS1_MII
(/2) PRUSS1_PRU
(/1) PRUSS1_UART
(/1) PRUSS2_eCAP
(/1) PRUSS2_ECAT
(/1) PRUSS2_MDIO
(/2) PRUSS2_MII
(/2) PRUSS2_PRU
(/1) PRUSS2_UART
1 (/1) QSPI ✓
(/1) RTCSS
(/1) SATA
(/1) SDMA
(/4) SPI
(/16) TIMER
(/10) UART
(/2) USB
1 (/6) VIN ✓
(/3) VOUT

VIN (1 of 6 Added)

Name:	MyVIN1
Use Case:	All pins of peripheral
Use Peripheral:	Any(VIN4)
Preferred Voltage:	Any
IO Set:	VIN4_IOSet_2
Mode:	Rise-Edge Capture

Fall-Edge Capture
Rise-Edge Capture

VIN Signals	VIN Pins	PU	PD	Rx
✓ a_clk0(vin4a_clk0)	Any(B11			✓
✓ a_d0(vin4a_d0)	Any(B7/			✓
✓ a_d1(vin4a_d1)	Any(B8/			✓
✓ a_d2(vin4a_d2)	Any(A7/			✓
✓ a_d3(vin4a_d3)	Any(A8/			✓
✓ a_d4(vin4a_d4)	Any(C9/			✓
✓ a_d5(vin4a_d5)	Any(A9/			✓
✓ a_d6(vin4a_d6)	Any(B9/			✓
✓ a_d7(vin4a_d7)	Any(A10			✓
✓ a_d8(vin4a_d8)	Any(E8/			✓
✓ a_d9(vin4a_d9)	Any(D9/			✓
✓ a_d10(vin4a_d10)	Any(D7/			✓
✓ a_d11(vin4a_d11)	Any(D8/			✓
✓ a_d12(vin4a_d12)	Any(A5/			✓

Category filter: All

boardPadDelay.h	SR1.1 - Platform Development Kit (PDK)
boardPadDelayDevice.c	SR1.1 - Platform Development Kit (PDK)
boardPadDelayInit.c	SR1.1 - Platform Development Kit (PDK)
boardPadDelayTune.h	SR1.1 - Platform Development Kit (PDK)
genericFileFormatIOdelay.txt	SR1.1 - Generic File Format
genericFileFormatPadConf.txt	SR1.1 - Generic File Format
PinmuxConfigSummary.csv	csv
7 Total Files	

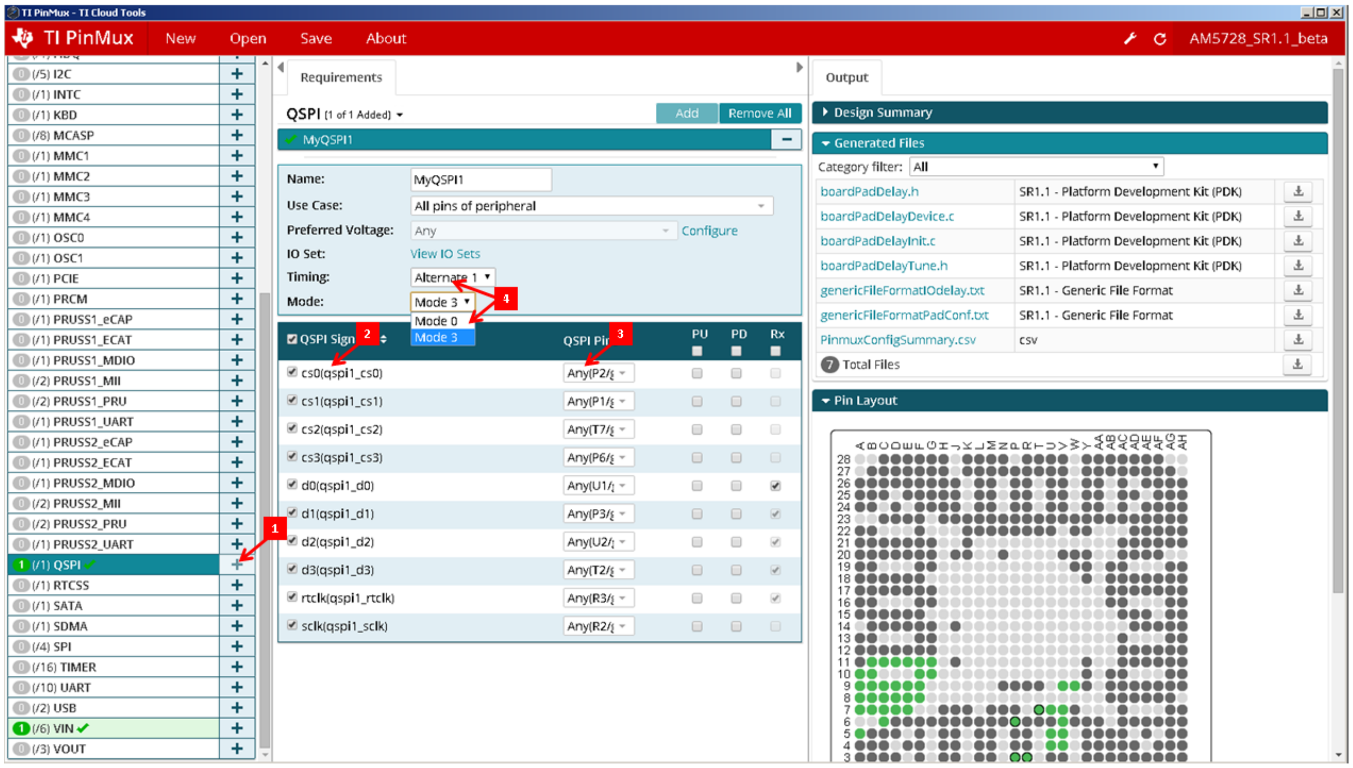
If using the QSPI configuration described earlier in this document, the Search View can be used to configure qspi1_d0 as follows:

1. QSPI can be added as a required peripheral.
2. qspi1_x signals can be selected in the Signal menu.
3. qspi1_x pins can be selected in the Pins menu.

Copyright © 2016–2017, Texas Instruments Incorporated

4. QSPI1 Mode 3 Alternate Timings 1 can be selected under the Mode and Timing drop-down menus, as shown in Figure 1-2.

Figure 1-2. Mode and Timing Drop-Down Menus



QSPI (1 of 1 Added)

MyQSPI1

Name:	MyQSPI1
Use Case:	All pins of peripheral
Preferred Voltage:	Any — Configure
IO Set:	View IO Sets
Timing:	Alternate 1
Mode:	Mode 3 (Mode 0, Mode 3)

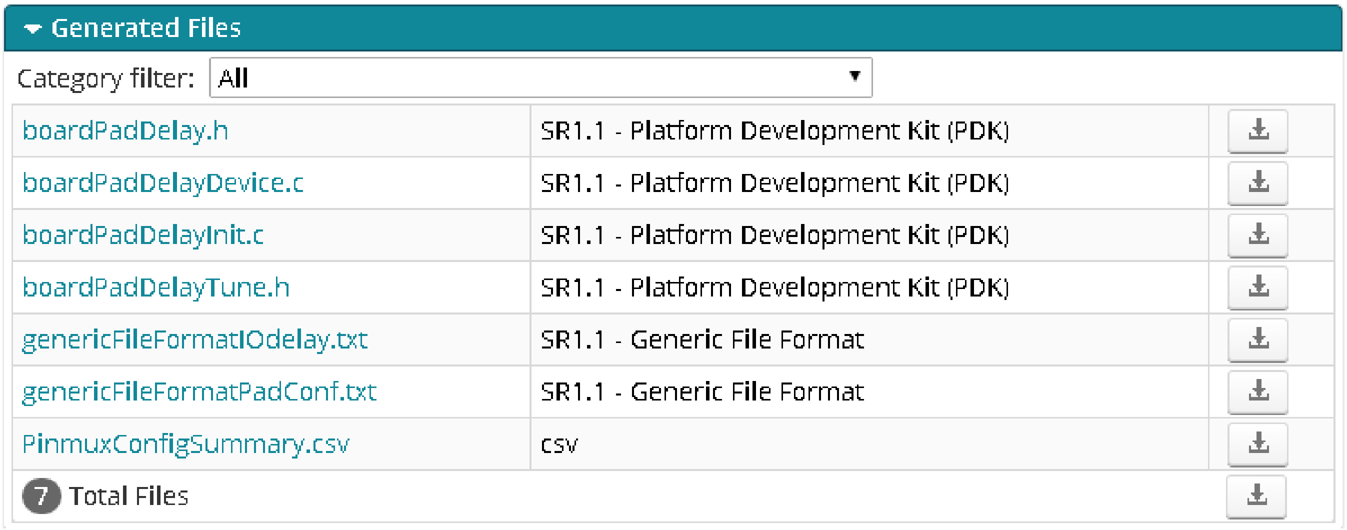
QSPI Sign	QSPI Pir	PU	PD	Rx
✓ cs0(qspi1_cs0)	Any(P2/	☐	☐	☐
✓ cs1(qspi1_cs1)	Any(P1/	☐	☐	☐
✓ cs2(qspi1_cs2)	Any(T7/	☐	☐	☐
✓ cs3(qspi1_cs3)	Any(P6/	☐	☐	☐
✓ d0(qspi1_d0)	Any(U1/	☐	☐	✓
✓ d1(qspi1_d1)	Any(P3/	☐	☐	✓
✓ d2(qspi1_d2)	Any(U2/	☐	☐	✓
✓ d3(qspi1_d3)	Any(T2/	☐	☐	✓
✓ rtclk(qspi1_rtclk)	Any(R3/	☐	☐	✓
✓ sclk(qspi1_sclk)	Any(R2/	☐	☐	☐

Category filter: All

boardPadDelay.h	SR1.1 - Platform Development Kit (PDK)
boardPadDelayDevice.c	SR1.1 - Platform Development Kit (PDK)
boardPadDelayInit.c	SR1.1 - Platform Development Kit (PDK)
boardPadDelayTune.h	SR1.1 - Platform Development Kit (PDK)
genericFileFormatIOdelay.txt	SR1.1 - Generic File Format
genericFileFormatPadConf.txt	SR1.1 - Generic File Format
PinmuxConfigSummary.csv	csv
7 Total Files	

Once all pads have been configured, the pin multiplexing and IO delay configuration settings can be exported into a variety of configuration file formats through the File Generation menu, as shown in Figure 1-3. See Section 1.6.1 for details on exported file formats.

Figure 1-3. File Generation Menu


Generated Files

Category filter: All

boardPadDelay.h	SR1.1 - Platform Development Kit (PDK)
boardPadDelayDevice.c	SR1.1 - Platform Development Kit (PDK)
boardPadDelayInit.c	SR1.1 - Platform Development Kit (PDK)
boardPadDelayTune.h	SR1.1 - Platform Development Kit (PDK)
genericFileFormatIOdelay.txt	SR1.1 - Generic File Format
genericFileFormatPadConf.txt	SR1.1 - Generic File Format
PinmuxConfigSummary.csv	csv
7 Total Files	

Copyright © 2016–2017, Texas Instruments Incorporated

1.6 TI EVM Example

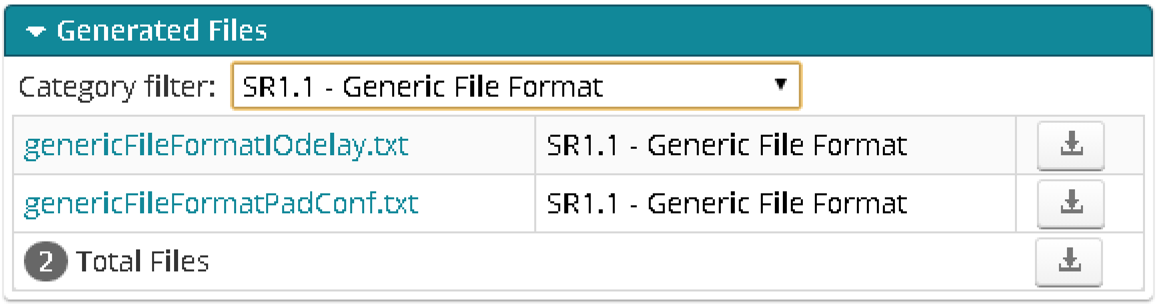
This section gives information about IO delay configuration with the TI AM572x GP EVM as an example. TI has provided PinMux configuration EVM files at http://www.ti.com/tool/tmdsevm572x.

1.6.1 PinMux Export File Types

1.6.1.1 Linux

For Linux® support, the PinMux tool exports the the Generic File Format package, as shown in Figure 1-4.

Figure 1-4. Generic File Format Package



genericFileFormatIOdelay.txt	SR1.1 - Generic File Format	
genericFileFormatPadConf.txt	SR1.1 - Generic File Format	
2 Total Files		

This file package contains a file for the padconf registers (genericFileFormatPadConf.txt) and a file containing the manual IO delay registers and values (genericFileFormatIOdelay.txt). The format is generic (similar to a .csv file), and can be post processed by a custom script to export whatever code format is needed by the final application.

genericFileFormatPadConf.txt: This is the main file containing the padconf settings in the following format:

- First column is the padconf register physical absolute address
- Second column is the register value (includes virtual mode configurations when applicable)
- Third column is the ball number
- Fourth column is the padconf register name
- Fifth column is the ball signal name (or muxmode 0 signal)
- Sixth column is the current muxmode signal selected

```
/* MMC1_VIRTUAL1 */
0x4A003754  0x101B0 W6   CTRL_CORE_PAD_MMC1_CLK  mmc1_clk    mmc1_clk
/* MMC1_VIRTUAL1 */
0x4A003758  0x501B0 Y6   CTRL_CORE_PAD_MMC1_CMD  mmc1_cmd    mmc1_cmd
/* MMC1_VIRTUAL1 */
0x4A00375C  0x501B0 AA6 CTRL_CORE_PAD_MMC1_DAT0 mmc1_dat0   mmc1_dat0
...
/* QSPI1_VIRTUAL1 */
0x4A003488  0x101B1 R2   CTRL_CORE_PAD_GPMC_A18  gpmc_a18    qspi1_sclk
/* QSPI1_VIRTUAL1 */
0x4A003474  0x501B1 R3   CTRL_CORE_PAD_GPMC_A13  gpmc_a13    qspi1_rtclk
/* QSPI1_VIRTUAL1 */
0x4A0034B8  0x101B1 P2   CTRL_CORE_PAD_GPMC_CS2  gpmc_cs2    qspi1_cs0
...
/* VIP2_4A_MANUAL1 */
0x4A0035D0  0x50103 B11 CTRL_CORE_PAD_VOUT1_FLD vout1_fld   vin4a_clk0
/* VIP2_4A_MANUAL1 */
0x4A0035CC  0x50103 B10 CTRL_CORE_PAD_VOUT1_DE  vout1_de    vin4a_de0
/* VIP2_4A_MANUAL1 */
0x4A0035C8  0x50103 D11 CTRL_CORE_PAD_VOUT1_CLK vout1_clk   vin4a_fld0
```

Copyright © 2016–2017, Texas Instruments Incorporated

genericFileFormatIOdelay.txt: This file is empty if there is no manual mode configuration. Pads not having a manual mode are also missing. This file contains the manual mode settings in the following format:

- First column is the manual mode register physical absolute address
- Second column is the A_DELAY value (ps)
- Third column is the G_DELAY value (ps)
- Fourth column is the manual mode register name
- Fifth is the current muxmode signal selected

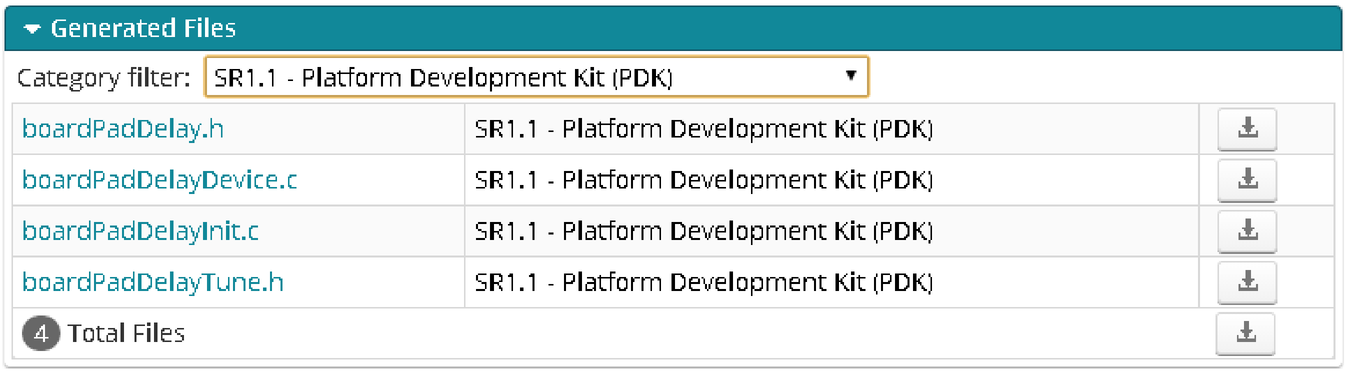
```
0x4844ACCC  0   0   CFG_VOUT1_FLD_IN    B11
0x4844ACC0  2347    0   CFG_VOUT1_DE_IN B10
0x4844AB94  2388    0   CFG_VOUT1_CLK_IN    D11
0x4844ACD8  2036    0   CFG_VOUT1_HSYNC_IN  C11
0x4844ACE4  1808    0   CFG_VOUT1_VSYNC_IN  E11
0x4844ABF4  2498    0   CFG_VOUT1_D16_IN    B7
0x4844AC00  2606    0   CFG_VOUT1_D17_IN    B8
0x4844AC0C  2410    0   CFG_VOUT1_D18_IN    A7
0x4844AC18  2074    0   CFG_VOUT1_D19_IN    A8
0x4844AC30  2617    91  CFG_VOUT1_D20_IN    C9
0x4844AC3C  2103    0   CFG_VOUT1_D21_IN    A9
...
```

Custom IO Delay Output Conversion Script for U-boot: The u-boot bootloader includes a perl script that takes the generic output from the PinMux tool (genericFileFormatPadConf.txt and genericFileFormatIOdelay.txt) and makes it usable in u-boot. This script (am57xx_generate_pin_config_data.pl) is available in Appendix B. Usage notes are included in Appendix C. These notes describe how to run the script, and where to include the generated .h files in the u-boot source.

1.6.1.2 TI-RTOS

For TI-RTOS support, the PinMux tool exports the Platform Development Kit (or PDK) files, as shown in Figure 1-5. These files are C-compilable files. The files can be logically separated into two groups: initial configuration and runtime configuration files.

Figure 1-5. Platform Development Kit Files



Category filter: SR1.1 - Platform Development Kit (PDK)

boardPadDelay.h	SR1.1 - Platform Development Kit (PDK)	
boardPadDelayDevice.c	SR1.1 - Platform Development Kit (PDK)	
boardPadDelayInit.c	SR1.1 - Platform Development Kit (PDK)	
boardPadDelayTune.h	SR1.1 - Platform Development Kit (PDK)	
4 Total Files		

NOTE: To utilize the files described below in TI-RTOS, refer to http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Board_Support for simple steps on how to add a custom board to the Processor SDK. RTOS.

Initial configuration – boardPadDelayTune.h: This file contains all manual and virtual IO delay mode definitions available per current PinMux GUI configuration.

Copyright © 2016–2017, Texas Instruments Incorporated

NOTE: This file is automatically configured by the PinMux tool with the desired functional modes uncommented based on the GUI configuration. For example, the tool uncomments functions used to define the proper IO delays. Reference the EVM configuration boardPadDelayTune.h file, found at http://www.ti.com/tool/tmdsevm572x.

```
/*#define MMC1_DS_PLB_SDR12_PLB_DEFAULT */ /* MMC1 DS (Pad Loopback) and SDR12 (Pad Loopback)
Default Timings */
/*#define MMC1_DDR50_PLB */ /* MMC1 DDR50 (Pad Loopback) Timings */
/*#define MMC1_SDR_104 */ /* MMC1 SDR104 Timings */
#define MMC1_HS_SDR12_ILB_SDR25  /* MMC1 HS (Internal Loopback and Pad Loopback), SDR12 (Internal
Loopback), SDR25 Timings (Internal Loopback and Pad Loopback) */
/*#define MMC1_SDR50_PLB */ /* MMC1 SDR50 (Pad Loopback) Timings */
/*#define MMC1_DS_ILB */ /* MMC1 DS (Internal Loopback) Timings */
/*#define MMC1_SDR50_ILB */ /* MMC1 SDR50 (Internal Loopback) Timings */
/*#define MMC1_DDR50_ILB */ /* MMC1 DDR50 (Internal Loopback) Timings */
/*#define QSPI1_MODE3 */ /* QSPI Mode 3 Default Timing Mode */
/*#define QSPI1_MODE0_DEFAULT */ /* QSPI Mode 0 Default Timing Mode */
/*#define QSPI1_MODE0_ALT1 */ /* QSPI Mode 0 Alternate Timing Mode 1 */
/*#define QSPI1_MODE0_ALT2 */ /* QSPI Mode 0 Alternate Timing Mode 2 */
#define QSPI1_MODE3_ALT1  /* QSPI Mode 3 Alternate Timing Mode 1 */
/*#define QSPI1_MODE3_ALT2 */ /* QSPI Mode 3 Alternate Timing Mode 2 */
/*#define VIP2_REC1 */ /* VIN3A/3B IOSET1 Rise-Edge Capture Mode */
/*#define VIP2_FEC2 */ /* VIN3A/3B IOSET1, VIN4A IOSET1/2 Fall-Edge Capture Mode */
#define VIP2_REC2  /* VIN4A IOSET1/2 Rise-Edge Capture Mode */
/*#define VIP2_FEC3 */ /* VIN4A IOSET1/2 Fall-Edge Capture Mode */
```

Further down in the file, there are mode redefinitions which link to manual/virtual IDs such as:

```
#ifdef MMC1_HS_SDR12_ILB_SDR25
     #define MMC1_VIRTUAL1
#endif
#ifdef QSPI1_MODE3_ALT1
     #define QSPI1_VIRTUAL1
#endif
#ifdef VIP2_REC2
     #define VIP2_4A_MANUAL1
#endif
```

Initial configuration – boardPadDelayInit.c: This file is the main initial C file that goes along with the boardPadDelayTune.h. It contains actual value arrays per pad. This file contains padconf and manual mode settings in the following format:

NOTE: Offset 0x0 and values of 0s indicates that there is no manual IN/OEN/OUT register configuration available.

- First column is the padconf register address offset
- Second column is the padconf register values
- Third column has a nested manual mode array with the following format:
  - CFG_x_IN register offset, aDelay value, gDelay value
- Fourth column has a nested manual mode array with the following format:
  - CFG_x_OEN register offset, aDelay value, gDelay value
- Fifth column has a nested manual mode array with the following format:
  - CFG_x_OUT register offset, aDelay value, gDelay value

```
/* MMC1 - mmc1_clk on W6 - MyMMC11 */
#ifdef MMC1_VIRTUAL1
     {0x1754, 0x101B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}},
#endif
...
/* QSPI1 - qspi1_sclk on R2 - MyQSPI1 */
#ifdef QSPI1_VIRTUAL1
     {0x1488, 0x101B1, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}},
#endif
```

Copyright © 2016–2017, Texas Instruments Incorporated

```
...
/* VIN4 - vin4a_de0 on B10 - MyVIN2 */
#ifdef VIP2_4A_MANUAL1
     {0x15CC, 0x50103, {0xCC0, 2347, 0}, {0x0, 0, 0}, {0x0, 0, 0}},
#endif
```

Runtime configuration – boardPadDelay.h: This file is primarily for MMC runtime configurations, as seen in the structure declarations (example excerpt below). The file contains comments explaining each field declaration. Refer to file inline comments to better understand the structure declaration.

```
typedef enum mmcMode
{
    MMC1_DEFAULT_PLB,
    /**< Default Pad Loopback mode of MMC1. */
    MMC1_HS_ILB,
    /**< High speed Internal Loopback mode of MMC1. */
    MMC1_HS_PLB,
    /**< High speed Pad Loopback mode of MMC1. */
    MMC1_SDR12_PLB,
    /**< SDR12 Pad Loopback mode of MMC1. */
    MMC1_SDR12_ILB,
    /**< SDR12 Internal Loopback mode of MMC1. */
    MMC1_SDR25_ILB,
    /**< SDR25 Internal Loopback mode of MMC1. */
    MMC1_SDR50_ILB,
    /**< SDR50 Internal Loopback mode of MMC1. */
    MMC1_SDR50_PLB,
    /**< SDR50 Pad Loopback mode of MMC1. */
...
    /**< High speed Internal Loopback mode of MMC2. */
    MMC3_DEFAULT,
    /**< Default speed mode of MMC3. */
    MMC3_HS,
    /**< High speed mode of MMC3. */
    MMC3_SDR12,
    /**< SDR12 mode of MMC3. */
    MMC3_SDR25,
    /**< SDR25 mode of MMC3. */
    MMC3_SDR50,
    /**< SDR50 mode of MMC3. */
    MMC4_DEFAULT,
    /**< Default speed mode of MMC4. */
    MMC4_HS,
    /**< High speed mode of MMC4. */
    MMC4_SDR12,
    /**< SDR12 mode of MMC4. */
    MMC4_SDR25,
    /**< SDR25 mode of MMC4. */
    MMC_MODE_INVALID = -1
    /**< Invalid MMC Mode */
}mmcMode_t;
```

Copyright © 2016–2017, Texas Instruments Incorporated

Runtime configuration – boardPadDelayDevice.c: This file is similar to boardPadDelayInit.c, but contains primarily MMC runtime configuration constants in the following format:

NOTE: Offset 0x0 and values of 0s indicates that there is no manual IN/OEN/OUT register configuration available.

- First column is the padconf register address offset
- Second column is the padconf register value
- Third column has a nested manual mode array with the following format:

  - CFG_x_IN register offset, aDelay value, gDelay value
- Fourth column has a nested manual mode array with the following format:

  - CFG_x_OEN register offset, aDelay value, gDelay value
- Fifth column has a nested manual mode array with the following format:

  - CFG_x_OUT register offset, aDelay value, gDelay value

```
#if defined(_TMS320C6X) || defined(__TI_ARM_V7M4__)
#pragma DATA_SECTION (gMmc1HsIlbPinmux, "BOARD_IO_DELAY_DATA");
const boardPadDelayCfg_t gMmc1HsIlbPinmux[] = {
#else
const boardPadDelayCfg_t gMmc1HsIlbPinmux[] __attribute__((section("BOARD_IO_DELAY_DATA"))) = {
#endif
    {0x1754, 0x101B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}}, /** MMC1 - Mmc1HsIlb:MMC1_VIRTUAL1 -
 mmc1_clk on W6 **/
    {0x1758, 0x501B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}}, /** MMC1 - Mmc1HsIlb:MMC1_VIRTUAL1 -
 mmc1_cmd on Y6 **/
    {0x175C, 0x501B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}}, /** MMC1 - Mmc1HsIlb:MMC1_VIRTUAL1 -
 mmc1_dat0 on AA6 **/
    {0x1760, 0x501B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}}, /** MMC1 - Mmc1HsIlb:MMC1_VIRTUAL1 -
 mmc1_dat1 on Y4 **/
    {0x1764, 0x501B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}}, /** MMC1 - Mmc1HsIlb:MMC1_VIRTUAL1 -
 mmc1_dat2 on AA5 **/
    {0x1768, 0x501B0, {0x0, 0, 0}, {0x0, 0, 0}, {0x0, 0, 0}} /** MMC1 - Mmc1HsIlb:MMC1_VIRTUAL1 -
 mmc1_dat3 on Y3 **/
};
```

Copyright © 2016–2017, Texas Instruments Incorporated

Further in the file, there are tables containing the MMC pad-to-mode mapping for each instance. The table lists the mode ID, the pointer to const table declared above, and the number of entities (pads) in the table.

```
mmcBoardPadCfgTable_t gMmc1PinmuxTable[] =
{
    { MMC1_DEFAULT_PLB, gMmc1DsPlbPinmux, 6 },
    /**< Pad configuration for Default Pad Loopback mode of MMC1. */
    { MMC1_HS_ILB, gMmc1HsIlbPinmux, 6 },
    /**< Pad configuration for High speed Internal Loopback mode of MMC1. */
    { MMC1_HS_PLB, gMmc1HsPlbPinmux, 6 },
    /**< Pad configuration for High speed Pad Loopback mode of MMC1. */
    { MMC1_SDR12_PLB, gMmc1Sdr12PlbPinmux, 6 },
    /**< Pad configuration for SDR12 Pad Loopback mode of MMC1. */
    { MMC1_SDR12_ILB, gMmc1Sdr12IlbPinmux, 6 },
    /**< Pad configuration for SDR12 Internal Loopback mode of MMC1. */
    { MMC1_SDR25_ILB, gMmc1Sdr25IlbPinmux, 6 },
    /**< Pad configuration for SDR25 Internal Loopback mode of MMC1. */
    { MMC1_SDR50_ILB, gMmc1Sdr50IlbPinmux, 6 },
    /**< Pad configuration for SDR50 Internal Loopback mode of MMC1. */
    { MMC1_SDR50_PLB, gMmc1Sdr50PlbPinmux, 6 },
    /**< Pad configuration for SDR50 Pad Loopback mode of MMC1. */
    { MMC1_DS_ILB, gMmc1DsIlbPinmux, 6 },
    /**< Pad configuration for Default speed Internal Loopback mode of MMC1. */
    { MMC1_DDR50_ILB, gMmc1Ddr50IlbPinmux, 6 },
    /**< Pad configuration for DDR50 Internal Loopback mode of MMC1. */
    { MMC1_DDR50_PLB, NULL, 0 },
    /**< Pad configuration for DDR50 Pad Loopback mode of MMC1. */
    { MMC1_SDR104, gMmc1Sdr104Pinmux, 6 },
    /**< Pad configuration for SDR104 mode of MMC1. */
    { MMC_MODE_INVALID, NULL, 0 }
    /**< Invalid MMC Mode */
};
```

Lastly, there is a lookup table listing all MMC instance pad configuration structures:

```
mmcBoardPadCfgTable_t* gMmcPadConfigTable[] =
{
    gMmc1PinmuxTable,
    /**< Pointer to the Pad configuration structure of MMC1 instance. */
    gMmc2PinmuxTable,
    /**< Pointer to the Pad configuration structure of MMC2 instance. */
    gMmc3PinmuxTable,
    /**< Pointer to the Pad configuration structure of MMC3 instance. */
    gMmc4PinmuxTable
    /**< Pointer to the Pad configuration structure of MMC4 instance. */
};
```

Copyright © 2016–2017, Texas Instruments Incorporated

References

Table A-1. References

Title	Literature Number
AM572x Sitara™ Processor Silicon Revision 2.0 Data Manual	SPRS953
AM572x Sitara™ Embedded Applications Processor 1.1 Data Manual	SPRS915
AM571x Sitara™ Processor Silicon Revision 2.0 Data Manual	SPRS957
AM571x Sitara™ Embedded Applications Processor Data Manual	SPRS919
AM572x Sitara™ Processor Silicon Revision 2.0, 1.1 Technical Reference Manual	SPRUHZ6
AM571x Sitara™ Processor Silicon Revision 2.0, 1.0 Technical Reference Manual	SPRUHZ7

Copyright © 2016–2017, Texas Instruments Incorporated

Downloading Custom IO Delay Output Conversion Script

The latest am57xx_generate_pin_config_data.pl script can be found on https://git.ti.com/pmt-generic-converter-tool/am57xx_uboot_pin_config

Copyright © 2016–2017, Texas Instruments Incorporated

Usage Notes for Custom IO Delay Output Conversion Script for U-boot

The custom script in Appendix B can read the generic output from the PinMux Tool (genericFileFormatPadConf.txt and genericFileFormatIOdelay.txt) and output files that can be added to the mux_data.h include file. To see usage, type in:

```
./am57xx_generate_pin_config_data -h
```

Usage:

```
./am57xx_generate_pin_config_data.pl [-h] -p padconf_file -d iodelay_file -o output_format
```

Where

- -h provides this help text
- -p padconf_file is the generic pad config output file provided by PMT(PinMux Tool)
- -d iodelay_file is the generic iodelay output file provided by PMT(PinMux Tool)
- -o output_format , where output_format is one of:
  - iodelay – Generate IO Delay data
  - iopad – Generate IO Pad data

As an example, to generate a text file called padconf.txt, type in:

```
./am57xx_generate_pin_config_data.pl -p genericPadConf.txt -d genericIOdelay.txt \
-o iopad >padconf.txt
```

To generate a text file called iodelay.txt, type in:

```
./am57xx_generate_pin_config_data.pl -p genericPadConf.txt -d genericIOdelay.txt \
-o iodelay >iodelay.txt
```

The two text files generated by the script can now be used to modify the contents of the mux_data.h header file that exists in board/ti/am57xx within the u-boot source directory. Note the default header file included with the Processor SDK splits the board configuration data into multiple data structures to support multiple AM57xx EVMs. The data structures are used by the board.c file within the same directory. The particular set of pad configuration settings used will depend on the results of board detection using the on-board EEPROM. Recompile u-boot after updating these files.

Copyright © 2016–2017, Texas Instruments Incorporated

Disclaimer

All programming models and use cases presented in this document are provided for educational purposes only, and may differ from or be optimized for your applications.

All peripheral devices presented in this document are provided for illustration purposes, and may be different from those in your system.

Copyright © 2016–2017, Texas Instruments Incorporated

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated (‘TI”) technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI’s standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated