Application Note

TDA4: Custom Board Bring Up Guide

Keerthy J, Karthik R, and Erick Narvaez

ABSTRACT

TDA4-EVM is a reference development platform with super set of peripherals to showcase the system-on-chip (SoC) capability. The customer develops board the based on the use cases that need to be supported on their platform. Typically, there is a cut down on the peripherals, change in the peripheral parts chosen, changes in the number of instances of a peripheral supported eventually leading to a different design. So, on top of the default SDK, the customer needs to implement custom pinmux, DDR, PMIC, console universal asynchronous receiver/ transmitter (UART) related changes as per the board design. This application note addresses the changes needed on top of the SDK and describes the common issues encountered during a custom TDA4 board bring up.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 PMIC and Power Custom Changes

Custom boards can have different PMICs sourcing the SoC voltage domains. Choosing different PMIC can be due to cost benefits, different power requirements and availability. The PMIC configuration is typically done in the safety aware MCU R5F0. R5 SPL primarily configures the CPU rail with AVS compensated voltages. Changes will be needed for custom PMIC in 2 places:

1. Device Tree: The device tree for WKUP_I2C0 needs to be populated with the right PMIC nodes. Example: arch/arm/dts/k3-j721e-r5-common-proc-board.dtsHook the right regulator node as the supply for VTM node.

```
&wkup_i2c0 {
        u-boot,dm-spl;
        tps659413a: tps659413a@48 {
                reg = <0x48>;
                compatible = "ti,tps659413";
                u-boot,dm-spl;
                pinctrl-names = "default";
                pinctrl-0 = <&wkup_i2c0_pins_default>;
                clock-frequency = <400000>;

                regulators: regulators {
                        u-boot,dm-spl;
                        buck12_reg: buck12 {
                                /*VDD_MPU*/
                                regulator-name = "buck12";
                                regulator-min-microvolt = <800000>;
                                regulator-max-microvolt = <1250000>;
                                regulator-always-on;
                                regulator-boot-on;
                                u-boot,dm-spl;
                        };
                };
        };

&wkup_vtm0 {
        vdd-supply-2 = <&buck12_reg>;
        u-boot,dm-spl;
};
```

2. CONFIG changes for PMIC in configs/j721e_evm_r5_defconfig. For example, below are the changes needed for TPS65941 PMIC:
3.
```
CONFIG_DM_PMIC=y
CONFIG_PMIC_TPS65941=y
CONFIG_DM_REGULATOR=y
CONFIG_SPL_DM_REGULATOR=y
CONFIG_DM_REGULATOR_TPS65941=y
```

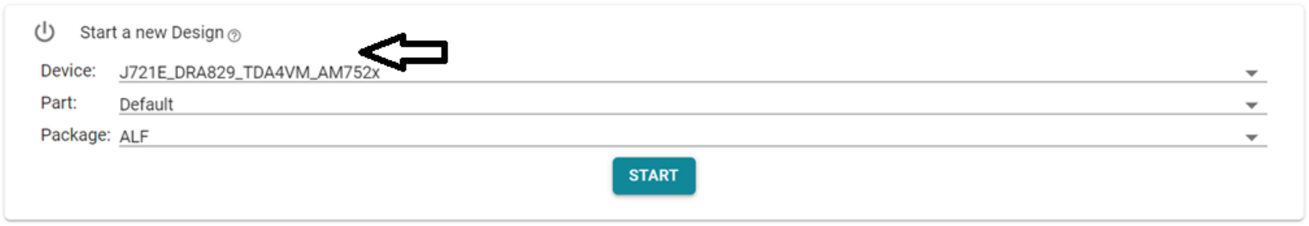
2 Pinmux

The recommended way of configuring the PinMux for Jacinto devices is by using the **PinMux tool.** This tool is as well as the installer are available online and can be used offline.

2.1 Steps to Configure the PinMux

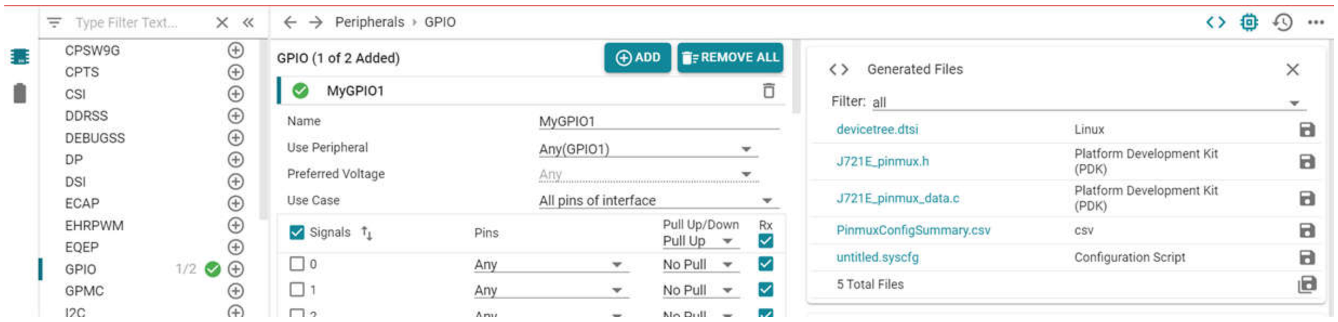
1. Go to https://dev.ti.com/sysconfig/?fromPinmux=true#/start.
2. In the “Start a new Design” Pick an existing design that matches your part number. For example, J721E_DRA829_TDA4VM_AM752x.

   Keep the “Part” tab default and “Package” ALF. Click “Start” button as shown in the screenshot below:



Device: J721E_DRA829_TDA4VM_AM752x
Part: Default
Package: ALF

3. Choose the peripheral by entering the corresponding name in the “Type Filter Text” (left top of the page). For example, GPIO and Click “+” sign. This adds a module on the center.



GPIO (1 of 2 Added)

MyGPIO1

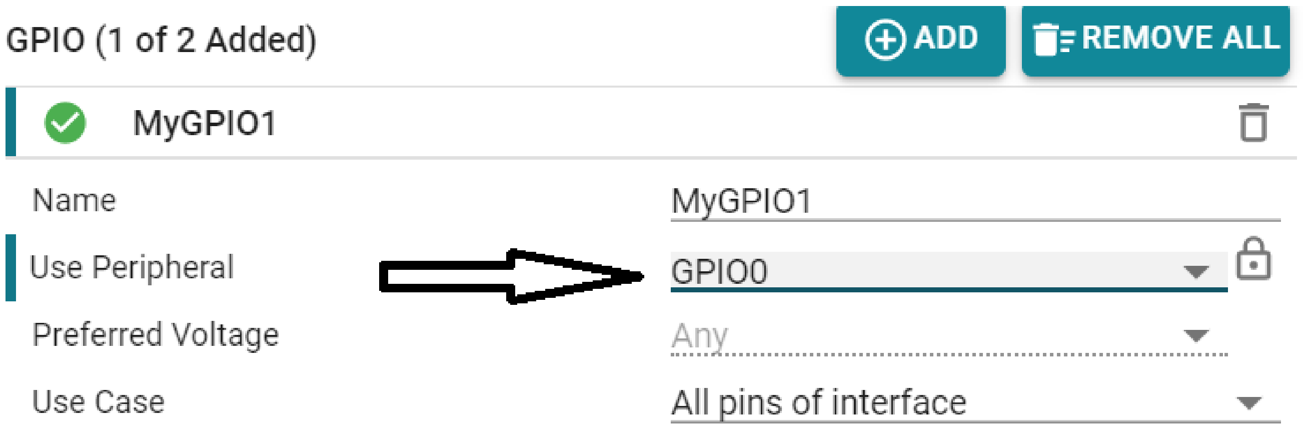
Name	MyGPIO1
Use Peripheral	Any(GPIO1)
Preferred Voltage	Any
Use Case	All pins of interface

Signals	Pins	Pull Up/Down	Rx
✓		Pull Up	✓
☐ 0	Any	No Pull	✓
☐ 1	Any	No Pull	✓
☐ 2	Any	No Pull	✓

Filter: all

devicetree.dtsi	Linux
J721E_pinmux.h	Platform Development Kit (PDK)
J721E_pinmux_data.c	Platform Development Kit (PDK)
PinmuxConfigSummary.csv	csv
untitled.syscfg	Configuration Script
5 Total Files	

4. MyGPIO1 has some defaults. Choose the exact module instance in the “Use peripheral” menu.By default it will be ANY. Select the appropriate instance using the drop down menu. For this case, GPIO0 is used.

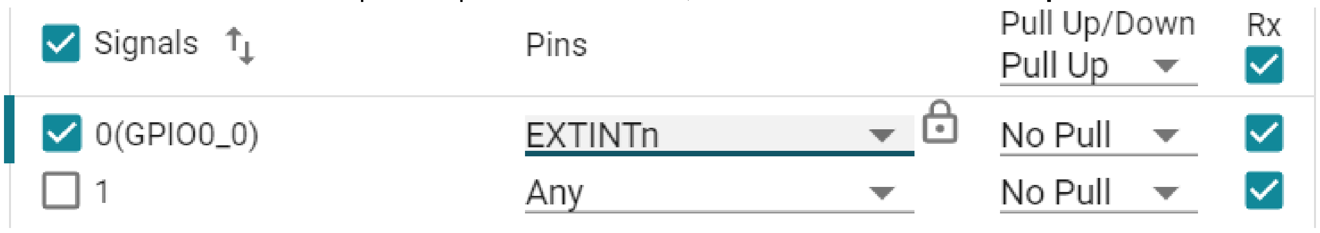


GPIO (1 of 2 Added)

MyGPIO1

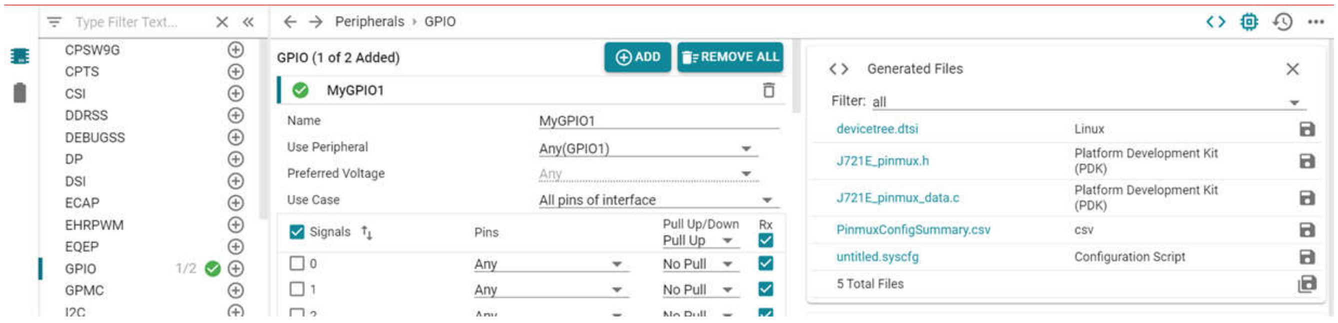
Name	MyGPIO1
Use Peripheral	GPIO0
Preferred Voltage	Any
Use Case	All pins of interface

5. With this, we are done with module level selection. Now we just need to pick the ball name that maps to the one present in the board schematics. This can be accomplished by choosing the right ball name in the "Pins" drop down menu. Based on schematics pick appropriate "Pull Up/Down" from the drop down menu. For information on how to pick the pin name EXTINTn, see the **Manual audit of the pinmux** details below.



Signals	Pins	Pull Up/Down	Rx
✓		Pull Up	✓
✓ 0(GPIO0_0)	EXTINTn	No Pull	✓
☐ 1	Any	No Pull	✓

6. From the "Generated Files" section, based on the choice of OS, download one of these files: - Linux: devicetree.dtsi: Just need to copy paste the node as is under the right parent pinmux node (see below for details) - PDK(RTOS): J721E_pinmux.h/ J721E_pinmux.c.



GPIO (1 of 2 Added)

MyGPIO1

Name	MyGPIO1
Use Peripheral	Any(GPIO1)
Preferred Voltage	Any
Use Case	All pins of interface

Signals	Pins	Pull Up/Down	Rx
✓		Pull Up	✓
☐ 0	Any	No Pull	✓
☐ 1	Any	No Pull	✓

Generated Files

Filter: all

devicetree.dtsi	Linux
J721E_pinmux.h	Platform Development Kit (PDK)
J721E_pinmux_data.c	Platform Development Kit (PDK)
PinmuxConfigSummary.csv	csv
untitled.syscfg	Configuration Script
5 Total Files	

a. For example, Linux: devicetree.dtsi.

   main gpio0 was chosen, hence, the pinmux node for the example mygpio1_pins_default should come under main_pmx0 node in arch/arm64/boot/dts/ti/j721-common-proc-board.dts file.

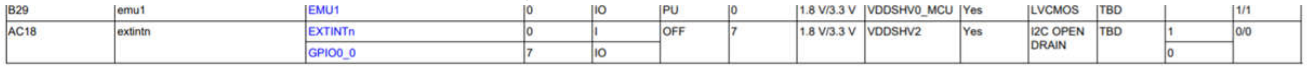
```
&main_pmx0 {
pinctrl-single,pins = < J721E_IOPAD(0x0, PIN_INPUT, 7) /* (AC18) EXTINTn.GPIO0_0 */ >;
};
};
```

The same can be followed with u-boot device tree. The very same example arch/arm/dts/j721-common-proc-board.dts file.

2.2 Manual Audit of the Pinmux

1. Check the schematics for the exact Ball number. For example, J721E/TDA4VM(SoC) à gpio0_0(Signal_name). Do a search in the corresponding Data Manual for TDA4VM/J721E:

   https://www.ti.com/lit/ds/symlink/tda4vm.pdf?ts=1593521450358&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTDA4VM
2. The first instance of search lands in the table that provides the Ball number and name, which is what you need to check in board schematics to see on which Ball the gpio0_0 is brought out on the board. For example: AC18 maps to for GPIO0_0. Page: 18 of the document link shared in #1 above.



B29	emu1	EMU1	0	IO	PU	0	1.8 V/3.3 V	VDDSHV0_MCU	Yes	LVCMOS	TBD		1/1
AC18	extintn	EXTINTn	0	I	OFF	7	1.8 V/3.3 V	VDDSHV2	Yes	I2C OPEN DRAIN	TBD	1	0/0
		GPIO0_0	7	IO								0	

3. Search for AC18 that maps on to a padconfig register as shown below. In the PADCONFIG0 register in the device-specific data manual search for AC18 and that is the mode that needs to be programmed. For example, gpio0_0 is mode7 and hence the pinmux files should have mode7 to bring out gpio0_0 signal out of ball AC18.Page: 138 of the document link pointed in #1 above.

   You can tally the devicetree.dtsi generated by the pinmux tool and manually audited register, and the bit fields and cross verify. For example, J721E_IOPAD(0x0, PIN_INPUT, 7) 0x0 corresponds to register offset 0x0 for PADCONFIG0 register and 7 corresponds to mux mode7.

3 Custom DDR Related Changes

Custom boards can have different DDR parts. Choosing different DDR parts can be due to cost benefits, different memory requirements and power design. All the DDR related configuration and initialization happens in R5 SPL. A custom DT file needs to be generated to address differences. arch/arm/dts/k3-j721e-ddr-evm-lp4-4266.dtsiwith needs to be replaced with a custom dts file corresponding to the chosen DDR part. This involves using the DDR config tool. The details are captured in the *Jacinto7 DDRSS Register Configuration Tool*.

4 Minimal Kernel DT to Start With

The first and primary goal is to bring up Linux on the custom board with TDA4. It is always better and easier to start with minimal device tree that covers just the boot media, console UART, Timers and other necessary DT nodes. The following zip file contains patches disabling the peripheral nodes that are NOT essential for booting Linux. The default device tree provided in the SDK is a super set of all the peripheral support for TDA4. Hence, optimization in DT is needed to avoid crashes/hangs due to a peripheral mismatch in custom board. Choose the minimal set of nodes and boot to Linux prompt with that. Post that customer can enable the nodes relevant for their custom board.

Here is a set of reference DT optimization patches for Linux on 8.0 SDK:

https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/791/0001_2D00_arm64_2D00_dts_2D00_j721e_2D00_Enable_2D00_Minimal_2D00_DT_2D00_configuration.patch

5 Boot Mode Support Summary

5.1 No-Boot Mode

The most supported boot-mode for hardware bring-ups. GEL files are the primary method for initializing the device and will bring the SoC to a state where cores can be loaded with programs and tests can be run with the most control during the boot process. GEL files are provided with the RTOS SDK of the device or also part of Code Composer Studio when the device target configuration is natively available. Lauterbach scripts are usually available as well.

No-Boot Mode keeps the boot process simple and should be used immediately after hardware check-out. For more complex applications it is recommended to migrate to boot solutions like SPL and SBL. These can be loaded in No-boot mode, however, it requires customization work and usually more steps in the debugger.

5.2 UART Boot Mode

UART boot mode allows loading boot binaries into memory that is already initialized by ROM. It is useful for being the simplest boot-mode that does not require any other modules except the UART. The DM firmware can be loaded, and this will take over and load the rest of the boot binaries required. The down-side to UART boot-mode is the speed: it will have a default speed set by ROM, and this can affect loading large applications. It is a great initial boot mode for initial checkout, but in later stages it is preferred to use other boot-modes.

5.3 OSPI/QSPI/SPI/xSPI/Serial NAND

Booting from a flash device is one of the most common production boot media choices. The SDK provides tools for flashing the media, but during a board bring-up, this adds one more failure point and dependency on tooling. It is recommended to have other boot modes available for back-up testing in case there are any issues with this boot-mode.

6 Commonly Encountered Issues During Custom Board Bring Up

1. **Custom dts files in U-Boot:** The U-Boot framework expects a k3-j721e-common-proc-board-u-boot.dtsi for A72 SPLcorresponding to k3-j721e-common-proc-board.dtsi.So if a customer board has DTS file named k3-j721e-*custom-board*.dts then one shouldadd k3-j721e-*custom-board-u-boot*.dts. That is mandatory for A72 SPL.
2. Switching the console to different UART. For example, how to switch console to MCU_UART from MAIN_UART.

   There are three stages according to the boot flow:

   - R5 SPL
   - A72 SPL
   - A72 u-boot.

   So, you need to change this at all levels. On top of that, Arm Trusted Firmware (ATF) also needs to change the console port as needed. Following are the dts changes on J7200 and similar changes for J721e that will work fine:

   FAQ Link: https://e2e.ti.com/support/processors/f/791/t/988278
3. How to debug with CCS when there are no console prints.

   Add an infinite loop at the start of the first instruction of R5 SPL.

   ```
   diff --git a/arch/arm/cpu/armv7/start.S b/arch/arm/cpu/armv7/start.S
   index 4f6327fe3ab..96c4554744c 100644
   --- a/arch/arm/cpu/armv7/start.S
   +++ b/arch/arm/cpu/armv7/start.S
   @@ -37,6 +37,7 @@

   #endif

    reset:
   +       b reset
           /* Allow the board to save important registers */
           b       save_boot_params
    save_boot_params_ret:
   ```

   Then we can attach to MCU R5F_0 using CCS. Move PC to the next instruction & step through to check where the crash is.

4. How to change the console for ATF and OPTEE for j721s2 and j784s4?

   Make CROSS_COMPILE64=aarch64-none-linux-gnu- CROSS_COMPILE=arm-none-linux-gnueabihf- PLATFORM=k3-j784s4 CFG_ARM64_core=y **CFG_CONSOLE_UART=0x8**

   Based on the custom board UART_INSTANCE X.

   CFG_CONSOLE_UART=0x8 will change