Technical White Paper

GPIO Virtualization and Interrupt Architecture on TI J7 Processors

Vinit Patel, Software Engineer, Jacinto Processors
Sudheer Kumar Doredla, Software Engineering Manager, Jacinto Processors
Brijesh Jadav, Jacinto Processors

Abstract

General-purpose input/output (GPIO) is a fundamental peripheral present in every embedded system. On TI's J7 family processors — TDA4VH (J784S4), TDA4VM (J721E), TDA4VL (J721S2), DRA821U (J7200) and TDA4AEN (J722S) — GPIO goes beyond the traditional single-instance model. These devices are implemented with a GPIO virtualization architecture that allows multiple software domains and CPU cores to independently own and interact with GPIO resources mapped to the same physical pins, without and interference. This white paper describes the hardware architecture of the GPIO subsystem on J7 devices, explains the interrupt delivery modes (bank and line), details the virtualization mechanism, and demonstrates validated multi-core GPIO coexistence. Software engineers designing safety-critical or multi-domain automotive systems can also see architectural guidance and implementation patterns that directly address GPIO resource sharing challenges.

Table of Contents

1 Introduction 2
1.1 GPIO in Automotive Applications 2
1.2 The J7 GPIO Virtualization Solution 2
2 GPIO Hardware Architecture 3
2.1 GPIO Module Structure 3
2.2 GPIO Instances Across J7 Devices 3
3 GPIO Interrupt Modes 4
3.1 Bank Interrupt Mode 4
3.2 Line (Per-Pin) Interrupt Mode 4
3.3 The BINTEN Register — A Critical Hardware Constraint 4
3.4 Interrupt Mode Comparison 5
4 GPIO Virtualization Architecture 6
4.1 The Virtual GPIO Concept 6
4.2 The PADCONFIG VGPIO_SEL Field 6
5 Interrupt Routing Architecture 7
5.1 The GPIO Interrupt Router 7
6 Multi-Core GPIO Virtualization — Validated Scenario 7
6.1 Test Configuration 7
6.2 Observed Results 7
7 Conclusion 8
8 References 8

Trademarks

Jacinto™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

Copyright © 2026 Texas Instruments Incorporated

1 Introduction

1.1 GPIO in Automotive Applications

Modern automotive system-on-chip (SoC) designs aggregate multiple processing domains — MCU, main application processor, digital signal processors, and accelerator clusters — onto a single device. Each domain can run independent real-time or safety-critical workloads that require direct access to physical I/O pins for:

- LED indicators and status signaling
- External sensor trigger lines
- Hardware watchdog toggle signals
- Safety monitoring outputs
- General control and status GPIO

In a traditional single-instance GPIO peripheral, only one software domain can own a given GPIO pin at a time. Any sharing requires explicit software coordination or off-chip multiplexing. This constraint complicates multi-domain designs and imposes unnecessary software coupling between otherwise independent workloads.

1.2 The J7 GPIO Virtualization Solution

TI's J7 family resolves this constraint through a GPIO virtualization architecture that instantiates multiple independent GPIO module instances within the same power domain, all sharing access to the same physical I/O pins. A per-pin selector in the pad configuration (PADCONFIG) register determines which GPIO module instance has interrupt routing authority for each pin at any given time. Combined with a flexible interrupt router, this design allows:

- Different CPU cores to independently control GPIO interrupt routing
- Multiple software domains to operate on GPIO without mutual interference
- Freedom-from-interference (FFI) helps support functional safety goals at the hardware level

This white paper explains the interrupt delivery architecture and the virtualization mechanism and provides implementation guidance for the J7 device family.

Copyright © 2026 Texas Instruments Incorporated

2 GPIO Hardware Architecture

2.1 GPIO Module Structure

Each GPIO module instance on J7 devices follows a standard block-structured layout. The module is organized into banks of 16 GPIO signals each.

- Pins per bank: 16
- Banks per module: 9 (Bank 0 through Bank 8)
- Maximum pins per instance: 144 (9 × 16), though actual usable pins depend on device pinout

Each GPIO pin within the module supports:

- Direction control (input or output) per pin
- Rising-edge trigger, falling-edge trigger, or both-edges trigger per pin
- Set and clear output control (atomic, no read-modify-write needed)
- Interrupt or DMA event generation in bank or per-pin (line) mode

2.2 GPIO Instances Across J7 Devices

J7 devices organize the GPIO instances across two power domains: the WKUP (Wake-Up) domain and the MAIN domain. The WKUP domain GPIO instances are always-on. The MAIN domain GPIO instances operate within the main application processor power island.

Table 2-1. GPIO Instances per J7 Device

Device	WKUP GPIO Instances	MAIN GPIO Instances	Total Instances
TDA4VH / J784S4(1)	WKUP_GPIO0, WKUP_GPIO1 (2)	GPIO0, GPIO2, GPIO4, GPIO6 (4)	6
TDA4VM / J721E(2)	WKUP_GPIO0, WKUP_GPIO1 (2)	GPIO0–GPIO7 (8)	10
TDA4VL / J721S2(1)	WKUP_GPIO0, WKUP_GPIO1 (2)	GPIO0, GPIO2, GPIO4, GPIO6 (4)	6
DRA821U / J7200(1)	WKUP_GPIO0, WKUP_GPIO1 (2)	GPIO0, GPIO2, GPIO4, GPIO6 (4)	6
TDA4AEN / J722S	WKUP_GPIO0, WKUP_GPIO1 (2)	GPIO0–GPIO1 (2)	4

(1) On J784S4, J721S2 and J7200, the MAIN GPIO instances use even-numbered names (GPIO0, GPIO2, GPIO4, GPIO6) reflecting the instance pair grouping in the virtualization mux.
(2) J721E exposes the full set of 8 MAIN GPIO instances (GPIO0 through GPIO7) reflecting the broader I/O subsystem

Table 2-2. GPIO Base Addresses (J784S4 Example)

Instance	Base Address	Domain
WKUP_GPIO0	`0x42110000`	WKUP
WKUP_GPIO1	`0x42100000`	WKUP
GPIO0	`0x00600000`	MAIN
GPIO2	`0x00610000`	MAIN
GPIO4	`0x00620000`	MAIN
GPIO6	`0x00630000`	MAIN

Copyright © 2026 Texas Instruments Incorporated

3 GPIO Interrupt Modes

GPIO modules on J7 devices support two fundamentally different interrupt delivery modes: **bank interrupt** mode and **line (per-pin) interrupt** mode. Understanding both is critical for designing multi-process or multi-core GPIO systems.

3.1 Bank Interrupt Mode

In bank interrupt mode, all 16 GPIO pins within a bank share a single interrupt output signal from the GPIO module. When any enabled pin in the bank transitions on the configured edge, the shared bank interrupt line is asserted.

Interrupt flow (bank mode):

GPIO Pin (any enabled pin in Bank N)

→ BINTEN register bit N must = 1 (bank enable gate)

→ GPIO module output: gpio_bank[N] (one signal per bank)

→ GPIO_INTRTR input (bank input range)

→ GPIO_INTRTR output (routed to target CPU VIM)

→ CPU interrupt handler

→ software reads GPIO_INTSTAT to determine which pin or pins fired

→ clears interrupt status

Limitation of bank interrupt mode: Because all 16 pins in a bank share a single interrupt line, routing pin 42 to Core A and pin 46 to Core B is not possible when both pins reside in the same bank (Bank 2 for pins 32–47). Furthermore, interrupts can be lost when multiple pins are enabled in the same bank.

3.2 Line (Per-Pin) Interrupt Mode

In line interrupt mode, each GPIO pin has a dedicated interrupt path from the GPIO module to the interrupt router. A specific pin fires only its own interrupt — no sharing with neighboring pins in the same bank.

Interrupt flow (line mode):

GPIO Pin N

→ BINTEN register must still be enabled (hardware gate — see Section 3.3)

→ GPIO module output: gpio[N] (per-pin line interrupt)

→ GPIO_INTRTR input at index N (line input range)

→ GPIO_INTRTR output (independently routed to any target CPU)

→ CPU interrupt handler (fires only for pin N)

For more information about the line interrupts and how to enable this mode, see [FAQ] TDA4VH-Q1: How to enable GPIO Line Interrupts on TDA4VH/VM/VL.

3.3 The BINTEN Register — A Critical Hardware Constraint

The GPIO_BINTEN (Bank Interrupt Enable) register controls interrupt output generation at the GPIO module boundary. GPIO_BINTEN is a physical hardware gate, not a software abstraction.

Important architectural fact: BINTEN must remain enabled regardless of whether bank or line interrupt mode is used. Although the register name implies "bank" control, BINTEN physically gates **all** interrupt pulse generation from that bank in the GPIO module — both bank-consolidated pulses and individual per-pin pulses. Disabling a BINTEN bit for a bank that has active GPIO line interrupt routes silences those interrupts.

This constraint is architectural and cannot be bypassed. The GPIO driver enables the relevant BINTEN bit whenever any pin in that bank is configured for interrupts, and this behavior must be preserved when implementing line interrupt support.

Copyright © 2026 Texas Instruments Incorporated

3.4 Interrupt Mode Comparison

Table 3-1. Bank Interrupt vs Line Interrupt

Attribute	Bank Interrupt	Line (Per-Pin) Interrupt
Interrupt source granularity	Per bank (16 pins → 1 interrupt)	Per pin (1 pin → 1 interrupt)
Interrupt index	Bank Num which is PinNum/16	PinNum
Can route different pins in same bank to different cores?	No	Yes
Identifies pin via	Reading GPIO_INTSTAT register	Direct (only that pin triggered it)
BINTEN required?	Yes	Yes
Resource consumption	Lower (fewer interrupt router outputs used)	Higher (one IR output per pin)
Multi-domain coexistence	Limited by bank Bank/register Same 32 pins controlled from same registers required H/W Semaphores for multi-core usage	Fully independent per pin

Copyright © 2026 Texas Instruments Incorporated

4 GPIO Virtualization Architecture

4.1 The Virtual GPIO Concept

All GPIO module instances within the same power domain — for example WKUP_GPIO0 and WKUP_GPIO1 in the WKUP domain — are mapped to the same physical I/O pins. The silicon does not dedicate different pins to different instances. Instead, a per-pin selector in the PADCONFIG register determines which interrupt path of an instance is active for each individual pin.

This is the GPIO virtualization mechanism and enables:

- Isolation: Core A uses WKUP_GPIO0 to monitor Pin 6; Core B uses WKUP_GPIO1 to monitor Pin 3. Each core interacts with a separate module instance and receives independent interrupts.
- Freedom from interference: The hardware mux prevents the interrupt traffic of one instance from polluting the interrupt path of another instance.
- Independent software ownership: Each GPIO module instance has a register space, a TISCI device ID, and interrupt router connections.

4.2 The PADCONFIG VGPIO_SEL Field

Each physical I/O pin has an associated PADCONFIG register in the Control Module MMR (CTRL_MMR). The PADCONFIG register contains a `**VGPIO_SEL**` field (bits [5:4] on J784S4) that selects which GPIO module instance the interrupt of the pin is routed through.

Table 4-1. VGPIO_SEL Encoding

VGPIO_SEL Value	Selected GPIO Instance
0b00	WKUP_GPIO0 (or GPIO0/GPIO1 in MAIN)
0b01	WKUP_GPIO1 (or GPIO2/GPIO3 in MAIN)
0b10	(MAIN: GPIO4/GPIO5)
0b11	(MAIN: GPIO6/GPIO7)

The default value is 0b00, meaning all pins default to GPIO instance 0 routing. To use WKUP_GPIO1 on a given pin, software must explicitly write `VGPIO_SEL = 1` to the PADCONFIG register of that pin.

Example: Configuring Pin 3 for WKUP_GPIO1 on J784S4

```
/* bits[2:0] = 7  → MUX_MODE = 7 (GPIO function)
 * bits[5:4] = 1  → VGPIO_SEL = 1 (route to WKUP_GPIO1)
 * bit[18]   = 1  → RX_ACTIVE = 1 (enable input buffer) */
```

Without this write, the interrupt for Pin 3 routes to WKUP_GPIO0 regardless of which GPIO module instance software programs the interrupt path for.

For mapping of GPIO Pin numbers to PADCONFIG refer to Technical datasheet of corresponding SoC.

Note

For TDA4VM (J721E), In the main domain, there are eight virtualized GPIOs. GPIO0, GPIO2, GPIO4 and GPIO6 share a set of pins while GPIO1, GPIO3, GPIO5 and GPIO7 share another set of pins. With 2 bits of VGPIO_SEL in PADCONFIG, the pin can be configured to be used by particular GPIO out of the four supported GPIOs.

Copyright © 2026 Texas Instruments Incorporated

5 Interrupt Routing Architecture

5.1 The GPIO Interrupt Router

GPIO module instances do not connect directly to CPU interrupt controllers. Interrupts are routed through a dedicated GPIO multiplexed interrupt router (`GPIO_INTRTR`) that aggregates all GPIO interrupt signals — both bank-consolidated and per-pin line interrupts — and allows flexible routing to any connected CPU core or processor cluster.

Two separate routers serve the two power domains:

Table 5-1.

Router	Domain	Inputs	Outputs	Architecture Type	Applies to
WKUP_GPIOMUX_INTRTR	WKUP	120	32	Selection Mux	J784s4, J721e, J721s2, J7200
MAIN_GPIOMUX_INTRTR	MAIN	304	64	Selection Mux	J721e
MAIN_GPIOMUX_INTRTR	MAIN	180	64	Selection Mux	J784s4, J721s2, J7200

The "Selection Mux" architecture means any of the M input events can be routed to any of the N output events through a programmed multiplexer. This architecture gives the system flexibility — any interrupt of the GPIO pin can be directed to any CPU core that has an allocated output line.

Additionally, SOCs supports cross-domain interrupt routing functionality. In this case, WKUP_GPIOMUX_INTRTR has dedicated output pins routing to the MAIN domain cores, so the router can be used when routing of MCU domain interrupt to MAIN domain core is required. Similarly, use MAIN2MCU_LVL_INTRTR0 (for level interrupts) or MAIN2MCU_PLS_INTRTR0 (for pulse interrupts) to route MAIN domain interrupt to MCU domain core.

6 Multi-Core GPIO Virtualization — Validated Scenario

6.1 Test Configuration

To validate the GPIO virtualization architecture, a multi-core application was built and executed on J784S4 hardware using a three-core SBL appimage:

Table 6-1.

Core	App	GPIO Instance	Pin	Physical Pin	Interrupt Mode
MCU1_0	Device Manager (Manages resources and Power, In this case resources like interrupt router mapping)	— (TISCI proxy)	—	—	—
MCU1_1	LED Blink example	WKUP_GPIO0	Pin 6	WKUP_GPIO0_6	Bank
MCU2_0	LED Blink example	WKUP_GPIO1	Pin 3	WKUP_GPIO1_3	Bank

Both GPIO pins (Pin 3 and Pin 6) are physical pins in the WKUP GPIO pad group. WKUP_GPIO0 and WKUP_GPIO1 share access to these same physical pads. The VGPIO_SEL setting in each PADCONFIG register of the pin determines which interrupt for each module instance fires when that pin transitions.

6.2 Observed Results

Upon booting the three-core appimage from OSPI:

MCU UART (WKUP_UART0) — combined sciserver + MCU1_1 GPIO application output

MAIN UART (MAIN_UART0) — MCU2_0 GPIO application output

Both GPIO applications execute concurrently and independently. MCU1_1 uses WKUP_GPIO0 (default VGPIO_SEL=0) and MCU2_0 uses WKUP_GPIO1 (VGPIO_SEL=1 programmed). No interference is observed between the two cores.

Copyright © 2026 Texas Instruments Incorporated

7 Conclusion

The GPIO subsystem on TI's J7 family of automotive processors provides significantly more capability than a traditional single-instance GPIO peripheral. Key architectural capabilities include:

- **Multiple virtual GPIO instances** per power domain, all sharing the same physical I/O pins, enabling per-domain and per-core GPIO ownership with hardware-enforced isolation
- **Two interrupt modes** — bank interrupt for low-overhead shared-bank routing, and line (per-pin) interrupt for fully independent per-pin routing to different CPU cores
- **PADCONFIG VGPIO_SEL** mechanism that allows the hardware interrupt mux to direct the physical interrupt of each pin to the GPIO instance associated with the owning software domain
- **A flexible GPIO interrupt router** (WKUP_GPIOMUX_INTRTR with 32 outputs, AIN_GPIOMUX_INTRTR with 64 outputs) that provides programmable routing from any GPIO input to any CPU output using the TISCI resource manager
- **Validated multi-core coexistence** on J784S4 hardware, with two different GPIO instances operating independently on two separate cores within a single multicore appimage

These capabilities enable automotive software architects to assign GPIO resources cleanly across safety, security, and application domains without off-chip multiplexing or cross-domain software coordination, directly supporting Freedom from Interference (FFI) requirements in ISO 26262 compliant system designs.

8 References

1. Texas Instruments, *J784S4 AM69x TDA4VH TDA4AH TDA4VP TDA4AP J742S2 AM69x-27 TDA4VPE TDA4APE Processors Texas Instruments Families of Products* technical reference manual
2. Texas Instruments, *J721E DRA829/TDA4VM/AM68P Processors Silicon Revision 1.1 Texas Instruments Families of Products* technical reference manual
3. Texas Instruments, *J721S2/TDA4VE/TDA4AL/TDA4VL/AM68 Processors Silicon Revision 1.0 Texas Instruments Families of Products* technical reference manual
4. Texas Instruments, *J722S/TDA4VEN/TDA4AEN/AM67 Processor Silicon Revision 1.0 Texas Instruments Families of Products* technical reference manual
5. Texas Instruments, *TDA4VH-Q1, TDA4AH-Q1, TDA4VP-Q1, TDA4AP-Q1 Jacinto™ Processors* datasheet
6. Texas Instruments, *TDA4VM Processors* datasheet
7. Texas Instruments, *TDA4VE TDA4AL TDA4VL Jacinto™ Processors, Silicon Revision 1.0* datasheet
8. Texas Instruments, *TDA4VEN, TDA4AEN Jacinto™ Processors* datasheet
9. Texas Instruments, J784S4, J721E, J721S2, J722S, J7200 Register Specification
10. Texas Instruments, GPIO Driver Source: Processor SDK RTOS 11.02

Copyright © 2026 Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025