Discrete I/O Fundamentals: GPIO configuration and electrical characteristics (EVKB-i.MX RT1050)

Discrete signals in avionics

- A discrete is a binary signal—physically an electrical line that is asserted (logic "1" / true) or de-asserted (logic "0" / false).
- In aircraft systems, discretes announce states such as WOW (Weight-On-Wheels), FLT/GRD (Flight/Ground), FIRE DETECT, DOOR CLOSED, or APU (Auxiliary Power Unit) READY.
- Software interacts with discretes via GPIO (General-Purpose Input/Output) pins.

GPIO features and pad configuration

- GPIOs can be configured as inputs or outputs with features like internal pull-ups/pull-downs, Schmitt triggers (hysteresis), open-drain mode, drive strength, and slew rate control.
- On the NXP i.MX RT1050, each physical pad is routed by a pin-mux to one of several peripheral functions (GPIO, UART, SPI, etc.).
- After selecting GPIO in the mux, you configure pad electrical characteristics (pulls, slew, drive strength, hysteresis) so logic levels are reliable, immune to noise, and compatible with surrounding hardware.

Safety-critical expectations (avionics)

- For safety-critical avionics,
- DAL (Design Assurance Level) software (per DO-178C) and hardware (per DO-254) expect discretes to be engineered for determinism
- diagnosability (BIT—Built-In Test),
- tolerance to EMI/EMC (Electromagnetic Interference/Compatibility) and ESD (Electrostatic Discharge) per DO-160.

Core GPIO and voltage terms

- GPIO (General-Purpose Input/Output): Configurable digital pin used as a binary input or output.
- VIH / VIL (Input High/Low Voltage): Minimum voltage recognized as logic "1" (VIH) and maximum recognized as logic "0" (VIL).
- VOH / VOL (Output High/Low Voltage): Guaranteed output voltages while driving high/low under load.
- Drive strength (mA): Sourcing/sinking capability at valid logic levels.
- Slew rate: Edge speed; slower edges reduce EMI.
- Schmitt trigger (hysteresis): Adds separate rising/falling thresholds to reject noise and slow edges.

Pad behavior, filtering, and shared lines

- Pull-up / Pull-down: Weak resistors to rails that define a default state for otherwise floating signals.
- Push-pull: Output stage can drive both high and low.
- Open-drain: Output can only pull low; a pull-up defines the high level—ideal for wired-OR and level translation.
- Pad / Mux: "Pad" is the electrical cell at the pin; "mux" selects which internal function connects to it.
- Debounce: Filtering that removes mechanical chatter and narrow spikes.
- Wired-OR / wired-AND: Multiple open-drain outputs share a line with a pull-up; any device pulling low dominates.

Logic levels and supplies (EVKB 3.3 V)

- The EVKB user I/O domain is 3.3 V (LVCMOS).
- Typical thresholds (always confirm in the data sheet) are $\sim$0.6–0.7$\times$VDDIO for VIH and $\sim$0.3–0.4$\times$VDDIO for VIL.
- Never drive GPIOs above their bank supply or below ground (beyond allowed diode/leakage currents) and observe absolute maximum ratings.

Inputs: default state and polarity

- Pull-ups/pull-downs: Prevent floating inputs.
- Use pull-up for active-low signals; pull-down for active-high.

Pull resistors: overview and level shifting

- Pull-up resistor increases the current, and the pull-down resistor is used to absorb the current.
- Increase the voltage level.
- When the TTL circuit drives the CMOS circuit, if the output high level of the TTL circuit is lower than the lowest high level of the CMOS circuit, then it is necessary to connect a pull-up resistor to the output terminal of the TTL to increase the value of the output high level.
- The OC gate circuit must add a pull-up resistor to increase the high-level value of the output.
- Increase the drive capability of the output pin.
- In order to enhance the drive capability of the output pins, pull-up resistors are often used on some single-chip pins.

Pull resistors: unused pins and EMI susceptibility

- The N/A pin (the pin not connected) should be anti-static and anti-interference.
- On the CMOS chip, in order to prevent damage caused by static electricity, the unused pins cannot be left floating.
- Generally, a pull-up resistor is connected to reduce the input impedance, provide a leakage path, and improve the anti-electromagnetic interference ability of the bus.
- Because the pin is left floating, it is easier to receive electromagnetic interference from the outside world.
- Resistance match

Pull resistors: resistance match and default potential

- In the long-line transmission, the resistance mismatch can easily cause the reflected wave interference.
- In addition, the pull-down resistor makes the resistance match, which can effectively suppress the reflected wave interference.
- Preset space state/default potential
- Pull-up or pull-down resistors are connected to some CMOS input terminals to preset the default potential.
- When these pins are not used, these input terminals are pulled down to low level or pulled up to high level.
- The state when idle on the bus such as I2C is obtained by the pull-up and pull-down resistors.

Pull resistors: improving noise tolerance

- Improve the noise tolerance of the chip input signal.
- If the input terminal is in a high-impedance state, or in a floating state, a pull-down or pull-down resistor needs to be added at this time, so as to avoid the random level.
- Similarly, if the output terminal is in a passive state, a pull-down or pull-down resistor needs to be added.
- For example, the output terminal is only the collector of a transistor, thereby improving the noise tolerance of the chip input signal and enhancing the anti-interference ability through a pull-up resistor or pull-down resistor.
- Schmitt trigger: Essential for long harnesses or RC-filtered discretes.

Glitch filtering

- Glitch filtering: In software (sampling/debounce) or hardware where available.

Push-pull output: how it works

- Push-pull output
- It is a circuit that uses a pair of devices to selectively source current to or sink current from a connected load.
- It often uses a pair of power transistors or MOSFETs with matching parameters, working in a push-pull manner within the circuit.
- In operation, only one of the two symmetrical switches is turned on at any given time, allowing the output to either source current to or sink current from the load.
- The push-pull output stage can both improve the circuit's load capacity and enhance switching speed.

Push-pull output: where it fits

- Useful for on-board logic at same voltage.
- Simply put, push-pull output can output both high and low levels, and it has driving capability at both levels.

Open-drain output: how it works

- Open drain
- outputs consist of NMOS transistors with their drains connected to the pin.
- On the board a termination resistor or pull-up resistor is connected between the pin and the power supply and this completes the path from power to ground, via the transistor.
- There is also on the pin a parasitic capacitor between the pin and ground.
- The capacitor models the combined capacitance of everything attached to output pin, including the PCB track capacitance and the pin capacitance of any connected devices.

Open-drain output: why it's useful

- Is useful when when sharing a line, implementing fail-safe wired-OR, or level shifting (with proper external circuitry).

Drive strength and slew rate

- Drive & slew:
- Use the lowest drive and slowest slew that still meets timing.

Protection and aircraft interfacing

- Aircraft systems are often 28 V nominal.
- Never connect these directly to MCU pins. Use resistor dividers, TVS (Transient Voltage Suppressor), RC filters, and isolation (optocoupler or digital isolator) as required.
- For outputs, use open-drain transistor/MOSFET stages into aircraft discrete networks and add EMI/ESD components per DO-160.

Pin mux, pad control, and GPIO driver APIs

- Pin mux & pad control: fsl_iomuxc.h -> IOMUXC_SetPinMux() and IOMUXC_SetPinConfig() (or tool-generated BOARD_InitPins() in pin_mux.c).
- GPIO driver: fsl_gpio.h ->
- gpio_pin_config_t (direction, default value, interrupt mode),
- GPIO_PinInit(GPIOx, pin, &cfg), GPIO_PinWrite(), GPIO_PinRead(),
- GPIO_PortSetInterruptConfig(), GPIO_PortClearInterruptFlags().

Board helpers and why pad names matter

- Board helpers: board.c/h, fsl_debug_console.h for PRINTF(), BOARD_InitBootPins(), BOARD_InitBootClocks().
- Exact LED & Button macros from your SDK's boards/evkbimxrt1050/.../board.h:
- The pad names above matter for electrical settings (pulls, hysteresis, open-drain). BOARD_InitBootPins() usually configures these for you; when you need to override, use the pad constants shown in Section 8 (Lab 4).

EVKB LED & button macros (from board.h)

- Exact LED & Button macros from your SDK's boards/evkbimxrt1050/.../board.h:

```c
#define LOGIC_LED_ON (0U)
#define LOGIC_LED_OFF (1U)
#define BOARD_USER_LED_GPIO GPIO1
#define BOARD_USER_LED_GPIO_PIN 9U // Pad: GPIO_AD_B0_09 GPIO1_IO09

#define BOARD_USER_BUTTON_GPIO GPIO5 // Pad: SNVS_WAKEUP GPIO5_IO00
#define BOARD_USER_BUTTON_GPIO_PIN (0U)
#define BOARD_USER_BUTTON_IRQ GPIO5_Combined_0_15_IRQn
#define BOARD_USER_BUTTON_IRQ_HANDLER GPIO5_Combined_0_15_IRQHandler
#define BOARD_USER_BUTTON_NAME "SW8"
```

Requirement and design choices

- Requirement: "Provide a debounced active-low input with noise immunity and a push-pull output to drive an on-board LED."
- Hardware choices: Input uses pull-up + Schmitt trigger; software debounces. Output uses push-pull at the lowest drive/slowest slew that still meets timing.

Software path: pins -> GPIO -> interrupts

- Software path: 1) Call BOARD_InitBootPins() (and clocks/console) to set mux & baseline pad config. 2) Initialize GPIO pins via GPIO_PinInit(). 3) If using interrupts on the input, configure GPIO_PortSetInterruptConfig() and enable its NVIC IRQ. 4) Read inputs (poll or ISR) and write outputs.

WOW discrete use case (with diagnostics)

- Scenario: An LRU (Line-Replaceable Unit) ingests a WOW discrete via an isolated interface, filters bounce/spikes, votes samples for truth, timestamps edges for maintenance logs to the CMC (Central Maintenance Computer), and asserts a FLT/GRD (Flight/Ground) status discrete on an open-drain output so multiple LRUs can share the line (wired-OR). Power-up and periodic BIT (Built-In Test) detect stuck-at faults.

Engineering choices and what to record

- Engineering choices: Input with internal pull-up + hysteresis + 5 ms debounce; output as open-drain (external pull-up on the backplane). Software records both raw and filtered states with timestamps.

Scenarios to simulate: robustness and diagnostics

- WOW with transient spikes: Inject brief pulses; show Schmitt + voting reject them. Log edge timestamps and report "Noise event ignored; no state change."
- Wired-OR FLT/GRD line: Tie two open-drain outputs with a single pull-up. Show either controller can assert ground. Demonstrate fault containment when one MCU resets.
- Stuck-at BIT: Short the WOW input to ground; detect stuck-low during maintenance stimulus and report to the CMC (Central Maintenance Computer) (log to console for the lab).

Scenarios to simulate: sharing lines, faults, and EMI

- EMI mitigation tradeoff: Observe ringing at high toggle rates; then reduce drive/slow slew via pad config and measure improvement.

Best practices: electrical and safety basics

- Match electrical domains: Never expose 28 V aircraft discretes directly to MCU pins. Use dividers, TVS, isolation.
- Fail-safe states: Choose pulls and output types so a broken wire or power-down yields the least hazardous interpretation.
- Enable hysteresis: Turn on Schmitt trigger for external discretes.
- Respect pull strengths: Disable MCU pulls if a strong external pull exists to avoid undesired dividers.
- Debounce deterministically: Use a stable time base (SysTick or GPT). Record raw and filtered states.
- BIT for stuck-at/open: Exercise inputs/outputs at power-up and periodically; use loopbacks when possible.

Best practices: software, process, and traceability

- Tame EMI: Use the lowest drive and slowest slew that meet timing; add small series resistors if needed.
- ISRs stay short: Clear flags first; defer work to main loop/task.
- Document bank supplies & mux: Keep a one-pager mapping pads -> banks -> rails; control pin_mux.c under configuration management.
- Traceability: Requirements -> pad config -> code -> tests; keep pin_mux.c settings under revision control.

Assessment checklist

- Asserted polarity and fail-safe state documented?
- Pad configs explicitly set pulls, hysteresis, slew, drive, and open-drain (as needed)?
- Debounce/filter rationale stated (e.g., 5 ms)?
- BIT covers stuck-high/low and open-circuit?
- Edge timestamps logged for maintenance?
- EMI/ESD robustness considered (lab + DO-160 assumptions)?

GPIOs around ARINC 429

- ARINC (Aeronautical Radio, Incorporated) 429 is a differential, unidirectional serial bus. While not a discrete itself, you frequently manage enables, health discretes, and label-present lines as GPIOs alongside the ARINC transceiver (e.g., on an ADK-8582 interfaced over UART). The GPIO concepts above—pad config, open-drain, debounce, BIT—apply directly to those companion lines.

Implementation guidance

- Prefer the board macros shown here instead of hardcoding banks/pins. This matches the SDK and reduces porting effort.
- Use the MCUXpresso Pins Tool to generate pin_mux.c and explicitly include the electrical options you require (pulls, hysteresis, drive, slew, open-drain). Keep the generated file under configuration control.