A09G-2 Comm Protocols (S25)

Your most important (high-risk) peripheral

Current Situation

The project involves designing a FreeRTOS-based embedded system using the Microchip SAMW25 microcontroller. The system performs real-time power monitoring and control for three actuators: an Antenna Tx module, a DC motor fan, and an LED. These actuators are powered through GPIO-controlled relays and monitored for voltage and current draw using the **INA3221 triple-channel current sensor** .

The INA3221 is responsible for sampling shunt voltage and bus voltage across all three actuator channels. These readings are used to calculate real-time current draw, which directly influences decisions for turning actuators ON or OFF based on power availability and load conditions. The INA3221 communicates with the SAMW25 via the I²C interface and operates continuously in background sampling mode.

The goal at this stage is to:

- Initialize the INA3221 sensor
- Interface it via I²C
- Continuously read raw voltage/current values from all three channels
- Print the readings via serial for verification and dashboard use

What your high-risk component is

Component Name	**INA3221 Triple-Channel High-Side Current and Bus Voltage Monitor**
Datasheet Section	Basic ADC Functions (Section 8.3.1-2 –*Alert Monitoring)*
Type	Power monitoring IC with analog front-end and digital I²C interface
Purpose	Measures both**shunt voltage (for current) and bus voltage** for up to three independent power channels
Key Registers	`0x01/03/05`: Shunt Voltage CH1/CH2/CH3 `–` 0x02/04/06`: Bus Voltage CH1/CH2/CH3
Operating Mode	Continuous Conversion Mode (`MODE = 111`) configured via Configuration Register (`0x00`)
Communication	I²C (Supports Fast up to 400 kHz and High-Speed up to 2.44 MHz)
Address	Default:`0x40` (when A0 pin tied to GND) Table 1 in datasheet
Supported Features	- Real-time voltage and current measurement `– Power-valid alert output`- Critical and warning alert thresholds `– Summation of channel currents`- Power sequencing validation (Timing Control)
Use in Project	- Monitor each actuator (Antenna, Fan, LED)`– Compute real-time power draw`- Make decisions for load control`- Log and visualize data on dashboard

What Makes INA3221 a High-Risk Component

Reason	Details	Datasheet Reference
Core to system functionality	The INA3221 is responsible for monitoring real-time current and voltage across all three actuators. Inaccurate readings can lead to incorrect load control, battery drainage, or even hardware damage.	Section 8.3.1 –*Basic ADC Functions*
Real-time data dependency	Load toggling logic and dashboard updates are directly tied to the values reported by this sensor. It influences control decisions at runtime.	Section 6 –*Software Requirements*(SRS 02, 04, 05)
Multi-channel, high-speed I²C communication	Requires reliable and synchronized communication with the MCU. Shared I²C bus with the BQ27441 increases risk of bus collisions or noise.	Section 8.5.1 –*Bus Overview*``Section 8.5.2 –*Writing To and Reading From*
Analog to digital conversion accuracy	Any errors in shunt voltage conversion can significantly impact the current calculation, especially since the actuators operate up to 1A.	Section 8.3.1 –*Shunt Voltage Measurement*``Register Maps –`0x01`,`0x03`,`0x05`
Register-based programming	The device uses 16-bit registers with MSB-first byte ordering and requires precise pointer management over I²C. Incorrect interpretation of raw values could lead to invalid readings.	Section 8.5.2 –*Read Word Format*(Figure 29)``Table 3 –*Register Maps*
Requires careful configuration	Initialization includes enabling all channels, setting averaging modes, and configuring conversion times. A	Section 8.3.1 –*Continuous Conversion Mode*``Register `0x00`–*Configuration Register*

Reason	Details	Datasheet Reference
	misconfigured mode (e.g., incorrect `MODE`bits in register `0x00`) can disable continuous sampling or one of the channels.	
Power-valid and alert output logic	Optional alert pins (PV, Critical, Warning) may introduce complexity if used later. Ensuring their correct use and electrical interfacing requires care.	Section 8.3.2 –*Alert Monitoring*`Figures 19, 20 – *Power-Valid Output*
No built-in self-test or calibration	There's no hardware fallback if readings are incorrect — verification has to be done entirely in software or with external test equipment.	General device limitation (implied across Section 8.3 and 8.5)

How it interfaces with the MCU and the rest of the system

Aspect	Details	Datasheet Reference
Communication Protocol	I²C interface (compatible with SMBus), supports Fast Mode (up to 400 kHz) and High-Speed Mode (up to 2.44 MHz).	Section 8.5.1 –*Bus Overview*
MCU Interface Pins	Connected to the SAMW25 via shared I²C lines (SCL, SDA). Pull-up resistors required on both lines.	Section 8.5.1.2 –*Serial Interface*
I²C Address	Configured using the A0 pin; default address is `0x40`when A0 is tied to GND.	Table 1 –*Slave Address Options*
Register Access	Uses a register pointer mechanism. Master sets the register pointer, then reads/writes 16-bit MSB-first values.	Section 8.5.2 –*Writing To and Reading From the INA3221*

Aspect	Details	Datasheet Reference
Channel Mapping	Each of the three INA3221 channels is connected to one actuator (CH1 → Antenna, CH2 → Fan, CH3 → LED).	Section 8.3.1 –*Basic ADC Functions*
Voltage Measurement	Bus voltage is sensed on the IN– pin of each channel, referenced to GND.	Section 8.3.1 –*Bus Voltage Measurement*
Current Measurement	Differential voltage measured across a precision shunt resistor between IN+ and IN–; converted to current in software.	Section 8.3.1 –*Shunt Voltage and Current Conversion*
Alert Logic (optional)	PV, Critical, Warning, and TC alert pins are available but not required for initial integration.	Section 8.3.2 –*Alert Monitoring*
Shared I²C Bus	INA3221 shares the I²C bus with the BQ27441 battery fuel gauge. Bus arbitration and noise immunity are critical.	Section 8.5.1 –*Bus Protocol and Arbitration*

Future Plans

Planned Feature / Task	Description	Relevant Datasheet Reference
Integration with actuator control	Use INA3221 current readings to dynamically enable/disable relays (via GPIO) for the antenna, fan, and LED based on predefined current thresholds.	Section 8.3.1 – *Basic ADC Functions*
Add critical alert thresholds	Program the**Critical Alert Limit registers**to detect when instantaneous current exceeds 1A (based on shunt voltage limit). Trigger relay shutdown or system warning.	Section 8.3.2.1 – *Critical Alert*

Planned Feature / Task	Description	Relevant Datasheet Reference
Add warning alert levels	Implement**Warning Alert Limit registers**to log excessive current draw before critical limits are reached. This enables proactive system behavior.	Section 8.3.2.2 – *Warning Alert*
Implement current summation monitoring	Enable**Summation Control**for all three channels to monitor total system current and trigger a shutdown if it exceeds 2.5A.	Section 8.3.2.1.1 –*Summation Control Function*
Enable Power-Valid logic	Use the**Power-Valid Upper and Lower Limit registers**to verify that all bus voltages are stable and within range (e.g., between 3V and 5.5V).	Section 8.3.2.3 – *Power-Valid Alert*
Optimize sensor polling task	Adjust FreeRTOS sampling intervals and conversion settings (averaging, timing) for better real-time responsiveness and reduced overhead.	Section 8.3.1 – *Operating Modes & Averaging*
Dashboard visualization	Map voltage and current readings to real-time dashboard via MQTT/Wi-Fi and visualize actuator behavior.	Project SRS (SRS 04)
Add sensor failure detection	Include checks for invalid readings (e.g., all zeros or stuck values) to detect disconnection or malfunction.	Section 8.3.1 – *Single-shot + Readback Logic*
Compare with multimeter values	Perform sensor calibration using external measurement devices (e.g., multimeter or oscilloscope) to verify accuracy.	Practical calibration plan

What tests do you plan to run to verify the functionality of your peripheral and library code (Hint: Look at your requirements to figure out your test cases.)

Test Name	Purpose	Description	Datasheet Reference / SRS
I²C Communication Test	Verify read/write over I²C	Use known registers (e.g., Configuration `0x00`) to test reading/writing 16-bit values.	Section 8.5.2 – *Serial Interface Transactions*
Channel Activation Test	Confirm all 3 channels are enabled and responding	Set `MODE`bits to `111`in Config Register (`0x00`) and check valid voltage/current values from CH1, CH2, CH3.	Section 8.3.1 – *Continuous Mode*
Current Reading Accuracy Test	Validate current calculation from shunt voltage	Connect a resistive dummy load (e.g., 50Ω), read shunt register, compute current and compare with multimeter.	Register `0x01/03/05`LSB = 40 μV
Bus Voltage Accuracy Test	Verify voltage readings	Power each actuator at a known voltage (e.g., 5V), read bus register and confirm accuracy.	Register `0x02/04/06`LSB = 8 mV
Load Toggle Behavior Test	Confirm readings respond to GPIO-controlled load changes	Toggle actuator ON/OFF via relay and confirm change in current/voltage reading.	SRS 05 –*Manual override commands*
Polling Interval Test	Ensure FreeRTOS task reads data within 1s ±100ms	Use `vTaskDelay()`and timestamps on serial prints to validate timing.	SRS 02 –*Sensor polling interval*
Alert Register Handling Test	Prepare for future	Program critical alert limits for shunt voltage	Section 8.3.2.1 – *Critical Alert*

Test Name	Purpose	Description	Datasheet Reference / SRS
	critical/warning logic	and simulate an overcurrent condition.	
Sum Current Test	Test total system current using summation register	Enable summation mode, apply load on all 3 channels, and verify sum register behavior.	Section 8.3.2.1.1 – *Summation Control*
Bus Voltage Drop Test	Simulate low voltage condition on one rail	Lower power supply on one actuator temporarily and check bus voltage + power-valid alert flag.	Section 8.3.2.3 – *Power-Valid Alert*

Pseudocode Implementation

I²C Communication Test

```
function test_i2c_communication():
    device_address = 0x40
    config_register = 0x00
    expected_config_value = 0x7127

    // Write config value to INA3221
    i2c_write_word(device_address, config_register, expected_config_value)

    // Read back config value
    actual_value = i2c_read_word(device_address, config_register)

    if actual_value == expected_config_value:
        print("I2C Test Passed")
    else:
        print("I2C Test Failed")
```

Read Current & Voltage from All Channels

```
// Pseudocode for periodic reading
function read_all_channels():
    for channel in [1, 2, 3]:
        shunt_reg = get_shunt_reg(channel)   // 0x01, 0x03, 0x05
        bus_reg = get_bus_reg(channel)       // 0x02, 0x04, 0x06
```

```
        raw_shunt = i2c_read_word(0x40, shunt_reg)
        raw_bus = i2c_read_word(0x40, bus_reg)

        v_shunt_mV = raw_shunt * 0.04        // 40 µV/bit
        current_mA = v_shunt_mV / 0.1        // Assuming 0.1Ω resistor

        v_bus_V = raw_bus * 0.008            // 8 mV/bit

        print("CH", channel, ": VBUS =", v_bus_V, "V | I =", current_mA,
"mA")
```

3. Polling Interval Test (FreeRTOS-style)

```
// Pseudocode for FreeRTOS Task
function vINA3221_Task():
    initialize_INA3221()  // Writes config register, enables channels

    while true:
        read_all_channels()
        delay(1000ms)     // vTaskDelay(pdMS_TO_TICKS(1000))
```

4. Load Toggle Behavior Test

```
function test_load_toggle_behavior():
    relay_pin = GPIO_PIN_RELAY_FAN

    set_gpio_high(relay_pin)  // Turn load ON
    delay(200ms)              // Wait for current to stabilize

    current_on = read_current(channel=2)

    set_gpio_low(relay_pin)   // Turn load OFF
    delay(200ms)

    current_off = read_current(channel=2)

    if current_on > current_off:
        print("Relay toggle behavior verified")
    else:
        print("Unexpected current readings")
```

Please Note: We did not receive our critical component yet so I just wrote the

pseudocode for my critical component sensor (INA3221)