IMU

The Inertial Measurement Unit (IMU) provides three-axis acceleration, angular velocity, and magnetic field measurements for attitude determination and motion analysis.

Hardware

Parameter	Value
Device	Xsens Avior (Xbus protocol, MTi-1-series compatible pipe interface)
Interface	I2C1, pins PB8 (IMU_I2C_Clock) / PB9 (IMU_I2C_Data), internal pull-ups enabled
I2C address	0x6B (7-bit, MTi-1 series default, override with -D XSENS_I2C_ADDR_7BIT)
Output rate	100 Hz requested for accel/gyro/mag (XSENS_OUTPUT_RATE_HZ)
I2C timeout	100 ms (XSENS_I2C_TIMEOUT_MS)

Important: the pipe opcodes, default I2C address, and Xbus data identifiers used by the driver are the documented Xsens MTi-1-series values. The Avior is Xbus-compatible, but confirm these against the Avior datasheet (mtidocs.xsens.com) and override with -D build flags if your unit differs. The driver was never validated against physical hardware.

Communication Protocol (Xbus over I2C)

Xbus messages move through "pipe" opcodes used as an 8-bit register address:

- 0x03 ControlPipe: write Xbus command messages
- 0x04 PipeStatus: read 4 bytes, notification size (LE16) and measurement size (LE16)
- 0x06 MeasurementPipe: read a pending MTData2 measurement message

```
Xbus frame: [0xFA][0xFF][MID][LEN][DATA...][CHK]
CHK makes (BID + MID + LEN + DATA + CHK) & 0xFF == 0
```

On the first poll the driver configures the device once: GoToConfig, SetOutputConfiguration (acceleration + rate of turn + magnetic field at 100 Hz, float32), GoToMeasure. If configuration fails, poll returns RESULT_ERR_COMMS (device not responding on I2C).

Data Structure

```
typedef struct {
    float accel[3];            /* [X, Y, Z] acceleration, m/s² */
    float gyro[3];             /* [X, Y, Z] angular velocity, °/s
                                  (converted from rad/s by the driver) */
    float mag[3];              /* [X, Y, Z] magnetic field, Xsens arbitrary
                                  units (~1.0 = local Earth field), NOT µT */
    uint32_t timestamp;        /* Current reading timestamp (HAL_GetTick ms) */
    uint32_t last_timestamp;   /* Previous reading timestamp */
} imu_data_t;
```

Unit notes: the gyroscope values are converted from rad/s to °/s inside the driver. The Xsens magnetic field output is in arbitrary units where roughly 1.0 equals the local Earth field; it is stored as-is and must be scaled externally if µT are needed.

Initialization & Usage

Initialize IMU

```
imu_data_t imu_data;
imu_sensor_init(&imu_data);   /* zeroes the structure */
```

Poll IMU Data

```
result_t imu_result = poll_imu_sensor(&imu_data);

if (imu_result == RESULT_OK) {
    /* Acceleration (m/s²) */
    float accel_x = imu_data.accel[0];
    float accel_y = imu_data.accel[1];
    float accel_z = imu_data.accel[2];

    /* Angular velocity (°/s) */
    float gyro_x = imu_data.gyro[0];
    float gyro_y = imu_data.gyro[1];
```

```
    float gyro_z = imu_data.gyro[2];

    /* Magnetic field (Xsens arbitrary units) */
    float mag_x = imu_data.mag[0];
    float mag_y = imu_data.mag[1];
    float mag_z = imu_data.mag[2];
}
/* RESULT_ERR_COMMS: device not responding, or no fresh sample ready yet */
```

Advanced Functions

Update with Raw Values

```
result_t imu_sensor_update(
    imu_data_t *imu,
    float ax, float ay, float az,  /* Accelerometer values */
    float gx, float gy, float gz,  /* Gyroscope values */
    float mx, float my, float mz,  /* Magnetometer values */
    uint32_t timestamp
);
```

Calculate Acceleration Magnitude

```
float acceleration_magnitude = imu_get_acceleration_magnitude(&imu_data);
/* |a| = sqrt(ax² + ay² + az²), useful for impact and free-fall detection */
```

Orientation Helpers (from the gravity vector)

```
float pitch_deg = imu_get_pitch(&imu_data); /* atan2(ay, sqrt(ax²+az²)) in degrees */
float roll_deg  = imu_get_roll(&imu_data);  /* atan2(ax, sqrt(ay²+az²)) in degrees */
/* Assumes the device is relatively stationary */
```

Gyroscope Drift Check

```
/* true when all gyro axes are below the threshold (device at rest) */
bool stable = imu_check_gyroscope_drift(&imu_data, 1.0f);
```

Copying Data for Another Context

```
imu_data_t imu_copy;
imu_sensor_read(&imu_data, &imu_copy);
/* Plain struct copy, no locking: safe only if the source is not
 * being updated concurrently */
```

Sensor Ranges Used by the Driver Validators

Measurement	Accepted Range	Units
Acceleration	±16 g (±156.9 m/s²)	m/s²
Angular Velocity	±2000	°/s
Magnetic Field	±4900	driver limit (see unit note above)

Conversion Reference

From	To	Factor
g	m/s²	× 9.80665
rad/s	°/s	× 180/π (applied inside the driver)
Gauss	µT	× 100

Validation Functions

```
/* Driver-level validators (used by the main loop) */
bool imu_validate_accelerometer_range(imu_data_t *imu);  /* ±16 g in m/s² */
bool imu_validate_gyroscope_range(imu_data_t *imu);      /* ±2000 °/s */
bool imu_validate_magnetometer_range(imu_data_t *imu);   /* ±4900 */

/* Utility-library validators (sensor_basics.h) */
result_t validate_accelerometer_value(float accel_value); /* ±160 m/s² */
result_t validate_imu_data(float accel_x, float accel_y, float accel_z);
```

Protobuf Message Format

```
message SensorBoardIMUInfo {
    float accel_x;
    float accel_y;
    float accel_z;
    float gyro_x;
    float gyro_y;
    float gyro_z;
    float mag_x;
    float mag_y;
    float mag_z;
    SensorState state;
    IMUErrorCode error_code;
}

enum IMUErrorCode {
    IMU_NO_ERROR = 0;
    IMU_COMMUNICATION_FAILURE = 1;
    IMU_ACCELEROMETER_ERROR = 2;
    IMU_GYROSCOPE_ERROR = 3;
    IMU_MAGNETOMETER_ERROR = 4;
}
```

Common Applications

Impact Detection

```
float mag = imu_get_acceleration_magnitude(&imu_data);
if (mag > IMPACT_THRESHOLD) {
    /* High acceleration detected */
}
```

Tilt Detection

```
float pitch = imu_get_pitch(&imu_data);
float roll  = imu_get_roll(&imu_data);
```

Motion Classification

```
/* Static vs dynamic based on gyro magnitude */
float gyro_mag = sqrtf(gyro_x*gyro_x + gyro_y*gyro_y + gyro_z*gyro_z);
```

Integration Notes

- Single IMU instance in the main application (dual IMU planned)
- All nine axes (accel, gyro, mag) transmitted as independent fields at the main loop interval
- First poll performs one-time device configuration; a failing device degrades to IDLE / DISCONNECTED without blocking the loop
- On successful poll the main loop runs the three range validators and sets IMU_ACCELEROMETER_ERROR, IMU_GYROSCOPE_ERROR, or IMU_MAGNETOMETER_ERROR accordingly
- Timestamp tracking (HAL_GetTick) enables dead reckoning applications
- Filter algorithms can be applied to the raw data for smoothing
- Unit tested on host: init defaults, update/read round-trip, magnitude, pitch/roll, range validators (test/sensor_board/test_imu_sensor)

Revision #5
Created 2026-04-14 15:34:08 UTC by Shishir Nambiar
Updated 2026-07-08 14:36:50 UTC by Shishir Nambiar