Carina VIO SDK — A1088 API Reference

Version: v2.9.7
Updated: 2026-06-02
Header: `api/carina_a1088.h`
Library: `libcarina_vio.so`

Table of Contents

---

1. Overview

C API for the A1088 VIO tracking device. Pure C interface, compatible with C and C++.

Device Variants

Type	device_type	Description
Single-chip	10	Stereo/quad, auto exposure, VIO tracking
Dual-chip	20	Quad camera, dual-chip combined output
ISP	30	Stereo, integrated ISP

Typical Lifecycle

```
init → set callbacks → start → [pause/resume...] → stop → release
  ↑                                                          |
  └──────────────── can re-init and reuse ──────────────────┘
```

Removed APIs

These old APIs have been removed:

Removed	Replacement
`carina_a1088_use_config`	Config passed via `carina_a1088_init`
`carina_a1088_trigger_cam_stop`	No manual trigger needed
`carina_a1088_trigger_cam_start`	No manual trigger needed
`carina_a1088_set_daisch_imu_pms`	Handled by calibration flow
`carina_a1088_get_daisch_imu_sn`	Handled by calibration flow
`carina_a1088_get_dev_info`	Handled by calibration flow
`carina_a1088_stm_get_device_version`	Handled automatically
`carina_a1088_stm_get_cpld_version`	Handled automatically
`carina_a1088_stm_suspend`	Handled automatically
`carina_a1088_stm_resume`	Handled automatically

---

2. Quick Start

```
#include "carina_a1088.h"
#include "carina_error.h"

// 1. Config as YAML string
const char* config = "device_type: 10\nmax_cameras: 2\nCamera.fps: 30\n...";

// 2. Initialize
int ret = carina_a1088_init(
    (char*)config,          // YAML config (required)
    "./database.bin",       // ORB vocabulary (required)
    -1, -1, -1              // USB params: -1 = auto-detect
);
if (ret != 0) {
    printf("Init failed: %d\n", ret);
    return -1;
```

```
}

// 3. Start VIO
ret = carina_a1088_start(
    my_pose_callback,      // pose callback
    NULL,                  // vsync (optional)
    my_imu_callback,       // imu callback
    my_image_callback,     // image callback
    NULL,                  // feature points (optional)
    NULL,                  // events (optional)
    NULL                   // user data
);
if (ret != 0) { /* handle error */ }

// 4. Stop & release
carina_a1088_stop();
carina_a1088_release();
```

> **Tip:** For dual-chip devices, use `carina_double_a1088_init` instead of `carina_a1088_init`. Everything else stays the same.

---

3. Data Structures

3.1 Device Mode

```
struct carina_device_mode {
    uint8_t cam_en;       // Camera enable mask
    uint8_t data_fps;     // Data frame rate
    uint8_t cam_fps;      // Camera frame rate
    uint32_t imu_fps;     // IMU frame rate
};
```

3.2 Feature Points

```
struct carina_orb_point {
    int octave;              // Image pyramid level
    float angle;             // Feature orientation (degrees)
    float response;          // Feature strength
    float x, y;              // Image coordinates
    unsigned char desc[32];  // ORB descriptor (256 bits)
    unsigned int id;         // Unique ID
};

struct carina_lk_point {
    float x, y;              // Image coordinates
    unsigned int id;         // Unique ID
};
```

```
struct carina_points {
    struct carina_lk_point **points_lk;    // LK points per camera
    struct carina_orb_point **points_orb;  // ORB points per camera
    int points_lk_rows;                    // Cameras with LK points
    int points_lk_cols[4];                 // LK count per camera
    int points_orb_rows;                   // Cameras with ORB points
    int points_orb_cols[4];                // ORB count per camera
};
```

3.3 Nordic Wireless Data (AR Glasses)

```
struct carina_nordic_pose_data {
    double timestamp;
    float fpose[16];    // 4×4 transformation matrix
    bool bsync;
};

struct carina_nordic_imu_data {
    double timestamp;
    float acc_data[3];   // acceleration
    float gyro_data[3];  // gyroscope
    bool bsync;
};
```

---

4. Callbacks

All callbacks run on **internal worker threads**. Pointers are valid **only during the callback** (see § 8).

```
// Pose — 32 floats
// pose[0..15]  = TWB transformation matrix (4×4, column-major)
// pose[16..18] = velocity (vx, vy, vz)
// pose[22..24] = gyro bias
// pose[25..27] = accel bias
// pose[28]     = static ratio
// pose[30]     = tracking state
typedef void (*CarinaA1088PoseCallBackType)(float *pose, double timestamp, vo
id *user_data);

// VSync
typedef void (*CarinaA1088VsyncCallBackType)(double timestamp, void *user_da
ta);

// IMU — 6 floats
// data[0..2] = acceleration (m/s²)
// data[3..5] = gyroscope (rad/s)
typedef void (*CarinaA1088ImuCallBackType)(float *imu_data, double
timestamp, void *user_data);

// Image — up to 4 grayscale images; NULL for unavailable cameras
```

```
typedef void (*CarinaA1088CameraCallBackType)(
    char *img0, char *img1, char *img2, char *img3,
    double timestamp, int w, int h, void *user_data
);

// Feature points (pass NULL if not needed)
typedef void (*CarinaA1088PointsCallBackType)(
    struct carina_points *points, double timestamp, void *user_data
);

// Events (pass NULL if not needed)
typedef void (*CarinaA1088EventCallBackType)(unsigned char event, void *user
_data);

// HMD suspend/wakeup: 0=suspend, 1=wakeup
typedef void (*CarinaA1088SuspendCallBackType)(int state, void *user_data);

// Firmware update progress (0.0 ~ 1.0)
typedef void (*CarinaA1088ProcessCallBackType)(float progress, void *user_da
ta);

// Nordic wireless data receive
typedef void (*CarinaA1088NordicDataCallBackType)(
    const char* data_val, int data_len,
    unsigned char nordic_trans_type, void* user_data
);

// UVC camera frame (MJPEG/NV12)
typedef void (*CarinaUvcFrameCallback)(
    const uint8_t* data, int size,
    int width, int height, int64_t timestamp, void* user_data
);
```

---

5. Core API

5.1 Lifecycle

```
// ———————— Init ————————

// Single-chip / stereo
// custom_config: YAML config string (required)
// vocab_file_path: ORB vocabulary (database.bin)
// fd/bus/addr: USB params, -1 = auto-detect
int carina_a1088_init(char *custom_config, char *vocab_file_path,
                      int fd = -1, int bus = -1, int addr = -1);

// Dual-chip / quad
int carina_double_a1088_init(char *custom_config, char *vocab_file_path,
                             int fd_main = -1, int bus_main = -1, int addr_
main = -1,
                             int fd_div = -1, int bus_div = -1, int
addr_div = -1);
```

```
// ──────────── Start / Stop ────────────

int carina_a1088_start(/* callbacks */);  // start VIO
int carina_a1088_stop();                  // stop
int carina_a1088_release();               // release all resources
int carina_a1088_pause();                 // pause (preserve state)
int carina_a1088_resume();                // resume

// ──────────── HMD callback ────────────

int carina_a1088_set_hmd_state_callback(
    CarinaA1088SuspendCallBackType cb, void* user_data);
```

5.2 Device Info

```
char *carina_a1088_get_sn();                 // serial number
char *carina_a1088_get_cam_param();          // camera calibration YAML
char *carina_a1088_get_config();             // current config YAML
char *carina_a1088_get_config_des(pass);     // decrypted config
char *carina_a1088_get_sdk_version();        // SDK version string
char *carina_a1088_get_firmware_version();   // firmware version string
bool  carina_a1088_is_device_connect();      // is device connected?
bool  carina_a1088_get_cur_status();         // is device operational?
int32_t carina_a1088_get_device_type();      // device type enum value
```

> `char*` return values point to **internal static buffers**. Do NOT free them.
> Subsequent calls overwrite the buffer — copy data if you need it to persist.

5.3 Pose

```
// pose[16] = 4×4 transformation matrix (column-major), IMU-predicted
int carina_a1088_get_imu_pose(float *pose, double predicttime);

// pose[32] = full state (velocity, bias), for OpenGL rendering
int carina_a1088_get_gl_pose(float *pose, double predicttime);
```

5.4 Device Control

```
// Display
int  carina_a1088_switch_display_mode(unsigned char mode);  // 0=2D, 1=3D
int  carina_a1088_set_display_level(unsigned char level);   // brightness 0~4
int  carina_a1088_get_display_level();                        // get
brightness
int  carina_a1088_set_led_color(unsigned char color);         // LED color
bool carina_a1088_detect_worn();                              // wear
detection
uint16_t carina_a1088_get_proximity_val();                    // proximity
sensor
```

```
uint16_t carina_a1088_get_key_status();                      // key/touch
state

// Sensor
int  carina_a1088_reset_pose();               // reset pose to origin
bool carina_a1088_check_loop_state();         // loop closure state
int  carina_a1088_set_sync_ts(bool enable);   // timestamp sync
void carina_a1088_set_log_level(unsigned int level);  // 0=off ~ 4=debug
int  carina_a1088_set_low_power_mode(bool enable);    // low power mode
int  carina_a1088_set_static_check_time(float sec);   // static detection
time

// Runtime config update
bool carina_a1088_update_custom_config(const char* config_val);
```

5.5 Exposure / Gain Control

```
// Set exposure or gain for specific camera(s)
// mode: 0x01=exposure, 0x02=gain
// cam_id: 0x01=CAM0, 0x02=CAM1, 0x04=CAM2, 0x08=CAM3
int carina_a1088_set_exposure_gain_with_id(
    unsigned char mode, unsigned char cam_id, int value);

// Auto exposure
int   carina_a1088_open_auto_exposure(int minG, int maxG);    // enable
int   carina_a1088_close_auto_exposure();                      // disable
int   carina_a1088_open_auto_exposure_v2(int minG, int maxG);  // 4-cam
independent

// Read
float carina_a1088_get_exposure_time(unsigned char camid);   // ms
int   carina_a1088_get_exposure_gain(unsigned char camid);   // gain
multiplier
int   carina_a1088_get_auto_exp_status();                    // 0=fixed,
1=auto,2=indep
```

5.6 USB Custom Data

```
// Send (USB2.0: max 512 bytes, USB3.0: max 1024 bytes)
int carina_a1088_send_custom_data(const char *data, int len);

// Read (returns static buffer, min 32 bytes)
char* carina_a1088_read_custom_data(int len);
```

5.7 UVC Camera

Fixed VID:PID = 2bc5:052b. MJPEG on Win/Linux, NV12 on macOS.

```
int  carina_a1088_uvc_open(int width, int height, int fps, int fd = -1);
void carina_a1088_uvc_close();
```

```
int  carina_a1088_uvc_start(CarinaUvcFrameCallback cb, void* user_data);
void carina_a1088_uvc_stop();   // stop stream, device stays open
```

5.8 Nordic Wireless (AR Glasses)

```
bool carina_a1088_nordic_send_imu_data(
    const struct carina_nordic_imu_data* data, int size,
    unsigned char datatype, int level);

bool carina_a1088_nordic_send_pose_data(
    const struct carina_nordic_pose_data* data, int size,
    unsigned char datatype, int level);

void carina_a1088_nordic_data_callback(
    CarinaA1088NordicDataCallBackType cb,
    int trx_mode, void* user_data);
// trx_mode: 0x01=TX, 0x10=RX, 0x11=TRX

float carina_a1088_nordic_time_diff();  // TX-RX time diff (seconds)
```

5.9 Firmware Update

```
void carina_a1088_set_process_callback(
    CarinaA1088ProcessCallBackType cb, void* user_data);

bool carina_a1088_update_chip_fw(
    const char* fw_path, int vid, int pid, int fd);
```

5.10 Factory / Calibration Helpers

```
int carina_a1088_set_sn(const char* sn);                // returns 1=ok
0=fail
int carina_a1088_set_dev_cali_info(const char* data,     // returns 1=ok
0=fail
                                   int len, int mode);
```

5.11 Custom Parameter Storage

```
bool carina_a1088_write_cus_para(const char* data, int len);
uint32_t carina_a1088_read_cus_para(char** out_data);
```

5.12 Debug Helpers

```
// Get data/analysis thread IDs (debugging)
void carina_a1088_get_thread_id(uint64_t* data_thread, uint64_t* analysis_th
read);
```

6. Configuration

6.1 Key Parameters

Key	Type	Description	Typical
`device_type`	int	Device type	10 / 20 / 30
`max_cameras`	int	Max camera count	2 ~ 4
`Camera.fps`	int	Camera FPS	30
`Imu.fps`	int	IMU rate	1000
`chip_tracking`	bool	On-chip tracking	true
`orb_database_path`	string	Vocabulary path	`./database.bin`

6.2 Templates

Config templates are in `config/templates/`:

- `a1088_base_template.yaml` — single-chip base config
- `four_cam_base_template.yaml` — quad-camera setup

7. Examples

7.1 Basic Usage

```
#include "carina_a1088.h"
#include <stdio.h>
#include <unistd.h>

void on_pose(float *pose, double ts, void *user) {
    float *TWB = pose;
    printf("Pos: [%.3f, %.3f, %.3f]\n", TWB[12], TWB[13], TWB[14]);
}

void on_imu(float *imu, double ts, void *user) {
    // imu[0-2] = acc, imu[3-5] = gyro
}

void on_image(char *img0, char *img1, char *img2, char *img3,
              double ts, int w, int h, void *user) {
    // grayscale images, valid only during this callback
```

```
}

int main() {
    const char *config = "device_type: 10\nCamera.fps: 30\nImu.fps: 1000\n";

    int ret = carina_a1088_init((char*)config, "./database.bin", -1, -1, -1);
    if (ret) return ret;

    ret = carina_a1088_start(on_pose, NULL, on_imu, on_image, NULL, NULL, NU
LL);
    if (ret) { carina_a1088_release(); return ret; }

    sleep(5);  // run for 5 seconds

    carina_a1088_stop();
    carina_a1088_release();
    return 0;
}
```

7.2 Dual-Chip Device

```
// Just change the init function — everything else is identical
carina_double_a1088_init((char*)config, "./database.bin", -1,-1,-1, -1,-1,-1)
;
```

7.3 Pause / Resume

Useful for focus switching, scene loading, etc.:

```
// Pause — preserve internal state, reduce CPU usage
carina_a1088_pause();

// Do something expensive (e.g. load scene)
load_scene();

// Resume
carina_a1088_resume();
```

7.4 Predicted Pose

```
#include <time.h>

float pose[16];
struct timespec ts;
clock_gettime(CLOCK_MONOTONIC, &ts);
double predict_time = ts.tv_sec + ts.tv_nsec * 1e-9;

// pose is a 4×4 matrix (column-major)
// [0,4,8,12] = rotation columns + translation
carina_a1088_get_imu_pose(pose, predict_time + 0.0055);
```

8. Thread Safety & Data Lifetime

8.1 Thread Safety

- All API functions are **thread-safe**
- Callbacks run on internal worker threads — protect shared data accordingly

8.2 Data Lifetime Quick Reference

Function	Return Type	Lifetime	Free Required
`get_sn`, `get_cam_param`, `get_config`	`char*`	Static buffer, overwritten on next call	**No**
`get_sdk_version`, `get_firmware_version`	`char*`	Same as above	**No**
`read_custom_data`	`char*`	Same as above	**No**
`read_cus_para`	`uint32_t`	Same as above	**No**
`get_device_type`	`int32_t`	Value	**No**
`get_cur_status`, `detect_worn`	`bool`	Value	**No**
`get_display_level`	`int`	Value	**No**
`get_proximity_val`, `get_key_status`	`uint16_t`	Value	**No**
Callback data (pose, imu, image...)	Pointer	**Valid during callback only**	**No**
UVC callback data	`const uint8_t*`	**Valid during callback only**	**No**

8.3 Callback Data Rules

```
// ✅ CORRECT: copy data
void on_pose(float *pose, double ts, void *user) {
    static float last_pose[32];
    memcpy(last_pose, pose, sizeof(last_pose));
}

// ❌ WRONG: storing pointer (invalid after callback returns)
void on_pose(float *pose, double ts, void *user) {
```

```
    static float *stored = pose;  // DANGEROUS!
}
```
