FIRMWARE EXTERNAL INTERFACE
SPECIFICATION — HOST SIDE

gene core ファームウェア 外部インターフェース 仕様書

本書は、gene + Solist-AI 開発キットに含まれる、gene core に出荷時に書き込み済みのファームウェアが USB シリアル (テキストプロトコル) 経由でホスト PC へ提供する外部インタフェース（コマンド API）を規定します。

文書情報 / Document Information

対象製品 / Product	gene core（ESP32-S3 + センサー 4 種 の 1 セット）
対象 FW / Firmware	gene firmware for Solist-AI v0.1.0（proto 1.0）
ホスト I/F / Interface	USB Type-C シリアル / UART 921600 bps（8N1）
対象ホスト OS	Windows / Linux
配布形態 / Distribution	バイナリ書込済みハードウェアとして提供（ソース非公開）。後に precompile ライブラリとユーザ拡張可能コードを提供予定。
版 / Version	Rev. 1.1

本書の位置づけ / SCOPE
本書は gene core（マスター / ホスト側）の物理・論理・機能と、PC から制御するための UART コマンド API を規定します。gene core が SPI 経由で制御する Solist-AI (ML63Q2537) スレーブ側の内部仕様は本書の対象外で、別資料で提供されます。

改訂履歴 / Revision History

版 / Rev.	日付 / Date	変更内容 / Changes
1.1	2026-07	製品位置づけの記述を更新（p-ban.com 提供の PoC 向けハードウェアとして整理）。§6.7 マイク録音の `mic.storage` / ultrasonic 運用注記を明確化。コマンドリファレンスを整理。
1.0	2026-07	初版発行。ファームウェア v0.1.0 (Phase 6) 実装に基づき、物理仕様・論理構成・機能・UART コマンド・設定キーを規定。

商標 / TRADEMARKS

Solist-AI™ は ROHM Co., Ltd. の商標です。gene は p-ban.com が提供する PoC 向けプラットフォームの名称です。その他、本書記載の会社名・製品名は各社の商標または登録商標です。

目次 / Contents

1

CHAPTER 01

製品概要

gene core の位置づけ、同梱物、出荷時の自動認識、主な特長と想定用途を示します。まず本章で全体像を把握してください。

1 製品概要

1.1 gene core とは

gene core は、p-ban.com が提供する PoC (概念実証) 向けハードウェアです。gene + Solist-AI 開発キットにおいては、Solist-AI 連携ファームウェアとセンサー一式を組み合わせ、すぐにエッジ AI を試せるセットとして提供します。

AI アクセラレータ Solist-AI (ML63Q2537) モジュールと ホスト PC を仲介する役割を持ちます。交換式のセンサー子基板から取得したデータを、AI アクセラレータ Solist-AI にリアルタイムで渡し、学習・推論・信号処理をオンデバイスで実行できます。ホスト PC とは USB シリアル (テキストプロトコル) で接続し、コマンドを送るだけで制御できます。

すぐ使える状態で出荷
ファームウェアは書き込み済み、センサーは起動時に自動認識される状態でお届けします。お客様によるビルドや書き込みは不要です。

1.2 セット同梱物

gene core は 1 セット販売です。以下が付属します（MicroSD カードのみ非同梱・お客様手配）。

#	品目	役割	備考
1	gene core 本体 (ESP32-S3)	メイン MCU / マスター制御	ファームウェア書込済み
2	IMU 子基板 (BMI088)	6 軸 加速度 + 角速度	自動認識対応
3	ToF 子基板 (VL53L5CX)	8×8 測距アレイ (ジェスチャ)	自動認識対応
4	MIC 子基板 (SPH0641LU4H-1)	デジタル MEMS マイク (PDM)	自動認識対応

#	品目	役割	備考
5	温湿度 子基板 (SHT40)	温度・相対湿度	自動認識対応
6	MicroSD カード	長時間 / ultrasonic 録音の保存先	非同梱（お客様手配）

> ⚠ **センサーは排他装着**
> 本体のセンサーコネクタには 一度に 1 枚を取り付けて使用します。用途に応じて子基板を差し替えてください。

1.3 出荷時の状態 (自動認識)

各センサー子基板は、コネクタに装着するだけで種別が自動判別されるよう出荷時に構成済みです。

- 起動時、gene core 本体が装着子基板の種別 (IMU / ToF / MIC / TH) を 自動判別します。
- 判別結果に応じて該当ドライバが立ち上がり、共有ピンの役割も自動設定されます。お客様による種別設定は不要です。
- 詳細な自動認識フローは § 3.2 を参照してください。

1.4 主な特長と想定用途

AI
オンデバイス AI ― 学習・推論を Solist-AI で実行。PC は司令塔に専念。

4-in-1
交換式マルチセンサー ― 動作・空間ジェスチャ・音・環境を 1 つの本体で。

ASCII
テキストベース制御 ― 改行区切りの ASCII コマンド。任意の言語・端末から扱える。

NVS
設定の永続化 ― パラメータは本体内の不揮発メモリに保存され電源断後も保持。

PoC
想定用途 ― 動作 / ジェスチャ認識、近接検知、異音検知、環境ロギング等の検証。

– 本書の範囲と別資料

範囲	扱い
gene core (マスター) の物理・論理・機能・制御 API	**本書で扱う**
各センサーの利用方法・データ形式	**本書で扱う**
Solist-AI (ML63Q2537, SPI スレーブ) 側の内部仕様	別資料で提供
ジェスチャ判定・音響解析などの上位アルゴリズム	お客様アプリ側 (本書対象外)
gene core ハードウェアの公開仕様	p-ban.com 公式情報を参照 (→ § 8)

2

CHAPTER 02

物理仕様

gene core を構成するハードウェアの物理的な仕様をまとめます。ピン配置はすべて公開可能な情報で構成しています。

2 物理仕様

2.1 本体・外部インターフェース

出典: p-ban.com gene 製品仕様 (公開情報)。

項目	仕様
MCU	ESP32-S3-WROOM-1-N16R8 (Xtensa デュアルコア LX7, 最大 240 MHz)
SRAM / PSRAM	512 KB / 8 MB
フラッシュ	最大 16 MB (Quad SPI)
無線	Wi-Fi 802.11 b/g/n, BLE 5 (本 FW では未使用)
バッテリー	3.7 V / 400 mAh
USB	USB Type-C (USB 2.0 Full Speed) — PC 接続 + 充電
ストレージ	内蔵 flash (既定) + MicroSD スロット × 1 (マイク録音 WAV)
センサーコネクタ	12 ピン (センサー子基板 1 枚を装着)
表示 / 操作	充電 LED × 1, リセットボタン × 1 (電源兼用)
サイズ / 動作電圧	40 × 40 × 19 mm / 3.0–3.6 V
マウント	アクションカムマウント対応

2.2 同梱センサー子基板 (4 種)

センサーは 排他装着です。コネクタには 1 枚のみ取り付けます。各子基板は装着するだけで種別が自動認識されます。

略称	部品 / メーカー	I/F	I2C アドレス (7-bit) / 機能
IMU	BMI088 / Bosch	I2C	acc 0x18 / gyro 0x68 — 6 軸

略称	部品 / メーカー	I/F	I2C アドレス (7-bit) / 機能
ToF	VL53L5CX / STMicro	I2C	0x29 — 8×8（or 4×4）測距
MIC	SPH0641LU4H-1 / Knowles	PDM	（なし）— MEMS マイク
TH	SHT40-AD1B-R3 / Sensirion	I2C	0x44 — 温度・相対湿度

IMU ボードの I²C 制御元切り替え
IMU 子基板のみ、ボード上の solder bridge（ジャンパ）をはんだ付けで切り替えることで、gene core ではなく Main から I²C 制御 する構成に変更できます（既定は gene core 制御）。他のセンサーには本オプションはありません。

2.3 ピン割り当て

I2C (SCL/SDA) は全センサー共通で起動時に即時利用できます。一方、共有スロットピン (IO1 / IO2 / IO3) は 装着子基板によって役割が変わります。本体は種別を判定するまでこれらのピンを設定しません（電気的衝突を防ぐため）。

共通バス (全機種共通) と共有スロットピン

GPIO	ToF	IMU	MIC	TH
GPIO8 / 9	I2C_SCL / I2C_SDA — センサー共通, 1 MHz			
GPIO1	I2C_RST (出力)	—	PDM CLK (出力)	—
GPIO2	LPn (出力)	—	PDM DATA (入力)	—
GPIO3	INT (入力)	—	—	—
GPIO46	—	ボタン入力	—	—

本体内蔵デバイス (MicroSD / SPI / UART)

GPIO	信号	用途 / 備考
GPIO10–13	SD_CS / MOSI / CLK / MISO	MicroSD (SDSPI / SPI3) — MIC 録音時のみ
GPIO21	SPI_CS	→ Solist-AI
GPIO35	SPI_MISO	← Solist-AI
GPIO36	SPI_SCLK	→ Solist-AI — Mode 0
GPIO37	SPI_MOSI	→ Solist-AI
GPIO38	SPI_READY	← Solist-AI (READY 信号)
GPIO43 / 44	UART0_TXD / RXD	PC ⇔ gene — 921600 bps

GENE ⇔ SOLIST-AI 配線 (参考)
本体と Solist-AI は基板内で結線済みで、お客様の配線は不要です。SPI は 8-bit / MSB first / Mode 0 (CPOL=Low) で動作します。GPIO46 (IMU ボタン) は Solist-AI 側 (P83) にも分岐接続され、同じ信号を両側で取得できます。Solist-AI 側の詳細仕様は別資料を参照してください。

3

CHAPTER 03

システム構成 (論理)

各要素の役割、起動時の自動認識フロー、データの流れ、動作状態を説明します。物理仕様（第 2 章）と機能仕様（第 4 章）をつなぐ概念モデルです。

3 システム構成 (論理)

3.1 役割分担と自動認識

要素	役割
PC (Win / Linux)	司令塔。シリアルでコマンドを送り、応答・イベントを受け取る。上位アルゴリズム（判定・解析）はここ。
gene core (ESP32-S3)	マスター。PC コマンドの解釈、センサー駆動、Solist-AI への SPI 制御、データ整形・配信。
Solist-AI (ML63Q2537)	AI コプロセッサ (SPI スレーブ)。学習・推論・FFT・フラッシュ書込みを実行。詳細は別資料。
センサー子基板 (1 枚)	データ源。IMU / ToF / MIC / TH のいずれか。装着するだけで自己識別。

3.2 起動時の自動認識フロー

gene core は 起動時に 1 回、装着センサー子基板を自動判別します。

1. **I2C バス初期化**
   SCL/SDA のみ立ち上げ（共有ピン IO1/IO2/IO3 はまだ触らない）。
2. **種別判定**
   共通 I2C バス経由で装着子基板の種別 (IMU / ToF / MIC / TH) を判別する。
3. **ドライバ起動**
   種別に応じて IMU / ToF / MIC / TH ドライバを初期化。共有ピンの役割も種別に合わせて設定。
4. **運用開始**
   センサー共通タスクが、装着センサー種別に分岐して動作。

> ⚠ **差し替え後は再起動または SENSOR_RESCAN**
> 種別判定は起動時に行われます。子基板を差し替えたら本体を 再起動するか、`>SENSOR_RESCAN` で再検出してください（再起動不要）。未対応 / 未書込のセンサーは「デバイス 0 台」として扱われ、ストリーミングはできません。

3.3 通信モデルとデータフロー

PC と gene core は テキストベースのシリアルプロトコルで通信します。すべて改行 (`\n`, LF) 終端、1 行 = 1 メッセージ。通信は 921600 bps, 8N1, フロー制御なし。

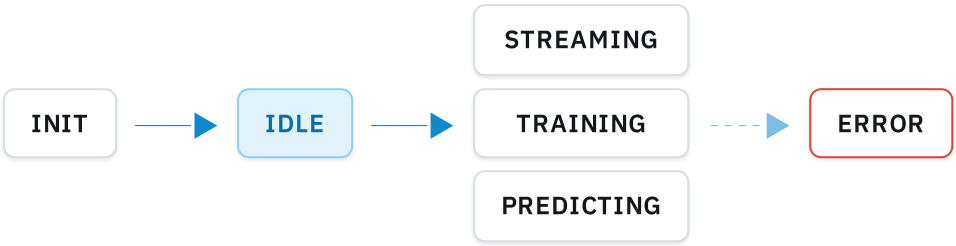
種別	先頭	例	方向
コマンド	`>`	`>START_STREAM 0`	PC → gene
成功応答	`<OK`	`<OK START_STREAM 0`	gene → PC
エラー応答	`<ERR`	`<ERR START_STREAM BUSY ...`	gene → PC
非同期イベント	`!`	`!SENSOR 0 12345 8 ...`	gene → PC (自発)
ログ / コメント	`#`	`# [12345] INFO: ...`	gene → PC (無視可)

コマンドには 同期応答 (`<OK` / `<ERR`) が必ず返ります。センサーデータはストリーミング中に 非同期イベント (`!SENSOR`) として届きます。データは大きく 3 つの経路に分かれます。

経路	流れ	制御コマンド
A	センサー → gene → PC（リアルタイム配信）	`START_STREAM` / `STOP_STREAM`
B	PC → gene →(SPI)→ Solist-AI → gene → PC（AI 演算）	`ML_*` / `FFT_*`
C	MIC → gene → 内蔵 flash / MicroSD（録音）、後で PC が取得	`MIC_REC_*` / `FILE_DOWNLOAD`

3.4 動作状態・BUSY・設定永続化

gene core は内部に動作状態を持ち、`STATUS` コマンドで確認できます。



状態: INIT（初期化中） / IDLE（待機） / STREAMING（配信中） / TRAINING（学習中） / PREDICTING（推論中） / ERROR（致命的エラー時、任意の状態から遷移しうる）。

BUSY (排他制御)

一部の操作は同時実行できず、競合時は `<ERR ... BUSY ...>` を返します。照会系コマンド（`STATUS` / `VERSION` / `PING` / `GET` / `SET` / `FILE_LIST`）はストリーミング中でも常時利用できます。

状態	入る契機	影響
Streaming	`START_STREAM 0` 成功	再度の START_STREAM は BUSY。配信中の FILE_DOWNLOAD も BUSY
Recording	`MIC_REC_START` 成功	再度の MIC_REC_START は BUSY。録音中ファイルへの FILE_DOWNLOAD は BUSY
ML busy	`ML_TRAIN` / `ML_PREDICT`	完了は ML_GET_BUSY で確認
Flash session	`FLASH_SESSION_BEGIN` 成功	FLASH_SESSION_END まで継続

設定の永続化 (NVS)

`SET` で変更した設定値は本体内の不揮発メモリ (NVS) に保存され、電源を切っても保持されます。次回起動時も同じ設定で立ち上がります。ストリーミング中の `SET` は対応キーであればリアルタイムに反映されます（→ §7）。

4

CHAPTER 04

機能仕様

gene core でできること ― センサーデータ取得、AI 演算、信号処理、録音・ファイル取得 ― を機能単位でまとめます。コマンドの厳密な書式は第 6 章を参照してください。

4 機能仕様

4.1 機能一覧

機能	概要	主なコマンド
センサーストリーミング	IMU / ToF / TH をリアルタイム配信	START_STREAM / STOP_STREAM
マイク録音	音声を WAV で内蔵 flash（既定）/ MicroSD に録音	MIC_REC_START / STOP
ファイル取得	録音ファイル等をシリアルで取得	FILE_LIST / FILE_DOWNLOAD
AI 学習・推論	Solist-AI で逐次学習 / 推論	ML_*
信号処理 (FFT)	Solist-AI で FFT 演算	FFT_*
フラッシュ操作	Solist-AI のフラッシュ書込み	FLASH_*
設定 / システム	パラメータ・状態・版数・接続確認	SET / GET / STATUS / VERSION / PING / RESET

4.2 センサーデータ取得 (ストリーミング)

`>START_STREAM 0` で配信を開始すると、装着センサーに応じた `!SENSOR` イベントが連続的に届きます。`>STOP_STREAM` で停止します。共通形式は `!SENSOR <device_id> <ts_ms> <ペイロード...>`（`device_id` は常に `0`、`ts_ms` は gene 起動からの経過ミリ秒）。

種別	`!SENSOR` ペイロード	補足
IMU	ax ay az gx gy gz	16-bit raw。換算は imu.range_g / range_dps で PC 側

種別	!SENSOR ペイロード	補足
ToF	res d0 d1 ... dN	res=4/8、距離 mm、個数 N=res×res、欠測 0
TH	t_milli_c rh_milli_pct	1/1000 単位。例 27345→27.345 ℃、56123→56.123 %RH

数値はすべて整数
すべての数値は 整数で送出されます。物理単位への換算は PC 側で行います（テキストプロトコルを整数のみに保つ設計）。MIC はストリーミング非対応で録音方式です（→ §4.3）。

4.3 マイク録音とファイル取得 (MIC)

MIC はリアルタイム配信ではなく 「録音 →（内蔵 flash / MicroSD）→ ダウンロード」方式です（PDM の高データレートを UART で送り切れないため）。典型フロー:

serial · MIC flow

```
>MIC_REC_START
<OK MIC_REC_START path=/sdcard/mic_rec_12345_1.wav
        …（録音中）…
>MIC_REC_STOP
<OK MIC_REC_STOP path=/sdcard/mic_rec_12345_1.wav
!MIC_REC_DONE path=/sdcard/mic_rec_12345_1.wav    ← 完了通知（ポーリング不要）
>FILE_DOWNLOAD /sdcard/mic_rec_12345_1.wav
<OK FILE_DOWNLOAD size=4500000 crc32=0xa1b2c3d4
<… size バイトのバイナリ列（エスケープ無し）…>
```

- WAV のサンプルレートは `mic.clock_hz` から自動算出（`clock_hz / 64`）。`1024000`→16 kHz（音声 / ウェイクワード）、`4800000`→75 kHz（超音波 / 機械異音検知）。
- 録音は `mic.max_record_seconds`（既定 60 秒）で自動停止。
- 録音中に電源が切れた場合、最後にストレージへ書き出した以降のデータは失われる可能性があります（`MIC_REC_STOP` / 自動停止での正常終了時に WAV が確定）。
- ダウンロードのバイナリは `<OK` 行の `size=` バイトをそのまま受信し、`crc32=`（IEEE 802.3）で検証します。
- MIC 装着時に `>START_STREAM 0` を送ると `NOT_SUPPORTED` を返します（録音コマンドを使用）。

4.4 AI・FFT・Flash (Solist-AI 連携)

gene core は SPI マスターとして Solist-AI を駆動し、逐次学習 (online learning) と推論、FFT、フラッシュ操作を実行します。`Solist-AI` 連携機能はセンサー子基板に依存しません。

serial · ML flow

```
>ML_INIT <inst> <input> <hidden> <output> ...        ← 全 11 引数は §6.7 参照
>ML_TRAIN <inst> input=[...] target=[...]
>ML_GET_BUSY                          ← 0 になるまで待つ
>ML_PREDICT <inst> input=[...]
>ML_GET_RESULT <inst>                 ← 推論結果を取得
>ML_GET_LOSS <inst>                   ← 損失を取得
```

- 重み (β) と共分散行列 (P) の取得・設定（`ML_GET/SET_WEIGHT_*`）に対応し、学習済みモデルの保存・復元が可能です。
- 浮動小数は SPI 転送時に bfloat16 に変換されます（高速・低帯域）。
- **FFT**: PCM サンプル列を Solist-AI に渡し FFT を実行（`FFT_INIT` → `FFT_START` → `FFT_GET_RESULT`、サイズ最大 1024）。
- **Flash**: `FLASH_SESSION_BEGIN`〜`FLASH_SESSION_END` のセッション内で消去・書込み・状態取得を行います。

4.5 機能と装着センサーの対応

機能	IMU	ToF	MIC	TH
`START_STREAM` ストリーミング	✅	✅	✗ 録音	✅
`MIC_REC_*` 録音	—	—	✅	—
`FILE_*` / `ML_*` / `FFT_*` / `FLASH_*` / 設定	✅ センサー非依存（どのセンサーでも利用可）			

5

CHAPTER 05 · GET STARTED

はじめに (Get Started)

gene core を PC につないで最初のコマンドを送るまでを案内します。対象ホスト OS は Windows / Linux です。

5 はじめに (Get Started)

5.1 準備するもの

- gene core 本体（ファームウェア書込済み）と、使用するセンサー子基板 1 枚
- USB Type-C ケーブル（データ通信対応のもの）
- MIC の音声録音は内蔵 flash で動作し MicroSD は不要。長時間録音 / ultrasonic 録音には MicroSD カードを使用する（**非同梱・お客様手配**、`mic.storage=sd` で切替）
- PC (Windows / Linux) とシリアルターミナル、または Python (任意)

5.2 接続とシリアル設定

1. **センサーを装着**
   使うセンサー子基板を本体コネクタに 1 枚取り付けます（差し替えた場合は再起動するか `>SENSOR_RESCAN` で再認識）。MIC 利用時は MicroSD も挿入。
2. **USB Type-C で PC に接続し電源 ON**
   PC 上にシリアルポートが現れます（Linux: `/dev/ttyUSB0` / `/dev/ttyACM0`、Windows: `COM3` 等）。
3. **STATUS で待機 (IDLE) を確認**
   シリアルポートを開き `>STATUS` を送り、`state=IDLE` が返れば待機完了です。ポートが現れない場合は USB-UART ブリッジ用ドライバ (CP210x / CH34x 等) を導入してください。

項目	値
ボーレート	921600（固定 — 不一致だと文字化け）
データ / パリティ / ストップ	8 / なし / 1（8N1）
フロー制御	なし
改行コード	LF（\n）を送信

5.3 ターミナル / Python で試す

Linux では `screen` / `minicom` / `picocom` 等、Windows では Tera Term / PuTTY が使えます（送信改行を LF に設定）。接続後、以下を 1 行ずつ送信します。

terminal

```
# picocom の例（改行を LF で送る）
picocom -b 921600 --omap crlf /dev/ttyUSB0

>PING 1
<OK PONG 1
>VERSION
<OK VERSION proto=1.0 fw=0.1.0 build="..." solist.input_max=256
>STATUS
<OK STATUS uptime=12345 state=IDLE solist=ok devices=1
<OK STATUS device:0 type=TOF bus=i2c addr=0x29 fw=VL53L5CX_2.0.1
```

スクリプトや自動評価には Python (pyserial) が便利です。`<` 行（同期応答）・`!` 行（イベント）・`#` 行（ログ）を先頭文字で振り分けて処理します。

python · pyserial

```
import serial
ser = serial.Serial("/dev/ttyUSB0", 921600, timeout=1)  # Windows は "COM3" 等

def cmd(line):
    ser.write((line + "\n").encode())
    while True:
```

```python
        resp = ser.readline().decode(errors="replace").strip()
        if resp.startswith("<"):  # 同期応答を待つ
            return resp

print(cmd(">PING 1"))      # -> <OK PONG 1
print(cmd(">STATUS"))      # -> <OK STATUS uptime=... state=IDLE ...
```

最初のストリーミングは `>START_STREAM 0` で開始、`>STOP_STREAM` で停止。`!SENSOR` の内容はセンサー種別で異なります（→ §4.2）。

5.4 困ったときは

症状	確認
応答が文字化け	ボーレートが 921600 か。8N1 / フロー制御なしか。
何も返らない	ポート選択は正しいか。コマンドが `>` 始まり・LF 終端か。
`devices=0`	センサーが装着・認識されているか。差し替え後は 再起動 か `>SENSOR_RESCAN`。
`START_STREAM` が `NOT_SUPPORTED`	MIC 装着時。録音コマンドを使用。
`BUSY` が返る	既にストリーミング / 録音中。停止してから再実行。

詳しくは §8.2 トラブルシューティングを参照してください。

6

CHAPTER 06 · COMMAND REFERENCE

シリアルコマンド リファレンス

gene core が受け付けるすべての UART コマンドの書式と応答をまとめます。本リファレンスはファームウェア v0.1.0 の実装に基づきます。

6 シリアルコマンド リファレンス

6.1 共通仕様・エラーコード

項目	値
物理 / ボーレート	USB Type-C シリアル / 921600 bps
フレーミング	8N1 / フロー制御なし / 行終端 \n（LF）
コマンド	>COMMAND [引数...]
成功 / エラー応答	<OK COMMAND [...] / <ERR COMMAND <CODE> <msg>
非同期イベント / ログ	!EVENT [...] / # [ts] LEVEL: msg（無視可）

数値・配列の表記: 配列引数は `input=[v1,v2,v3,...]`（角括弧・カンマ区切り）、配列応答も `[v1,v2,...]` 形式。ML 系の浮動小数は SPI 転送時に bfloat16（上位 16bit）へ変換されます。

プロトコル共通規約

- **文字コード:** コマンド・応答・イベント・ログ、およびファイル名はすべて ASCII 印字可能文字のみ。非 ASCII（日本語等）は用いない。
- **行終端:** 双方向とも 1 行 = 1 メッセージで `\n` (LF) 終端（PC→gene・gene→PC のいずれも）。
- **<msg> フィールド:** `<ERR>` やログの `<msg>` は ASCII 印字可能文字のみで、LF・タブを含まない。最大長は 1 行 **256 バイト**。
- **命名の例外:** 応答の先頭語は原則コマンド名と一致するが、`PING` のみ応答が `<OK PONG ...` となる。
- **バイナリ転送の例外:** 「1 行 = 1 メッセージ」原則の例外として、バイナリを伴うコマンド（現状 `FILE_DOWNLOAD` のみ）は `<OK` 行の LF 直後から `size=` バイトの生バイナリが続き、その直後に次のメッセージが始まる。バイナリ区間にエスケープ・区切りは無い。
- **crc32 表記:** `crc32=0x<hex>` は常に 0 埋め 8 桁（IEEE 802.3, poly 0xEDB88320）。

エラーコード一覧

コード	意味
INVALID_PARAM	引数の不足 / 形式不正
INVALID_KEY	未知の設定キー
NOT_FOUND	デバイス / ファイルが見つからない
NOT_READY	サブシステム未初期化
INVALID_STATE	現在の状態では実行不可（前提コマンド未実行・セッション未開始等）
NOT_SUPPORTED	この状況では非対応の操作
NOT_STREAMING / NOT_RECORDING	対象が動作中でない
BUSY	使用中につき実行不可
DEVICE_ERROR	SPI / I2C / ハードウェア障害
IO_ERROR	ファイル / ストレージ障害
NO_MEM	メモリ不足
SIZE_EXCEEDED	データがバッファ上限超過
VERIFY_FAILED	書き込み後の照合不一致

6.2 システム

PING >PING <ts>

System

目的	接続確認。`<ts>` はラウンドトリップ計測用の任意整数（エコーされる）
応答	`<OK PONG 123`

VERSION >VERSION

System

目的	プロトコル / ファームウェア版数の取得 + Solist-AI capability (`input_max`) の取得
応答	`<OK VERSION proto=1.0 fw=<major>.<minor>.<build> build="<日付>" solist.input_max=<uint|unknown>`
補足	`solist.input_max`: Solist-AI が受け付ける `>ML_INIT` の `input_size` 上限（bf16 element 数）。Solist-AI 未接続 / capability 取得失敗時は文字列 `unknown`。

STATUS >STATUS

System

目的	システム状態 + 接続センサー。1 行目がサマリ、以降が接続デバイス（1 台ごと）
応答	`<OK STATUS uptime=<ms> state=<STATE> solist=ok devices=<N>` `<OK STATUS device:<id> type=<TYPE> bus=<bus> addr=<addr> fw=<ver>`
補足	`state`=INIT/IDLE/STREAMING/TRAINING/PREDICTING/ERROR、`type`=TOF/IMU/MIC/TH。`devices=0` のときは 1 行目のみ。

RESET >RESET <soft|hard>

System

目的	`soft` = 状態を INIT に戻す / `hard` = 本体再起動	
エラー	未知の reset type	INVALID_PARAM

RESCAN	>SENSOR_RESCAN	System
目的	装着センサーを再検出する（再起動不要）。子基板を差し替えた後に使用。現在のセンサーを解放し、装着子基板を判別し直して新しい種別のドライバを起動する	
応答	<OK SENSOR_RESCAN kind=<KIND> detected=<KIND> [device:0 type=... bus=... [addr=...] fw=...]	
補足	`kind`=再検出後に有効になったセンサー（IMU/TOF/MIC/TH/NONE）、`detected`=検出された種別（`kind ≠ detected` はドライバ初期化失敗）	
エラー	配信中 / 録音中	BUSY（先に STOP_STREAM / MIC_REC_STOP）
	応答なし	TIMEOUT

STORAGE	>STORAGE_INFO	System
目的	マイク録音の保存先ストレージ（`mic.storage` の設定に対応）の状態と、現在の録音設定での残り録音可能時間を取得する	
応答	<OK STORAGE_INFO backend=<internal|sd> path=</int|/sdcard> free_bytes=<n> rec_seconds_left=<n>	
補足	`rec_seconds_left` は現在の `mic.clock_hz`（PCM レート）と空き容量から算出した目安値。`backend=sd` で MicroSD 未挿入・不良のときは `<ERR STORAGE_INFO IO_ERROR ...>`。	

6.3 設定 (SET / GET)

SET	>SET <key> <value>	NVS 永続化
目的	設定変更（NVS に永続化）。キー一覧は §7 参照。例: `>SET imu.rate 100` → `<OK SET imu.rate 100`	
エラー	未知のキー	INVALID_KEY
	範囲外 / 形式不正	INVALID_PARAM

GET	>GET <key> / >GET *	System
目的	設定取得。`*` で全キー取得。例: `>GET imu.rate` → `<OK GET imu.rate 100`	

6.4 ストリーミング

START `>START_STREAM <device_id>` — Stream

目的	センサーデータ配信を開始する。配信中は `!SENSOR` イベントが連続で届く
引数	`<device_id>`: 常に `0`（将来の複数センサー同時装着に備えた予約）
イベント	`!SENSOR <id> <ts> <payload>`
エラー	引数不正 / デバイス無し → `INVALID_PARAM / NOT_FOUND` 既に配信中 → `BUSY` MIC 装着時 → `NOT_SUPPORTED`

STOP `>STOP_STREAM` — Stream

目的	配信停止
イベント	なし（配信停止。以降 `!SENSOR` は届かない）
エラー	配信していない → `NOT_STREAMING`

6.5 マイク録音 (MIC)

REC `>MIC_REC_START [basename]` — MIC

目的	録音開始。`basename` 省略時は自動命名 → `<OK MIC_REC_START path=<フルパス>`
エラー	MIC 未装着 → `NOT_FOUND` 録音中 → `BUSY` 開始失敗 → `IO_ERROR`

REC >MIC_REC_STOP	MIC
目的	録音停止（WAV 確定まで待機）。完了時に `!MIC_REC_DONE path=<フルパス>` を発火
エラー	録音していない NOT_RECORDING ストレージ障害 IO_ERROR

6.6 ファイル (FILE)

FILE >FILE_LIST <path>	File
目的	ディレクトリ内のファイル一覧（通常ファイルのみ）を取得する
応答	<OK FILE_LIST count=<N> <OK FILE_LIST entry=<name> size=<bytes> …（entry 行を N 行、1 行 1 エントリ）
補足	`count` 行の entry が 1 行 1 ファイルで返る。`name` は空白を含まない ASCII。ホストは `count` と受信した entry 行数の一致で完了を判定できる
エラー	path 不足 / 不在 / IO INVALID_PARAM / NOT_FOUND / IO_ERROR

FILE >FILE_DOWNLOAD <path>	File
目的	ファイルをバイナリ転送。`<OK FILE_DOWNLOAD size=<bytes> crc32=0x<8hex>` の後に size バイトの生バイナリが続く
補足	`crc32`=IEEE 802.3 (poly 0xEDB88320)。バイナリにはエスケープ・区切りが無く、`size=` が正となります。
エラー	配信中 / 録音中ファイル BUSY 不在 / IO / メモリ NOT_FOUND / IO_ERROR / NO_MEM

6.7 AI 学習・推論 (ML) Solist-AI 連携

Solist-AI 上のニューラルネットに対する逐次学習・推論の中核コマンド群です。1 台あたり複数のネットワーク（`<inst>` = インスタンス番号）を保持でき、各コマンドで対象を指定します。共通エラー: `NOT_READY`（未初期化）/ `DEVICE_ERROR`（SPI 障害）/ `SIZE_EXCEEDED`（配列が転送上限超過）。

INIT >ML_INIT <inst> <in> <hidden> <out> <act> <loss> <forget> <seed> <scale_a> <scale_g> <leak>

ML

目的	インスタンス <inst> のオンライン学習ネットワークを構築する。以降の学習・推論の前提となる。全 11 引数をスペース区切りで指定する
引数	<inst> インスタンス番号（0 / 1。同時に扱える数は構成依存 → 備考） <in> 入力次元数（inputSize） <hidden> 隠れ層ユニット数（hiddenSize） <out> 出力次元数（outputSize） <act> 活性化関数（activationFunction）: 0=Linear / 1=Sigmoid / 2=ReLU <loss> 損失関数（lossFunction）: 0=MAE / 1=MSE <forget> 学習率 / 忘却係数（forgettingFactor, 実数 → bfloat16。典型値 0.01） <seed> 重み初期化の乱数シード（seed, 整数） <scale_a> 重み α スケール（scaleAlpha, 実数 → bfloat16） <scale_g> 重み γ スケール（scaleGamma, 実数 → bfloat16。ESN モード用） <leak> リークレート（leakRate, 実数 → bfloat16。ESN モード用）
補足	scale_g / leak は Echo State Network 系モデル設定用。通常の前向きネットワークとして使う場合は意味を持たず、既定値のままでよい。実数引数（forget / scale_a / scale_g / leak）はファーム側で bfloat16 に変換して保持される
備考	in / hidden の上限は構成に依存する（構成 A: インスタンス 0 のみ・in ≤ 512・hidden ≤ 64 / 構成 B: インスタンス 0,1・in ≤ 256・hidden ≤ 64）。初期化に成功すると以降ファームが in / hidden / out を保持し、ML_GET_RESULT ・サイズ取得の応答長はこれらから自動計算される
応答	<OK ML_INIT <inst>
エラー	引数不足 / 範囲外（in / hidden 上限超過・0, out=0 等） INVALID_PARAM Solist-AI 通信失敗 DEVICE_ERROR

RESET >ML_RESET <inst> / >ML_CLEAR_RAM

ML

目的	`ML_RESET` = 指定インスタンスの重み・状態を初期化（構成は保持）。 `ML_CLEAR_RAM` = 全インスタンスを破棄し RAM を解放
応答	`<OK ML_RESET <inst>` / `<OK ML_CLEAR_RAM`
エラー	未初期化 NOT_READY 不正な inst INVALID_PARAM

TRAIN >ML_TRAIN <inst> input=[...] target=[...]

ML

目的	1 サンプルで逐次学習（1 ステップ）。`input` は入力ベクトル、`target` は教師ベクトルで、それぞれ `ML_INIT` の入力／出力ユニット数と一致させる。演算は非同期で、完了は `ML_GET_BUSY` が `0` を返すことで確認する
応答	`<OK ML_TRAIN <inst>`
エラー	未初期化 NOT_READY 長さ不一致 / 形式不正 INVALID_PARAM 配列が上限超過 SIZE_EXCEEDED 演算中 BUSY

PREDICT >ML_PREDICT <inst> input=[...]

ML

目的	入力ベクトルから推論を実行。演算は非同期で、完了は `ML_GET_BUSY` で確認し、結果は `ML_GET_RESULT` で取得する
応答	`<OK ML_PREDICT <inst>`
エラー	未初期化 NOT_READY 長さ不一致 INVALID_PARAM 演算中 BUSY

STATUS >ML_GET_BUSY

ML

目的	Solist-AI が学習／推論を実行中かを返す（`1` = 演算中 / `0` = 完了）。`ML_TRAIN` / `ML_PREDICT` 後のポーリングに用いる
応答	<OK ML_GET_BUSY <flag>

RESULT >ML_GET_RESULT <inst>

ML

目的	直近の `ML_PREDICT` の出力ベクトルを取得。1 行目にサマリ、続けて出力データ行が届く
応答	<OK ML_GET_RESULT <inst> size=<n> <データ行>
エラー	結果未確定 / 演算中 — INVALID_STATE / BUSY 未初期化 — NOT_READY

LOSS >ML_GET_LOSS <inst>

ML

目的	直近の学習ステップの損失値を取得。学習の収束確認に用いる
応答	<OK ML_GET_LOSS <inst> <loss>
エラー	未初期化 / 未学習 — NOT_READY / INVALID_STATE

WEIGHT >ML_SET_WEIGHT_BETA <inst> input=[...] / >ML_GET_WEIGHT_BETA <inst>

ML

目的	重み係数 β の設定／取得。学習済みモデルの保存・復元（ホスト側でのシリアライズ）に用いる。取得時は 1 行目のヘッダに続けてデータ行が届く
応答	<OK ML_SET_WEIGHT_BETA <inst> <OK ML_GET_WEIGHT_BETA <inst> size=<n> + データ行
エラー	未初期化 — NOT_READY 長さ不一致 — INVALID_PARAM 配列が上限超過 — SIZE_EXCEEDED

WEIGHT	>ML_SET_WEIGHT_P <inst> input=[...] / >ML_GET_WEIGHT_P <inst>	ML
目的	共分散行列 P の設定／取得。β と対で保存・復元することで逐次学習の内部状態をまるごと引き継げる。取得時は 1 行目のヘッダに続けてデータ行が届く	
応答	<OK ML_SET_WEIGHT_P <inst> <OK ML_GET_WEIGHT_P <inst> size=<n> + データ行	
補足	P の総バイト数は `hidden × hidden × 2`。`ML_GET_WEIGHT_P_SIZE` で事前に取得できる	
エラー	未初期化 NOT_READY 長さ不一致 INVALID_PARAM 配列が上限超過 SIZE_EXCEEDED	

WEIGHT	>ML_GET_WEIGHT_BETA_SIZE <inst> / >ML_GET_WEIGHT_P_SIZE <inst>	ML
目的	β / P 配列の総バイト数を取得。取得コマンドの前に受信バッファを確保する用途に使う（β = `hidden × out × 2` / P = `hidden × hidden × 2`）	
応答	<OK ML_GET_WEIGHT_BETA_SIZE <inst> <size> <OK ML_GET_WEIGHT_P_SIZE <inst> <size>	
エラー	未初期化 NOT_READY	

6.8 信号処理 (FFT) Solist-AI 連携

入出力とも int16 の実数系 FFT。点数（サイズ）は 1〜1024、窓関数は 矩形（なし）または Hann。`FFT_INIT` でサイズ・窓を設定し、`FFT_START` → 完了待ち → `FFT_GET_RESULT` の順で使う（同じ設定で繰り返す場合は再初期化不要）。演算アクセラレータは AI と共有のため、AI 演算と FFT 演算は同時に実行できない。

INIT >FFT_INIT <size> <window>

FFT

目的	FFT の点数と窓関数を設定する（1 回のみ。同一設定で繰り返す場合は再初期化不要）
引数	<size> FFT 点数（1〜1024） <window> 窓関数: 0=なし（矩形） / 1=Hann
応答	<OK FFT_INIT
エラー	size=0 / >1024, window>1 INVALID_PARAM

START >FFT_START input=[...]

FFT

目的	int16 PCM を投入して FFT 演算を開始する。`input` の要素数は `FFT_INIT` の `size` と一致させる。演算は非同期で、完了は `FFT_GET_BUSY` で確認する
応答	<OK FFT_START
エラー	FFT_INIT 未実行 INVALID_STATE 長さ不一致 / int16 範囲外 INVALID_PARAM AI/FFT 演算中 BUSY

STATUS >FFT_GET_BUSY

FFT

目的	FFT 演算が実行中かを返す（1=演算中 / 0=完了）。AI と FFT は同一アクセラレータを共有するため、この busy 状態は `ML_GET_BUSY` と同一の値を返す
応答	<OK FFT_GET_BUSY <flag>

RESULT >FFT_GET_RESULT		FFT
目的	直近の FFT 結果（int16 配列）を取得する。1 行目にサイズ、続けてデータ行が届く	
応答	<OK FFT_GET_RESULT size=<n> <データ行>	
エラー	FFT_INIT 未実行 / 結果未確定 演算中	INVALID_STATE BUSY

6.9 フラッシュ操作 (FLASH) Solist-AI 連携

Solist-AI 内蔵の不揮発メモリを扱う。用途は学習済み重みの保存・アプリ設定の保存・データロギング。すべての消去・書込みは `FLASH_SESSION_BEGIN` ～ `FLASH_SESSION_END` の セッション内でのみ実行でき、セッション外で発行すると `INVALID_STATE` を返す。書込みに必要なアクセスコードはファームが内部保持するため、ホストが意識する必要はない。

フラッシュ構成

領域	全体サイズ	書込単位	Sector 消去	Block 消去
Program flash	256 KB	ワード (4 B)	2 KB	32 KB
Data flash	8 KB	バイト (1 B)	256 B	8 KB

書込み前に対象領域の消去が必要。書込アドレスは書込単位（ワード / バイト）に整列させること。

SESSION	>FLASH_SESSION_BEGIN <target> / >FLASH_SESSION_END	Flash
目的	一連の Flash 操作をまとめるセッションを開始／終了する。`<target>` で対象領域（Program / Data flash）を選択する。セッション中はアクセスコードがファーム内部で保持される	
応答	<OK FLASH_SESSION_BEGIN / <OK FLASH_SESSION_END	
エラー	既に開始済 / 未開始で END 不正な target	INVALID_STATE INVALID_PARAM

FLASH	>FLASH_INIT / >FLASH_OPEN / >FLASH_CLOSE	Flash
目的	Flash ペリフェラルの初期化・オープン・クローズ。通常は `FLASH_SESSION_BEGIN` / `_END` が代行するため、個別呼び出しは不要	
応答	<OK FLASH_INIT / <OK FLASH_OPEN status=<hw> / <OK FLASH_CLOSE status=<hw>	
エラー	ハードウェア障害	DEVICE_ERROR

ERASE	>FLASH_ERASE_SECTOR_PROG <addr> / >FLASH_ERASE_SECTOR_DATA <addr>	Flash
目的	Sector 単位の消去（Program flash = 2 KB / Data flash = 256 B）。`<addr>` はファーム側で sector 境界に丸められる。消去完了は `FLASH_GET_STATUS` で確認する	
HW 所要	Program ~50 ms / Data ~6 ms	
エラー	セッション未開始 ハードウェア busy 消去失敗	INVALID_STATE BUSY DEVICE_ERROR

ERASE `>FLASH_ERASE_BLOCK_PROG <addr> / >FLASH_ERASE_BLOCK_DATA <addr>` — Flash

目的	Block 単位の消去（Program flash = 32 KB / Data flash = 8 KB）。`<addr>` は block 境界に丸められる
HW 所要	Program 32 KB は **最大 ~3 秒** / Data 8 KB ~400 ms
エラー	セッション未開始 `INVALID_STATE` ハードウェア busy `BUSY` 消去失敗 `DEVICE_ERROR`

> ⚠ **32 KB ブロック消去とウォッチドッグ**
> Program flash の 32 KB ブロック消去は最大 ~3 秒かかり、Solist-AI 内部のウォッチドッグ（2 秒）を超える。ホストは消去中、`FLASH_GET_STATUS` を **500 ms 以下の間隔で発行し続ける**こと。ポーリングを怠るとデバイスがリセットされる恐れがある。

WRITE `>FLASH_WRITE_DATA_BYTE <addr> input=[...] / >FLASH_WRITE_PROG_WORD <addr> input=[...]` — Flash

目的	消去済み領域への書込み。Data flash はバイト単位、Program flash はワード（4 B）単位。`<addr>` は書込単位に整列させる。書込み前に対象領域を消去しておくこと
応答	`<OK FLASH_WRITE_DATA_BYTE` / `<OK FLASH_WRITE_PROG_WORD`
エラー	セッション未開始 `INVALID_STATE` アドレス不整列 / 範囲外 `INVALID_PARAM` 書込み失敗 `DEVICE_ERROR`

STATUS **>FLASH_GET_STATUS / >FLASH_CONTROL_SELFPROG <on|off>** Flash

目的	`FLASH_GET_STATUS` = ハードウェアステータスを取得（消去・書込みの完了確認。セッション外でも可）。`FLASH_CONTROL_SELFPROG` = セルフプログラム機能の有効／無効を制御する
応答	`<OK FLASH_GET_STATUS status=0x<NN>`
補足	`status` はビットフラグ: `0x00` =アクセス可 / `0x01` =Data 消去中 / `0x02` =Data 書込中 / `0x04` =Program 消去中 / `0x08` =Program 書込中

7

CHAPTER 07 · CONFIGURATION

設定キー リファレンス

`>SET <key> <value>` で変更、`>GET <key>` で取得します。値は NVS に保存され電源断後も保持。ストリーミング中の `SET` は対応キーであればリアルタイムに反映されます。

7 設定キー リファレンス

範囲外・形式不正の値は `<ERR SET INVALID_PARAM ...>`、未知のキーは `<ERR SET INVALID_KEY <key>>` を返します。`>GET *` で全キーをまとめて取得できます。

7.1 IMU（BMI088）

キー	型	既定	取り得る値 / 説明
imu.rate	int	200	25/50/100/200/400 Hz（BMI088 ODR にマップ）
imu.range_g	int	6	3/6/12/24（加速度レンジ ±g）
imu.range_dps	int	500	125/250/500/1000/2000（角速度レンジ ±dps）

7.2 ToF（VL53L5CX）

キー	型	既定	取り得る値 / 説明
tof.resolution	int	8	4 / 8（グリッド解像度 4×4 / 8×8）
tof.freq	int	15	1–60 Hz（8×8 は実質 15 まで）
tof.sharpener	int	5	0–99 %（エッジ先鋭化）
tof.target_order	str	closest	closest / strongest
tof.integration_ms	int	5	1–1000（autonomous モード時のみ有効。continuous モードでは無視）
tof.ranging_mode	str	continuous	continuous / autonomous

7.3 温湿度（SHT40）

キー	型	既定	取り得る値 / 説明
th.rate	int	1	1–10 Hz（取得レート）

キー	型	既定	取り得る値 / 説明
th.repeatability	str	high	low / medium / high（high ≈ 8.2 ms / low ≈ 1.6 ms）

7.4 マイク（SPH0641LU4H-1）

キー	型	既定	取り得る値 / 説明
mic.clock_hz	int	1024000	1024000–4800000。PCM レート = clock_hz / 64（1024000→16 kHz 音声 / 4800000→75 kHz 超音波）。既定は voice（16 kHz）
mic.max_record_seconds	int	60	1–3600（録音の自動停止までの最大秒数）
mic.storage	str	internal	`internal` = オンチップ flash（`/int`、SD 不要・~125 s@voice）、`sd` = MicroSD（`/sdcard`、長時間用）。残り録音可能時間は `>STORAGE_INFO` で取得

> ⚠ **内蔵 FLASH + ULTRASONIC (4.8 MHZ) は非推奨**
> 150 KB/s の連続書き込みに内蔵 flash が追いつかずサンプルを取りこぼし、再生が速く・音質が劣化します。ultrasonic は `mic.storage=sd` を使ってください。

7.5 ストリーム共通 / 反映タイミング

キー / キー群	型	説明 / 反映
tof.* / imu.* / th.*	—	ストリーミング中 SET 可。値変更時に再構成・次フレームから反映
mic.*	—	次回録音開始時にスナップショット

すべてのキーは NVS に保存され、再起動後も保持されます。

8

APPENDIX

付録

出荷時設定、トラブルシューティング、用語集、仕様サマリ、参照リンク、別資料と注意事項をまとめます。

8 付録

8.1 出荷時設定 (デフォルト値)

項目	値
ファームウェア版数	`0.1.0`（プロトコル `1.0`）
ボーレート	`921600 bps（8N1）`
センサー識別	起動時に自動認識
設定値	§7 の「既定」列のとおり

8.2 トラブルシューティング

症状	原因の可能性	対処
文字化けする	ボーレート不一致	921600 / 8N1 / フロー制御なしを確認
応答が返らない	ポート / 改行 / プレフィックス	正しいポート選択、`>` 始まり・LF 終端を確認
`STATUS` が `devices=0`	センサー未認識	装着確認。差し替え後は 再起動か `SENSOR_RESCAN`
差し替えが反映されない	再検出未実行	本体を再起動するか `SENSOR_RESCAN` を実行
`START_STREAM` → `NOT_SUPPORTED`	MIC 装着中	`MIC_REC_START` を使用
`BUSY` が返る	ストリーミング / 録音中	停止後に再実行
録音できない / `IO_ERROR`	MicroSD 未挿入 / 不良	カード挿入・フォーマット確認
`FILE_DOWNLOAD` が `BUSY`	配信中 or 録音中ファイル	停止後、または別ファイルで実行

症状	原因の可能性	対処
ML 系が `NOT_READY`	初期化前	`ML_INIT` を先に実行
ML 系が `DEVICE_ERROR`	Solist-AI との SPI 障害	再起動。別資料 (Solist-AI 側) を確認

8.3 用語集・仕様サマリ・参照リンク

gene
p-ban.com 提供の PoC 向け ESP32-S3 ベースハードウェア

gene core
本製品。p-ban.com の PoC 向けハードウェア + 4 センサーのセット（MicroSD は非同梱・お客様手配）

Solist-AI `ML63Q2537`
ROHM の AI アクセラレータ搭載 MCU (SPI スレーブ)

ToF `Time-of-Flight`
光の往復時間で距離を測る方式 (VL53L5CX)

IMU
慣性計測装置。加速度 + 角速度 (BMI088)

PDM
Pulse Density Modulation。デジタルマイクの出力方式

bfloat16
16bit 浮動小数フォーマット (float32 の上位 16bit)

NVS
ESP32 の不揮発設定ストレージ

ODL `On-Device Learning`
オンデバイス逐次学習

仕様サマリ

項目	値
メイン MCU	ESP32-S3（Xtensa デュアルコア, 最大 240 MHz）
AI コプロセッサ	ROHM ML63Q2537（Solist-AI, SPI スレーブ）
PC 接続	USB Type-C / UART 921600 bps（テキスト）
センサー I/F	I2C 1 MHz（IMU/ToF/TH） / PDM（MIC）
ストレージ / サイズ	内蔵 flash + MicroSD（SDSPI） / 40 × 40 × 19 mm

項目	値
対象ホスト OS	Windows / Linux

参照リンク・別資料
gene (p-ban.com) 製品仕様 / ROHM Solist-AI ブランド・ML63Q2537 製品ページ、および主要センサー（Bosch BMI088 / ST VL53L5CX / Knowles SPH0641LU4H-1 / Sensirion SHT40）のメーカー公開データシートを参照。Solist-AI (ML63Q2537) SPI スレーブ側の詳細仕様は別資料で提供します。

免責事項 / Disclaimer

本書に記載された情報は、発行時点のものであり、予告なく変更されることがあります。本書の内容については正確を期していますが、**当社は本書の記載内容に関して明示・黙示を問わずいかなる保証も行わず**、本書の使用または使用不能から生じるいかなる損害（逸失利益、事業の中断、データの損失等を含む）についても責任を負いません。本製品は評価・開発（PoC）用途を目的としており、医療機器・航空宇宙・原子力など、人命に関わる用途や高い信頼性が要求される用途への使用を想定していません。

本製品の仕様、外観、付属品は改良のため予告なく変更される場合があります。お客様が本製品を組み込んだ最終製品の適合性・安全性については、お客様ご自身の責任において十分に評価・検証してください。本製品のファームウェアは 非公開バイナリ です。ソースコードの提供・改変は行いません。記載のピン配置・仕様は 公開可能な情報 の範囲です。

注記・特記事項 / Notes & Special Remarks

※1 本書中の数値（アドレス・サイズ・タイミング・コマンド書式等）は本インタフェース仕様を規定するものであり、ホスト側（PC アプリケーション）はこれらの記載に基づいて実装してください。本書に記載のない挙動には依存しないでください。

※2 Solist-AI™ の推論精度は学習データ・動作環境に依存し、特定の性能を保証するものではありません。

※3 ファームウェア更新により、将来コマンドの追加や機能拡張が行われる場合があります。本書はファームウェア v0.1.0 時点の内容です。

※4 輸出に際しては、関連する輸出関連法令を遵守してください。該当する場合は事前に許可が必要です。

著作権表示 / Copyright

本書の著作権は当社に帰属します。当社の事前の書面による許可なく、本書の全部または一部を複製・転載・改変・翻訳・再配布することを禁じます。