ND03 SDK User Guide

- ND03 SDK User Guide
  - 1. 介绍 Introduction
  - 2. 平台接口实现 User Platform API Implementation
  - 3. I2C通讯 I2C Communication
  - 4. ND03 初始化与测距流程 ND03 Initialization And Ranging
  - 5. 标定 Calibration
    - 5.1. Offset标定 Offset Calibration
    - 5.2. 串扰标定 Crosstalk Calibration
  - 6. 低功耗 Lower Power
    - 6.1. 低功耗模式的进入与唤醒 Sleep and Wake up
    - 6.2. 深度休眠模式的进入与唤醒 Deep Sleep and Wake up

1. 介绍 Introduction

本文档对ND03 SDK的使用流程做基本介绍，如需更详细的信息可查看源码注释或者联系支持人员

- SDK 代码目录结构

```
|--ND03_SDK
    |-- inc 头文件
     |-- nd03_calib.h
     |-- nd03_comm.h
     |-- nd03_data.h
     |-- nd03_def.h
     |-- nd03_dev.h
    |-- src 源码
     |-- nd03_calib.c
     |-- nd03_comm.c
     |-- nd03_data.c
     |-- nd03_dev.c
    |-- example
        |-- ND03_Ranging_Example.c
        |-- nd03_platform.c
        |-- nd03_platform.h
```

2. 平台接口实现 User Platform API Implementation

用户需要实现 `example/nd03_platform.c` 中的接口函数，以适配目标平台，详情见 `example/nd03_platform.h` 。

```
int32_t i2c_read_nbytes(uint8_t i2c_addr, uint16_t i2c_read_addr, uint8_t *i2c_read_data,
uint8_t len);
int32_t i2c_write_nbytes(uint8_t i2c_addr, uint16_t i2c_write_addr, uint8_t
*i2c_write_data, uint8_t len);
void delay_1ms(uint32_t count);
void delay_10us(uint32_t count);
void set_xshut_pin_level(uint32_t level);
```

3. I2C通讯 I2C Communication

- **默认设备地址**:

```
0x5B
```

- **修改I2C地址方式**:

  ND03支持修改i2c设备地址，但由于模组没有对修改后的地址进行保存，重新上电则自动恢复默认i2c地址0x5B.
  Note: ND03设备的i2c地址后，本地同时要更新新的地址.

```
int32_t ND03_SetSlaveAddr(ND03_Dev_t *pNxDevice, uint8_t addr)
```

4. ND03 初始化与测距流程 ND03 Initialization And Ranging

完整测距流程可参考`example/ND03_Ranging_Example.c`

- **平台接口配置 Platform API Registering**

```
typedef struct{
int32_t(*I2C_WriteNBytesFunc)(uint8_t, uint16_t, uint8_t *, uint8_t); /*!< i2c写函数 */
int32_t(*I2C_ReadNBytesFunc)(uint8_t, uint16_t, uint8_t *, uint8_t); /*!< i2c读函数 */
void(*Delay1msFunc)(uint32_t );   /*!< 延时1ms函数 */
void(*Delay10usFunc)(uint32_t );  ///< 延时10us函数
}ND03_Func_Ptr_t;

/** 模组设备 */
ND03_Dev_t g_nd03_device = {.i2c_dev_addr = ND03_DEFAULT_SLAVE_ADDR,
.SetXShutPinLevelFunc = set_xshut_pin_level};

/* 函数指针结构体 */
ND03_Func_Ptr_t dev_op = {NULL, NULL, NULL, NULL};

/* 初始化函数指针结构体 */
dev_op.Delay10usFunc = delay_10us;
dev_op.Delay1msFunc = delay_1ms;
dev_op.I2C_ReadNBytesFunc = i2c_read_nbytes;
dev_op.I2C_WriteNBytesFunc = i2c_write_nbytes;

/* 将host端功能函数注册到SDK中 */
ND03_RegisteredPlatformApi(dev_op);
```

- **启动 Boot Up**

  初始化ND03之前，首先需要调用`ND03_WaitDeviceBootUp()`函数启动模组;

```
int32_t ND03_WaitDeviceBootUp(ND03_Dev_t *pNxDevice)
```

- **初始化 Initialization**

```
int32_t ND03_InitDevice(ND03_Dev_t *pNxDevice)
```

- **测距 Ranging**

  使用`ND03_StartMeasurement`发起一次测距

```
int32_t ND03_StartMeasurement(ND03_Dev_t *pNxDevice)
```

  使用`ND03_GetRangingData`获取测距数据

```
int32_t ND03_GetRangingData(ND03_Dev_t *pNxDevice, ND03_RangingData_t *pData)
```

5. 标定 Calibration

通常，ND03需要进行两次标定，详见**《ND03标定手册》**

5.1. Offset标定 Offset Calibration

模组焊接到PCB后，需要进行一次offset标定，详见**《ND03标定手册》**

- **默认标定距离 Calibration at default position**

  在默认offset标定环境下(模组放在距90%反射板150 mm处)，调用`ND03_OffsetCalibration`接口，标定完成后标定数据会保存在模组内部flash中，开机自动读取flash中的标定数据。

  ```
  int32_t ND03_OffsetCalibration(ND03_Dev_t *pNxDevice)
  ```

- **指定标定距离 Calibration at custom position**

  如果用户有在其他距离处进行offset标定的需求，则需要使用`ND03_OffsetCalibrationAtDepth`接口，参数`calib_depth_mm`用来指定模组到目标处距离。

  ```
  int32_t ND03_OffsetCalibrationAtDepth(ND03_Dev_t *pNxDevice, uint16_t calib_depth_mm)
  ```

5.2. 串扰标定 Crosstalk Calibration

如需在ND03模组上方增加盖板，则需要进行串扰标定，详见**《ND03标定手册》**

- **默认串扰标定距离 Crosstalk calibration at default distance**

  在默认串扰标定的环境下(标定Chart 距离模组 800 mm)，调用`ND03_XTalkCalibration`接口，标定完成后标定数据会保存在模组内部flash中，开机自动读取flash中的标定数据。

  ```
  int32_t ND03_XTalkCalibration(ND03_Dev_t *pNxDevice)
  ```

- **串扰标定距离 Crosstalk calibration at custom position**

  如果用户有在其他距离处进行串扰标定的需求，则需要使用`ND03_XTalkCalibrationAtDepth`接口，参数`xtalk_calib_depth_mm`用来指定模组到目标处距离。

  ```
  int32_t ND03_XTalkCalibrationAtDepth(ND03_Dev_t *pNxDevice, uint16_t
  xtalk_calib_depth_mm)
  ```

- **串扰值获取 Crosstalk Amplitude**

  调用`ND03_GetXTalkAmp`函数接口，可获取串扰值，如果该值比较大或者返回错误，请联系支持人员协助。

  ```
  int32_t ND03_GetXTalkAmp(ND03_Dev_t *pNxDevice, uint32_t *xtalk_amp)
  ```

6. 低功耗 Lower Power

6.1. 低功耗模式的进入与唤醒 Sleep and Wake up

- **进入低功耗 Sleep**

  调用`ND03_Sleep`函数，ND03模组即可进入低功耗模式。

  ```
  int32_t ND03_Sleep(ND03_Dev_t *pNxDevice)
  ```

- **低功耗唤醒 Wake up**

  调用`ND03_Wakeup`函数，ND03模组即可从低功耗模式中唤醒。

```
int32_t ND03_Wakeup(ND03_Dev_t *pNxDevice)
```

6.2. 深度休眠模式的进入与唤醒 Deep Sleep and Wake up

- **进入深度休眠 Deep Sleep**

  调用`ND03_DeepPowerSleep`函数，ND03模组即可进入深度休眠模式。

```
int32_t ND03_DeepPowerSleep(ND03_Dev_t *pNxDevice)
```

- **从深度休眠唤醒 Wake up**

  调用`ND03_DeepPowerWakeup`函数，ND03模组即可通过从深度休眠模式中唤醒。

```
int32_t ND03_DeepPowerWakeup(ND03_Dev_t *pNxDevice)
```