文档编号：

版本：V1.0

生效日期：2018-01-17

深圳市通力科技开发有限公司

TCL通力电子（惠州)有限公司

西安TCL软件开发有限公司



星诺齐SX1系统软件详细设计说明书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档作者：邓正午、邱伟盛**  **项目经理：李强** | |  | **日期：2020-08-15** |  |
| **审核：** |  | | **日期：** |  |
| **批准：** |  | | **日期：** |  |
|  |  | | **日期：** |  |

**通力电子股份有限公司 版权所有**

**内部资料 注意保密**

文档修订记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **版本号** | **变化状态** | **变更（+/-）说明** | **作者** | **日期** |
| 1 | V0.1 | C |  | 邱伟盛、邓正午 | 2020-08-15 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

\*变化状态：C――创建，A——增加，M——修改，D——删除

目录

[星诺齐SX1系统软件详细设计说明书 1](#_Toc55944470)

[文档修订记录 2](#_Toc55944471)

[1、 简介 3](#_Toc55944472)

[1.1 文档目的 3](#_Toc55944473)

[1.2 项目背景 4](#_Toc55944474)

[1.3 设备功能 4](#_Toc55944475)

[1.4 适用范围 4](#_Toc55944476)

[1.5 模块名称 5](#_Toc55944477)

[1.6 术语和缩略语 5](#_Toc55944478)

[1.7 参考资料 5](#_Toc55944479)

[2、 系统总体设计 6](#_Toc55944480)

[2.1 系统架构图 6](#_Toc55944481)

[2.2 软件架构图 6](#_Toc55944482)

[2.3 MCU架构图 7](#_Toc55944483)

[2.4 系统流程图 8](#_Toc55944484)

[3、 模块设计 9](#_Toc55944485)

[3.1 模块功能 9](#_Toc55944486)

[4、 接口说明 11](#_Toc55944487)

[4.1 按键模块接口 11](#_Toc55944488)

[4.2 充电模块接口 13](#_Toc55944489)

[4.3 ATS模块接口 15](#_Toc55944490)

[5、 数据结构 18](#_Toc55944491)

[5.1 用户基本信息数据结构 18](#_Toc55944492)

[6、 系统出错处理 19](#_Toc55944493)

# 简介

## 文档目的

本文档主要说明星诺齐SX1自动系鞋带项目的软件设计思路，以软件系统架构对项目进行陈述，以便于了解项目的设计框架及具体的实现机理。项目软件编写人员，项目相关领导应阅读此文件，用以了解项目软件设计架构及思路。

## 项目背景

SX1项目是新诺奇给中国乔丹品牌制造的一款自动系鞋带引擎。该产品主打在满足松紧鞋带基本功能的情况下，做到最低成本。产品安装在运动鞋跟，使用充电电池驱动，可以使用磁吸充电线充电。

使用时通过两个按键来实现。另外驱动装置采用丝杆配合一个微型的行星减速箱来驱动，所以驱动部分尺寸可以做到很小。产品有左右脚之分，男式4款和女式3款，每款还有7种颜色

## 设备功能

1、按收紧键驱动电机收紧鞋带

2、按放松键驱动电机放松鞋带

3、通过双击收紧\放松按键实现自适应收紧，放松鞋带

4、长按放松收紧键进入按键锁模式，即按按键电机不启动

5、通过不同的LED灯效显示当前的程序状态

6、有低电量提醒以及高低温的保护

7、具有充电功能

## 适用范围

本文档适用于项目软件开发人员、业务规范设计人员、产品测试人员及用户代表。

## 模块名称

LED模块：tonly\_led.h

按键模块：tonly\_button.h

电机模块：motor\_control.h

Eeprom模块：main.h

ATS模块：ats\_function.h

充电模块：charging.h

叶片开关模块：leaf\_switch.h

## 术语和缩略语

| **缩略语/术语** | **全称** | **说明** |
| --- | --- | --- |
| 叶片开关 | 用于鞋带圈数计数的弹片开关 |  |
| 拨片 | 电机转轴上的两个拨片，拨片会接触叶片开关计数 |  |
|  |  |  |

## 参考资料

STM8S003官方SDK

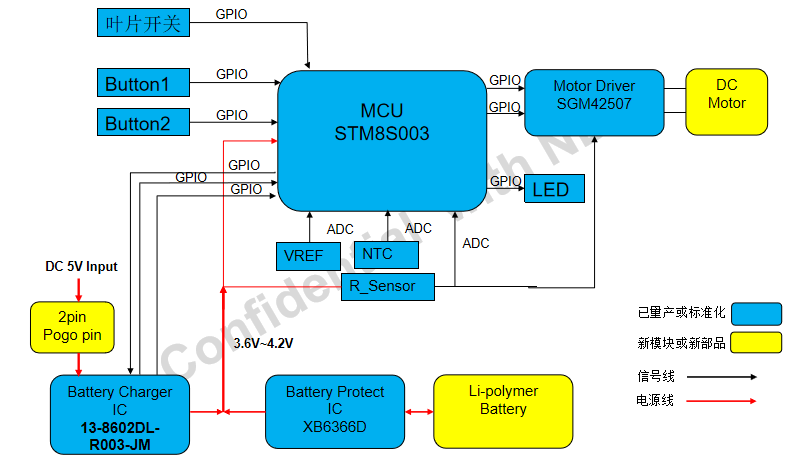
SGM42507 REV.A\_20181128.pdf//电机规格书

STM8S\_datasheet.pdf //MCU规格书

# 系统总体设计

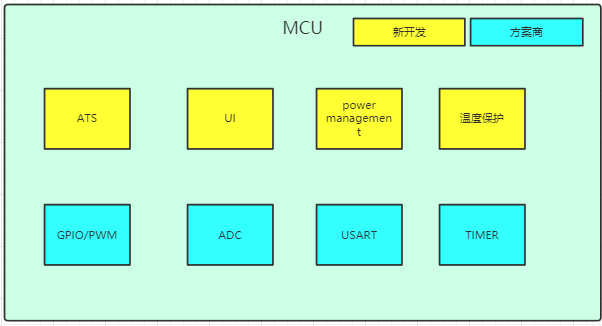
## 系统架构图

本项目由stm8s003f3 为主芯片，通过button或者charger pin input 给MCU信号，由MCU分别对motor、led、charge enable 进行控制，通过叶片开关计数记录鞋带当前旋转收紧或放松长度。

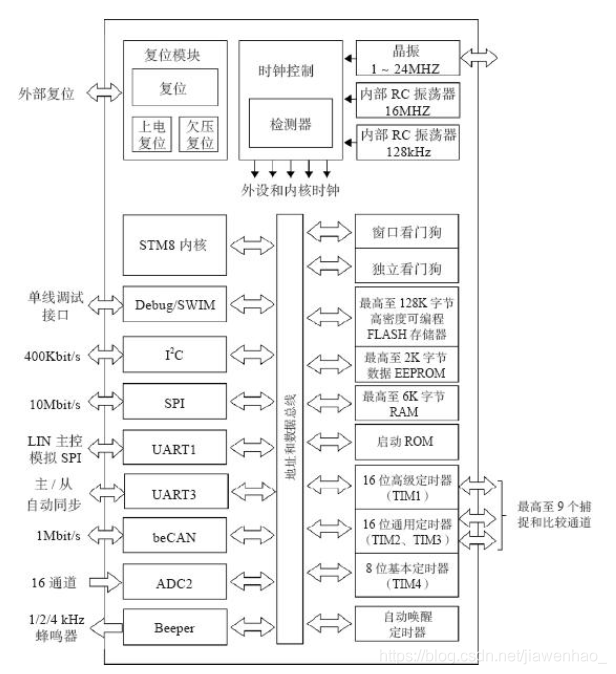


## 软件架构图

软件架构就分为平台层以及应用层，所有平台层的封装均是由方案商ST公司进行SDK的封装库。Tonly负责应用层的软件编写.

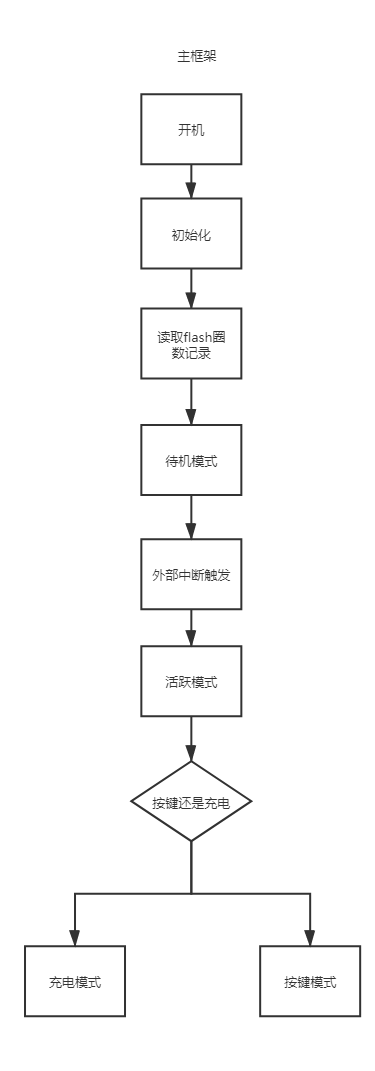


## MCU架构图



## 系统流程图

本系统通过外部消息输入进行控制的，首先通过外部中断的触发将MCU从睡眠模式唤醒，MCU唤醒之后通过软件逻辑层选择进入对应逻辑，通过距离上次外部中断的时间长短来判断系统是否需要重新进入睡眠模式，系统主框架如下：

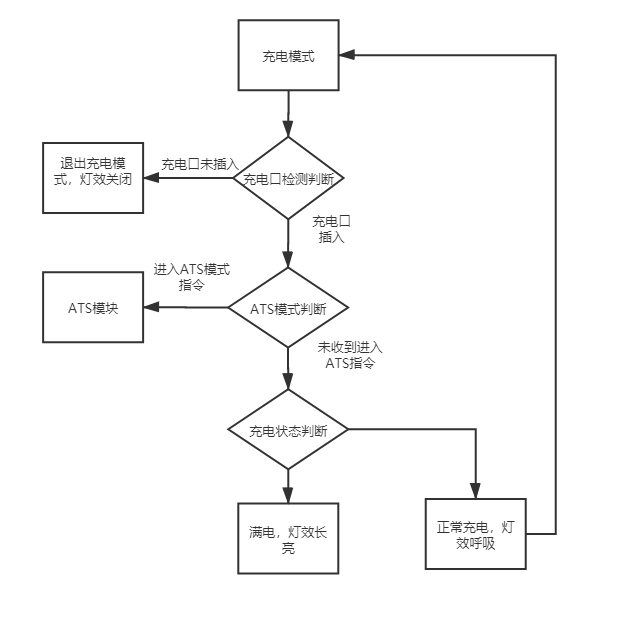


# 模块设计

## 模块功能

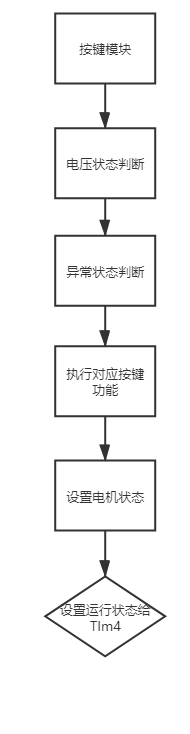
充电模块：

获得充电口插入的中断信息，唤醒mcu，通过读取IO口电平判断充电口是否插入，再根据充电state口的电平信号显示对应灯效，执行过程中同时判断是否有进入ATS的指令。



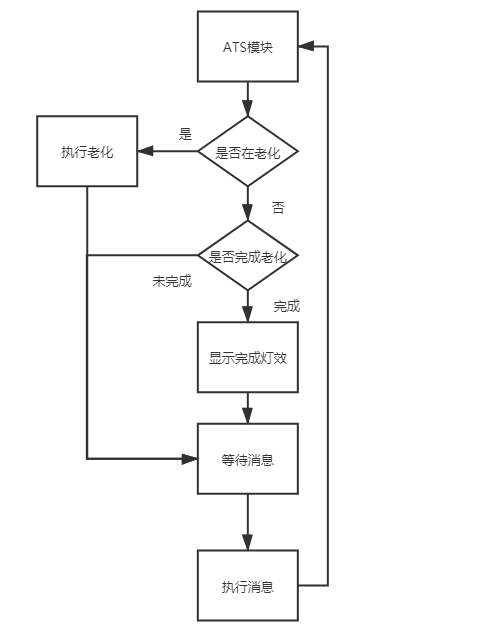
按键模块：

按键模块采用GPIO中断触发，选取上下边沿电平触发，按键按下后首先经过按键消抖，然后处理按键功能，同时将按键状态上报到TIM4中断，由TIM4内对按键状态进行判断，然后启动或者停止电机。



ATS模块：

ATS模块用于PC端与MCU端的通讯，采用UART通讯，MCU端依据通讯协议返回对应信息。



# 接口说明

接口说明分按键模块，LED模块，Trical模块来写。

## 按键模块接口

button\_callback 函数

|  |  |
| --- | --- |
| 函数原型 | **void button\_callback(void)** |
| 功能描述 | 按键响应 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | forward\_button.button\_state  reverse\_button.button\_state  reverse\_button.press\_count  forward\_button.press\_count |
| 返回值 | 无 |

函数实现描述（自然语言）：

**void button\_callback(void)**

{

按键模式响应；

}

button\_motor\_dalay\_control 函数

|  |  |
| --- | --- |
| 函数原型 | **void button\_motor\_dalay\_control(void)** |
| 功能描述 | 电机延迟响应 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | 无 |
| 返回值 | 无 |

函数实现描述（自然语言）：

**void button\_motor\_dalay\_control(void)**

{

电机延迟响应；

}

motor\_delay\_update函数

|  |  |
| --- | --- |
| 函数原型 | **void motor\_delay\_update()** |
| 功能描述 | 电机状态延迟更新 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | g\_button\_opposite |
| 返回值 | 无 |

函数实现描述（自然语言）：

**void motor\_delay\_update()**

{

电机状态延迟响应检测；

}

## 充电模块接口

enter\_charge\_mode函数

|  |  |
| --- | --- |
| 函数原型 | **void enter\_charge\_mode(void)** |
| 功能描述 | 进入充电模式 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | reverse\_button.button\_state = EM\_BUTTON\_RELEASED;  forward\_button.button\_state = EM\_BUTTON\_RELEASED;  reverse\_button.press\_count = 0;  forward\_button.press\_count = 0;  g\_button\_opposite = 0;  g\_tim\_button\_behavior = 0;  g\_A\_button\_pressed = 0;  g\_D\_button\_pressed = 0;  g\_button\_locked = FALSE;  g\_charge\_or\_ats\_led\_state = 100; //invalid value |
| 返回值 | 无 |

函数实现描述（自然语言）：

Void enter\_charge\_mode(void)

{

进入充电模式

}

exit\_charge\_mode函数

|  |  |
| --- | --- |
| 函数原型 | **void exit\_charge\_mode(void)** |
| 功能描述 | 退出充电模式 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | g\_tim4\_charge\_or\_aging\_count = 0;  g\_device\_mode = EM\_ACTIVE\_MODE; |
| 返回值 | 无 |

函数实现描述（自然语言）：

Void exit\_charge\_mode(void)

{

退出充电模式；

}

charge\_insert\_check函数

|  |  |
| --- | --- |
| 函数原型 | **void charge\_insert\_check(void)** |
| 功能描述 | 充电插口检测 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | 无 |
| 返回值 | 无 |

函数实现描述（自然语言）：

Void charge\_insert\_check(void)

{

充电引脚检测

}

charge\_state\_check函数

|  |  |
| --- | --- |
| 函数原型 | **void charge\_state\_check(void)** |
| 功能描述 | 充电状态检测 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | 无 |
| 返回值 | 无 |

函数实现描述（自然语言）：

Void charge\_state\_check(void)

{

充电状态检测；

}

## ATS模块接口

app\_usart\_irq\_receive\_handle函数

|  |  |
| --- | --- |
| 函数原型 | **void app\_usart\_irq\_receive\_handle(uint8\_t data)** |
| 功能描述 | 串口数据接收 |
| 输入参数 | 串口数据 |
| 输出参数 | 无 |
| 全局变量值的变化 | g\_msg\_fin = 1；  g\_USART\_BUFFER[g\_USART\_IDX++] = data; |
| 返回值 | 无 |

函数实现描述（自然语言）：

void app\_usart\_irq\_receive\_handle(uint8\_t data)

{

串口数据接收；

}

app\_usart\_data\_handle\_loop函数

|  |  |
| --- | --- |
| 函数原型 | **void app\_usart\_data\_handle\_loop(void)** |
| 功能描述 | 串口消息处理 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | g\_msg\_fin = 0；  g\_motor\_circles\_counts = 0；  g\_USART\_IDX = 0； |
| 返回值 | 无 |

函数实现描述（自然语言）：

void app\_usart\_data\_handle\_loop(void)

{

串口消息处理；

}

app\_usart\_transmit\_handle函数

|  |  |
| --- | --- |
| 函数原型 | **void app\_usart\_transmit\_handle(uint8\_t state, uint8\_t\* data, uint8\_t data\_len)** |
| 功能描述 | 串口消息回复 |
| 输入参数 | 状态，数据，数据长度 |
| 输出参数 | 无 |
| 全局变量值的变化 | 无 |
| 返回值 | 无 |

函数实现描述（自然语言）：

void app\_usart\_transmit\_handle(uint8\_t state, uint8\_t\* data, uint8\_t data\_len)

{

串口消息回复；

}

CRC16\_CCITT\_FALSE函数

|  |  |
| --- | --- |
| 函数原型 | **uint16\_t CRC16\_CCITT\_FALSE(uint8\_t\* puchMsg, uint8\_t usDataLen)** |
| 功能描述 | CRC校验 |
| 输入参数 | 数据，数据长度 |
| 输出参数 | U16返回值 |
| 全局变量值的变化 | 无 |
| 返回值 | U16返回值 |

函数实现描述（自然语言）：

uint16\_t CRC16\_CCITT\_FALSE(uint8\_t\* puchMsg, uint8\_t usDataLen)

{

数据校验；

Return 校验值；

}

enter\_ats\_init函数

|  |  |
| --- | --- |
| 函数原型 | **void enter\_ats\_init()** |
| 功能描述 | 进入ATS模式 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | g\_motor\_circles\_counts = 0;  g\_tim4\_charge\_or\_aging\_count = 0;  g\_adc\_high\_temperature\_flag = FALSE;  g\_aging\_res\_num = 0;  g\_action\_num = 2;  g\_tim4\_delay\_ms = 255;  motor\_state = EM\_MOTOR\_STOP;  g\_aging\_state\_res = 2;  g\_device\_mode = EM\_ATS\_MODE; |
| 返回值 | 无 |

函数实现描述（自然语言）：

void enter\_ats\_init()

{

进入ATS

}

ats\_mode\_run函数

|  |  |
| --- | --- |
| 函数原型 | **void ats\_mode\_run()** |
| 功能描述 | ATS模式运行 |
| 输入参数 | 无 |
| 输出参数 | 无 |
| 全局变量值的变化 | 无 |
| 返回值 | 无 |

函数实现描述（自然语言）：

**void ats\_mode\_run()**

{

ATS模式运行；

}

ats\_aging\_run 函数

|  |  |
| --- | --- |
| 函数原型 | **void ats\_aging\_run(uint8\_t order, uint8\_t num )** |
| 功能描述 | 老化模式运行 |
| 输入参数 | 命令，次数 |
| 输出参数 | 无 |
| 全局变量值的变化 | g\_action\_num = 0;  g\_action\_flag = 0;  g\_tim4\_charge\_or\_aging\_count = 0;  g\_tim4\_delay\_ms = 255;  motor\_state = EM\_MOTOR\_STOP;  g\_aging\_state\_res = 0;  g\_device\_mode = EM\_AGING\_MODE; |
| 返回值 | 无 |

函数实现描述（自然语言）：

**void ats\_aging\_run(uint8\_t order, uint8\_t num )**

{

老化模式运行；

}

# 数据结构

## 用户基本信息数据结构

typedef struct

{

button\_state\_e button\_state;

uint8\_t press\_count;

}button\_t;

|  |  |
| --- | --- |
| 变量名 | 含义 |
| button\_state | 按键状态 |
| press\_count | 按键在窗口期按下次数，双击功能使用 |

typedef enum

{

EM\_BUTTON\_MOTOR\_REST,

EM\_BUTTON\_MOTOR\_PRESSED,

EM\_BUTTON\_MOTOR\_REVERSE,

EM\_BUTTON\_MOTOR\_FORWARD,

EM\_BUTTON\_MOTOR\_STOP

}button\_motor\_e;

|  |  |
| --- | --- |
| 变量名 | 含义 |
| EM\_BUTTON\_MOTOR\_REST | 电机复位 |
| EM\_BUTTON\_MOTOR\_PRESSED | 按键被按下 |
| EM\_BUTTON\_MOTOR\_REVERSE | 电机反转 |
| EM\_BUTTON\_MOTOR\_FORWARD | 电机正转 |
| EM\_BUTTON\_MOTOR\_STOP | 电机停止 |

# 系统出错处理

|  |  |
| --- | --- |
| 故障情况 | 应对措施 |
| 程序跑出异常，系统挂死 | 增加看门狗复位 |
| 电机持续转动温度过高 | 增加过热保护 |
| EEPROM第一次上电读取失败 | 增加10msdelay |
| 拨片卡在叶片开关上导致计数错乱 | 电机回退机制 |
| 按键电平窜入正常逻辑 | 增加双按键标志位 |
| 双击按键进自适应反转 | 缓存buffer存入，延缓执行 |