## 设计方案 & 设计进度记录

[设计方案 & 设计进度记录 1](#_Toc144456648)

[一、 时间节点记录 & 方案书写说明 3](#_Toc144456649)

[二、 基本思路梳理 3](#_Toc144456650)

[**1.** **状态分类** 3](#_Toc144456651)

[**1)** **“WAIT”等待状态：** 3](#_Toc144456652)

[**2)** **“INPUT”输入密码状态：** 3](#_Toc144456653)

[**3)** **“INPUT”最后的结果：** 3](#_Toc144456654)

[**4)** **“ALARM”报警状态：** 4](#_Toc144456655)

[**5)** **按时间主动退回等待状态的情况：** 4](#_Toc144456656)

[**2.** **按键“BUTTON”** 4](#_Toc144456657)

[**1)** **专用按键1，“ADMINISTRATOR”：** 4](#_Toc144456658)

[**2)** **专用按键2，“RELIEF”：** 4](#_Toc144456659)

[**3)** **输入完密码后的确定键 + 解锁后回到等待状态的确定键，“OK”：** 4](#_Toc144456660)

[**4)** **退格键，“BACKSPACE”：** 4](#_Toc144456661)

[**3.** **密码的拨码开关组织** 5](#_Toc144456662)

[**1)** **密码格式：** 5](#_Toc144456663)

[**2)** **@Q——√ 拨动相应的开关进入“INPUT”状态”的问题，有以下两种解决方案：** 5](#_Toc144456664)

[**3)** **“SWITCH”设置：** 5](#_Toc144456665)

[**4)** **@Q——√“SWITCH”输入 “CODE”不等于 4 bit 的问题，有以下思考（注意！考虑所有状态，完备性问题！）：** 5](#_Toc144456666)

[**A.** **当少于 4 bit ：** 5](#_Toc144456667)

[**B.** **当多于 4 bit：** 5](#_Toc144456668)

[**A.** **当输入位数int“CODE\_BIT”< 4，那么状态仍将停留在“INPUT”的显示：** 6](#_Toc144456669)

[**a)** **“SWITCH”down** 6](#_Toc144456670)

[**b)** **“OK”** 6](#_Toc144456671)

[**c)** **“BACKSPACE”** 6](#_Toc144456672)

[**d)** **10s不操作** 6](#_Toc144456673)

[**B.** **当输入位数int“CODE\_BIT”== 4，临界情况：** 6](#_Toc144456674)

[**a)** **“SWITCH” down** 6](#_Toc144456675)

[**b)** **“OK”** 6](#_Toc144456676)

[**c)** **“BACKSPACE”** 6](#_Toc144456677)

[**d)** **10s不操作** 6](#_Toc144456678)

[**C.** **当输入位数“CODE\_BIT”> 4，此时显示的位数由于有 4 bits 的限制，之后的输入将不会改变原有 4 bits 密码的显示：** 7](#_Toc144456679)

[**a)** **“SWITCH” down / up** 7](#_Toc144456680)

[**4.** **内部存储数据：** 7](#_Toc144456681)

[**1)** **外部输入的密码“CODE”：** 7](#_Toc144456682)

[**2)** **系统内置的正确密码，“KEY”：** 7](#_Toc144456683)

[**3)** **系统内置的已输入密码位数，“CODE\_BIT”：** 7](#_Toc144456684)

[**5.** **LED显示** 7](#_Toc144456685)

[**1)** **等待状态“WAIT”：** 7](#_Toc144456686)

[**A.** **段码“SEGMENTS”：** 7](#_Toc144456687)

[**B.** **16 × LED：** 7](#_Toc144456688)

[**C.** **2 × 三色LED：** 8](#_Toc144456689)

[**D.** **AUDIO\_OUT：** 8](#_Toc144456690)

[**2)** **输入状态“INPUT”：** 8](#_Toc144456691)

[**A.** **段码“SEGMENTS”：** 8](#_Toc144456692)

[**B.** **16 × LED：** 8](#_Toc144456693)

[**C.** **2 × 三色LED：** 8](#_Toc144456694)

[**D.** **AUDIO\_OUT：** 8](#_Toc144456695)

[**3)** **输入结果状态“ERROR[]”：** 8](#_Toc144456696)

[**A.** **段码“SEGMENTS”：** 8](#_Toc144456697)

[**B.** **16 × LED：** 8](#_Toc144456698)

[**C.** **2 × 三色LED：** 8](#_Toc144456699)

[**D.** **AUDIO\_OUT：** 9](#_Toc144456700)

[**4)** **输入结果状态“ALARM”：** 9](#_Toc144456701)

[**A.** **段码“SEGMENTS”：** 9](#_Toc144456702)

[**B.** **16 × LED：** 9](#_Toc144456703)

[**C.** **2 × 三色LED：** 9](#_Toc144456704)

[**D.** **AUDIO\_OUT：** 9](#_Toc144456705)

1. **时间节点记录 & 方案书写说明**
2. **黑色——2023.8.30讨论结果，方案1.0**
3. **橙色——2023.8.31，开题汇报 & 反馈建议，提供之后方案调整的指向**
4. **蓝色——2023.8.31，对于方案的修订和补充，方案1.1**
5. **基本思路梳理**
6. **状态分类**
7. **“WAIT”等待状态：**
8. **“INPUT”输入密码状态：**
9. **“INPUT”最后的结果：**

**“LOCK”——未解锁；“UNLOCK”——解锁。**

**也是状态，但和其他状态有明显的区别，其他状态作为持续量，而这两个结果状态是作为瞬时量出现。仍然将其作为状态的原因是，在这两种情况下需要产生别的操作，e.g.，LED的显示。**

1. **“ALARM”报警状态：**

**为了避免和“CODE”位数一样出现状态计数问题，分为 3 个 “ALARM” 状态，“ALARM\_1”——错误一次 ，“ALARM\_2”——错误两次，“ALARM\_3”——错误三次，i.e.，最终报警状态（除了LED，可以搭配蜂鸣器）**

**开题汇报 & 反馈建议：“INPUT”最后的结果状态和其他的状态分开，不要单独设置，“UNLOCK”的状态等价于“ERROR[]”、“ALARM”，并且将“ERROR[]”区分两个状态，“ALARM”独立出来作为一个状态，不要将“ERROR[]”和“ALARM”混淆。**

1. **按时间主动退回等待状态的情况：**

**猜想可能涉及到在相应的 always@ 块中做 #10 延时的问题，但具体的块、具体的状态转换流程仍需要确定。**

**开题汇报 & 反馈建议：在 always@ 块中直接写延时 #10，有可能最后生成不可综合的电路，所以应该单独设置针对 posedge clk 的计数器，对于每个操作进行计时。对于计数信号的问题，可认为是内部信号，综合是不存在问题的。**

1. **按键“BUTTON”**
2. **专用按键1，“ADMINISTRATOR”：**

**管理员权限象征，用于改密码**

1. **专用按键2，“RELIEF”：**

**用于触发警报后管理员解除警报。（不使用）**

**若按键个数或者功能有限制，可以考虑只使用一个“ADMINISTRATOR”**

**开题汇报 & 反馈建议：用两个也可以，i.e.，1/2个都可以实现，从合理性 & 可行性上来讲都没有问题，只是硬件使用资源的差别。**

1. **输入完密码后的确定键 + 解锁后回到等待状态的确定键，“OK”：**
2. **退格键，“BACKSPACE”：**

**用于用户输入密码未完成时的退格**

1. **密码的拨码开关组织**
2. **密码格式：**

**\_ \_ \_ \_，四位十进制密码。**

**类比数字键盘，每个“SWITCH”从0~9一一对应，拨动一个“SWITCH”就相当于输入十进制中的一位。**

1. **@Q——√ 拨动相应的开关进入“INPUT”状态”的问题，有以下两种解决方案：**
2. **此“SWITCH”分立实现：**

**“WAIT”到“INPUT”的转换，而不作为“CODE”被存入系统——（不需要分离）**

1. **此“SWITCH”同时实现：**

**“WAIT”到“INPUT”的转换 & 作为“CODE”的第一位被存入**

**开题汇报 & 反馈建议：同时实现的思路，i.e.，一个“SWITCH”直接触发“INPUT”& 计入一位密码。**

1. **“SWITCH”设置：**

**由于 4 bit CODE 可能有重复数字，在 CODE 每一位输入时，必须保证——各个 SWITCH 全都是0的初始化情况。所以采用置1 SWITCH 作为 CODE 一位输入，置0 同一个SWITCH 作为切换到下一位的信号。**

1. **@Q——√“SWITCH”输入 “CODE”不等于 4 bit 的问题，有以下思考（注意！考虑所有状态，完备性问题！）：**
2. **当少于 4 bit ：**

**相当于仍处于“LOCK”，在功能规定中有处理方式（直接退回“WAIT”）**

1. **当多于 4 bit：**

**假设用户并不知道输入了几位，那么在输入 4 bit 后还将继续输入（操作过程是“SWITCH”拨上—拨下—拨上），因为 always@ 块不能嵌套，状态的切换（ always@（negedge 4 “SWITCH”对应的信号））、“CODE”的输入（always@（posedge 4 “SWITCH”对应的信号））两个 always@ 块将是并行的，在每个块内进行的状态切换都需要考虑完备性问题（闭合图），不能在最后一个状态结束后不做处理，否则系统工作状态转入未知情况。**

**开题汇报 & 反馈建议：同 状态数量的计数问题，此处也可以设置计数器记录已经输入“CODE”的数目“CODE\_BIT”，从而利用“CODE\_BIT”实现足够完备的状态跳转。重新考虑完备性问题，对于上述可能的情况排列如下：**

1. **当输入位数int“CODE\_BIT”< 4，那么状态仍将停留在“INPUT”的显示：**

**可能的触发操作有：**

1. **“SWITCH”down**

**—下一位**

1. **“OK”**

**—“ERROR[]”/“ALARM”**

1. **“BACKSPACE”**

**—退回原始“INPUT”的显示，注意不是“WAIT”！**

1. **10s不操作**

**—退回“WAIT”状态**

1. **当输入位数int“CODE\_BIT”== 4，临界情况：**

**可能的触发操作有：**

1. **“SWITCH” down**

**—“ERROR[]”/“ALARM**

1. **“OK”**

**—“UNLOCK”/“ERROR[]”/“ALARM”**

1. **“BACKSPACE”**

**—“CODE\_BIT”==3**

1. **10s不操作**

**—退回“WAIT”状态**

1. **当输入位数“CODE\_BIT”> 4，此时显示的位数由于有 4 bits 的限制，之后的输入将不会改变原有 4 bits 密码的显示：**

**可能的触发操作有：**

1. **“SWITCH” down / up**

**—“ERROR[]”/“ALARM**

**∵ 假定——“当输入位数“CODE\_BIT”> 4”，**

**∴ 触发条件只有“SWITCH”，i.e.，在尝试输入更多位密码，其他的触发条件认为都可以归结到“当输入位数int“CODE\_BIT”== 4”的情况下做处理。**

1. **内部存储数据：**
2. **外部输入的密码“CODE”：**

**应该用reg信号保存，i.e.，将对应拨码开关的引脚和 .V 文件中的信号做相应的约束绑定**

1. **系统内置的正确密码，“KEY”：**

**用于最后的匹配**

1. **系统内置的已输入密码位数，“CODE\_BIT”：**

**涉及状态的跳转，需要保存在系统内**

1. **LED显示**
2. **等待状态“WAIT”：**
3. **段码“SEGMENTS”：**

**中间一段常亮，显示范围——四个密码位**

1. **16 × LED：**

**流水模式点亮，连续循环。具体设计找资料再定**

1. **2 × 三色LED：**

**全暗，不涉及时间操作。**

1. **AUDIO\_OUT：**

**不鸣响。**

1. **输入状态“INPUT”：**
2. **段码“SEGMENTS”：**

**下面一段常亮，显示范围——四个密码位**

1. **16 × LED：**

**全暗**

1. **2 × 三色LED：**

**每一次操作之后按颜色顺序开始闪烁，将计时过程可视化。**

1. **AUDIO\_OUT：**

**不鸣响。**

1. **输入结果状态“ERROR[]”：**
2. **段码“SEGMENTS”：**

**显示字符“ERROR”，用五位段码管，**

**点亮后闪烁三次，结束后回到造成“ERROR[]”的密码状态**

1. **16 × LED：**

**用以区分第一次、第二次“ERROR[]”：**

**1st——段码显示“ERROR”并闪烁同时，16 × LED全暗；**

**2nd——段码显示“ERROR”并闪烁同时，16 × LED全亮也在同一时间周期内交替闪烁三次（“ERROR”亮—LED暗，“ERROR”暗—LED亮）**

1. **2 × 三色LED：**

**在系统对“ERROR[]”状态做出表征时，2 × 三色LED不计时，全暗；**

**表征结束后，操作权回到用户，此时2 × 三色LED重新开始计时**

1. **AUDIO\_OUT：**

**鸣响“密码错误”**

1. **输入结果状态“ALARM”：**
2. **段码“SEGMENTS”：**

**显示字符“ALARM”，并不断闪烁持续报警，直到管理员作出处理**

1. **16 × LED：**

**与“ERROR[]”的处理方式类似，和段码字符“ALARM”明暗交替显示**

1. **2 × 三色LED：**

**∵ 理论上管理员不做处理，“ALARM”状态会持续发生**

**∴ 专用于计时的2× 三色LED全暗？**

1. **AUDIO\_OUT：**

**鸣响报警音**