中南大学

条形码实验报告

GS1-128 条码识别软件

学生姓名 刘 婕、王 睿、冼日胜

指导教师 贺建飚

学　　院 计算机学院

专业班级 物联网工程 1701、1702 班

完成时间 2020 年 5 月 15 日

第一章 实验概述

1.1 实现功能

1.可自动识别 SSCC 条码；

2.可自动识别包含表 6-88 中的 AI 标识符；

3.可自动识别含有 401、402、403、410、413、420、421、310n、330n 等 AI 标识符；

4.可自动识别具有链接功能的 GS-128 条码；

5.可双向自动识别 GS1-128 条码。

1.2 实验原理

GS1-128 条码结构包含起始符(startA、startB、startC)、FNC1、字符串(应用标识符+数据域)、符号校验字符、终止符。若应用标识符 AI 为表中的两位标识符，则其有对应的预定义字符串长度，并且保持不变；若 AI 为 401、402、310n 等，则没有预定义的长度。GS1-128 可以将多个字符串链接起来。当预定义长度字符串在链接中，其后不需要使用数据分隔符，每个字符串紧跟下一个应用标识符，直到校验符和终止符。若是可变长度字符串连接，需要使用 FNC1 作为数据分隔符，FNC1 紧跟在可变长度数据串的后面，FNC1 后面紧跟下一个字符串的应用标识符。当预定义长度字符串与其他字符串混合链接，建议将预定义长度字符串放在可变长度字符串前面，减少链接需要的条码字符。

第二章 需求分析

2.1 数据库需求分析

2.1.1 数据字典

表 1 数据项表

编号	数据项名	数据项含义	存储结构
01	A	字符集 A	VARCHAR
02	B	字符集 B	VARCHAR
03	C	字符集 C	VARCHAR
04	num	单元宽度(模块数)	INT

2.2 功能流程

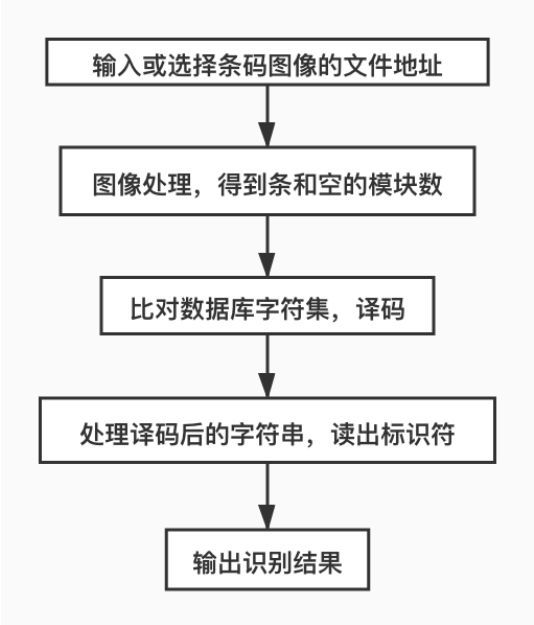


图 2.2 GS1-128 条码识别流程图

第三章 总体设计

3.1 软件设计主流程

整体系统区分为三个大模块完成：图像处理部分、字符集译码部分和标识符识别部分。考虑到译码过程中，同时进行数据库比对和标识符识别时，逻辑容易出现混乱和重复，因此将数据库译码部分和标识符及输出结果分为两部分进行，先将图像处理后的结果整体译码，再根据标识符识别进行结果的输出。

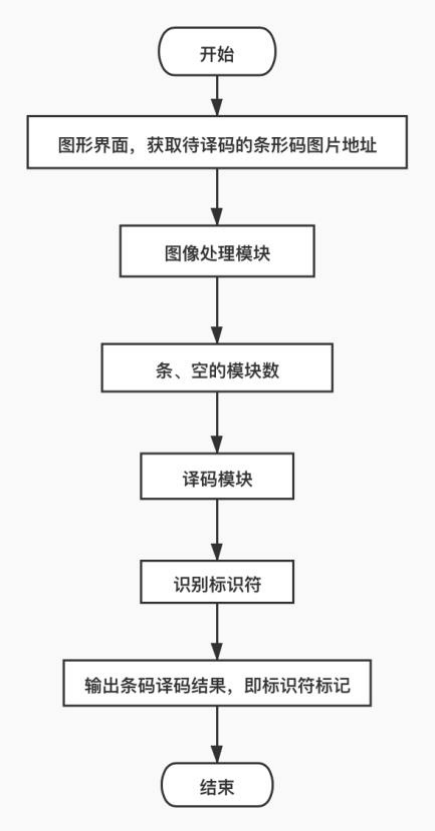


图 3.1 系统主流程图

3.2 系统功能模块

3.2.1 图像处理模块

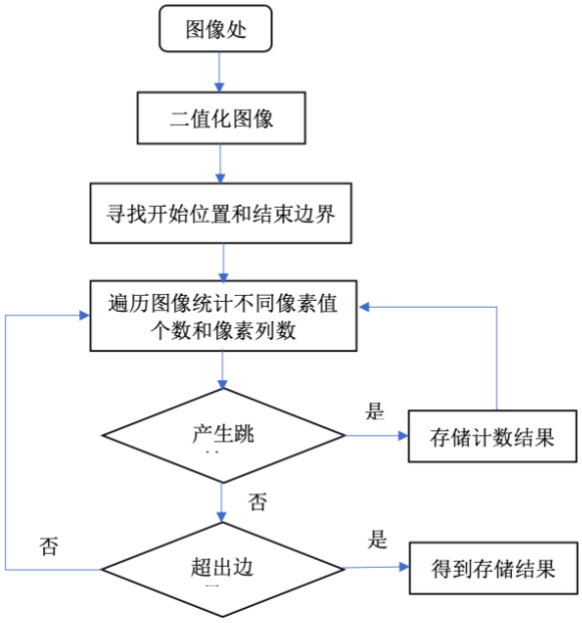


图 3.2 图像处理流程图

读取图像文件，先将图像像素二值化成为只有两个数值，以便于后续操作。对二值化图像进行按像素遍历，首先遍历找到条码开始的位置以及反向遍历找到条码结束的位置作为遍历的结束条件。按像素对图像进行遍历时，采取纵向扫描的方法，统计当前像素列各像素颜色的个数，分别为黑色像素个数和白色像素个数，遍历结束当前像素列时，比较哪个像素个数较多，将较多个数的像素设置为当前像素列的像素值。若当前像素列为 0，则不需要和前面像素列颜色进行比较，否则都要和前一个像素列进行比较，如果当前像素列的像素值等于前一个像素列的像素值，计数值加一，并水平平移一个像素点继续循环；否则意味着像素值产生了跳转，此时将当前计数值存入数组并将计数值重新置 1，继续新的循环，直到将所有黑/白像素列的计数值存入数组。得到黑/白像素列计数值的数值之后，设置标准宽度为数组第一个值的 1/2，因为 GS1—128 条码中，起始符和终止符的第一个条都是标准宽度的两倍，之后根据标准宽度将数组按照误差范围内进行更新，最后得到的是一个 int 类型的数组，数组中存储的是图像中条码内容的标准模块数。

3.2.2 字符集译码模块

字符集译码模块完成将条码模块数转换成相应字符的任务。模块开始时带入由图像处理得到的模块数数组，结束时生成字符数组，供后续识别标识符。

1.扫码方向及结构完整性判断

GS1-128 条码的双字起始符包括一个起始符（Start A，Start B 或 Start C）和 FNC1 字符，终止符结构是“2331112”。因系统要实现双向识别 GS1-128 条码，故需在开始译码前判断方向。首先假设模块数数组是由正向扫描得到的，获取前 6 个模块数，若是 Start A，Start B 或 Start C 其中之一，则为正向扫描。否则获取前 7 个模块数，若是“2111332”（终止符的镜像），则第一个字符是终止符，即为逆向扫描。此时需将模块数数组倒置后再进行译码。第二个字符 FNC1 用于标识 GS1 标准体系，故也需要确保其正确性。

判断方向函数 direction:

```
static char direction(String[] arr) {
    char c;n = getint(0,6,arr);
    if(n==211412)  { c='A';  storevalue(n);  }
    else if(n==211214)  { c='B';  storevalue(n);  }
    else if(n==211232)  { c='C';  storevalue(n);  }
    else {n = getint(0,7,arr);if(n==2111332)c='F';//终止符 else c='E';//error}
    return c;}
```

2.数据域译码

以6个模块数为一组进行循环译码。译码结果可能出现两种：特殊字符和普通字符。特殊字符进行相应的字符集切换，普通字符直接存储到数组。需要注意的是，切换字符CODE应用于切换字符后面的所有字符，直至符号结束或遇到另一个特殊字符；而转换字符SHIFT将转换字符之后的一个字符从字符集A转换到B或从B转换到A。

3.校验符验证

校验符由其之前的字符以一种算法计算而得，可以在译码时验证数据的正确性。算法如下：

（1）查表得到字符对应的字符值；

（2）将每个字符的值乘以相应权数。起始符和FNC1的权数为1，以后依次为2，3，4，…，n；

（3）将（2）中的乘积求和，对103取余数，结果即为校验符的字符值。

程序将译码得出的字符集数组进行上述计算，与扫描出的校验码进行对比，若一致则条码数据正确，可进一步识别标识符。

```
static boolean test(String[] s){
        int i = num1*6;  //校验码起始位置
        int tn = getint(i,6,s);  //校验码的模块数
        storevalue(tn);
        int tvalue = testvalue[num1-1];  //校验符的字符值
        int sum = testvalue[0]+testvalue[1];
        for(int k=2;k<num1-1;k++){sum = sum + k*testvalue[k];}
        if((sum%103)==tvalue)  //验证数据正确性
        {return true;}else return false;}
```

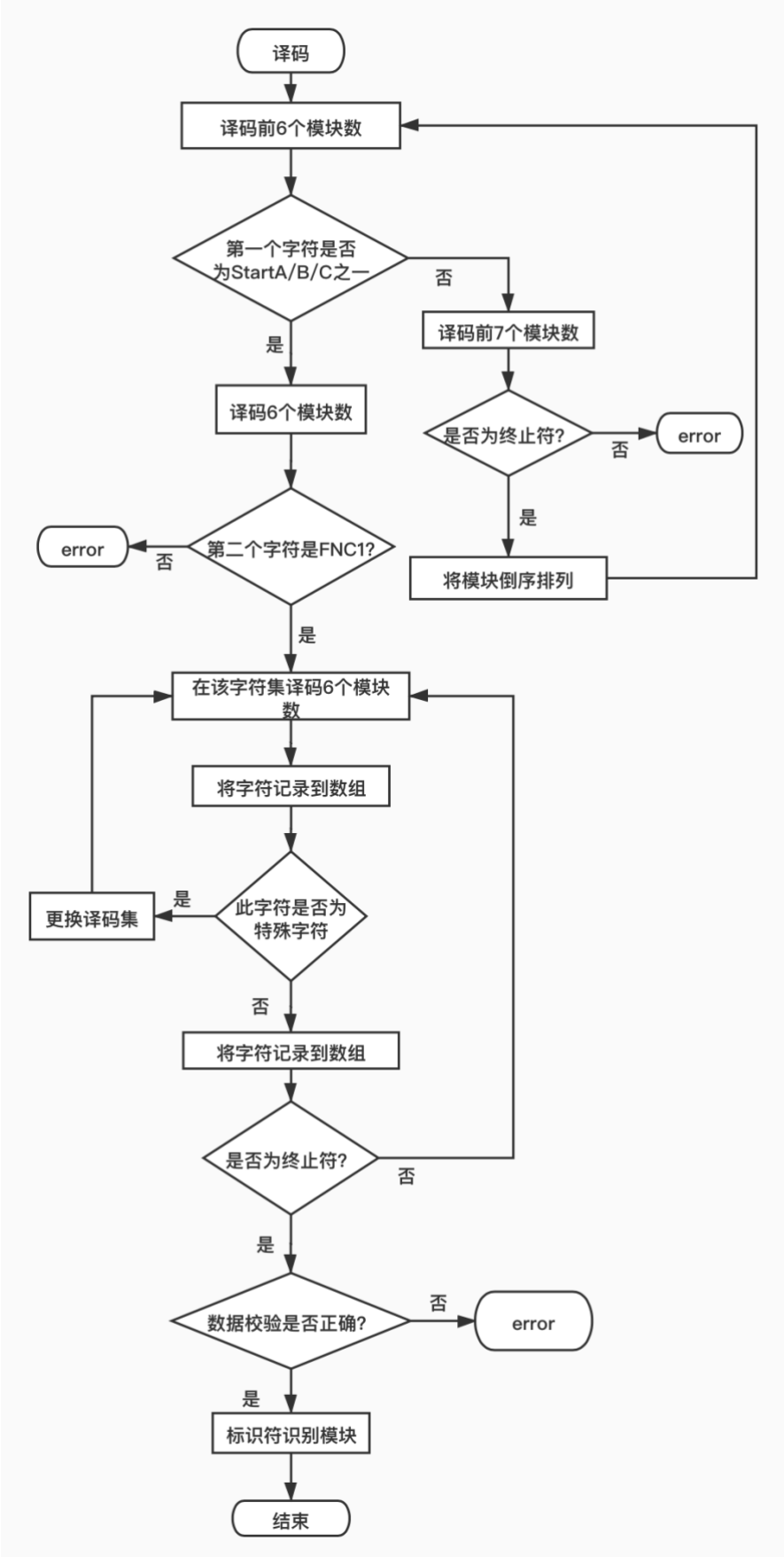


图 3.3 字符集译码流程图

3.2.3 标识符识别模块

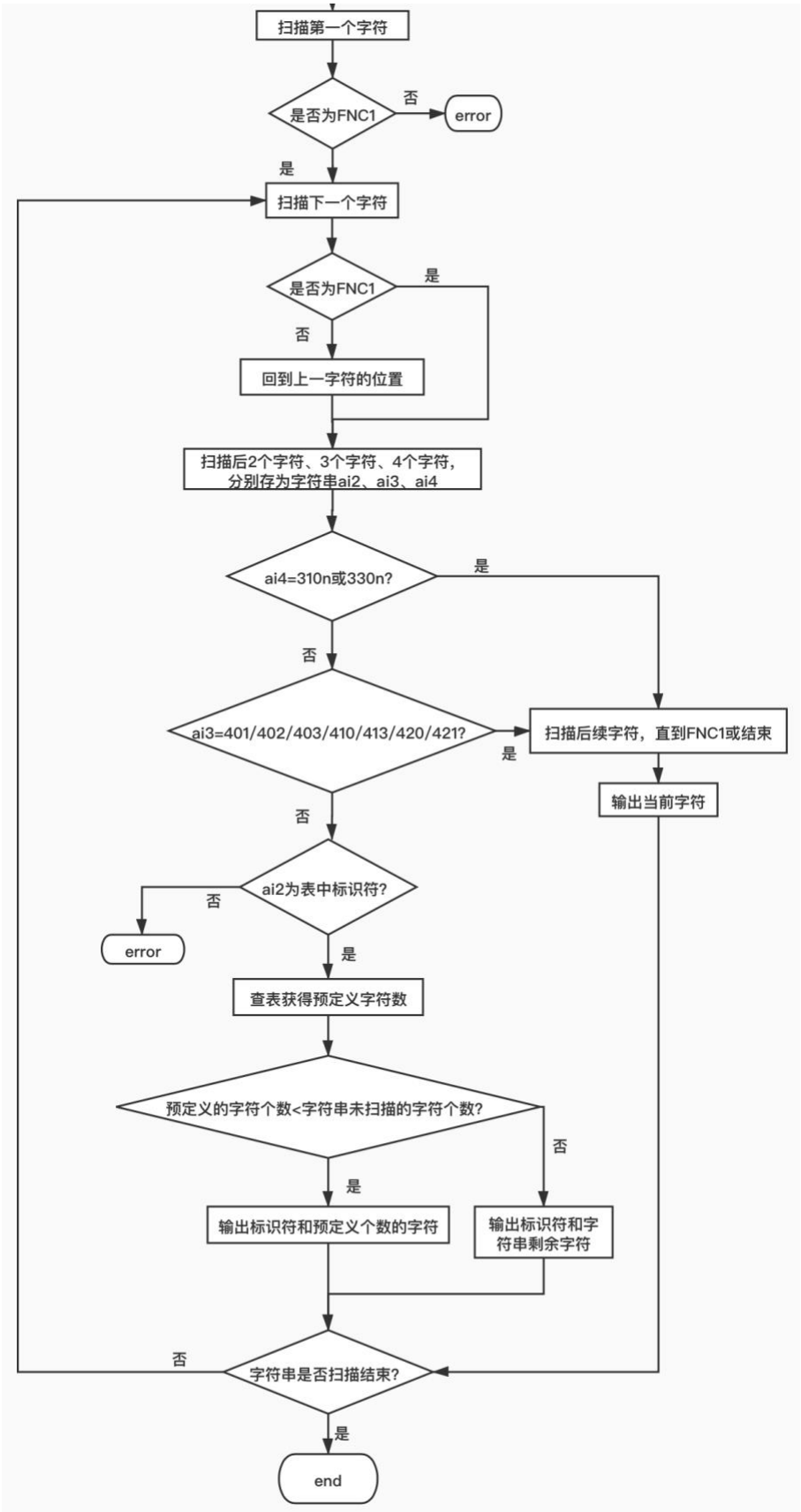

扫描第一个字符
是否为FNC1
否
error
是
扫描下一个字符
是否为FNC1
是
否
回到上一字符的位置
扫描后2个字符、3个字符、4个字符，
分别存为字符串ai2、ai3、ai4
ai4=310n或330n?
是
否
ai3=401/402/403/410/413/420/421?
是
扫描后续字符，直到FNC1或结束
输出当前字符
否
ai2为表中标识符?
否
error
是
查表获得预定义字符数
预定义的字符个数<字符串未扫描的字符个数?
否
是
输出标识符和预定义个数的字符
输出标识符和字
符串剩余字符
否
字符串是否扫描结束?
是
end

图 3.4 标识符识别流程图

在译码模块，返回的结果将是一组如{FNC1,0,1,9,9,3,1,0,0,9,5,5,FNC1,4,0,1,0,1}形式的字符串数组。在标识符识别模块则需要分别对两位的预定义长度标识符和三位、四位的不定长标识符做出识别，并完成链接条码的识别。

因链接条码中，需要以标识符作为引导，将条码区分为多组，而在每一组的识别过程，是重复的。因此，设置一个全局变量 k，用于标记每一次遍历数组的起点，并且在每一次遍历后，将其加上此次遍历的长度，以便于下一次遍历时从固定位置开始。以下遍历过程均以一个标识符引导的条码为一组。

为了便于区分四位、三位、两位标识符，在识别每组条码时，先取出数组的前四、三、二位，将其记录为 ai4、ai3、ai2 三种字符串，再依次进行判断：

```
if(!getai4(t,ai3,ai4, arr2)) {if(!getai3(t,ai3, arr2)) {if(!getai2(t,ai2, arr2)) {
                        System.out.println("条码有误，无法识别");}}}}
//判断 2 位标识符
    static boolean getai2(int t,String ai2,String [] arr2)
    {
        int charnum=getnum(ai2);   //获得 2 位标识符对应的字符长度
        if(charnum != 0)
        {
            String print1 = getstring(t, charnum, arr2);
            printfirst.add("\n"+"第"+(m+1)+"个标识符为："+ai2+"\n"+print1);
              m++;
            k=t+charnum;   //下次开始扫描的位置
            return true;
        }
        else
        {
           String res=getprint(t,arr2);
            printfirst.add("\n"+"第"+(m+1)+"个标识符为："+ai2+"\n"+res);
            System.out.println("test3:"+m+printfirst.get(m));
            m++;
            return true;
        }
```

```
        }
        //判断 4 位标识符
        static boolean getai4(int t,String ai3,String ai4 ,String []arr2)
        {
            if(ai3.equals("310")||ai3.equals("330"))
            {
                String res=getprint(t,arr2);
                printfirst.add("\n"+"第"+(m+1)+"个标识符为："+ai4+"\n"+res);
                m++;
                return true;
            }
            return false;
        }
        //判断 3 位标识符
        static boolean getai3(int t,String ai3,String [] arr2)
        {
 if(ai3.equals("401") || ai3.equals("402") || ai3.equals("403") || ai3.equals("410") || ai3.equals("413") ||
ai3.equals("420") || ai3.equals("421"))
            {
                String res = getprint(t, arr2);
                printfirst.add("\n"+"第"+(m+1)+"个标识符为："+ai3+"\n"+res);
                m++;
                return true;
            }
            return false;
        }
```

当识别到两位标识符时，需要读取数组的预定义长度或读取到结尾处：

```
//输出长字符串结果时，读取数组中的 n 个字符(或读到结尾)，输出格式为 string
    static String getstring(int t,int n,String[] src)
    {   //t 为起始位置，n 为获取的字符串长度，src 为源数组
        String b=src[t];
        for(int i=t+1;(i<t+n)&&(i<src.length);i++)
        {
```

```
            String a=src[i];
            b=b+a;
        }
        return b;
    }
```

当识别到三位或四位不定长标识符时，需要读取到数组的 FNC1 或结尾处：

```
//扫描并输出字符串，直到遇到 FNC1 或结束
static String getprint(int i,String[] src)
{    //i 为起始位置，src 为源数组
    String d=src[i];
    while(!src[i].equals("FNC1")&&!(i==src.length-1))
    {
        //依次扫描，直到遇到 FNC1 或到结尾结束
        i++;
        String c=src[i];
        if(!c.equals("FNC1"))
            d=d+c;
    }
    k=i+1;  //下次开始扫描的位置
    return d;
}
```

第四章 运行及使用

4.1 程序使用说明

1.导入 sql 文件，创建数据库 GS1-128

2.修改 DBConnect 的源码，将数据库连接密码改为本机设置的密码

4.2 运行效果

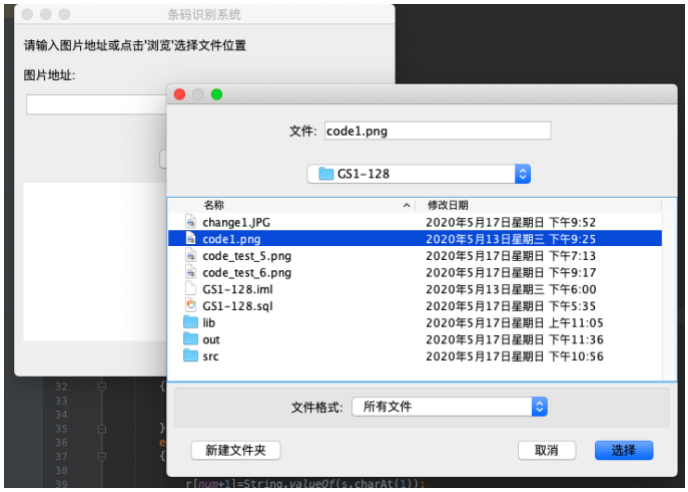
1.由三组链接而成的条码，标识符均为表内预定义长度的标识符

条码图像：


(01)99310095000358(15)950827(30)03

识别结果：

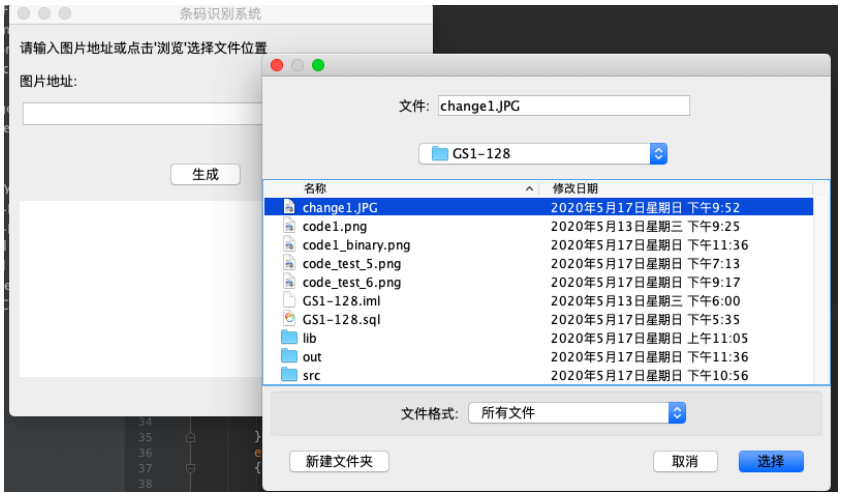

条码识别系统
请输入图片地址或点击'浏览'选择文件位置
图片地址:
浏览文件
生成


条码识别系统
请输入图片地址或点击'浏览'选择文件位置
图片地址:
文件: code1.png
GS1-128
名称
修改日期
change1.JPG 2020年5月17日星期日 下午9:52
code1.png 2020年5月13日星期三 下午9:25
code_test_5.png 2020年5月17日星期日 下午7:13
code_test_6.png 2020年5月17日星期日 下午9:17
GS1-128.iml 2020年5月13日星期三 下午6:00
GS1-128.sql 2020年5月17日星期日 下午5:35
lib 2020年5月17日星期日 上午11:05
out 2020年5月17日星期日 下午11:36
src 2020年5月17日星期日 下午10:56
文件格式: 所有文件
新建文件夹
取消
选择

2.由 1 中图像镜像反转后，反向识别

条码图像：

识别结果：



3.由一组预定义长度的条码和一组未定义长度的条码组合

条码图像：

识别结果：

4.用校验码验证，条码内容是否错误

条码图像：


(00)350123450123450123

识别结果：


条码识别系统
请输入图片地址或点击'浏览'选择文件位置
图片地址:
jie/Desktop/GS1-128/code_test_6.png
浏览文件
生成
数据有误，无法识别