上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

嵌入式开放性实验

EMBEDDED SYSTEM PROJECT

基于 M4 板的 BASIC 语言解析器的实现

IMPLEMENTION OF BASIC INTERPRETER ON CORTEX-M4

课题报告

PROJECT REPORT

学生姓名: 刘子凡 罗 晶

学生学号: 5140219311 5140219349

授课教师: 朱弘恣

专 业: 计算机科学

学 院: 电子信息与电气工程学院

目录

一、项目概述

1.1 项目名称

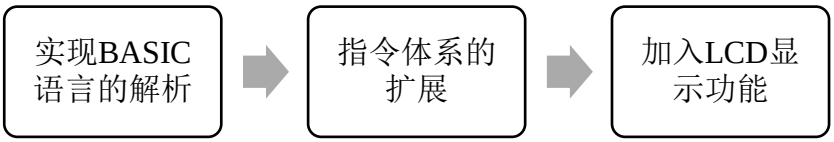
基于 M4 板的 BASIC 语言解析器的实现

1.2 项目背景

EK-TM4C1294XL 开发板：开发板数据手册见附件一。

BASIC 语言：BASIC 语言的介绍见附件二，使用 HELP 指令可以查看支持的语句，显示如下：

1.3 研发思路

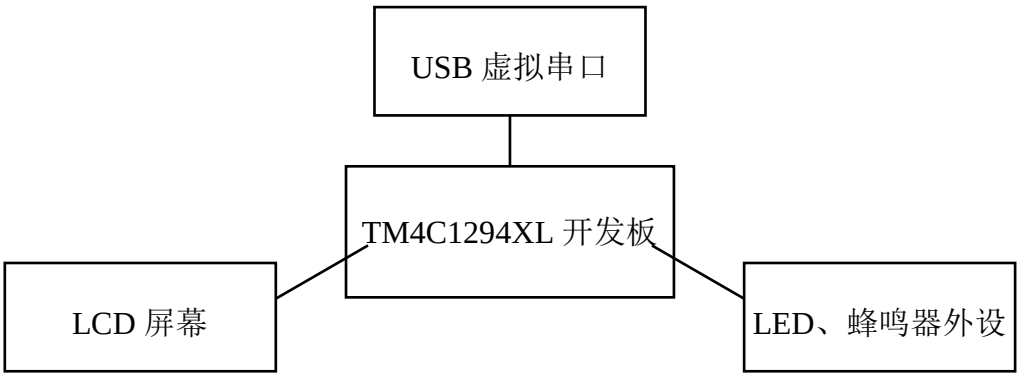


实现 BASIC 语句的解析：通过虚拟串口输入 BASIC 语句，经 M4 板处理后返回语句执行结果。

指令体系的扩展：加入 HIGH、LOW、DELAY 语句，控制开发板的引脚的高低电平。

加入 LCD 显示功能：引入 12864B 型 LCD，显示指令及运行结果。

1.4 系统总框图



1.5 项目分工

项目成员	分工方向	说明
刘子凡	底层设计	UART 的使用，GPIO 的控制，LCD 的设计
罗晶	顶层调用	BASIC 解释器的实现，BASIC 语句的扩展

二、研发历程（详见 ppt）

2.1 底层开发

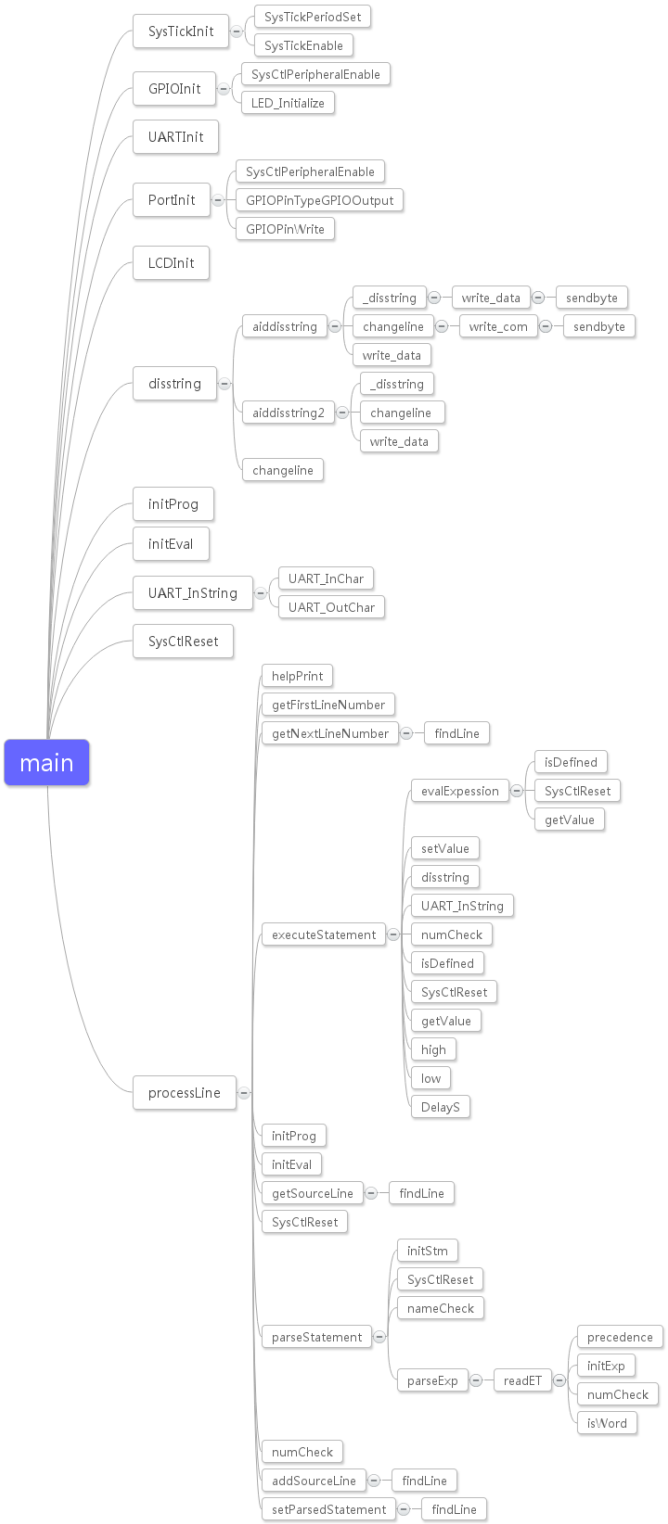
- UART
- GPIO 的控制
- DELAY 的实现
- 12864B 型 LCD 的开发
  - 主要参数
  - 引脚功能
  - 串行连接时序图
  - 控制代码
  - 演示

2.2 上层实现

- BASIC 解析的 C 语言代码框架
- BASIC 解析器的实现细节
  - 表达式树的建立与运算
  - 指令的储存和运行
- 代码的完备性和健壮性
  - 对 Basic 程序语句和控制指令的支持和扩展
  - 允许乱序输入行号
  - Error Report 机制

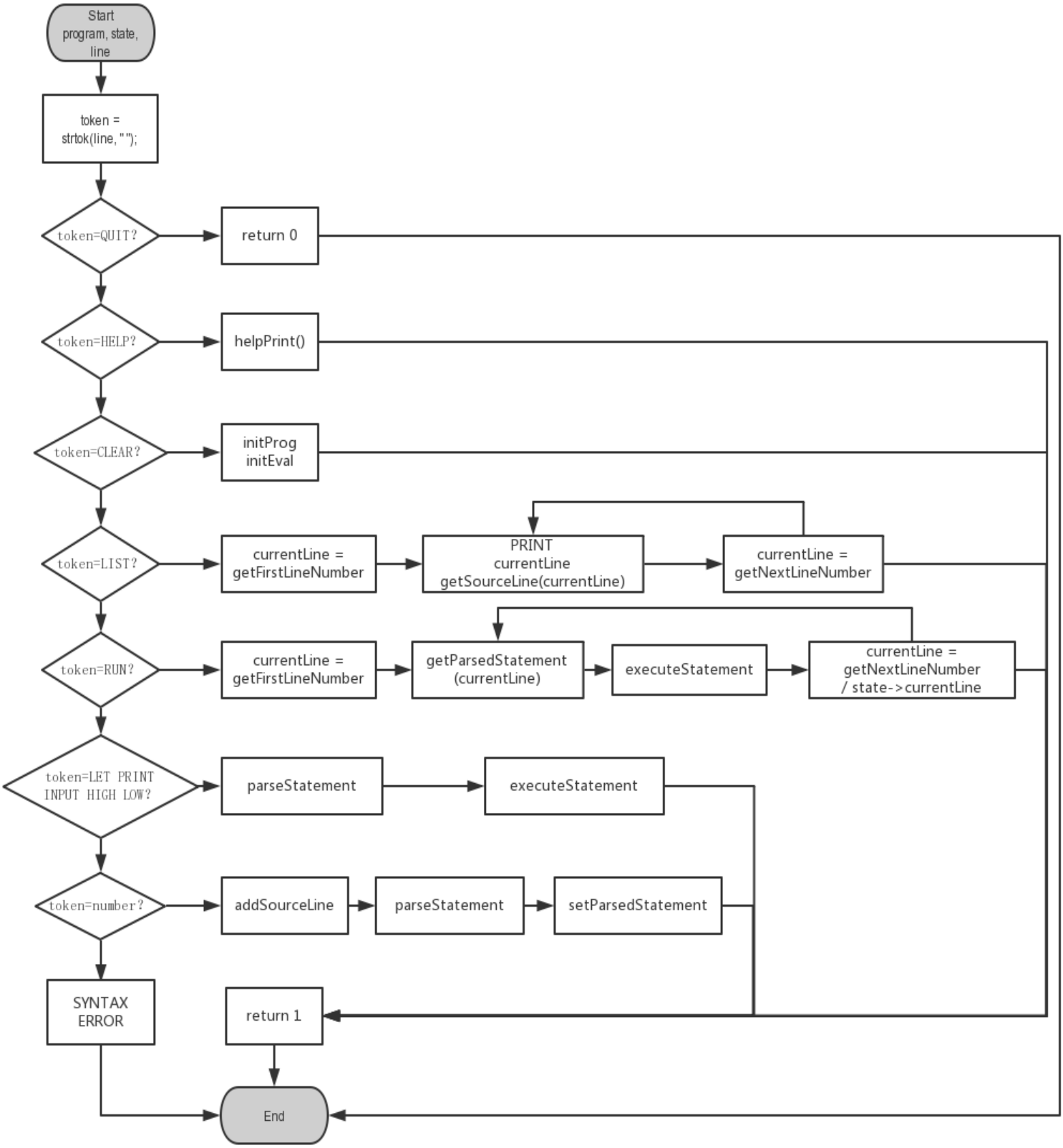
三、工程代码调用的函数说明

3.1 函数调用关系图（图片太大导致不清晰，见附件三）

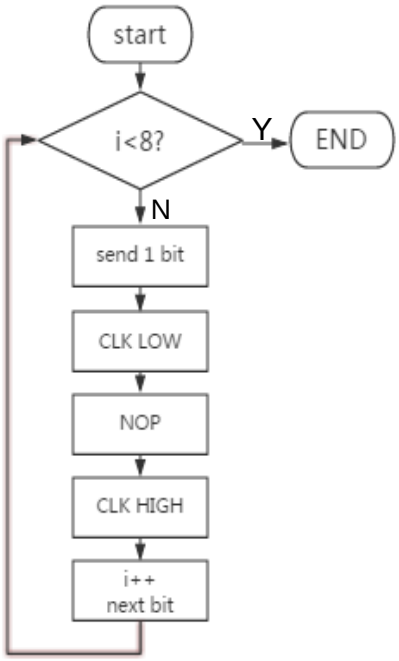


3.2 重要函数流程框图

- processLine 函数（图片太大导致不清晰，见附件四）



- sendbyte 函数（输出一个字节到 LCD）



3.3 函数输入输出及功能说明表（底层设计部分）

void SysTickInit()	
参数	无
返回	无
说明	无
功能	Systick 初始化设置与使能
void GPIOInit()	
参数	无
返回	无
说明	相关的 GPIO：portF portK portL portM portN portP portQ
功能	GPIO 初始化设置与使能
void UARTInit()	
参数	无
返回	无
说明	波特率设置为 115200，一次收发 8bit，无奇偶校验位，一个停止位
功能	UART 初始化设置与使能
void PortInit()	
参数	无
返回	无
说明	用于 LCD 屏幕
功能	PL1,PL0 的设置与使能

void LCDInit()	
参数	无
返回	无
说明	设置为：一次收发 8 位数据，显示字符从左到右移位，不显示光标，DDRAM 地址归位，显示地址初始设为第一行
功能	LCD 初始化设置与使能
void _disstring(char *s)	
参数	s 是字符数组的基地址
返回	无
说明	至多显示十五个字符，多余字符会被忽略
功能	将字符串显示在 LCD 屏幕上
void aiddisstring(char* str) / void aiddisstring2(char* str)	
参数	s 是字符数组的基地址
返回	无
说明	是 disstring 的辅助函数
功能	向左流动显示字符串，并设置缓存，使屏幕可以向上滚动显示
void changeline(int i)	
参数	i 是目标行，i 必须为 1,2,3,4
返回	无
说明	无
功能	LCD 屏幕换行
void UART_InString(char*buff)	
参数	buff 是字符数组的基地址
返回	无
说明	通过 UART 的方式收发
功能	读取字符串同时发送出去
int high(char ch,int num)	
参数	ch，num 共同表示所控制的端口，比如 ch = ‘N’, num = 1 表示 PN1
返回	1 表示正常，0 表示所选端口不存在或不被允许
说明	只能控制被允许的端口
功能	给特定 GPIO 输出高电平
int low(char ch,int num)	
参数	ch，num 共同表示所控制的端口，比如 ch = ‘N’, num = 1 表示 PN1
返回	1 表示正常，0 表示所选端口不存在或不被允许
说明	只能控制被允许的端口
功能	给特定 GPIO 输出低电平
void DelayS(int ms)	
参数	ms 表示需延迟的毫秒数
返回	无

说明	通过 Systick 实现，较为精确
功能	较为精确的延迟

3.4 函数输入输出及功能说明表（顶层调用部分）

tool.h	
int isWord(char* tmp)	
参数	待检查的字符串
返回	1 或 0
说明	如果是字符串都是字符或数字，返回 1，否则返回 0
功能	工具函数，检查一个字符串是否可以作为一个变量名
void nameCheck(char* name)	
参数	待检查的字符串
返回	无
说明	检查 LET、PRINT 语句的变量是否符合合法，例如，LET GOTO = 1 语句中，变量名 GOTO 不合法，报错 INVALID NAME
功能	检查变量名是否违规
int numCheck(char* tmp)	
参数	待检查的字符串
返回	1 或 0
说明	字符串是数字返回 1，否则返回 0
功能	检查字符串是否为数字
char* subLine(char* line, int pos, int len)	
参数	待截取的字符串
返回	字符串
说明	用于一些语句的句法设置
功能	从 line 的特定位置 pos 截取长为 len 的字符串
int precedence(char* op)	
参数	待检查的运算符 op
返回	该运算符的优先级
说明	=的优先级为 1，+和-的优先级为 2，*和/的优先级为 3
功能	返回一个运算符优先级，在表达式建立中需要用到
void helpPrint()	
参数	无
返回	无
说明	输出 HELP 语句对应的信息
功能	同上
evalstate.h	
#define maxNum 30	

typedef struct{ int currentLine; char mapSymbol[maxNum][10]; int mapValue[maxNum]; int numSymbol; }EVALSTATE;	
void initEval(EVALSTATE* state)	
参数	EVALSTATE* state
返回	无
说明	将 numSymbol 设置为 0
功能	初始化 EVALSTATE
void setValue(EVALSTATE* state, const char* var, int value)	
参数	EVALSTATE* state，变量名 const char* var，变量值 int value
返回	无
说明	state 中如果存在 var 变量，则更改它的值，否则新增一个变量 var 并设值
功能	改变或设置变量值
int getValue(EVALSTATE* state, const char* var)	
参数	EVALSTATE* state，变量名 const char* var，变量值 int value
返回	变量值
说明	变量不存在报错 VARIABLE NOT DEFINED，存在则返回其值
功能	返回变量值
int isDefined(EVALSTATE* state, const char* var)	
参数	EVALSTATE* state，变量名 const char* var
返回	1 或 0
说明	查看变量 var 是否被定义
功能	同上
exp.h	

struct _EXPRESSION{
int type;//1 for CONSTANT; 2 for IDENTIFIER; 3 for COMPOUND
int value;//for CONSTANT
char* name;//for IDENTIFIER
char op;//for COMPOUND
struct _EXPRESSION *lhs,*rhs;//for COMPOUND
};
typedef struct _EXPRESSION EXPRESSION;	

void initExp(EXPRESSION* exp)	
参数	表达式 exp
返回	无
说明	初始化表达式

功能	同上
int evalExpession(EVALSTATE* state, EXPRESSION* exp)	
参数	变量对应表 state，表达式 exp
返回	表达式值
说明	根据表达式不同的类型返回不同的值，如 type 为 1，直接返回常量值，type 为 3，迭代的方式计算 left 和 right 的值，再根据 op 的情况返回这个组合表达式的值
功能	表达式求值
EXPRESSION* readET(int type, LINE* line, int prec)	
参数	选择函数功能的 type,一行语句 LINE* line，上一个表达式读取时字符的优先级 prec
返回	表达式指针
说明	$\mathbf{E} \rightarrow \mathbf{T}$　　$\mathbf{T} \rightarrow$ *constant* $\mathbf{E} \rightarrow \mathbf{E}\ op\ \mathbf{E}$　　$\mathbf{T} \rightarrow$ *identifier* $\mathbf{T} \rightarrow (\mathbf{E})$ 表达式树的构建。type 为 0 代表读取 Expression，type 为 1 代表读取的是 Term。这是一个内部迭代、反复调用的函数。
功能	构建表达式树，返回表达式
EXPRESSION* parseExp(LINE line)	
参数	字符串 line
返回	表达式
说明	因为 readET 用到了递归，所以通过 EXPRESSION* exp = readET(0, &line, 0)开启这个递归。
功能	readET 的包装函数
statement.h	

typedef struct{
int type;//1 for REM; 2 for LET; 3 for INPUT; 4 for PRINT; 5 for GOTO; 6 for IF; 7 for END; 8 for FOR; 9 for NEXT£»10 for HIGH; 11 for LOW; 12 for DELAY
char* name;
EXPRESSION* exp;
int nextLine;
EXPRESSION* exp2;
char op;
}STATEMENT;	

void initStm(STATEMENT* statement)	
参数	STATEMENT* statement
返回	无
说明	初始化 STATEMENT
功能	同上
STATEMENT* parseStatement(LINE originalLine)	

参数	字符串 originalLine
返回	STATEMENT 指针
说明	这个是针对一行去掉了行号的 BASIC 语言代码进行语法分析，建立并设置一个 statement，用到了 token 分解
功能	分解语句得到 STATEMENT
void executeStatement(EVALSTATE* state, STATEMENT* statement, int NextLine)	
参数	EVALSTATE* state, STATEMENT* statement，针对 FOR NEXT 语句才会用到的 NextLine
返回	无
说明	针对不同的 statement type 进行运行，NextLine 只在 NEXT 语句才用到，如果 NEXT 语句后的变量仍小于 FOR 语句设定的范围，那么 state->currentLine = NextLine，NextLine 即为 FOR 语句的下一条语句的行号。
功能	运行语句，得到相应的结果，如输出值、更改 state 里的变量值、更改 state 里的 currentLine

program.h

```
struct _sunion{
    LINE line;//原代码
    STATEMENT* statement;//分解后得到的 STATEMENT
    int index;//index 是在 vec 中这个数组中的下标
};

typedef struct _sunion* sunion;

#define maxLine 100
typedef struct{
    int ln[maxLine];//行号数组
    sunion su[maxLine];//代码行构成的结构体数组
    int vec[maxLine];//行号的有序数组
    int numLine;//代码总行数
}PROGRAM;
```

void initProg(PROGRAM* program)	
参数	PROGRAM* program
返回	无
说明	设置 numLine 为 0
功能	初始化
int findLine(PROGRAM* program, int lineNumber)	
参数	PROGRAM* program, int lineNumber
返回	下标
说明	根据行号找到 ln[]数组的下标，这个也是 su[]数组的下标，一一对应

功能	分解语句得到 STATEMENT
void addSourceLine(PROGRAM* program, int lineNumber, LINE line)	
参数	PROGRAM* program, 行号 lineNumber, LINE line
返回	无
说明	给 program 添加代码元素
功能	同上
LINE getSourceLine(PROGRAM* program, int lineNumber)	
参数	PROGRAM* program, 行号 lineNumber,
返回	LINE
说明	LIST 指令调用此函数，根据行号打印源码
功能	寻找源语句
int getFirstLineNumber(PROGRAM* program)	
参数	PROGRAM* program
返回	第一句运行指令的行号
说明	返回最先运行代码的行号
功能	同上
int getNextLineNumber(PROGRAM* program, int lineNumber)	
参数	PROGRAM* program, 行号 lineNumber,
返回	下一句运行指令的行号
说明	返回下一句运行代码的行号，如果已经是最后一句，返回-1 代表已经执行到最后
功能	同上
STATEMENT* getParsedStatement(PROGRAM* program, int lineNumber)	
参数	PROGRAM* program, 行号 lineNumber,
返回	STATEMENT 指针
说明	根据行号分解语句，得到 STATEMENT
功能	分解语句，供 RUN 时运行
void setParsedStatement(PROGRAM* program, int lineNumber, STATEMENT *statement)	
参数	PROGRAM* program, 行号 lineNumber, STATEMENT 指针
返回	无
说明	将行号和 STATEMENT 对应
功能	同上
int processLine(PROGRAM* program, EVALSTATE* state, LINE line)	
参数	PROGRAM* program, EVALSTATE* state, LINE line
返回	1 或 0
说明	整个 BASIC 解析代码的接口函数，对输入的一行代码的处理，包罗万象。
功能	处理 BASIC 代码

四、项目总结

4.1 项目中遇到的困难

1) 相比 C++，C 语言缺乏面向对象编程的功能

2) malloc 动态分配导致堆空间不足

3) LCD 显示字符需要时间延迟，影响 SSCOM 文件流的输入

4.2 项目收获

1) 锻炼了 C 语言编程能力

2) 熟练掌握了 M4 板的基本操作

3) 培养了翻阅数据手册查找关键信息的能力

4) 培养了团队合作能力