```
/***********************************************
 * File: scan.c         Project by: David Hodgdon *
 * Class: ECE476        Instructor: Bruce Land    *
 ***********************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "scan.h"
#include "parse.h"
#include "lcd.h"
#include "serial.h"

#ifndef ATMEL
#define memcmpf memcmp
#endif

#ifdef ATMEL
#include "serial.h"
#endif

#ifdef ATMEL
#define exit(x) while(1);
#endif

float scanRealValue;    // floating point number appearing in code
int scanIntValue;               // integer value appearing in code +-32K, 16bit
integer
char scanIsFloat;               // 1=number is FP, 0=number is INT
char equalCount;        // must be reset to ZERO for each "line"!!!!!!!
char tempStr[TEMPLEN+1];

#ifdef ATMEL
#define GETC    SerialGetChar()
#else
#define GETC    getc(f)
#endif
#define SCANVAR {\
        switch(c) {\
                case '%':       s=ScanIntegerVar;c=GETC;break;\
                case '$':       s=ScanStringVar;c=GETC;break;\
                default:        s=ScanRealVar;break;\
        }\
}
char c, cBackup;                        // current character in code


#ifndef ATMEL
FILE *f; FILE *fBackup;                 // file pointer, only for testing on PC
#endif

#ifdef ATMEL
void ScanDebug(flash char *err) {
#else
void ScanDebug(char *err) {
#endif
        sprintf(lcd_buffer, "(%d)");
        LCDPrintString(lcd_buffer);
        sprintf(lcd_buffer, err);
        LCDPrintString(lcd_buffer);
}


#ifdef ATMEL
void ScanError(flash char *err) {
#else
void ScanError(char *err) {
#endif
        ScanDebug(err);
#ifdef ATMEL
```



```
        while(1);
#else
        exit(1);
#endif
}

static void ScanIdentifier() {
        register unsigned char len;
        for(len=0; (('A'<=c && c<='Z') || ('0'<=c && c<='9')) /* upper character
s */; len++) {
                if(len<TEMPLEN) tempStr[len]=c;
                c=GETC;/*getc(f);*/
        }
        if(len>=TEMPLEN) ScanError("Scan: Identifier too long\n");
        tempStr[len]='\0';      // finish string
}

// This could be either a integer or a floating point number
static void ScanNumber() {
        register unsigned char len;
        scanIsFloat=0;
        for(len=0; ('0'<=c && c<='9') || (c=='.'); len++) {
                if(c=='.') scanIsFloat=1;       // num is FP, so use FP string f
unc
                if(len<TEMPLEN) tempStr[len]=c;
                c=GETC;
        }
        if(len>=TEMPLEN) ScanError("ScanNumber: Number too long\n");
        tempStr[len]='\0';      // terminate string
        if(scanIsFloat) scanRealValue=(float)atof(tempStr);
        else                    scanIntValue=atoi(tempStr);
}

static void ScanString() {
        register unsigned char len;
        for(len=0; c!='"'; len++) {
                if(len<TEMPLEN) tempStr[len]=c;
                c=GETC;
        }
        if(len>=TEMPLEN) ScanError("ScanString: String too long\n");
        tempStr[len]='\0';
}

enum ScanSymbol ScanGetSymbol() {
        register enum ScanSymbol s;
        for(s=0;(c!=EOF)&&(c==' '||c=='\t');c=GETC);
        switch(c) {
                case EOF: s=ScanEOF; break;
                case '\n':s=ScanEOL; c=GETC; break;
                case ':': s=ScanNewStatement; c=GETC;break;
                case '+': s=ScanPlus; c=GETC;break;
                case '-': s=ScanMinus;c=GETC;break;
                case '*': s=ScanTimes;c=GETC;break;
                case '/': s=ScanDiv;      c=GETC;break;
                case '^': s=ScanExpon;c=GETC;break;
                case '=':
                        s=ScanEqual;
                        c=GETC;
                        if(c=='<') {            // =< symbol
                                s=ScanLessThanEqual;
                                c=GETC;
                        }
                        else if(c=='>') {       // => symbol
                                s=ScanGreaterThanEqual;
                                c=GETC;
                        }
                        break;
                case '<': s=ScanLessThan; c=GETC;
                        if(c=='=') { s=ScanLessThanEqual; c=GETC;}
```

```
                else if(c=='>') {s=ScanNotEqual; c=GETC;}
                break;
        case '>': s=ScanGreaterThan; c=GETC;
                if(c=='=') { s=ScanGreaterThanEqual; c=GETC;}
                else if(c=='<') {s=ScanNotEqual; c=GETC;}
                break;
        case ',': s=ScanComma;  c=GETC;break;
        case '(': s=ScanLeftParen;c=GETC;break;
        case ')': s=ScanRightParen;c=GETC;break;
        case ';': s=ScanSemicolon; c=GETC;break;
        case '&': s=ScanPRINT; c=GETC; break;
        case '"': s=ScanStr; c=GETC; ScanString(); c=GETC; break;
        /*********************************/
        case '0':case '1':case '2':case '3':case '4':case '5':
        case '6':case '7':case '8':case '9':case '.':
                s=ScanNum;
                ScanNumber();
                break;
        case 'A': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "AND",3)==0) s=ScanAND;
                else {SCANVAR}
                break;
        case 'D': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "DATA",4)==0) s=ScanDATA;
                else if(memcmpf(tempStr, "DIM",3)==0) s=ScanDIM;
                else {SCANVAR};
                break;
        case 'F': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "FOR",3)==0) s=ScanFOR;
                else {SCANVAR}
                break;
        case 'G': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "GOSUB",5)==0) s=ScanGOSUB;
                else if(memcmpf(tempStr, "GOTO",4)==0) s=ScanGOTO;
                else {SCANVAR}
                break;
        case 'H': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "HOME",4)==0) s=ScanHOME;
                else {SCANVAR}
                break;
        case 'I': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "IF",2)==0) s=ScanIF;
                else if(memcmpf(tempStr, "INPUT",5)==0) s=ScanINPUT;
                else if(memcmpf(tempStr, "INT",3)==0) s=ScanINT;
                else {SCANVAR}
                break;
        case 'L': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "LET",3)==0) s=ScanLET;
                else if(memcmpf(tempStr, "LEN",3)==0) s=ScanLEN;
                else {SCANVAR}
                break;
        case 'N': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "NEXT",4)==0) s=ScanNEXT;
                else if(memcmpf(tempStr, "NOT",3)==0) s=ScanNOT;
                else {SCANVAR}
                break;
        case 'O': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "OR",2)==0) s=ScanOR;
                else {SCANVAR}
                break;
        case 'P': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "PRINT",5)==0) s=ScanPRINT;
                else {SCANVAR}
                break;
        case 'R': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "REM",3)==0) s=ScanREM;
                else if(memcmpf(tempStr, "READ",4)==0) s=ScanREAD;
                else if(memcmpf(tempStr, "RAND",4)==0) s=ScanRAND;
                else {SCANVAR}
```



```
                break;
        case 'S': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "STEP",4)==0) s=ScanSTEP;
                else if(memcmpf(tempStr, "SND",3)==0) s=ScanSND;
                else {SCANVAR}
                break;
        case 'T': s=ScanIdent; ScanIdentifier();
                if(memcmpf(tempStr, "THEN",4)==0) s=ScanTHEN;
                else if(memcmpf(tempStr, "TO",2)==0) s=ScanTO;
                else {SCANVAR}
                break;
        default:
                if('A'<=c && c<='Z') {
                        s=ScanIdent;
                        ScanIdentifier();
                        SCANVAR;
                        break;
                }
    }
    return s;
}

void ScanSkipLine() {
    register enum ScanSymbol s;
    do{
            s=ScanGetSymbol();
    }while(s!=ScanEOL && s!=ScanEOF);
    s=ScanGetSymbol();
}

void ScanInit() {
#ifdef ATMEL
    char i;
#if 0
    i=SerialFOpen(fname);
#else
  i=SerialFOpen(InputGetString());
#endif
    if(i!=0) ScanError("ScanInit: File could not be opened.\n");
#else
    f=fopen(fname, "r+t");
    if(f==NULL) ScanError("ScanInit: File could not be opened.\n");
#endif
    c=GETC;
}

#ifdef ATMEL
void ScanReopenForATMEL() {
            SerialReopenFile(newPC);
            c=GETC;
}
#endif

void ScanClose(){
#ifndef ATMEL
    fclose(f);
#endif
}
```