# User's Guide

## 1.Variable and Register

One unsigned 8-bit variable **B** (Brightness) and sixteen 16-bit registers **R0** to **R15**, the range of register value is -32768 to 32767.

**Purpose:**

Set the global brightness **B** and the value stored in registers **R0** to **R15**.

**Examples:**

```
B = 255
'Set the global brightness B to 255.
B = R0
R1 = 123
'Set the register R1 to 123.
R3 = R1
R4 = B
```

## 2.Arithmetic Operators

Expressions combine variables and constants to produce new values. Arithmetic operators are used to perform many of the familiar arithmetic operations that involve the calculation of numeric values represented by variables.

You can add two values in an expression together with the + operator or subtract one from another with the - operator, as the following example demonstrates.

```
R1 = R2 + 123
R1 = R2 + R3
R4 = R5 - 123
R4 = 123 - R5
R6 = R7 - R8
```

Negation also uses the - operator, but with only one operand, as the following example demonstrates.

```
R0 = -123
```

Multiplication and division use the * operator and / operator, respectively, as the following example demonstrates.

```
R0 = R0 * 10
R1 = R2 * R3
```

```
R2 = R0 / 10
R2 = R1 / R3
```

## 3.Logical Operations

Logical operations are performed between two data bits (except for *NOT*). Bits can be either "1" or "0", and these operations are essential to performing digital math operations.

The *NOT* operator performs logical negation. It yields the logical opposite of its operand. The following example illustrates this.

```
R1 = 1
R2 = NOT R1
'The preceding statement set R2 to 0.
```

The *AND* operator performs logical conjunction,the *OR* operator performs logical disjunction, and the XOR operator performs logical exclusion.The following example illustrates the *AND*,*OR* and *XOR* operators.

```
R1 = 1
R2 = 2
R3 = R1 AND R2
'The preceding statement set R3 to 0.
R1 = 1
R2 = 2
R3 = R1 OR R2
'The preceding statement set R3 to 3.
R1 = 1
R2 = 2
R3 = R1 XOR R2
'The preceding statement set R3 to 3.
```

## 3.CLS Statement

**Purpose:**

Set the brightness of all LED to 0.

Syntax:

**CLS**

Examples:

CLS

Set the brightness of all LED to 0.

## 4.PRINT Statement

**Purpose:**

To output a display to the LED.

**Syntax:**

**PRINT** *<Index>/<Ri>* [,*<Bright>/<Rb>*]

**Comments:**

Index is a string, it constants in the list must be enclosed in double quotation marks. For function keys, please see the following table:

| Key | Expression | Key | Expression |
|---|---|---|---|
| Esc | <ESC> | F1 | <F1> |
| F2 | <F2> | F3 | <F3> |
| F4 | <F4> | F5 | <F5> |
| F6 | <F6> | F7 | <F7> |
| F8 | <F8> | F9 | <F9> |
| F10 | <F10> | F11 | <F11> |
| F12 | <F12> | Print Screen | <PRTSC> |
| Scroll Lock | <SCROLL> | Pause | <PAUSE> |
| Backspace | <BSP> | Insert | <INS> |
| Home | <HOME> | Page Up | <PGUP> |
| Num Lock | <NUM> | Tab | <TAB> |
| Delete | <DEL> | End | <END> |
| Page Down | <PGDN> | Caps Lock | <CAPS> |
| Enter | <ENT> | Left Shift | <LSHIFT> |
| Right Shift | <RSHIFT> | UP | <UP> |
| Left | <LEFT> | Down | <DOWN> |
| Right | <RIGHT> | Left Ctrl | <LCTRL> |
| Left Win | <LWIN> | Left Alt | <LALT> |

| | | | |
|---|---|---|---|
| Space | <SPACE> | Right Alt | <RALT> |
| Right Win | <RWIN> | Apps | <APP> |
| Right Ctrl | <RCTRL> | Keypad / | <K/> |
| Keypad * | <K*> | Keypad - | <K-> |
| Keypad + | <K+> | Keypad 0 | <K0> |
| Keypad 1 | <K1> | Keypad 2 | <K2> |
| Keypad 3 | <K3> | Keypad 4 | <K4> |
| Keypad 5 | <K5> | Keypad 6 | <K6> |
| Keypad 7 | <K7> | Keypad 8 | <K8> |
| Keypad 9 | <K9> | Keypad Enter | <KENT> |

If *[,<Bright>/<Rb>]* is omitted, global brightness will be used.

If [,*<Bright>/<Rb>*] is included, separated by commas, the values of the expressions are displayed. Expressions in the list may be numeric(brightness) or register **R0** to **R15**, the range of brightness is 0 to 255.

**Examples:**

PRINT "WASD<F1><K1>"
*'Display LED with index(Key W,A,S,D,F1 and numeric keypad 1), and global brightness ( variable **B**) will be used.*
PRINT "A",255
*'Display LED with index(Key A), and the value of brightness is 255.*
PRINT "A",R1
*'Display LED with index(Key A), and the value of register **R1** will be used as brightness.*

## PRINT ALL[,*<Bright>/<Rb>*]

Comments:

Display all LED in brightness <Bright> or brightness stored in Rb.

**Examples:**

PRINT ALL,255
*'Display all LED in brightness 255.*
PRINT ALL,0

*'Turn off all LED.*

## 5.PSET Statements

**Purpose:**

To display LED at a specified place on the keyboard.

**Syntax:**

**PSET** *X/Rx , Y/Ry [,<Bright>/<Rb>]*

**Comments:**

*X ,Y (X* and *Y* is an integer*)* or the value of register *(**Rx , Ry**)* represents the coordinates of the key.

If *[,<Bright>/<Rb>]* is omitted, global brightness will be used.

If [,*<Bright>/<Rb>*] is included, separated by commas, the values of the expressions are displayed. Expressions in the list may be numeric(brightness) or register **R0** to **R15**, the range of brightness is 0 to 255.

The relationship between position and key is as below:

| X , Y | 0 | 1 | ... | 21 | 22 |
|---|---|---|---|---|---|
| 0 | Esc | NULL | ... | PN | Toggle |
| 1 | ` | 1 | ... | Keypad * | Keypad - |
| 2 | Tab | Q | ... | Keypad 9 | Keypad + |
| 3 | Caps Lock | A | ... | Keypad 6 | Keypad + |
| 4 | Left Shift | Z | ... | Keypad 3 | Keypad Enter |
| 5 | Left Ctrl | Left Win | ... | Keypad . | Keypad Enter |

**Examples 1:**

PSET 0,0

*'Display LED with position (0,0), and global brightness ( variable **B**) will be used.*

**Examples 2:**

```
R1 = 10
R2 = 0
R3 = 255
PSET R1,R2,R3
'Display LED with position Rx,Ry(10, 0) in brightness stored in R3.
```

## 6. IF ... THEN ... ELSE ... END IF Statement

**Purpose:**

To make a decision regarding program flow based on the result returned by an expression.

**Syntax:**

```
IF condition THEN
statement(s) 1
[ELSE
 statement(s) 2
]
END IF
```

**Comments:**

If the result of *condition* (only one *condition* can be specified) is nonzero (logical true), *statement(s) 1* is executed.

If the result of *condition* is zero (false), the *statement(s) 1* is ignored and the *statement(s) 2,* if present, is executed. Otherwise, execution continues with the next executable statement.

THEN and ELSE may be followed by one or more statements to be executed.

**Examples 1:**

The next statement causes the LED of key A to be displayed, depending on the value of register R0. If R0 is zero, goes to the line 2; otherwise, goes to the terminal.

```
IF R0 = 0 THEN
   PRINT “A”,255
END IF
```

**Examples 2:**

```
IF R0 >= 0 THEN
   PRINT “A”,255
ELSE
   PRINT “B”,255
END IF
```

## 7.GOTO Statement

**Purpose:**

Goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.

**Syntax:**

```
GOTO label
. . .
. . .
label:
statement(s)
```

**Comments:**

Label is an identifier. When, the control of program reaches to goto statement, the control of the program will jump to the label: and executes the code after it.

**Examples:**

```
Reset:
IF R0 < 10 THEN
   R0 = R0 + 1
   GOTO Reset
END IF
```

## 8.WAIT Statement

**Purpose:**

To suspend program execution.

Syntax:

**WAIT <Integer>/Rm**

Comments:

Wait for **<Integer>** or **Rm** millisecond. The range of **<Integer>** is 0 to 9999.

Examples:

```
WAIT 1000
'Wait for 1000 milliseconds.
R0 = 500
WAIT R0
'Wait for 500 milliseconds.
```