**8086微机系统仿真说明文档**

# 1.命名规则

**·一般变量、函数命名原则：**采用小驼峰式命名法。第一个单词以小写字母开始；从第二个单词开始以后的每个单词的首字母都采用大写字母。

**·特殊变量命名原则：**1）**引脚名：**低电平有效小写，高电平有效大写；

**·类命名原则：**采用大驼峰式命名法，首字母大写，其余同小驼峰式命名法。

**·枚举量命名原则：**使用大驼峰式命名法，另外应封装枚举以避免重名。

**·常量及系统变量：**全部大写。

**·空间：**一般使用项目名称。

# 2.数据类型

·枚举量

- 引脚电平（voltage）

- 引脚名（pinName）

·变量

- 8或16位寄存器变量：short

·字符串：QString/string？

# 3.设计规则

## 3.1顶层设计

**·将基类、1级派生类声明为抽象类**

在基类中声明如下纯虚函数，在派生类重写：

- 设置引脚电平 void setVoltage（enum ,short）;

- 读取引脚电平 voltage readVoltage（enum）;

在1级派生类中视情况声明纯虚函数，在2级派生类中重写；

**·在“方法类”中声明并实现具体指令操作**

每一执行汇编指令的函数均被声明为该类的成员函数。图形化界面使用的硬件对象将在该类中生成，且硬件对象的方法将在该类的构造函数中被调用。

**·利用函数重载整合代码**

## 3.2详细设计

**·将基类、1级派生类声明为抽象类**

在基类中声明如下纯虚函数，在派生类重写：

- 设置引脚电平 void setVoltage（enum ,short）;

# 附录一 类一览

**1.基类**

1）类名：

硬件（Hardwares）

2）属性：

硬件名（hardwareName） [QString/string]

3）方法：

·设置类名

void setHardwareName

·设置引脚电平

virtual void setVoltage(MicroCom::Pins,Voltage)

**1.1 1级派生类1**

1）类名：

8086CPU（CPUs）

2）属性：

·内部寄存器

|  |  |  |  |
| --- | --- | --- | --- |
| unsigned short | ax | unsigned short | ss |
| unsigned short | bx | unsigned short | bp |
| unsigned short | cx | unsigned short | sp |
| unsigned short | dx | unsigned short | si |
| unsigned short | cs | unsigned short | di |
| unsigned short | ds | unsigned short | ip |
| unsigned short | es | unsigned short | flags |

·引脚电平

|  |  |  |  |
| --- | --- | --- | --- |
| Voltage | AD[16] | Voltage | ALE |
| Voltage | AS[4] | Voltage | READY |
| Voltage | Mio | Voltage | RESET |
| Voltage | wr | Voltage | NMI |
| Voltage | rd | Voltage | inta |
| Voltage | den | Voltage | INTR |
| Voltage | bhe | Voltage | test |
| Voltage | DTr | Voltage | CLK |

3）方法：

·引脚选择器，返回指向引脚的枚举指针

Voltage\* selectPin(MicroCom::Pins);

·获取引脚电压值

Voltage getPinVoltage(MicroCom::Pins);

·设置引脚电压值

bool setVoltage(MicroCom::Pins, Voltage pinVol);

·寄存器选择器，返回指向寄存器的指针（注：ah等返回的是ax）

unsigned short\* selectRegister(MicroCom::Regs);

·获取寄存器的值

unsigned short getRegValue(MicroCom::Regs);

·获取寄存器某一位的值

Voltage getRegValue(MicroCom::Regs, short pos);

·设置寄存器的值

void setRegValue(MicroCom::Regs, short value);

·设置寄存器某一位的值

void setRegValue(MicroCom::Regs, Voltage biValue, short pos);

·将原码转换成补码(short->unsigned)

unsigned short toCompForm(short value, MicroCom::RegsLen len = MicroCom::dbyte);

·将补码转换成有符号数（默认长度为16位）

unsigned short toUnsignedTrueForm(unsigned short value, MicroCom::RegsLen len = MicroCom::dbyte);

·将补码转换成无符号数（默认长度为16位）

short toSignedTrueForm(unsigned short value, MicroCom::RegsLen len = MicroCom::dbyte);

**·1级派生类2**

类名：

存储器（Memory）

属性：

存储单元（data） [结构体 memoryUnit]

//结构体存储单元叫什么？

引脚电平（G等） [枚举 voltage]

方法：

设置引脚电平（void setVoltage）

读取引脚电平（voltage readVoltage）

设置存储单元值（bool setMemoryUnit）

读取存储单元值（short readMemoryUnit）

设置存储单元地址（bool setMemoryUnitAddr）

读取存储单元地址（short readMemoryUnitAddr）

**·1级派生类3**

类名：

组合逻辑电路（Combinational logic circuit）

属性：

方法：

·2级派生类3.1

类名：

译码器（Decoder）

属性：

方法：

·2级派生类3.2

类名：

编码器（Encoder）

属性：

方法：

·2级派生类3.3

类名：

缓冲器（Buffer）

属性：

方法：

·2级派生类4

类名：

锁存器（Latch）

属性：

方法：

**·1级派生类4**

类名：

可编程芯片（Programmable chip[FPGA]）

属性：

方法：

·2级派生类4.1

类名：

I/O接口芯片8255A（io Interface chip）

属性：

方法：

·2级派生类4.2

类名：

计数器8053/8254（Counter）

属性：

方法：

·2级派生类4.3

类名：

中断控制器8259A（Interrupt controller）

属性：

方法：

**·1级派生类5**

类名：

逻辑门（Logic Gate）

属性：

方法：

·2级派生类5.1

类名：

与门（AND Gate）

属性：

方法：

·2级派生类5.2

类名：

或门（OR Gate）

属性：

方法：

·2级派生类5.3

类名：

非门（NOT Gate）

属性：

方法：

·2级派生类5.4

类名：

与非门（NAND Gate）

属性：

·2级派生类5.5

类名：

或非门（NOR Gate）

属性：

·2级派生类5.6

类名：

异或门（XOR Gate）

属性：

**·1级派生类6**

类名：

输入设备（Input Unit）

属性：

方法：

·2级派生类6.1

类名：

键盘（Keyboard）

属性：

方法：

·2级派生类6.2

类名：

开关（on-off switch）

属性：

方法：

**·1级派生类7**

类名：

输出设备（Output Unit）

属性：

方法：

·2级派生类7.1

类名：

LED灯（LED）

属性：

方法：

·2级派生类7.2

类名：

控制台（Console）

属性：

方法：

·2级派生类7.3

类名：

示波器（Oscilloscope）

属性：

方法：

·2级派生类7.4

类名：

8段数码管（Digital Tube）

属性：

方法：

**·1级派生类8**

类名：

辅助装置（Auxiliary）

属性：

方法：

·2级派生类8.1

类名：

电阻（Resistance）

属性：

方法：

·2级派生类8.2

类名：

电源（Power Source）

属性：

方法：

·2级派生类8.3

类名：

电源（Ground Electrode）

属性：

方法：

# 附录二 枚举量一览

1. enum MicroCom::Pins

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant** | **Value** | **Constant** | **Value** |
| MicroCom::AD1 | 1 | MicroCom::AS18 | 18 |
| MicroCom::AD2 | 2 | MicroCom::AS19 | 19 |
| MicroCom::AD3 | 3 | MicroCom::AS20 | 20 |
| MicroCom::AD4 | 4 | MicroCom::rd | 21 |
| MicroCom::AD5 | 5 | MicroCom::wr | 22 |
| MicroCom::AD6 | 6 | MicroCom::Mio | 23 |
| MicroCom::AD7 | 7 | MicroCom::ALE | 24 |
| MicroCom::AD8 | 8 | MicroCom::DTr | 25 |
| MicroCom::AD9 | 9 | MicroCom::den | 26 |
| MicroCom::AD10 | 10 | MicroCom::bhe | 27 |
| MicroCom::AD11 | 11 | MicroCom::NMI | 28 |
| MicroCom::AD12 | 12 | MicroCom::INTR | 29 |
| MicroCom::AD13 | 13 | MicroCom::inta | 30 |
| MicroCom::AD14 | 14 | MicroCom::READY | 31 |
| MicroCom::AD15 | 15 | MicroCom::RESET | 32 |
| MicroCom::AD16 | 16 | MicroCom::test | 33 |
| MicroCom::AS17 | 17 | MicroCom::CLK | 34 |
| MicroCom::VCC | 51 | MicroCom::gnd | 50 |

2. enum MicroCom::Regs

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant** | **Value** | **Constant** | **Value** |
| MicroCom::ax | 1 | MicroCom::ip | 13 |
| MicroCom::bx | 2 | MicroCom::flags | 14 |
| MicroCom::cx | 3 | MicroCom::al | 15 |
| MicroCom::dx | 4 | MicroCom::bl | 16 |
| MicroCom::cs | 5 | MicroCom::cl | 17 |
| MicroCom::ds | 6 | MicroCom::dl | 18 |
| MicroCom::es | 7 | MicroCom::io | 19 |
| MicroCom::ss | 8 | MicroCom::ct | 20 |
| MicroCom::bp | 9 | MicroCom::ah | 30 |
| MicroCom::sp | 10 | MicroCom::bh | 31 |
| MicroCom::si | 11 | MicroCom::ch | 32 |
| MicroCom::di | 12 | MicroCom::dh | 33 |

3. enum MicroCom:: RegsLen

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant** | **Value** | **Constant** | **Value** |
| dbyte | 16 | Byte | 8 |

4. enum Voltage

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant** | **Value** | **Constant** | **Value** |
| high | 5 | low | 0 |
| Himped | -1 |  |  |