**8086微机系统仿真说明文档**

# 1.命名规则

（1）一般变量、函数

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。

（2）特殊变量

·引脚名：低电平有效小写，高电平有效大写；

（3）类

采用大驼峰式命名法，首字母大写，其余同小驼峰式命名法。

（4）枚举量

使用大驼峰式命名法，另外应封装枚举以避免重名。

（5）常量及系统变量

全部大写。

（6）空间

一般使用项目名称。

# 2.数据类型

## 2.1 枚举

（1）引脚号（Pins）

包括所有硬件的引脚号。8086CPU的引脚号范围为：1-34

（2）引脚电平（Voltage）

包括high和low两种；

（3）寄存器号（Regs）

对所有原件的寄存器号进行枚举，包括8086CPU内部寄存器、8255A控制字寄存器等；

## 2.2 变量

（1）8位或16位寄存器变量

为真实反映计算机内部存储数据的特点，同时便于利用位运算直接对变量进行操作，统一使用unsigned short（16位无符号整型变量）。

（2）字符串

使用QString或string（暂时未碰到，待商榷）

（3）类私有变量

将所有用于表示寄存器或引脚的变量均用数组表示，这样避免使用switch语句增加程序负担。

# 3.设计规则

## 3.1顶层设计

**（1）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：

- 选择引脚 virtual Voltage\* selectPin(MicroCom::Pins);

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；

声明纯虚函数的好处：可以在父类中直接调用而不必给出实现方法，在子类中视情况对其进行改写，进而实现针对不同的子类实现不同的方法。

**（2）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。

**（3）在“方法类”中使用“信号与槽”的机制实现对连线的仿真**

## 3.2详细设计

# 附录一 类一览

## 1.基类

1）类名：

·硬件（Hardwares）

2）属性：

|  |  |
| --- | --- |
| Data Type | Name |
| QString/string | hardwareName |

3）方法：

·设置类名

void setHardwareName

·选择引脚

virtual Voltage\* selectPin(MicroCom::Pins)=0;

·处理外部引脚电平变化

virtual void handleOuterVolChange(MicroCom::Pins, Voltage)=0;

·设置引脚电平

bool setVoltage(MicroCom::Pins, Voltage);

·读取引脚电平

Voltage getPinVoltage(MicroCom::Pins);

·10进制数转换为2进制数

void toBinary(int denary, short binary[]);

·2进制数转换为10进制数

unsigned short toDenary(short binary[]);

·原码转换为补码

unsigned short toCompForm(short value, MicroCom::RegsLen);

·补码转换为原码

short toTrueForm(unsigned short value, MicroCom::RegsLen);

### 1.1 1级派生类1

1）类名：

·8086CPU（CPUs）

2）属性：

·内部寄存器

|  |  |
| --- | --- |
| Data Type | Name |
| Unsigned short[14] | innerReg |

·引脚电平

|  |  |
| --- | --- |
| Data Type | Name |
| Voltage[34] | pins |

3）方法：

·引脚选择器，返回指向引脚的枚举指针

Voltage\* selectPin(MicroCom::Pins);

·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（有符号数）

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

·设置寄存器的值（无符号数）

void setRegUnsignedValue(MicroCom::Regs, unsigned short);

·设置寄存器某一位的值

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

·获取数据线的值

unsigned short getDataValue(MicroCom::RegsLen);

·读总线周期

unsigned short readBusCycle(int phyAddr);

·写总线周期

void writeBusCycle(int phyAddr, unsigned short value);

### ·1.2 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::si | 13 |
| MicroCom::bx | 2 | MicroCom::di | 14 |
| MicroCom::cx | 3 | MicroCom::io | 15 |
| MicroCom::dx | 4 | MicroCom::ct | 16 |
| MicroCom::cs | 5 | MicroCom::al | 17 |
| MicroCom::ds | 6 | MicroCom::bl | 18 |
| MicroCom::es | 7 | MicroCom::cl | 19 |
| MicroCom::ss | 8 | MicroCom::dl | 20 |
| MicroCom::ip | 9 | MicroCom::ah | 21 |
| MicroCom::flags | 10 | MicroCom::bh | 22 |
| MicroCom::bp | 11 | MicroCom::ch | 23 |
| MicroCom::sp | 12 | MicroCom::dh | 24 |

**3. enum MicroCom:: RegsLen**

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

**4. enum MicroCom:: Mode**

|  |  |  |  |
| --- | --- | --- | --- |
| **Constant** | **Value** | **Constant** | **Value** |
| MicroCom::write | 0 | MicroCom::read | 1 |

**5. enum Voltage**

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