```cpp
class StateMachine:public CompositeState {
private:
  Stopped* stopped;
  Operating* operating;
public:
  bool transition(Stopped* state, On* event);
  bool transition(Operating* state, Off* event);
  void setIniDefaultState();}
```

```cpp
class Stopped: public State {
private:
  StateMachine* ancestor;
public:
  virtual bool processEvent(On* event) {
    return this->ancestor
            ->transition(this,event);
  }
}
```

```cpp
class Operating: public State {
private:
  StateMachine* ancestor;
public:
  virtual bool processEvent(Off* event) {
    return this->ancestor
            ->transition(this,event);
  }
  virtual void onEntryAction() {
    this->context->Prepare();)
  virtual void onExitAction() {
    this->context->Disable();)
}
```
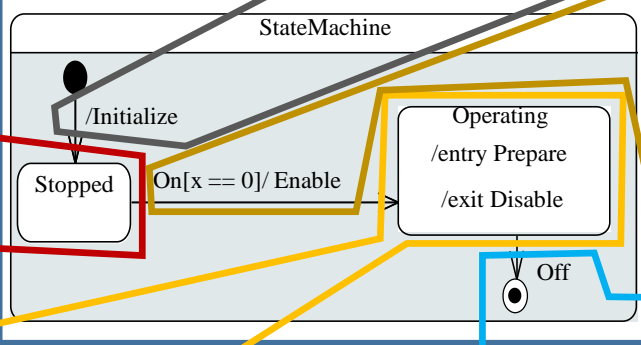
```cpp
void StateMachine::setIniDefaultState(){
  this->context->Initialize();
  this->activeSubState = stopped;
  this->activeSubState->entry();
}
```

```cpp
bool StateMachine::transition(
        Stopped* state, On* event) {
 if(this->context->guard(event)){
   this->activeSubState->exit();
   this->context->Enable(event);
   this->activeSubState = this->operating;
   this->activeSubState->entry();
   return true;}
 return false;
}
```

```cpp
bool StateMachine::transition(
    Operating* state, Off* event) {
   this->activeSubState->exit();
   //no action defined
   this->activeSubState = NULL;
   return true;
  }
```

StateMachine

/Initialize

Stopped

On[x == 0]/ Enable

Operating

/entry Prepare

/exit Disable

Off