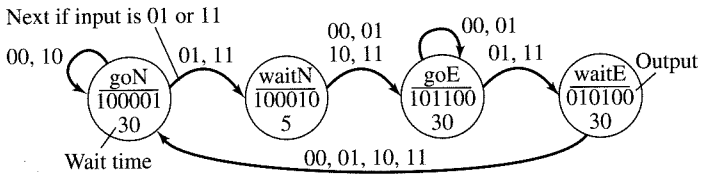

Figure 2.25
Graphical form of a Moore FSM that implements a traffic light.

State \ Input	00	01	10	11
goN, 100001,30	goN	waitN	goN	waitN
waitN, 100010,5	goE	goE	goE	goE
goE, 001100,30	goE	goE	waitE	waitE
waitE, 010100,5	goN	goN	goN	goN

Table 2.12
Tabular form of a Moore FSM that implements a traffic light.

The first step in designing the software is to decide on the sequence of operations.

1. Initialize timer and directions registers
2. Specify initial state
3. Perform FSM controller
   a) Output to traffic lights, which depends on the state
   b) Delay, which depends on the state
   c) Input from sensors
   d) Change states, which depends on the state and the input

The second step is to define the FSM graph using a data structure. Program 2.11 shows two possible implementations of the Moore FSM. The implementation on the left uses a table

Program 2.11
Two 6812 C implementations of a Moore FSM.

```
// Table implementation
const struct State {
  unsigned char Out;
  unsigned short Time;
  unsigned char Next[4];};
typedef const struct State STyp;
#define goN   0
#define waitN 1
#define goE   2
#define waitE 3
STyp FSM[4]={
 {0x21,3000,{goN,waitN,goN,waitN}},
 {0x22, 500,{goE,goE,goE,goE}},
 {0x0C,3000,{goE,goE,waitE,waitE}},
 {0x14, 500,{goN,goN,goN,goN}}};
unsigned char Input;
void main(void){
unsigned char n; // state number
  Timer_Init();
  DDRB = 0xFF;
  DDRA &= ~0x03;
  n = goN;
  while(1){
    PORTB = FSM[n].Out;
    Timer_Wait10ms(FSM[n].Time);
    Input = PORTA&0x03;
    n = FSM[n].Next[Input];
  }
}
```

```
// Pointer implementation
const struct State {
  unsigned char Out;
  unsigned short Time;
  const struct State *Next[4];};
typedef const struct State STyp;
#define goN   &FSM[0]
#define waitN &FSM[1]
#define goE   &FSM[2]
#define waitE &FSM[3]
STyp FSM[4]={
 {0x21,3000,{goN,waitN,goN,waitN}},
 {0x22, 500,{goE,goE,goE,goE}},
 {0x0C,3000,{goE,goE,waitE,waitE}},
 {0x14, 500,{goN,goN,goN,goN}}};
STyp *Pt;  // state pointer
unsigned char Input;
void main(void){
  Timer_Init();
  DDRB = 0xFF;
  DDRA &= ~0x03;
  Pt = goN;
  while(1){
    PORTB = Pt->Out;
    Timer_Wait10ms(Pt->Time);
    Input = PORTA&0x03;
    Pt = Pt->Next[Input];
  }
}
```