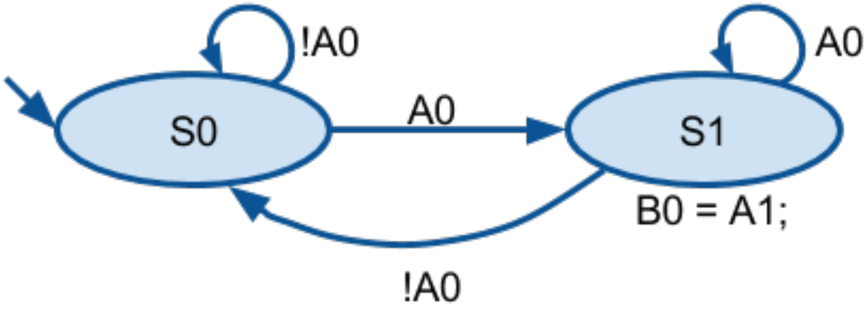
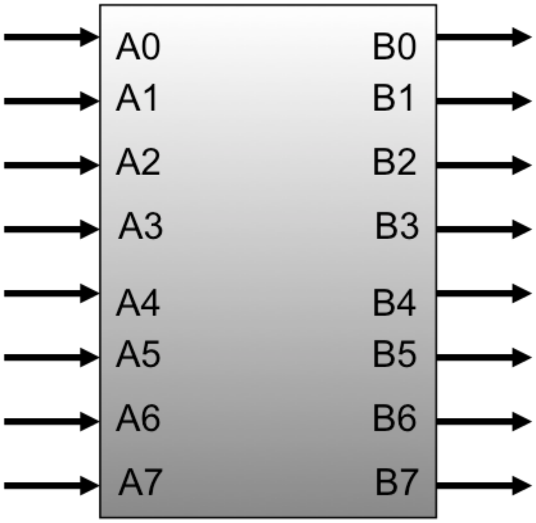
SM Basics

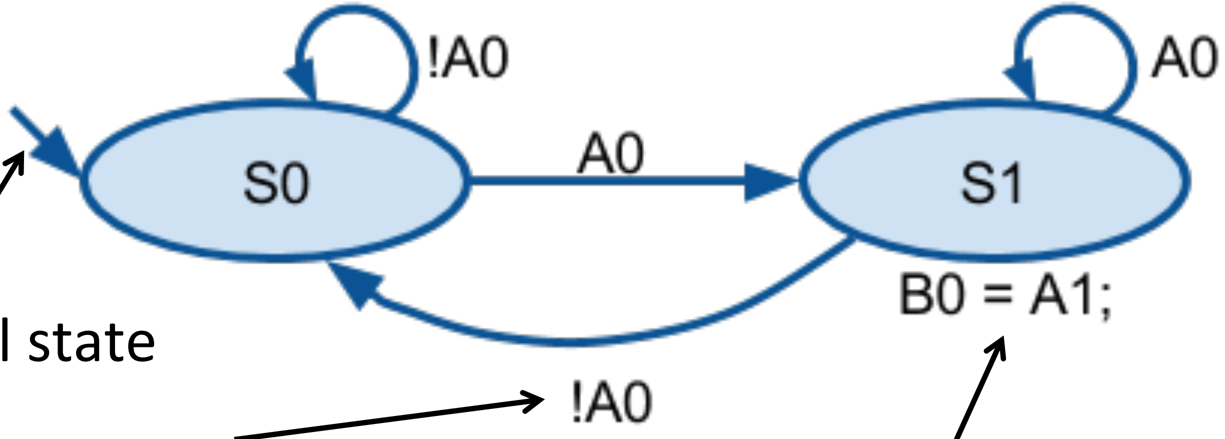
State Machine (SM)



RIMS

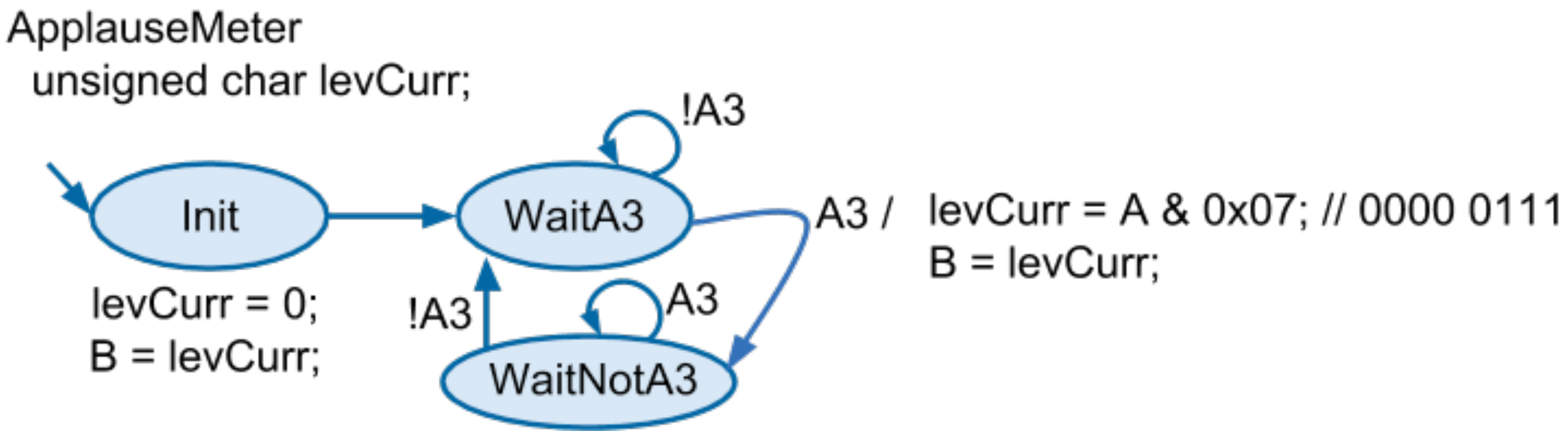


SM Basics



- Initial state
- Transitions between states
  - Controlled by a Boolean function
  - Often involves RIMS Input variables
- State Actions
  - Execute when control transfers to the state
  - Arbitrary C code, RIMS output, etc.

Example: Applause Meter SM



- New features in this example
  - State-holding variable (levCurr)
  - More complex state actions
  - Actions on transitions

C Code SM Template

```
#include "RIMS.h”
enum States { … } state;

void Tick() {

        switch(state) { // Transitions
                …
        } // Transitions

        switch(state) { // State actions
                …
        } // State actions
}

void main() {
         B = 0x00; // Initialize outputs
         state = …; // Indicates initial call
         while(1) { Tick(); }
}
```

C Code SM Template

List the states

```
#include "RIMS.h”
enum States { … } state;

void Tick() {

        switch(state) { // Transitions
                …
        } // Transitions

        switch(state) { // State actions
                …
        } // State actions
}

void main() {
         B = 0x00; // Initialize outputs
         state = …; // Indicates initial call
         while(1) { Tick(); }
}
```

C Code SM Template

List the states

Declare global variables here

- Must be visible to all states

```
#include "RIMS.h”
enum States { … } state;

void Tick() {

    switch(state) { // Transitions
        …
    } // Transitions

    switch(state) { // State actions
        …
    } // State actions
}

void main() {
    B = 0x00; // Initialize outputs
    state = …; // Indicates initial call
    while(1) { Tick(); }
}
```

C Code SM Template

List the states

Declare global variables here

- Must be visible to all states

Set the initial state

```
#include "RIMS.h”
enum States { … } state;

void Tick() {

        switch(state) { // Transitions
            …
        } // Transitions

        switch(state) { // State actions
            …
        } // State actions
}

void main() {
        B = 0x00; // Initialize outputs
        state = …; // Indicates initial call
        while(1) { Tick(); }
}
```

C Code SM Template

List the states

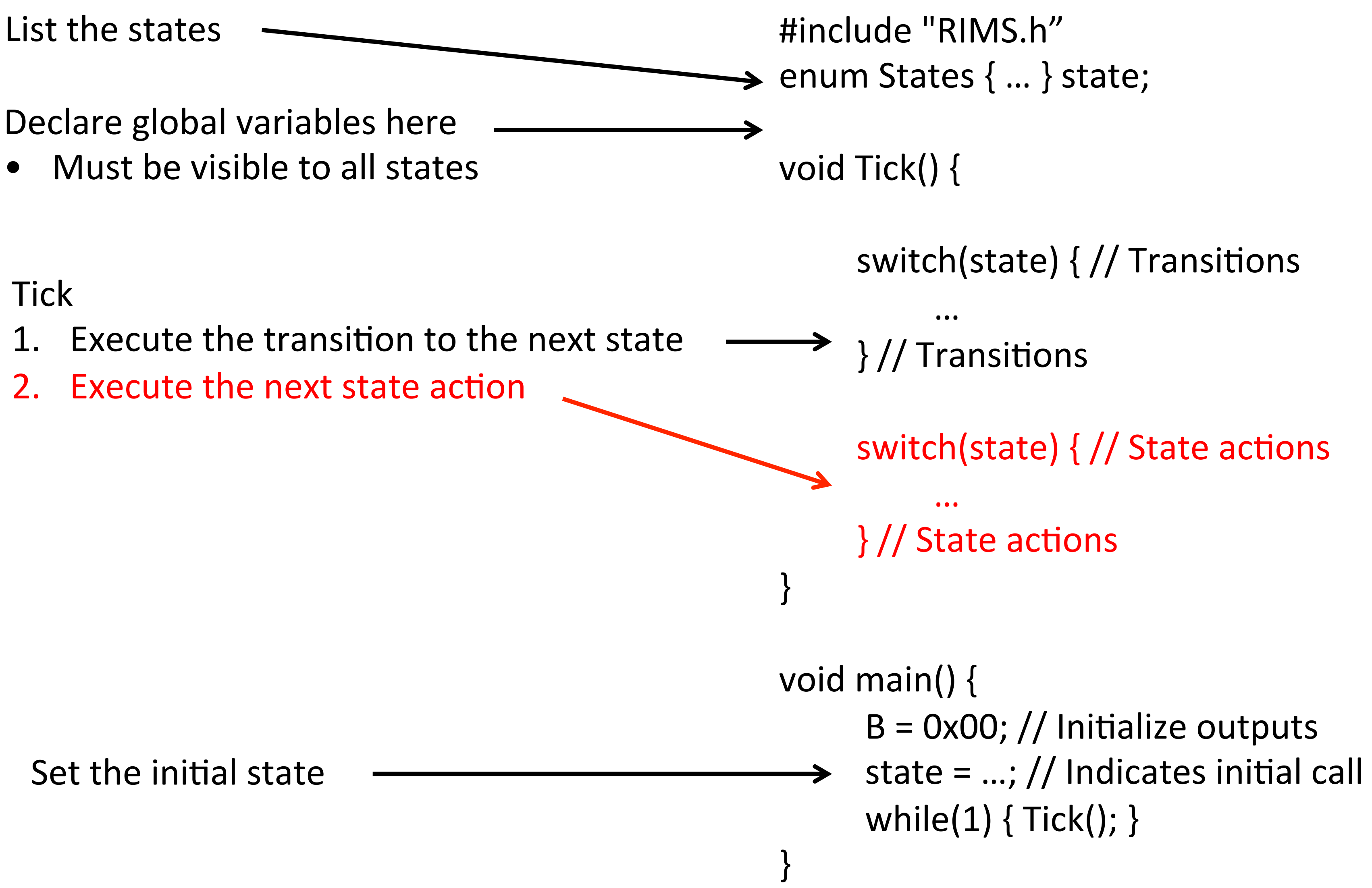
Declare global variables here
- Must be visible to all states

Tick
1. Execute the transition to the next state

Set the initial state

```
#include "RIMS.h”
enum States { … } state;

void Tick() {

        switch(state) { // Transitions
                …
        } // Transitions

        switch(state) { // State actions
                …
        } // State actions
}

void main() {
        B = 0x00; // Initialize outputs
        state = …; // Indicates initial call
        while(1) { Tick(); }
}
```

C Code SM Template



List the states

Declare global variables here

- Must be visible to all states

Tick

1. Execute the transition to the next state
2. Execute the next state action

Set the initial state

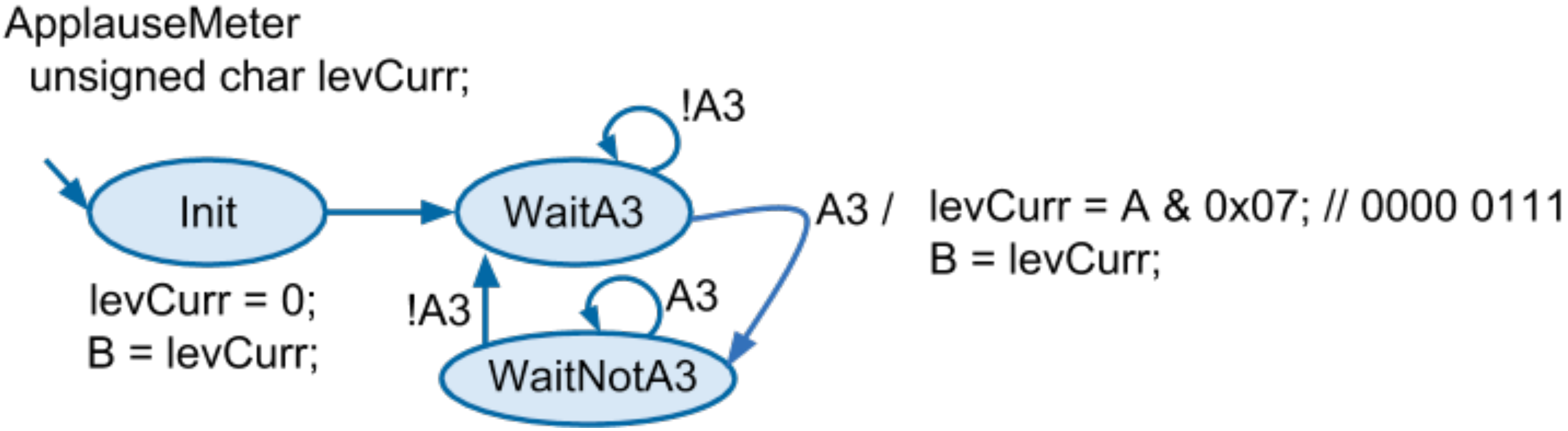
```
#include "RIMS.h”
enum States { … } state;

void Tick() {

        switch(state) { // Transitions
                …
        } // Transitions

        switch(state) { // State actions
                …
        } // State actions
}

void main() {
        B = 0x00; // Initialize outputs
        state = …; // Indicates initial call
        while(1) { Tick(); }
}
```

My Expectations of You

1. Memorize the template
2. Convert a RIBS/RIMS compatible SM to C

• Fill in the states	enum States { … } state;
• List SM variables	(global)
• Set initial state	state = …; (in “main”)
• Specify transitions	(in “Tick”)
• Specify state actions	(in “Tick”)

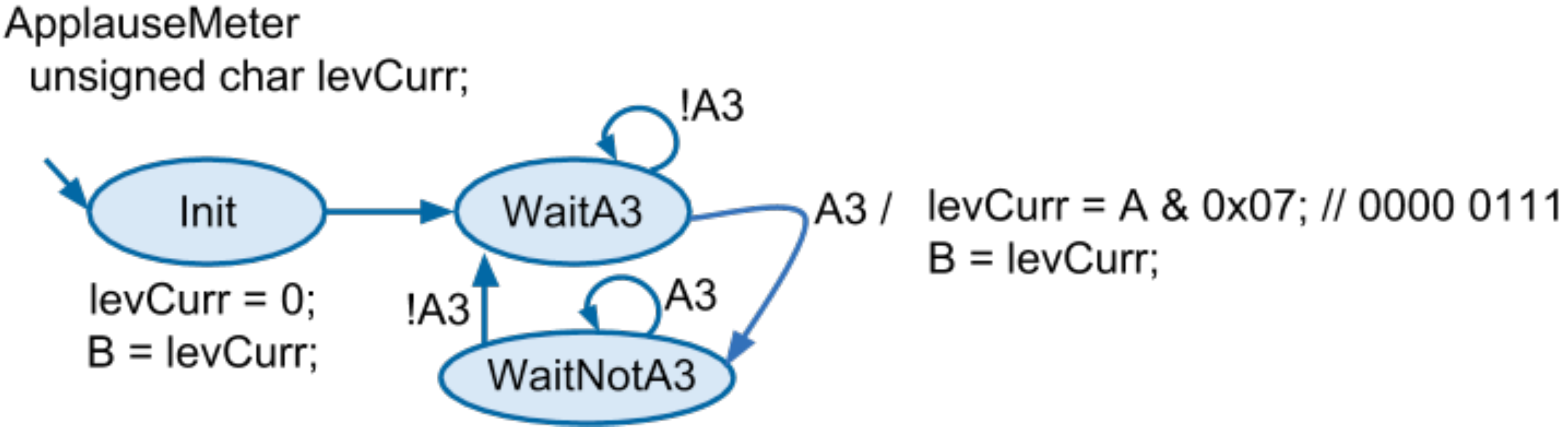
An Example to Get You Started


ApplauseMeter
unsigned char levCurr;

levCurr = 0;
B = levCurr;

levCurr = A & 0x07; // 0000 0111
B = levCurr;

Fill in the States

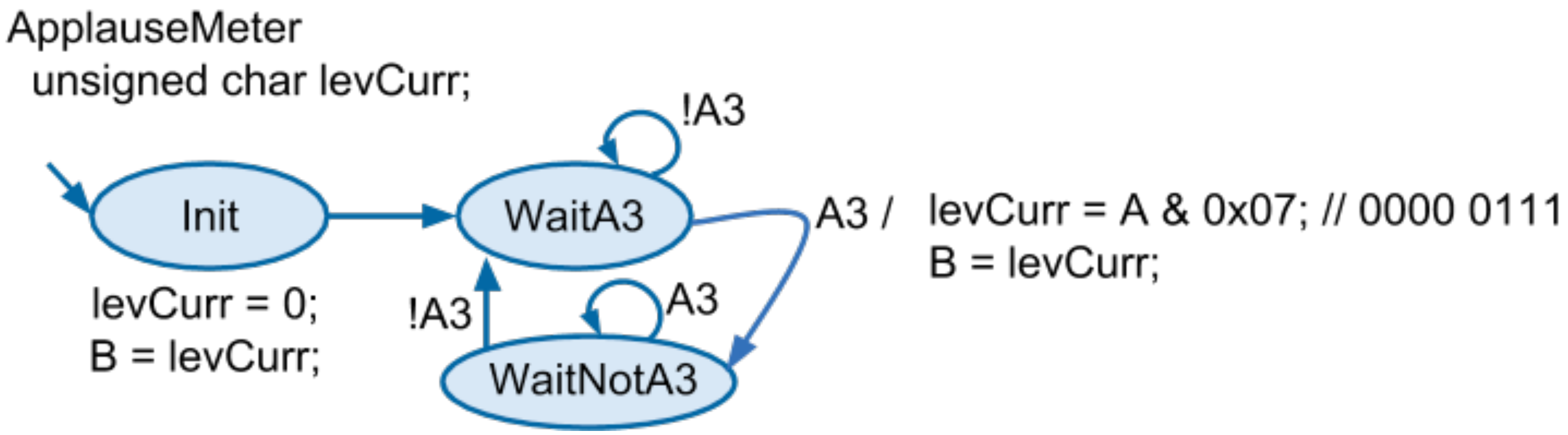


ApplauseMeter
  unsigned char levCurr;

levCurr = 0;
B = levCurr;

A3 / levCurr = A & 0x07; // 0000 0111
B = levCurr;

Fill in the States



ApplauseMeter
unsigned char levCurr;

Init

levCurr = 0;
B = levCurr;

WaitA3

!A3

A3 / levCurr = A & 0x07; // 0000 0111
B = levCurr;

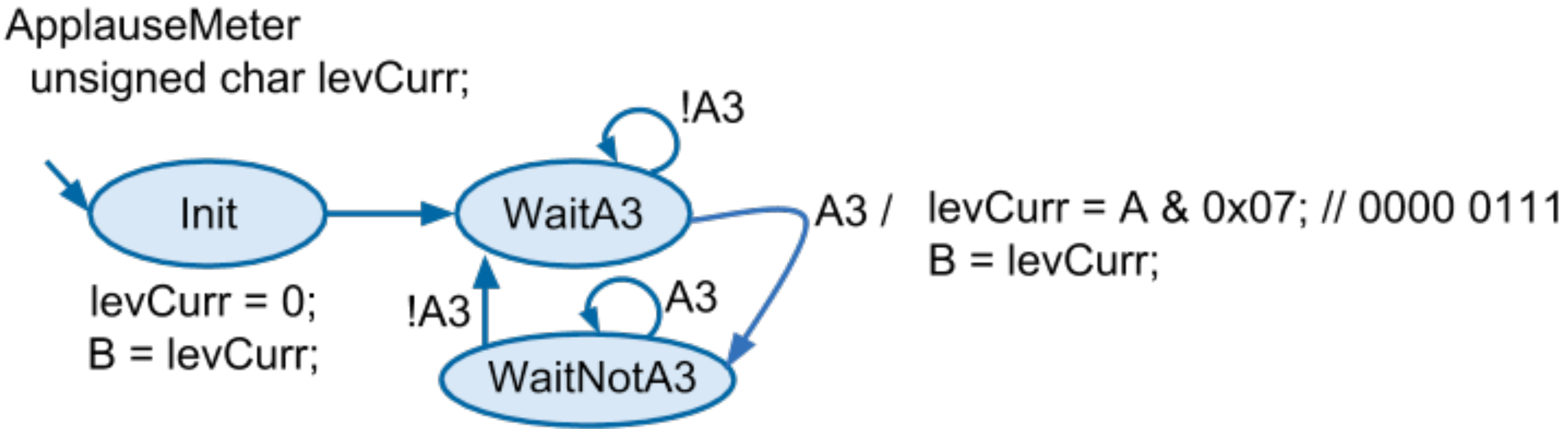
!A3

A3

WaitNotA3

enum States {   } state;

Fill in the States



```
unsigned char levCurr;
```

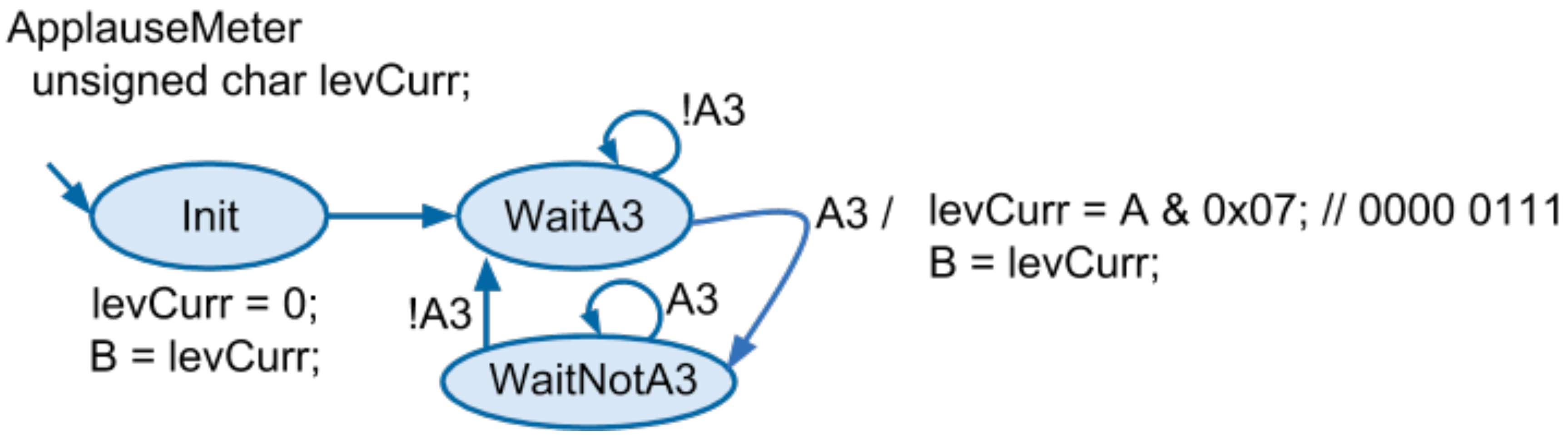
```
levCurr = 0;
B = levCurr;
```

```
levCurr = A & 0x07; // 0000 0111
B = levCurr;
```

enum States {Start, Init, WaitA3, WaitNotA3} state;

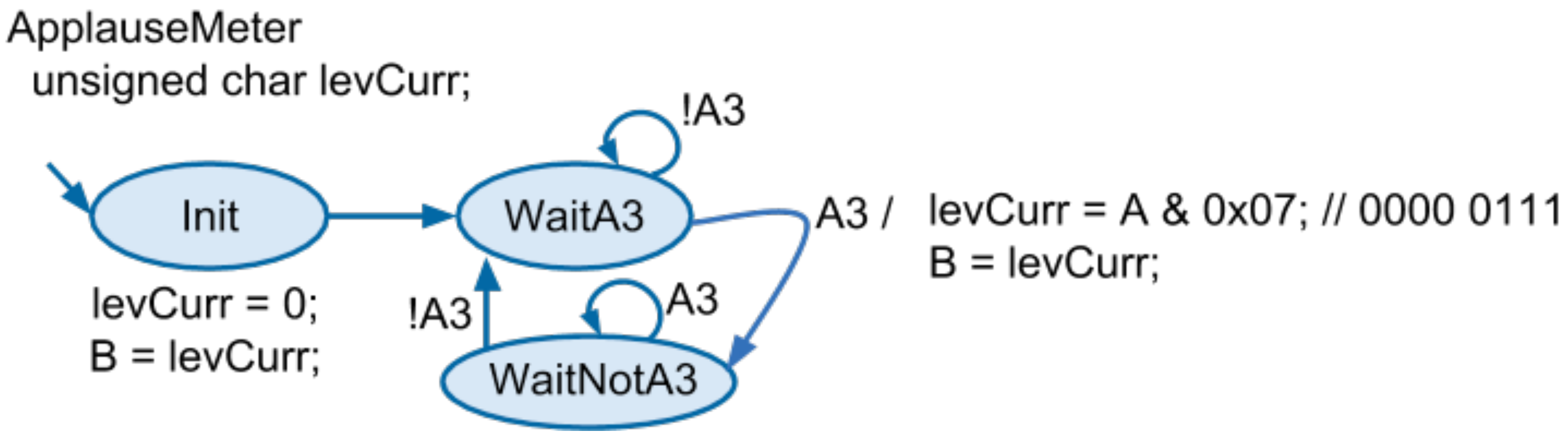
Extra state always added for initialization

List SM Variables



```
enum States {Start, Init, WaitA3, WaitNotA3} state;
```

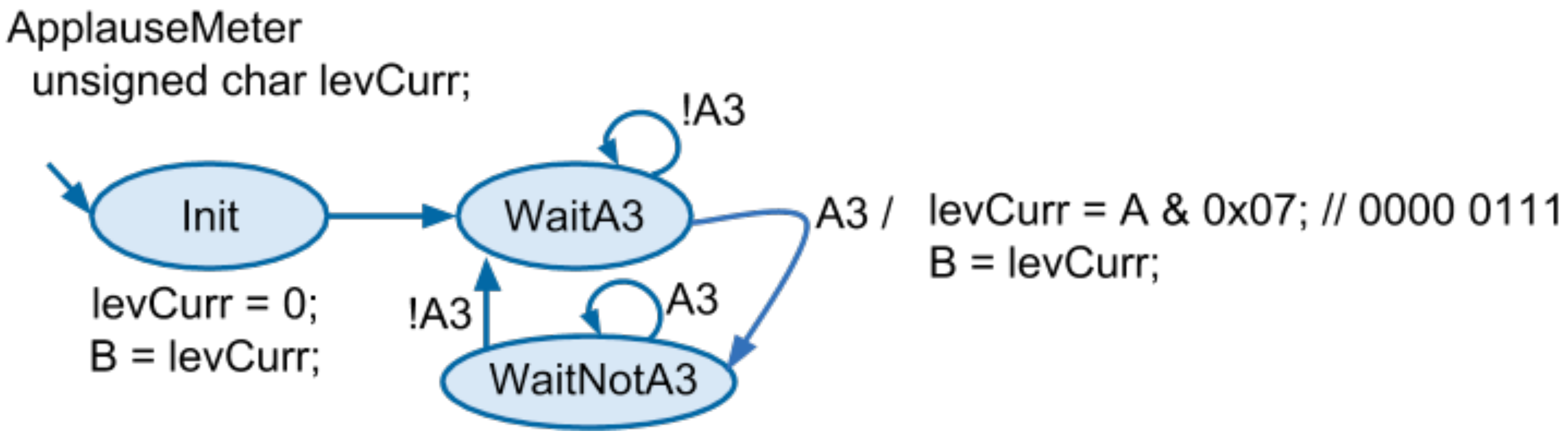
List SM Variables



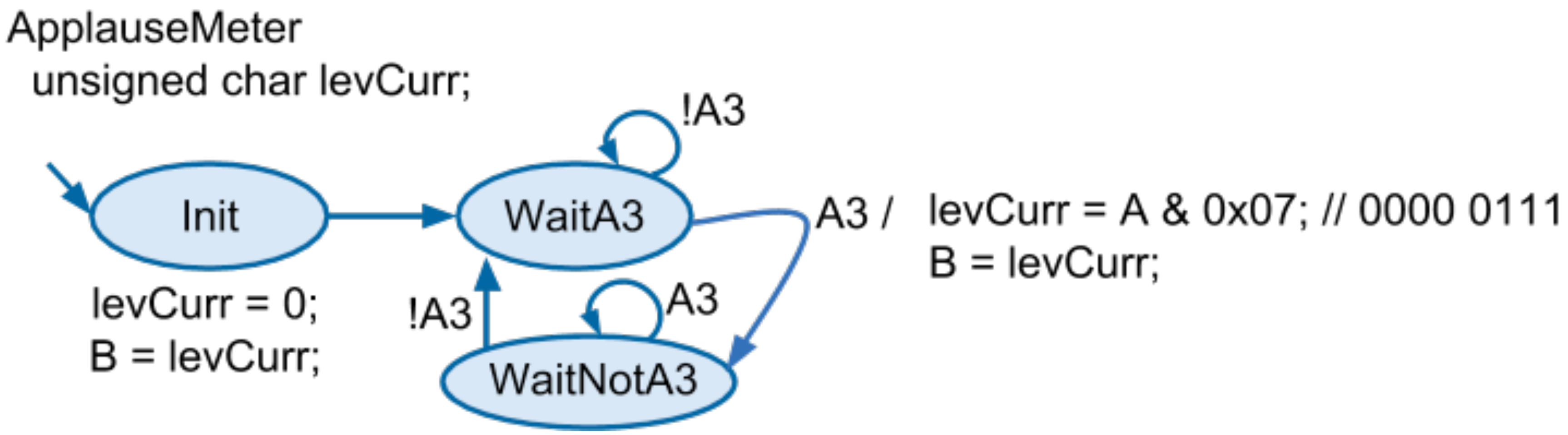
enum States {Start, Init, WaitA3, WaitNotA3} state;

unsigned char levCurr;

Set Initial State

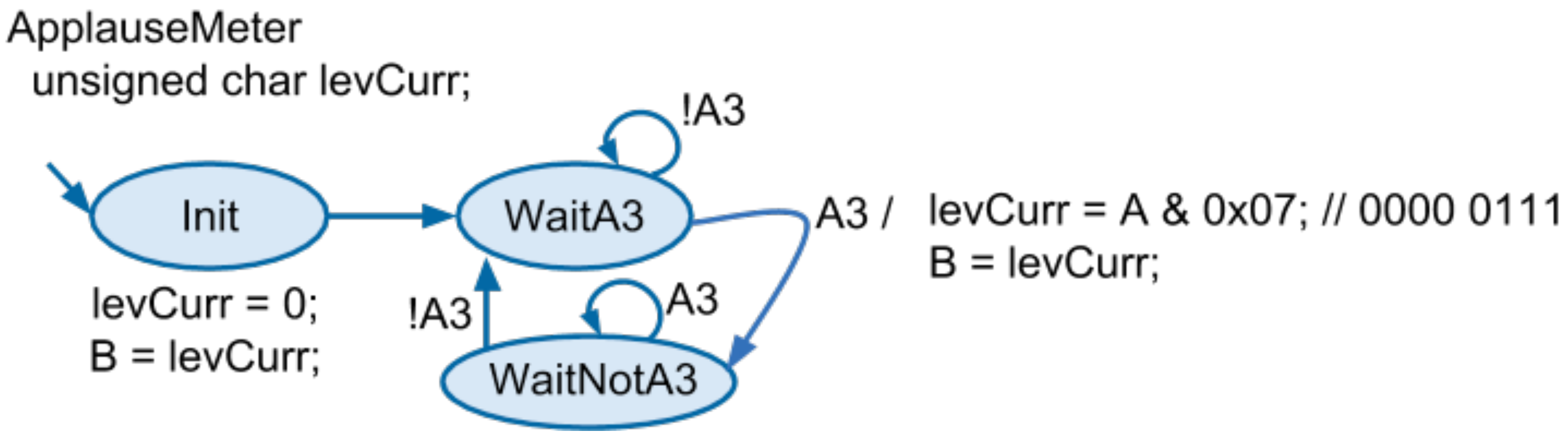


Set Initial State



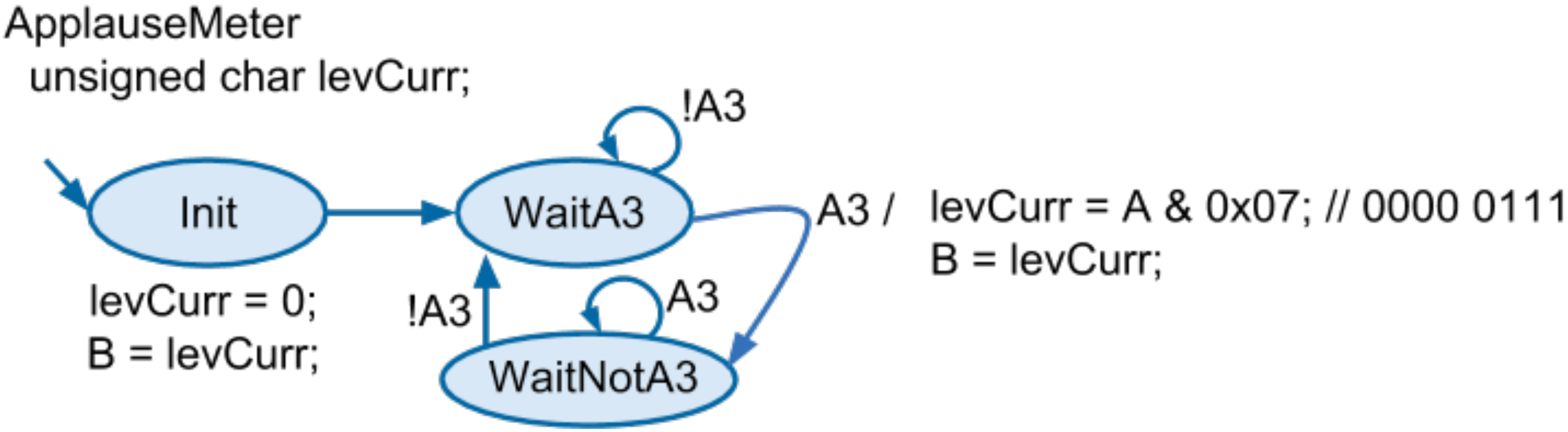
```
void main() {
     B = 0x00;
     state =           ;
     while(1) { Tick(); }
}
```

Set Initial State



```
void main() {
    B = 0x00;
    state = Start;
    while(1) { Tick(); }
}
```

Specify Transitions



ApplauseMeter
unsigned char levCurr;

Init

levCurr = 0;
B = levCurr;

WaitA3

!A3

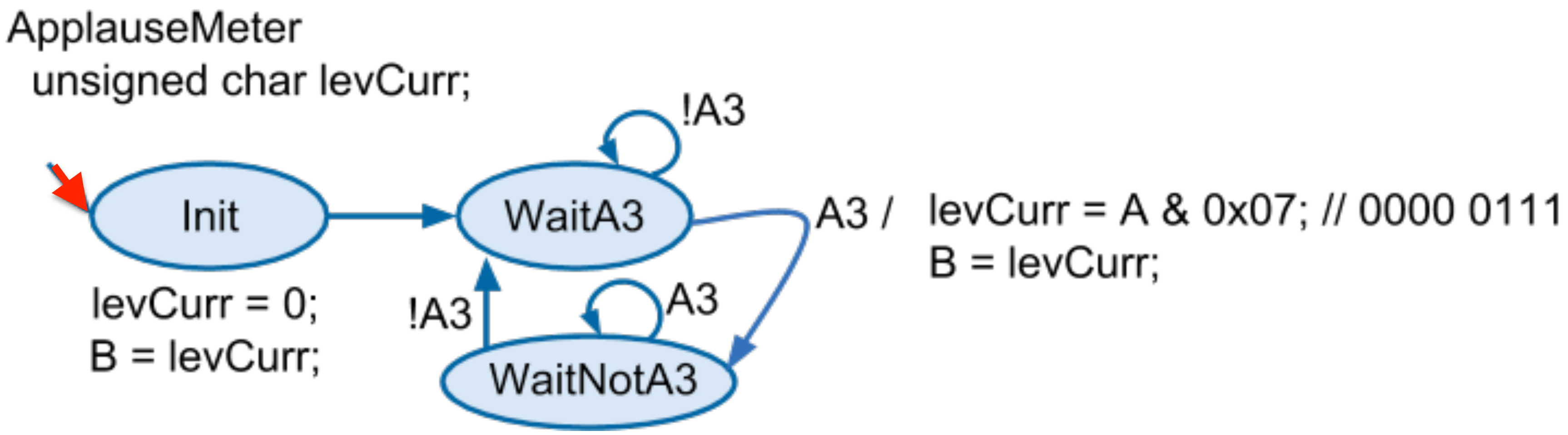
A3 / levCurr = A & 0x07; // 0000 0111
B = levCurr;

WaitNotA3

A3

!A3

Specify Transitions



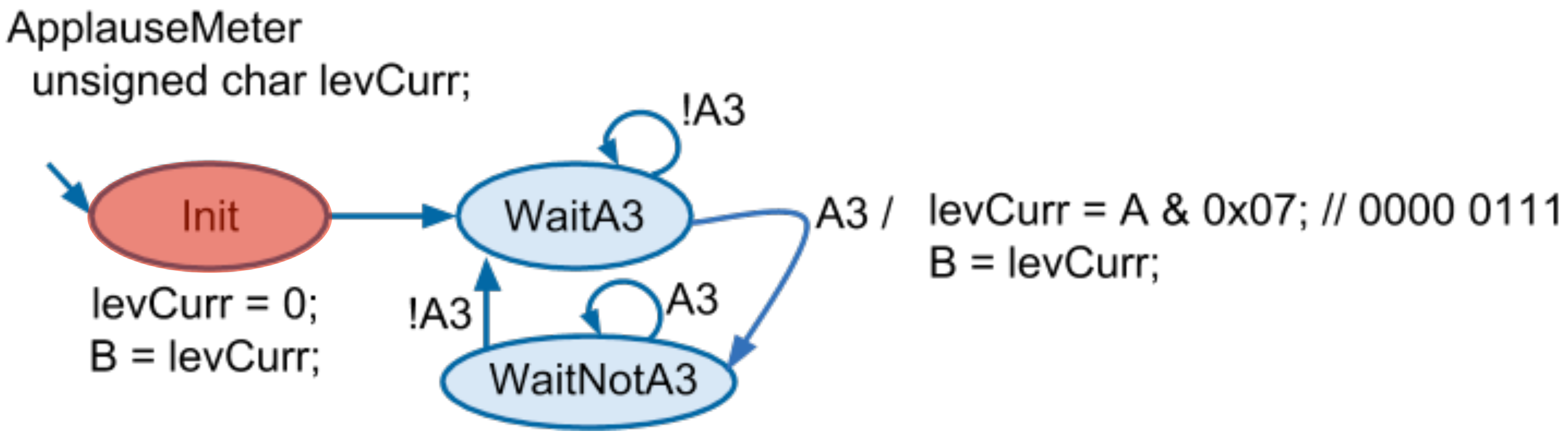
ApplauseMeter
unsigned char levCurr;

levCurr = 0;
B = levCurr;

levCurr = A & 0x07; // 0000 0111
B = levCurr;

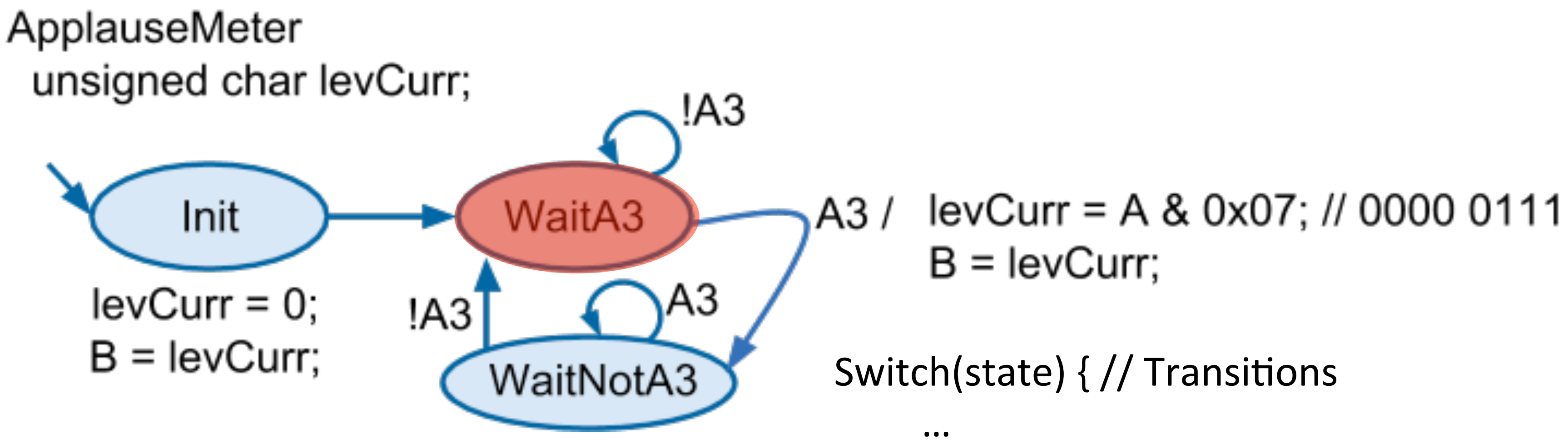
```
Switch(state) { // Transitions
    case Start:  // Initial transition
        state = Init;
        break;
    …
    default:
        state = Start;
        break;
 } // Transitions
```

Specify Transitions



```
Switch(state) { // Transitions
        …
        case Init:
                state = WaitA3;
                break;
        …
 } // Transitions
```

Specify Transitions



ApplauseMeter
unsigned char levCurr;

Init

levCurr = 0;
B = levCurr;

WaitA3

!A3

A3 / levCurr = A & 0x07; // 0000 0111
B = levCurr;

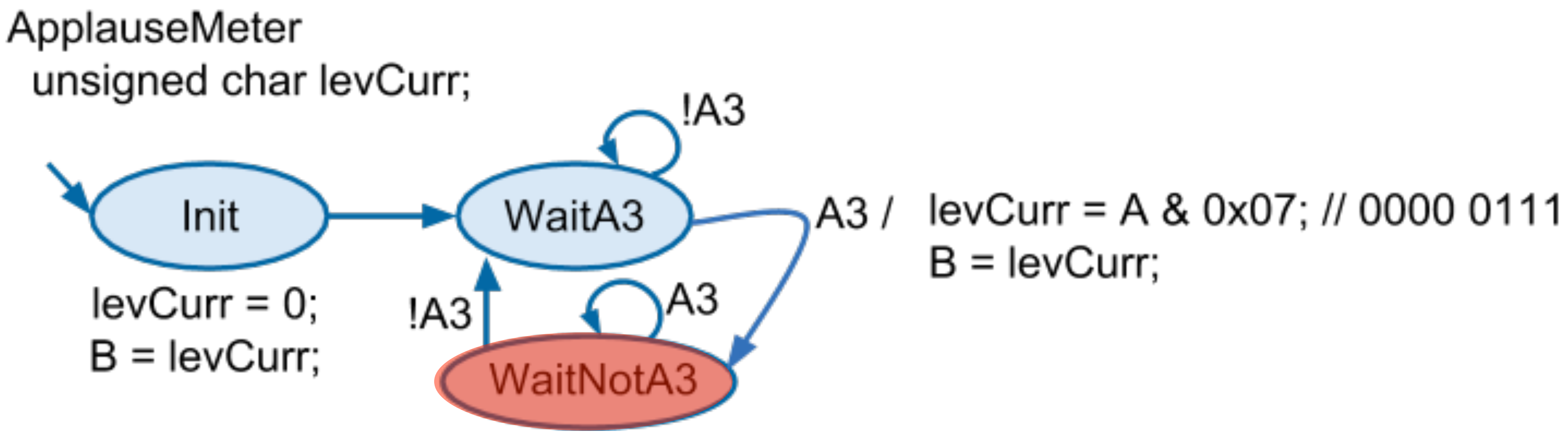
!A3

A3

WaitNotA3

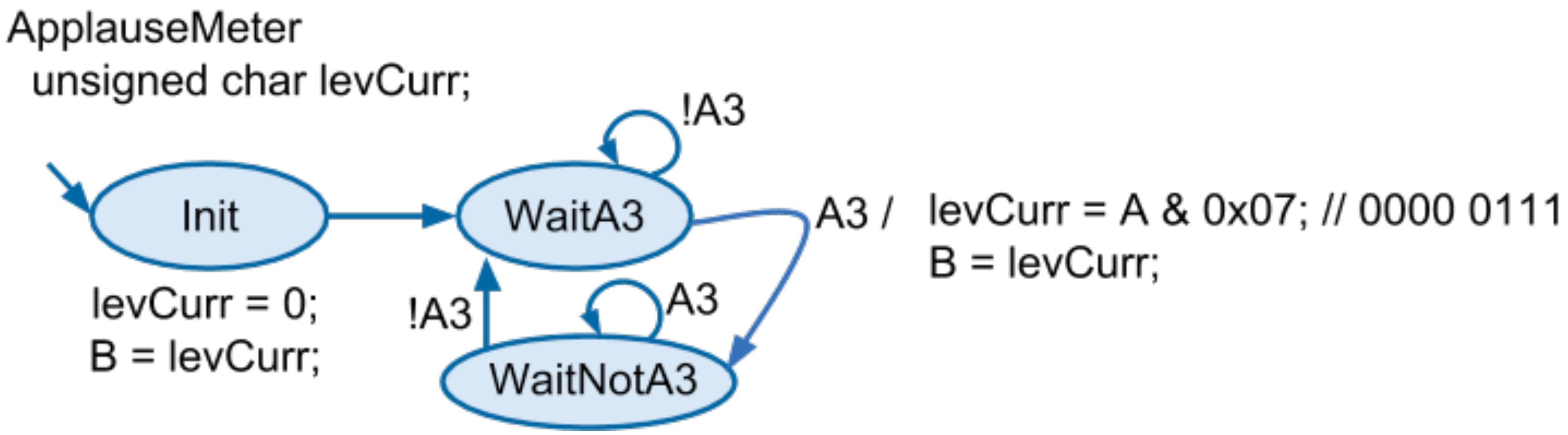
```
Switch(state) { // Transitions
        …
        case WaitA3:
                if(!A3) state = WaitA3;
                else {
                        state = WaitNotA3;
                        levCurr = A & 0x07;
                        B = levCurr;
                }
                break;
        …
  } // Transitions
```

Specify Transitions



```
Switch(state) { // Transitions
        …
        case WaitNotA3:
                state = A3 ? WaitNotA3 : WaitA3;
                break;
        …
 } // Transitions
```

Specify State Actions

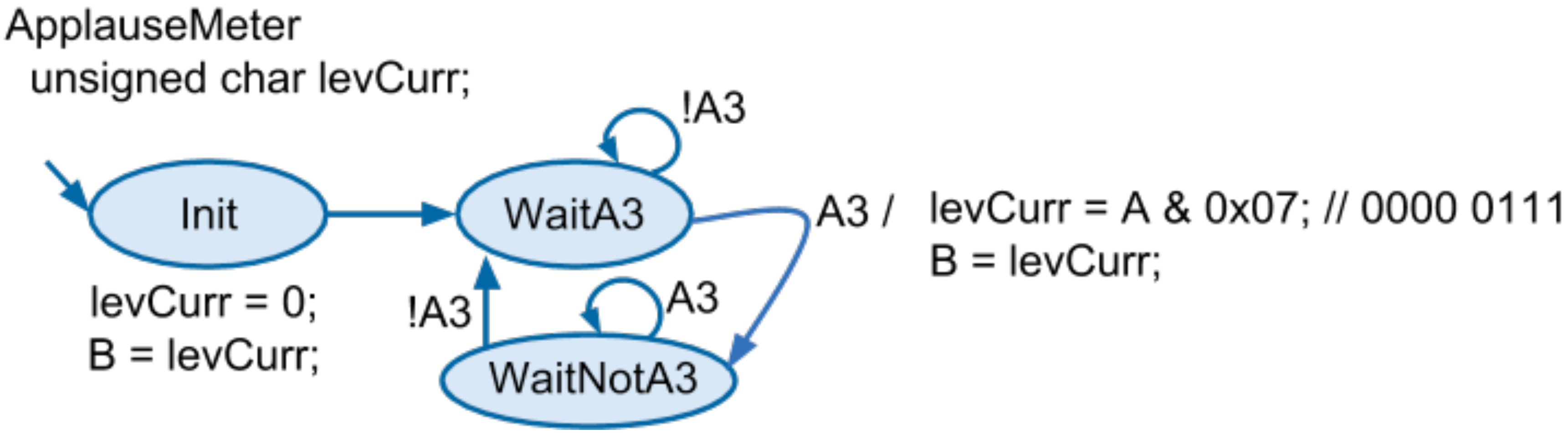


ApplauseMeter
unsigned char levCurr;

levCurr = 0;
B = levCurr;

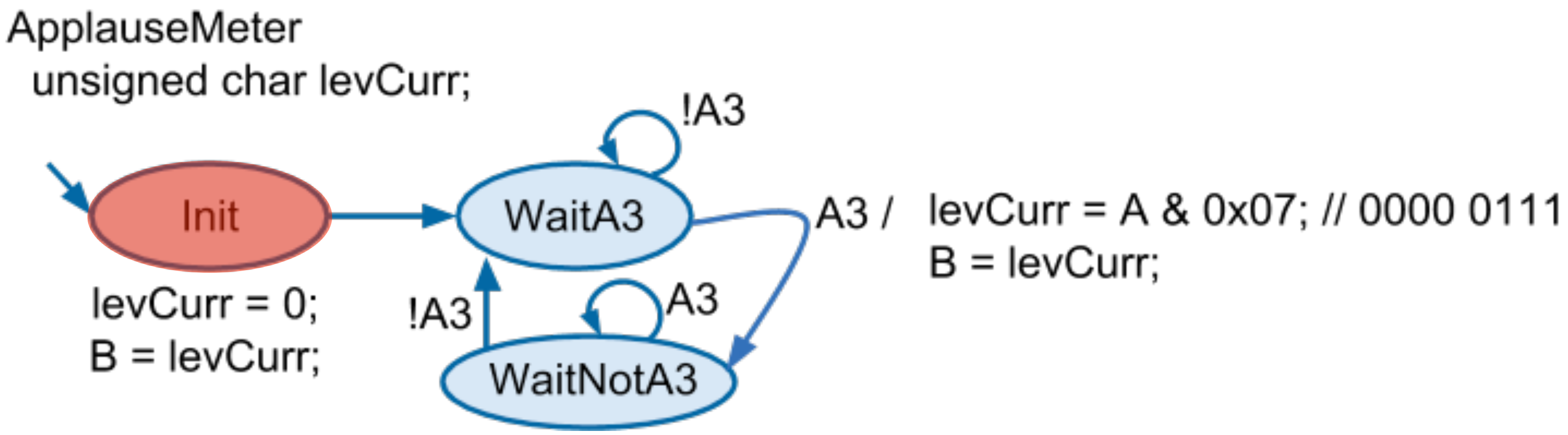
A3 / levCurr = A & 0x07; // 0000 0111
B = levCurr;

Specify State Actions



```
Switch(state) { // State actions
        ....
        default:
                break;
} // State actions
```

Specify State Actions

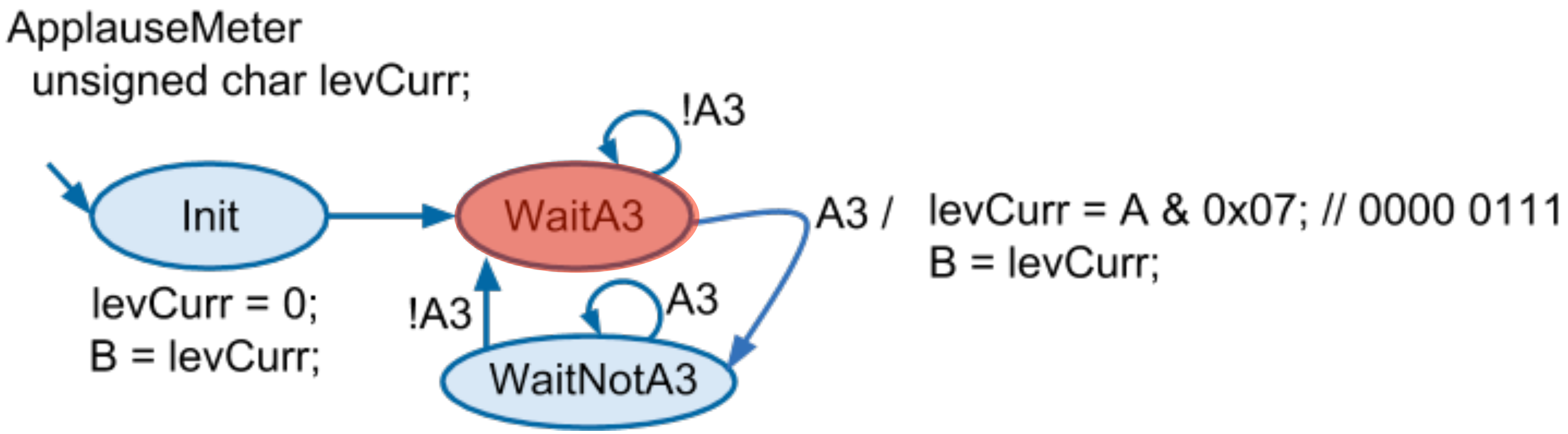

ApplauseMeter
unsigned char levCurr;

levCurr = 0;
B = levCurr;

levCurr = A & 0x07; // 0000 0111
B = levCurr;

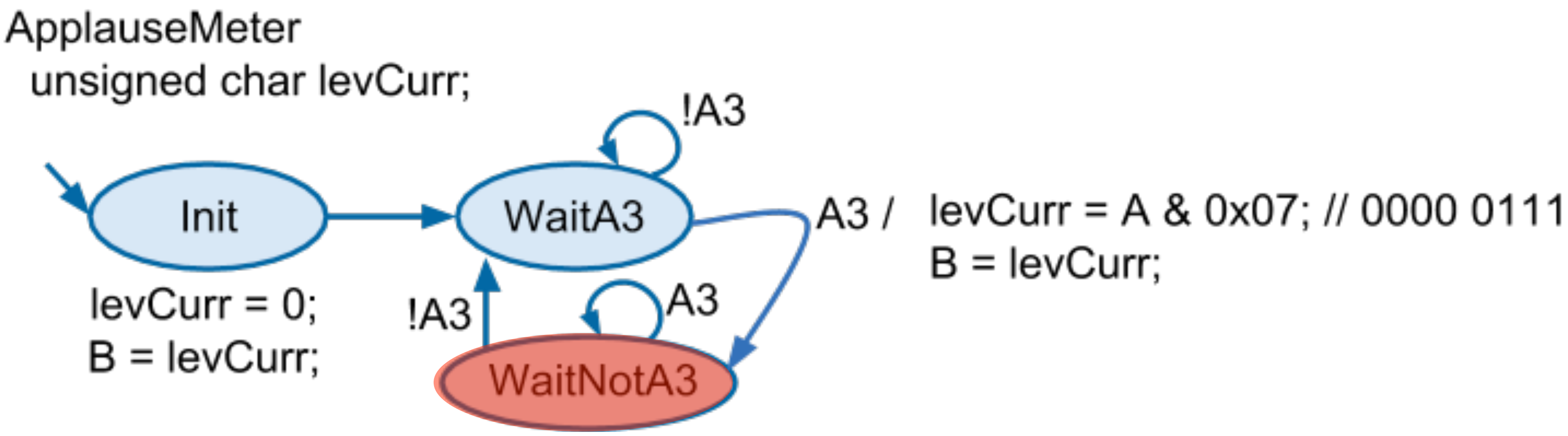
```
Switch(state) { // State actions
        ....
        case Init:
                levCurr = 0;
                B = levCurr;
                break;
        ....
} // State actions
```

Specify State Actions



```
Switch(state) { // State actions
        ....
        case WaitA3:
                break;
        ....
} // State actions
```

Specify State Actions



```
Switch(state) { // State actions
        ....
        case WaitNotA3:
                break;
        ....
} // State actions
```

Now, It’s Your Turn

- Three design examples for you to work out on paper
- Consult with your friends
- Work together if you want
- Ask questions
- Peer-grade one another