

Lecture 8 objectives

- linked data structures in ROM,
- Mealy and Moore finite state machines,
- fixed time delay using TCNT,
- adding output pins, adding input pins,
- running the FSM in the background using interrupts

2.4. Abstraction**2.4.1. Definitions.****Software abstraction**

define a complex problem with a set of basic abstract principles
 construct our software system using these building blocks
 better understanding of the problem
 separate **how we are getting it done** from **what we are doing**
 separate **mechanisms** from **policies**
 easier to optimize
 proof of correct function,
 simplifies both extensions and customization

A good example of abstraction is the **Finite State Machine (FSM)** implementations. The **abstract principles** of FSM development

inputs,
outputs,
states, and
state transitions.

If we can take a complex problem and map it into a FSM model, then we can solve it with a simple FSM software tools.

Finite State Machine (FSM.)

inputs, outputs, states, and state transitions
 state graph defines relationships of inputs and outputs

The three advantages of this abstraction are

- 1) it can be faster to develop
- 2) it is easier to debug (prove correct) and
- 3) it is easier to change

What is a state (system not changing)?

Description of current conditions
 Actions required to keep the system from changing

What is a state graph?

Graphical interconnection between states

What is a controller?

Software that inputs, outputs, changes state
 Accesses the state graph

What are the parts of a finite state machine?

Input sensors
 Output actuators
 Controller
 State graph

Moore

Output depends on state
Next state depends on input and current state
use Moore if the output means being in that state
need the output to be in that state
E.g., traffic light.

Mealy

Output depends on input and current state

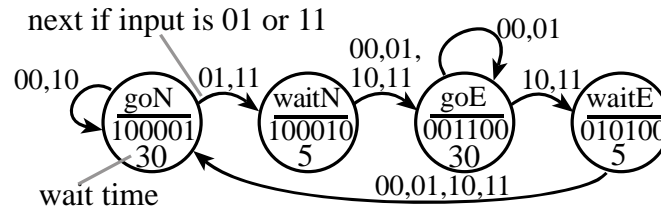
Next state depends on input and current state

use Mealy if the output causes the state to change

do not need a specific output to be in that state

E.g., four-legged robot

Moore Finite State Machine



```

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, 30, {GON, WAITN, GON, WAITN}},
    {0x22,  5, {GOE, GOE, GOE, GOE}},
    {0x0C, 30, {GOE, GOE, WAITE, WAITE}},
    {0x14,  5, {GON, GON, GON, GON}}};
void main(void){
    unsigned char n; // state number
    unsigned char Input;
    DDRT = 0xFF;
    DDRM &= ~0x03;
    n = GON;
    while(1){
        PTT = FSM[n].Out;
        Wait1sec(FSM[n].Time);
        Input = PTM&0x03;
        n = FSM[n].Next[Input];
    }
}

```

```

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, 30, {GON, WAITN, GON, WAITN}},
    {0x22,  5, {GOE, GOE, GOE, GOE}},
    {0x0C, 30, {GOE, GOE, WAITE, WAITE}},
    {0x14,  5, {GON, GON, GON, GON}}};
void main(void){
    STyp *Pt; // state pointer
    unsigned char Input;
    DDRT = 0xFF;
    DDRM &= ~0x03;
    Pt = GON;
    while(1){
        PTT = Pt->Out;
        Wait1sec(Pt->Time);
        Input = PTM&0x03;
        Pt = Pt->Next[Input];
    }
}

```

```

void Wait(unsigned short delay) {
    unsigned short startTime;
    startTime = TCNT;
    while((TCNT-startTime) <= delay){}
}
void InitOC3(void){
    TIOS |= 0x08; // OC3, but not armed for interrupts
    TSCR1 = 0x80; // activate TCNT
}
void Wait(unsigned short delay) {
    TFLG1 = 0x08; // clear C3F
    TC3 = TCNT+delay; // TC3 is TCNT at end of wait
    while((TFLG1&0x08)==0){}
}

```

separate **how we are getting it done** from **what we are doing**
 separate **mechanisms** from **policies**

Mealy Finite State Machine: Walking Robot

One input meaning (PM0)

- 0 stop
- 1 walk forward

There are four legs in this walking robot
 Each leg is controlled by four motors (PA7-0, PB7-0)

- back
- fwrđ
- up
- down

For example, if the back motor is active for 2 seconds, the leg moves back.

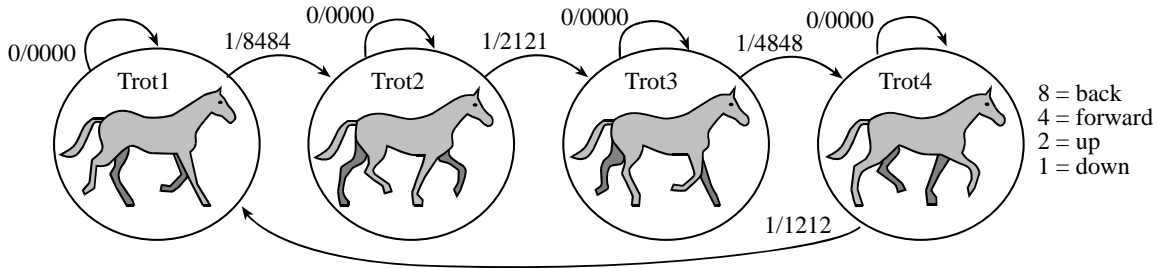
If no motors are active the leg remains stationary.

There are nine appropriate actions for each leg

- 0000 0 no change
- 1000 8 move leg back
- 0100 4 move leg forward
- 0010 2 lift leg up
- 0001 1 set leg down
- 0110 6 move leg forward and lift up
- 0101 5 move leg forward and set down
- 1010 A move leg back and lift up
- 1001 9 move leg back and set down

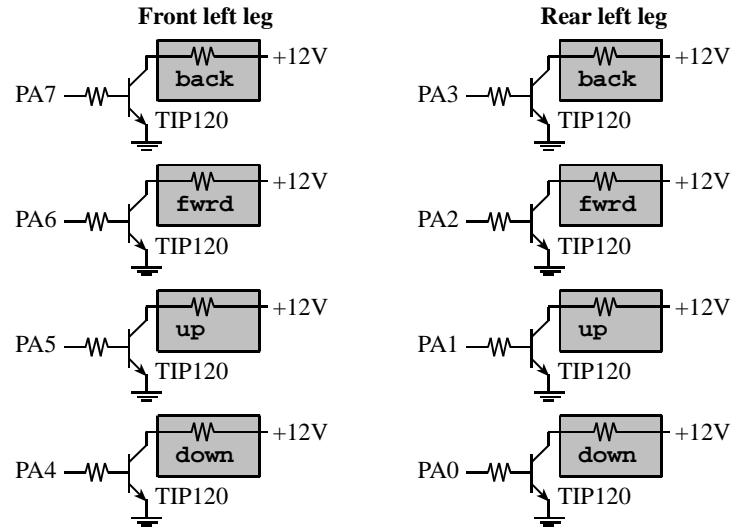
(hexadecimal codes 8 4 2 1 are used as a shorthand in the FSM)

This is a Mealy machine because
 the state defines the position of the robot, and
 the outputs are used to change the position



Run FSM in background using Output compare

```
const struct State {
    unsigned short Out[2]; // 16-bit PORTAB
    unsigned short Time;
    const struct State *Next[2];};
typedef const struct State sTyp;
#define Trot1 &FSM[0]
#define Trot2 &FSM[1]
#define Trot3 &FSM[2]
#define Trot4 &FSM[3]
STyp FSM[4]={
    {{0,0x8484},20,{Trot1,Trot2}},
    {{0,0x2121},20,{Trot2,Trot3}},
    {{0,0x4848},20,{Trot3,Trot4}},
    {{0,0x1212},20,{Trot4,Trot1}}};
```



```

STyp *Pt; // state pointer
unsigned short Count;
void FSMinit(void){
    asm sei
    TIOS |= 0x08; // activate TC3 as output compare
    TSCR1 = 0x80; // Enable TCNT, E clock 24MHz
    TSCR2 = 0x07; // divide by 128 TCNT, TOI disarm
    PACTL = 0; // timer prescale used for TCNT
    TIE |= 0x08; // arm OC3
    TC3 = TCNT+50; // first interrupt right away
    DDRAB = 0xFFFF; // 16-bit output port
    DDRM &= ~0x01;
    Pt = Trot1;
    PORTAB = 0;
    Count = 10; // starts in 1 sec
    asm cli
}
interrupt 11 void TC3handler(void){
// executes at 24000000/128/18750 = 10 Hz
unsigned char Input;
    TFLG1 = 0x08; // acknowledge OC3
    TC3 = TC3+18750; // every 100 ms
    Count--;
    if(Count==0){
        Input = PTM&0x01; // 1) Input
        PORTAB = Pt->Out[Input]; // 2) Output
        Count = Pt->Time; // 3) Wait
        Pt = Pt->Next[Input]; // 4) Next
    }
}

```

Mealy Finite State Machine

One input function called feelings() that returns my mood

```

0 OK
1 tired
2 curious
3 anxious

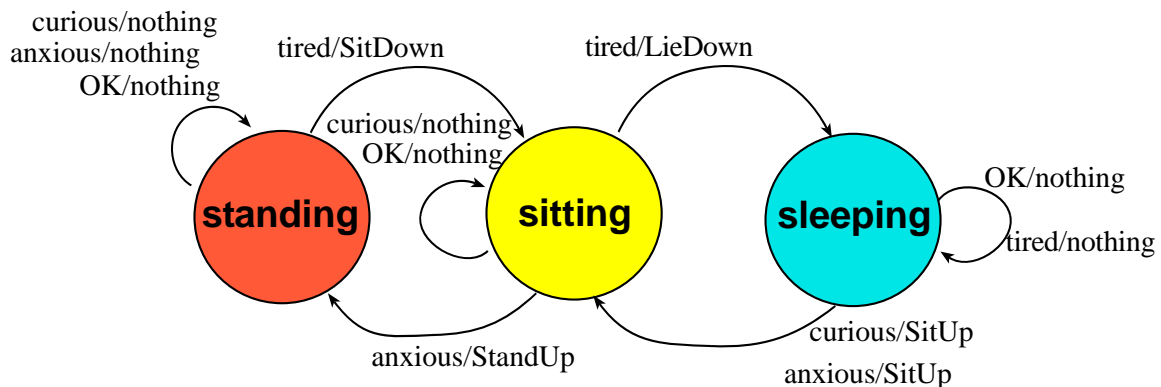
```

Five output functions that cause actions to occur

```

nothing()
StandUp()
SitDown()
SitUp()
LieDown()

```



```
const struct State{
```

```

    void (*CmdPt)[4](void);    // functions to execute
    const struct State *Next[4]; // Next if 0,1
};
typedef const struct State StateType;
#define STANDING &fsm[0]
#define SITTING &fsm[1]
#define SLEEPING &fsm[2]
StateType FSM[3]={
    {{&nothing,&SitDown,&nothing,&nothing}}, // STAND
    {{STANDING,SITTING, STANDING,STANDING}},
    {{&nothing,&LieDown,&nothing,&StandUp}}, // SIT
    {{SITTING,SLEEPING,SITTING, STANDING}},
    {{&nothing,&nothing,&SitUp, &SitUp}}, // SLEEP
    {{SLEEPING,SLEEPING,SITTING, SITTING}}
};
void main(void){
    StatePtr *Pt; /* Current State */
    unsigned char Input;
    Pt = STANDING; // Initial State
    while(1){
        Input = feelings(); // 1) Input=0 1,2,or 3
        (*Pt->CmdPt[Input])(); // 2) execute function
        Pt = Pt->Next[Input]; // 3) Move to next state
    }
}

```

Add walk light and walk button to traffic light

```

const struct State {
    unsigned char Out; // 7 bits
    unsigned short Time;
    const struct State *Next[8]; // 3 inputs => 8 next states
}
typedef const struct State STyp;
#define goN &fsm[0]
#define waitN &fsm[1]
#define goE &fsm[2]
#define waitE &fsm[3]
#define walk &fsm[4]
#define flash0 &fsm[5]
#define flash1 &fsm[6]
#define flash2 &fsm[7]
STyp fsm[7]={
    {0x21, 30, {waitN,waitN,waitN,waitN,goN,waitN,goN,waitN}},
    {0x22, 5, {walk,walk,walk,walk,goE,goE,goE,goE}},
    {0x0C, 30, {waitE,waitE,waitE,waitE,goE,goE,waitE,waitE}},
    {0x14, 5, {walk,walk,walk,walk,goN,goN,goN,goN}},
    {0x64, 10, {walk,walk,walk,walk,flash0,flash0,flash0,flash0}},
    {0x24, 2, {flash1,flash1,flash1,flash1,flash1,flash1,flash1,flash1}},
    {0x64, 2, {flash2,flash2,flash2,flash2,flash2,flash2,flash2,flash2}},
    {0x24, 2, {goN,goE,goN,goN,goN,goE,goN,goN}}};
void main(void){
    STyp *Pt; // state pointer
    unsigned char Input;
    DDRT = 0xFF;
    DDRM &= ~0x07;
    Pt = goN;
    while(1){
        PTT = Pt->Out;
        Wait1sec(Pt->Time);
        Input = PTM&0x07;
        Pt = Pt->Next[Input];
    }
}

```