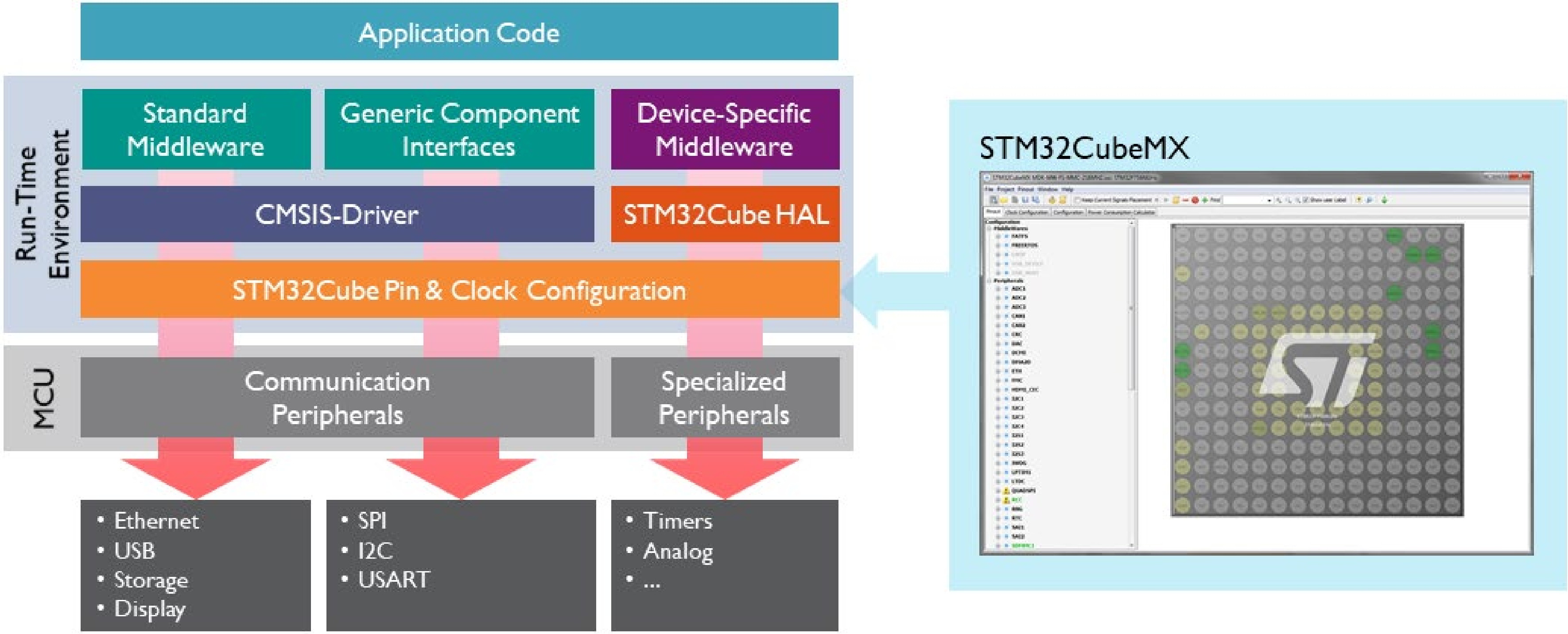
Microcomputer Engineering TMIK13 Lecture 3

FINITE STATE MACHINES (FSM)

ANDREAS AXELSSON (ANDREAS.AXELSSON@JU.SE)

Microchip – Abstraction Layers



Application Code

Run-Time Environment

Standard Middleware

Generic Component Interfaces

Device-Specific Middleware

CMSIS-Driver

STM32Cube HAL

STM32Cube Pin & Clock Configuration

MCU

Communication Peripherals

Specialized Peripherals

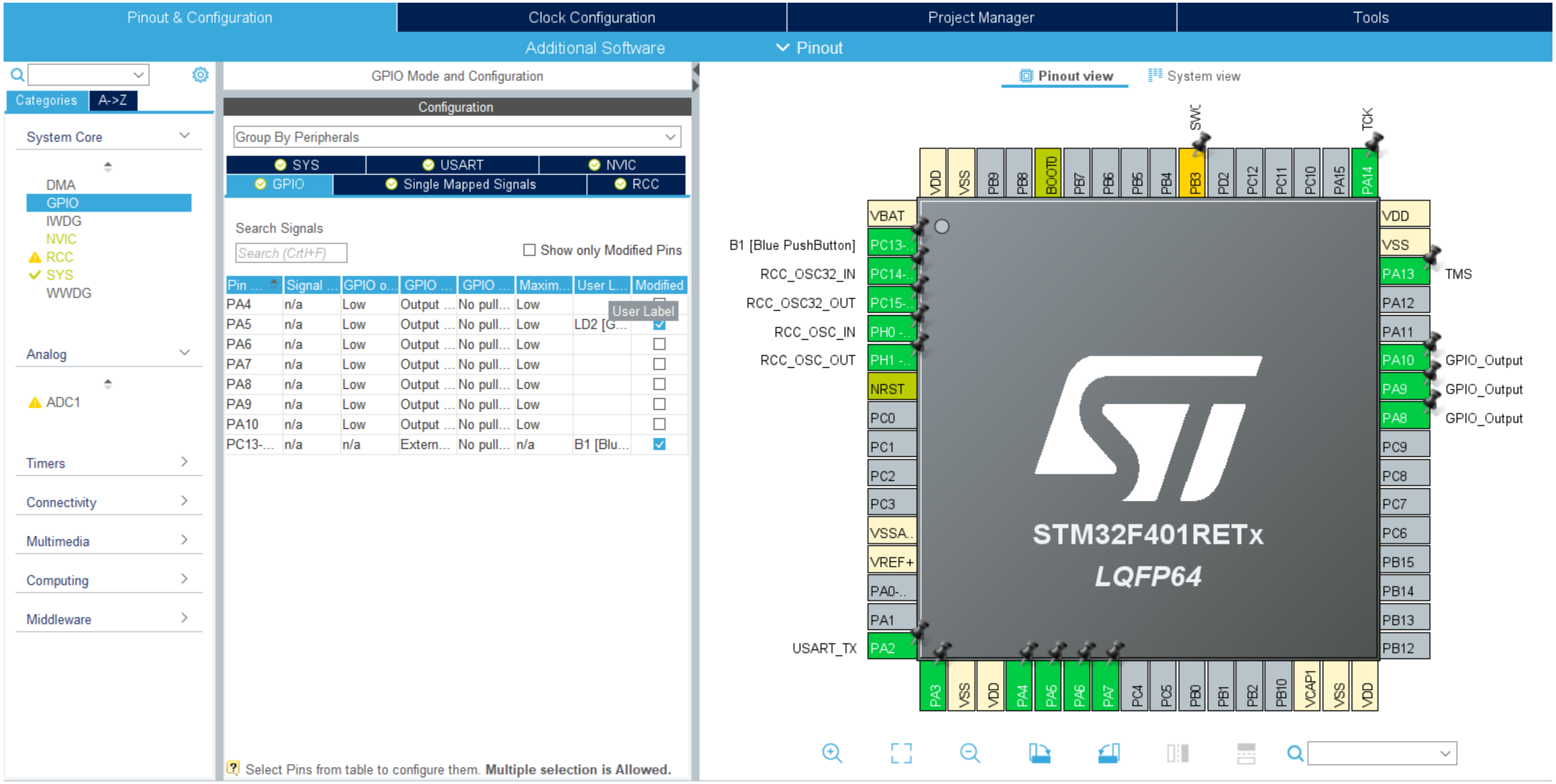
- Ethernet
- USB
- Storage
- Display

- SPI
- I2C
- USART

- Timers
- Analog
- ...

STM32CubeMX

STM32CubeMX – Peripherals



GPIO Mode and Configuration

Configuration

Group By Peripherals

SYS | USART | NVIC | GPIO | Single Mapped Signals | RCC

Search Signals

Search (Crtl+F)

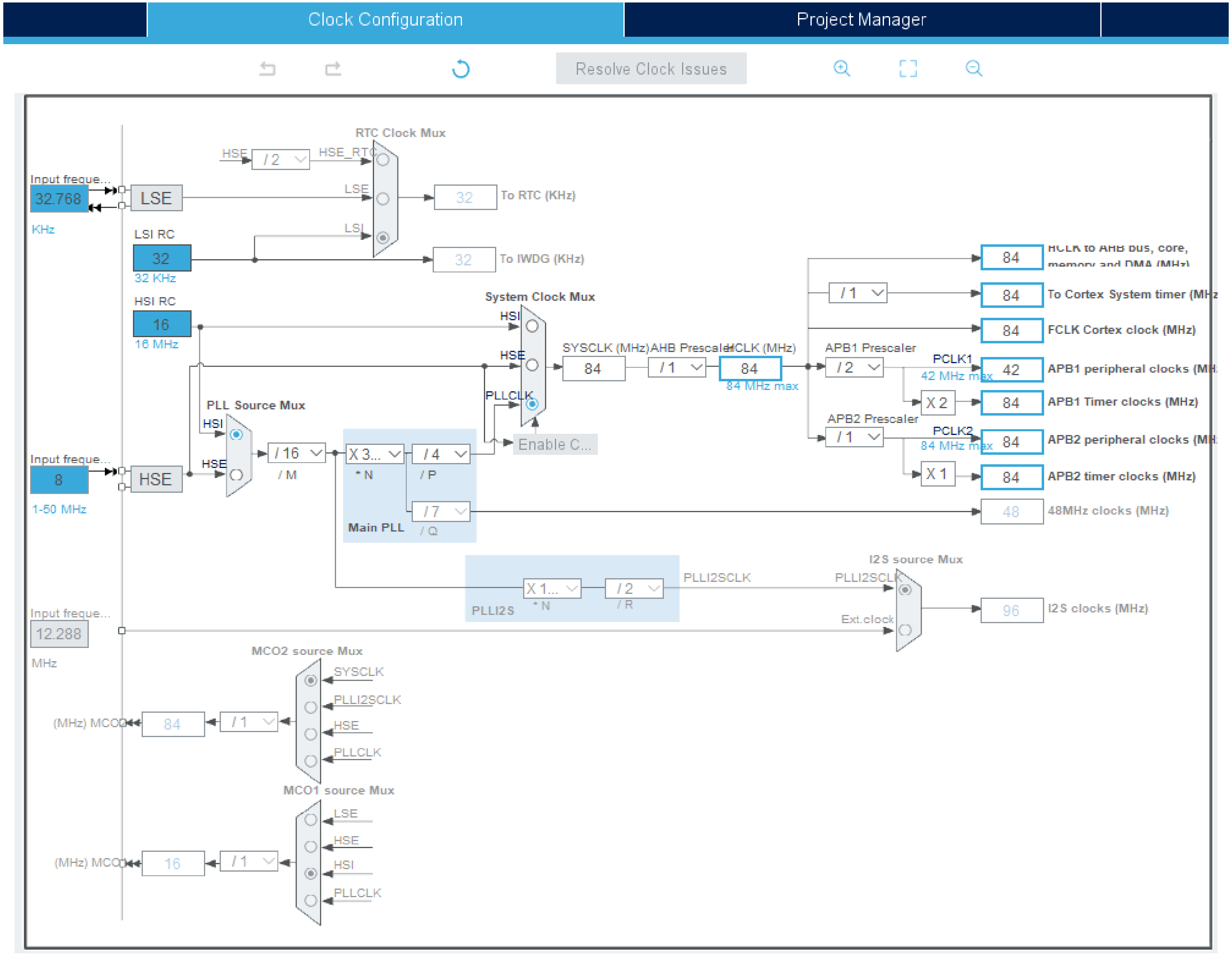
☐ Show only Modified Pins

Pin ...	Signal ...	GPIO o...	GPIO ...	GPIO ...	Maxim...	User L...	Modified
PA4	n/a	Low	Output ...	No pull...	Low		☐
PA5	n/a	Low	Output ...	No pull...	Low	LD2 [G...	☑
PA6	n/a	Low	Output ...	No pull...	Low		☐
PA7	n/a	Low	Output ...	No pull...	Low		☐
PA8	n/a	Low	Output ...	No pull...	Low		☐
PA9	n/a	Low	Output ...	No pull...	Low		☐
PA10	n/a	Low	Output ...	No pull...	Low		☐
PC13-...	n/a	n/a	Extern...	No pull...	n/a	B1 [Blu...	☑

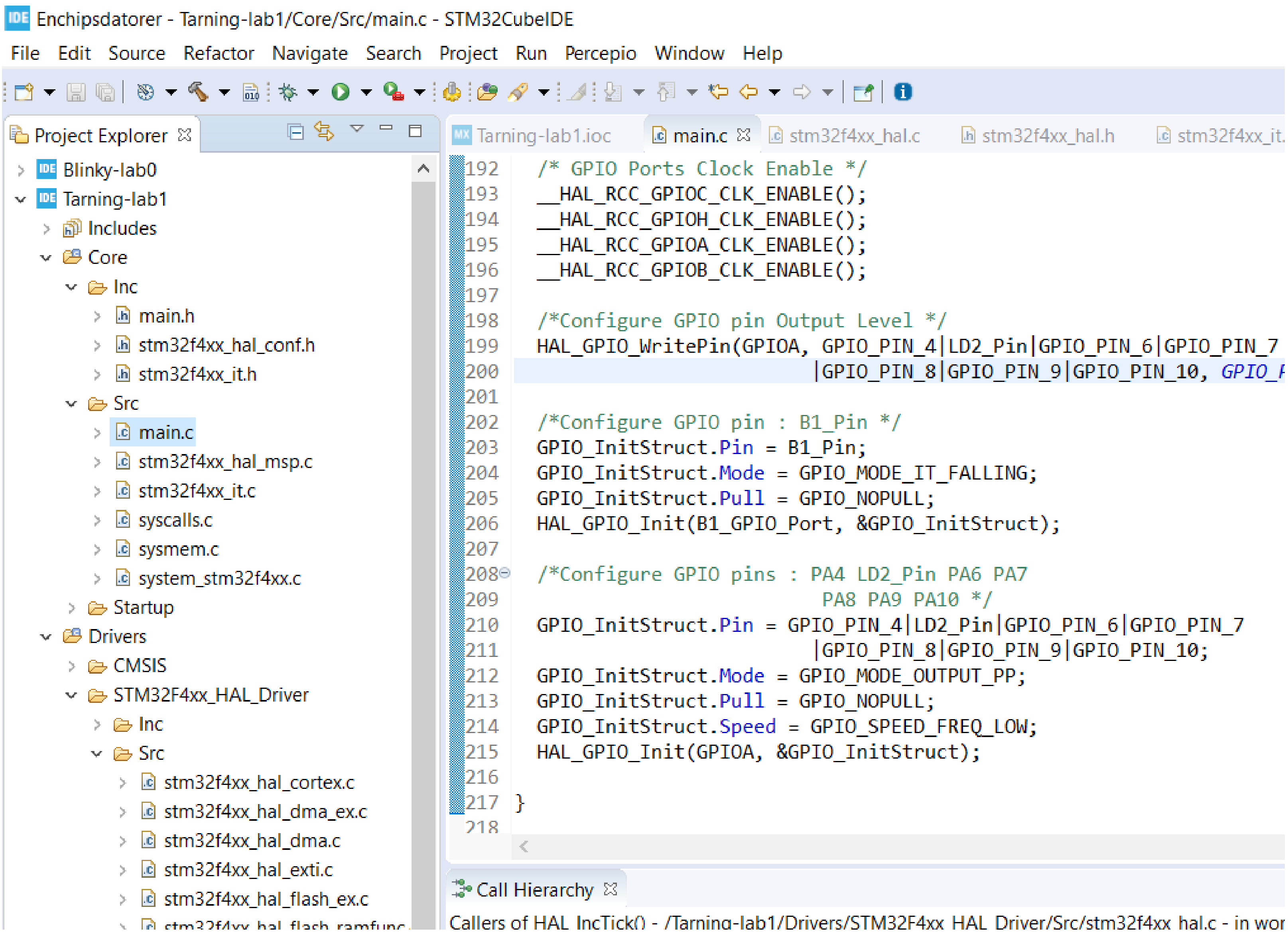
User Label

Select Pins from table to configure them. **Multiple selection is Allowed.**

STM32CubeMX – Clock Config



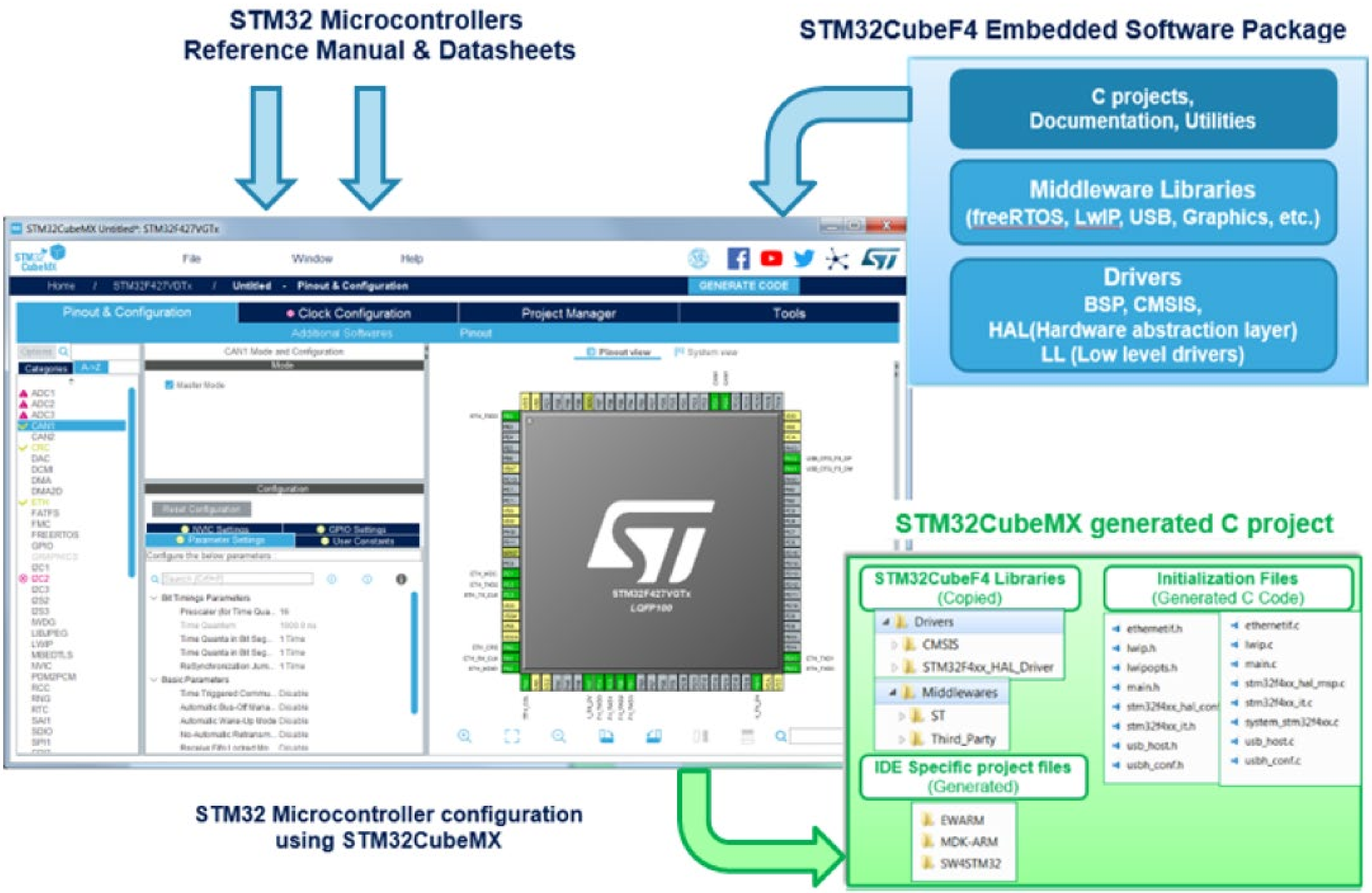
STM32CubeIDE – Generated Code



```c
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4|LD2_Pin|GPIO_PIN_6|GPIO_PIN_7
                          |GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10, GPIO_P

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PA4 LD2_Pin PA6 PA7
                           PA8 PA9 PA10 */
  GPIO_InitStruct.Pin = GPIO_PIN_4|LD2_Pin|GPIO_PIN_6|GPIO_PIN_7
                          |GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}
```

Callers of HAL_IncTick() - /Tarning-lab1/Drivers/STM32F4xx_HAL_Driver/Src/stm32f4xx_hal.c - in wor

STM32CubeMX



STM32 Microcontrollers
Reference Manual & Datasheets

STM32CubeF4 Embedded Software Package

C projects,
Documentation, Utilities

Middleware Libraries
(freeRTOS, LwIP, USB, Graphics, etc.)

Drivers
BSP, CMSIS,
HAL(Hardware abstraction layer)
LL (Low level drivers)

STM32CubeMX generated C project

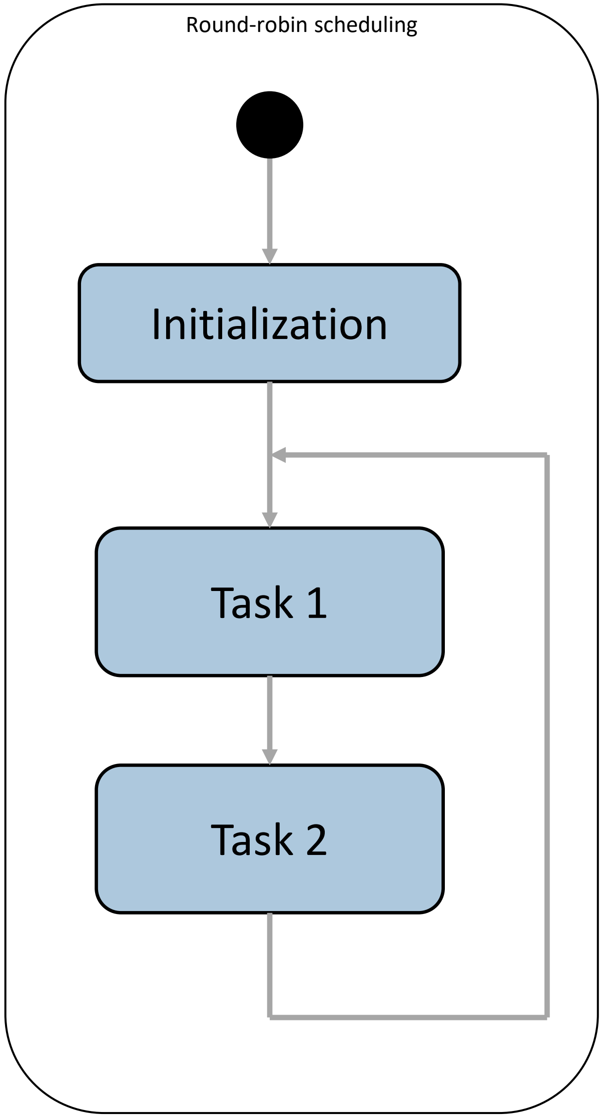
STM32CubeF4 Libraries
(Copied)

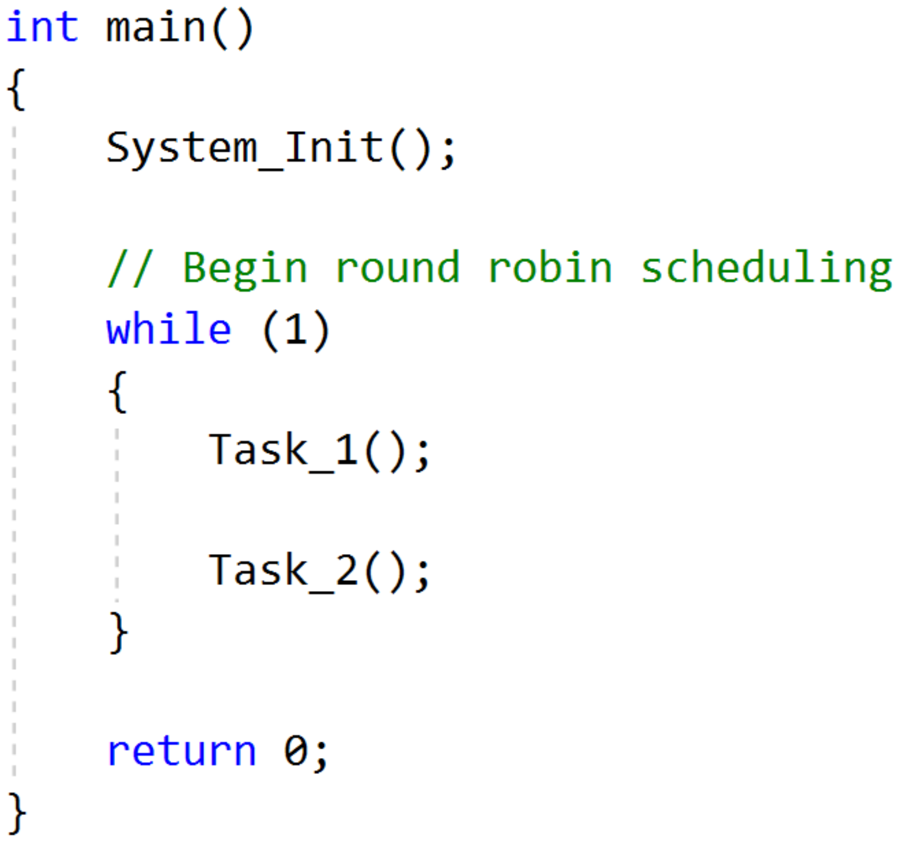
Initialization Files
(Generated C Code)

IDE Specific project files
(Generated)

STM32 Microcontroller configuration
using STM32CubeMX

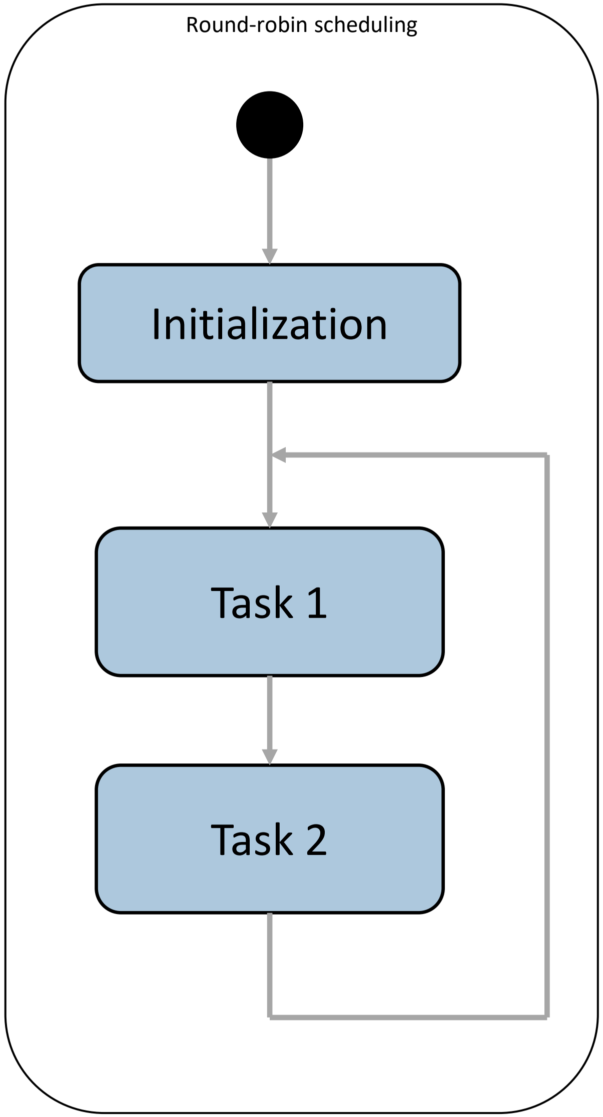
Simple Scheduling – Super Loop

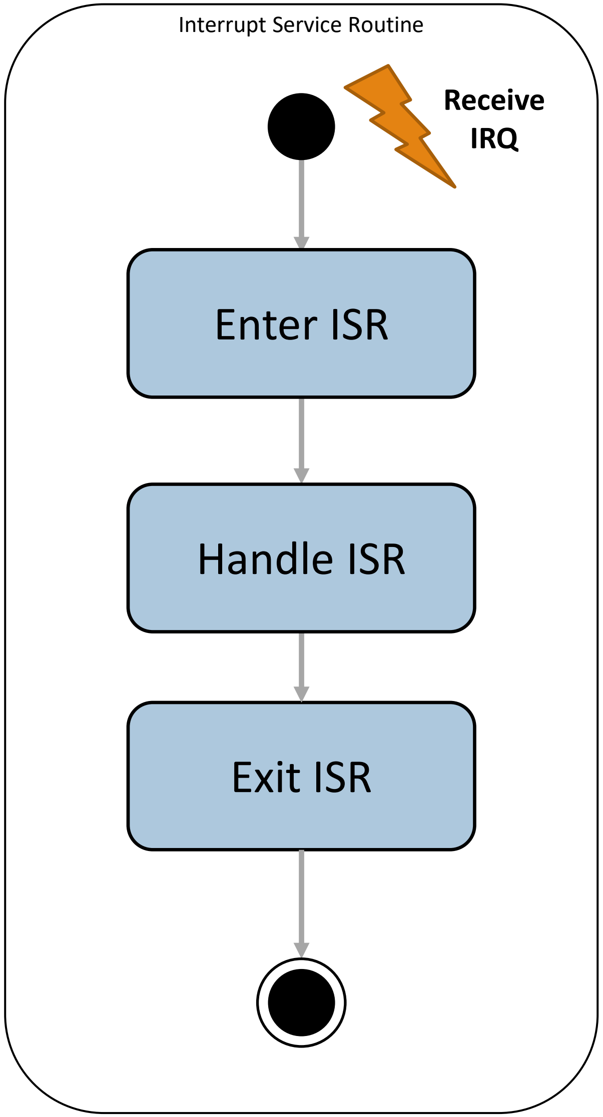




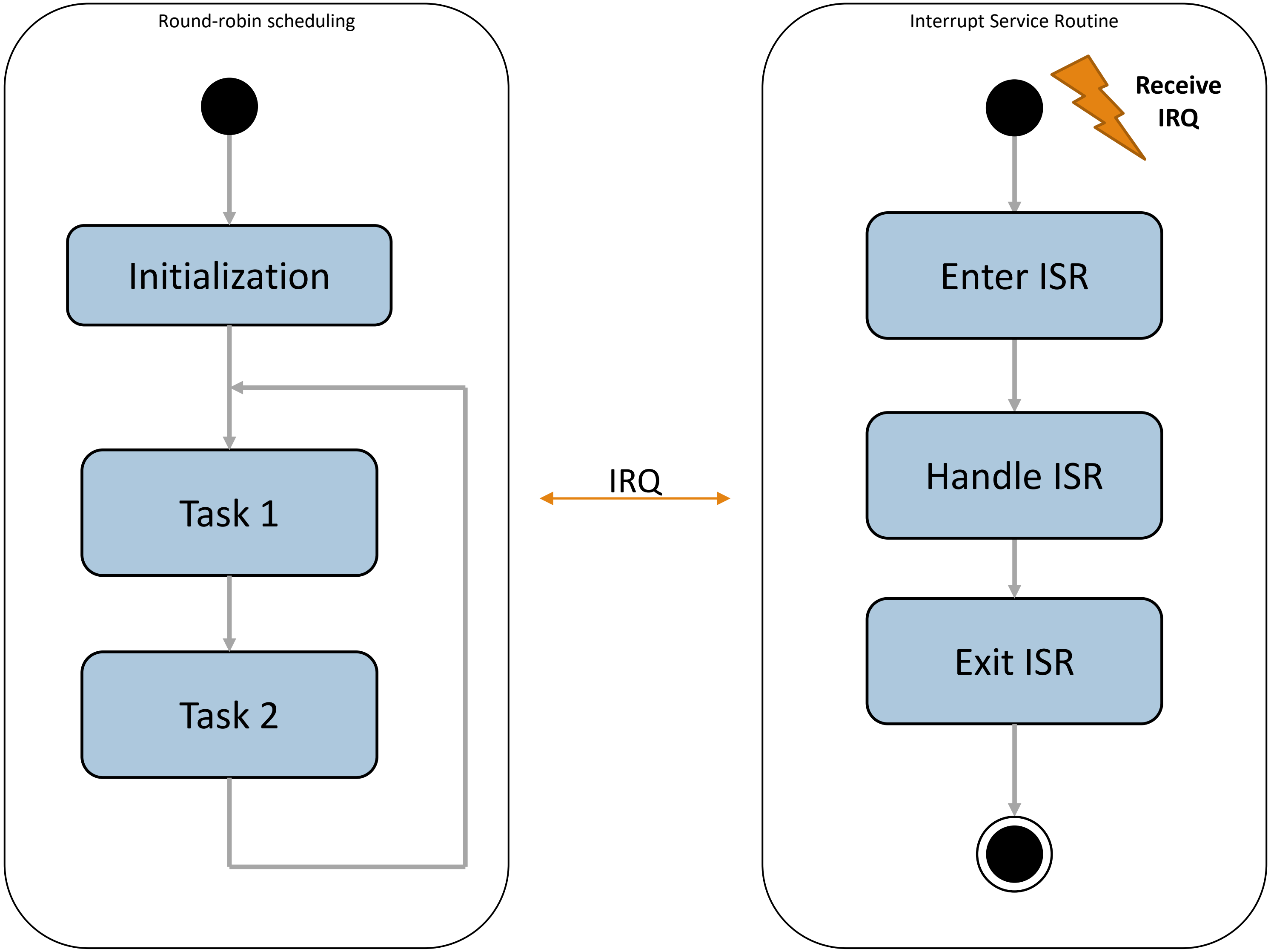
```
int main()
{
    System_Init();

    // Begin round robin scheduling
    while (1)
    {
        Task_1();

        Task_2();
    }

    return 0;
}
```

Simple Scheduling – Interrupts





Simple Scheduling – Interrupts



Round-robin scheduling

Initialization

Task 1

Task 2

IRQ

Interrupt Service Routine

Receive IRQ

Enter ISR

Handle ISR

Exit ISR

State Machines

“State Machines Reduce Spagetti Code”

A State Machine captures the logic of the system in a structured way

Key concepts are

Event Some external or internal inputs which the system shall respond to.

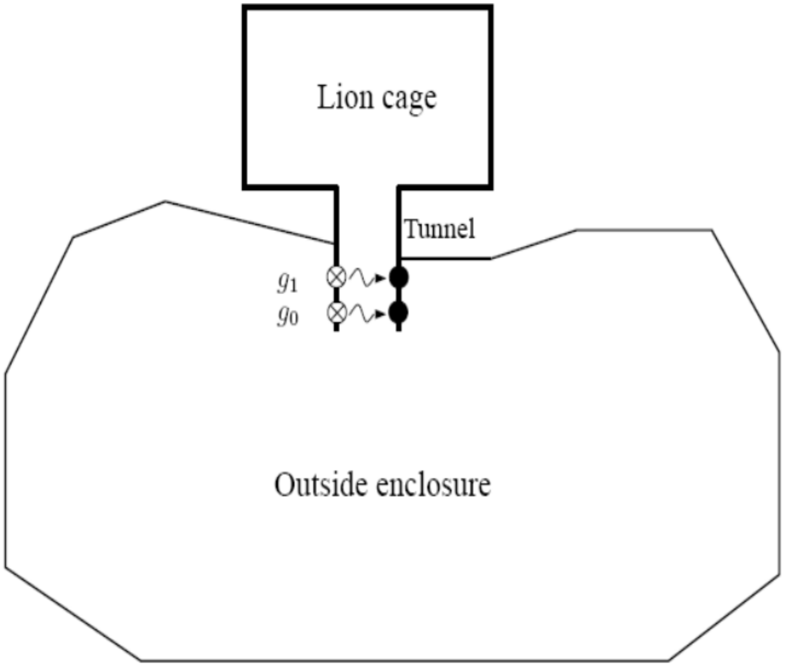
State An encapsulation of the history of past events

The State Machine describes how the *next state* is determined by the *current state* and incoming *events*

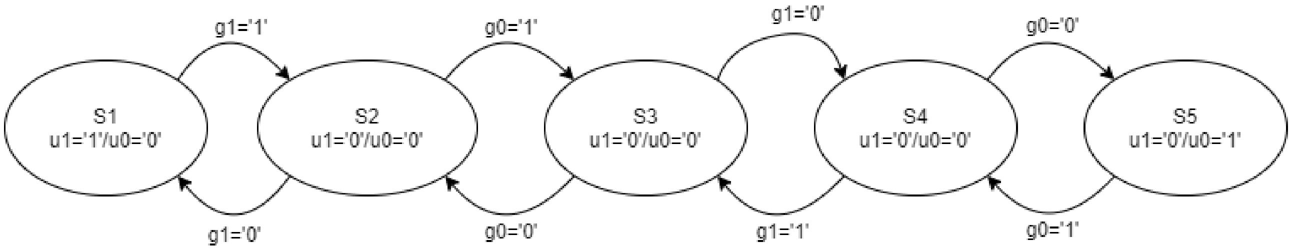
Lion Cage Example

1. Inputs: Sensors g1 and g0
2. Outputs: u1 and u0

u1 = ’1’ if indoors otherwise ’0’
u0 = ’1’ if outside otherwise ’0’



S1 = ”Lion in the cage”
S2 = ”Lion almost in the cage”
S3 = ”Lion in the tunnel”
S4 = ”Lion almost outdoors”
S5 = ”Lion outdoors”



State Machines – Moore vs Mealy

Moore

- A Moore machine performs actions when entering a state. Each state may have it's own entry action.

Mealy

- A Mealy machine performs actions on transitions. Each transition in a state machine may invoke a unique action.

Note: Both Moore and Mealy can be used at the same time.

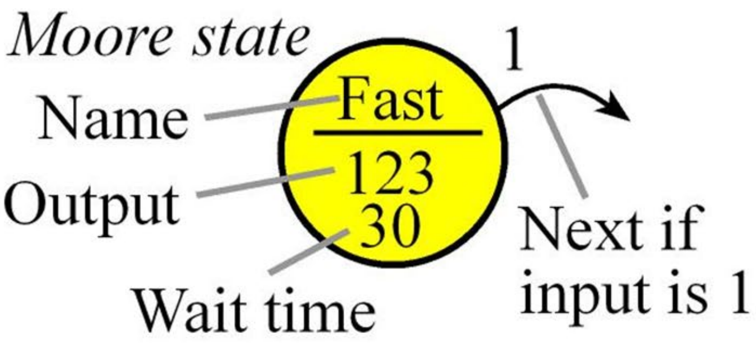
State Machines – Moore vs Mealy

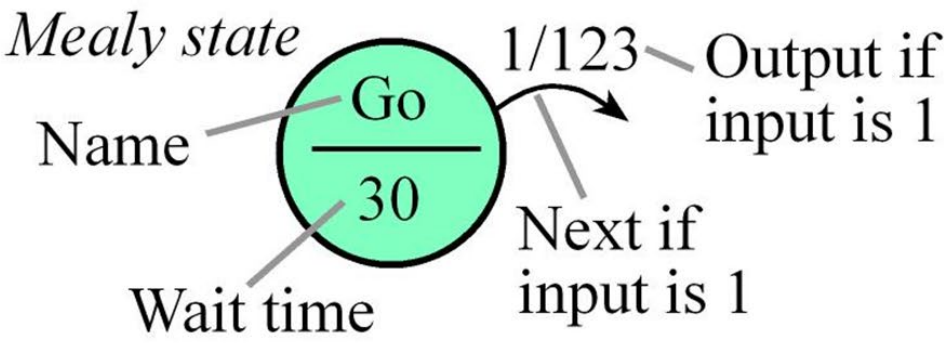
Moore State Machine	Mealy State Machine

Outputs depend on current state only
outputs = F(currentState);
nextState = F(inputs, currentState);
	Outputs depend on current state and the inputs
	outputs = F(inputs, currentState);
	nextState = F(inputs, currentState);

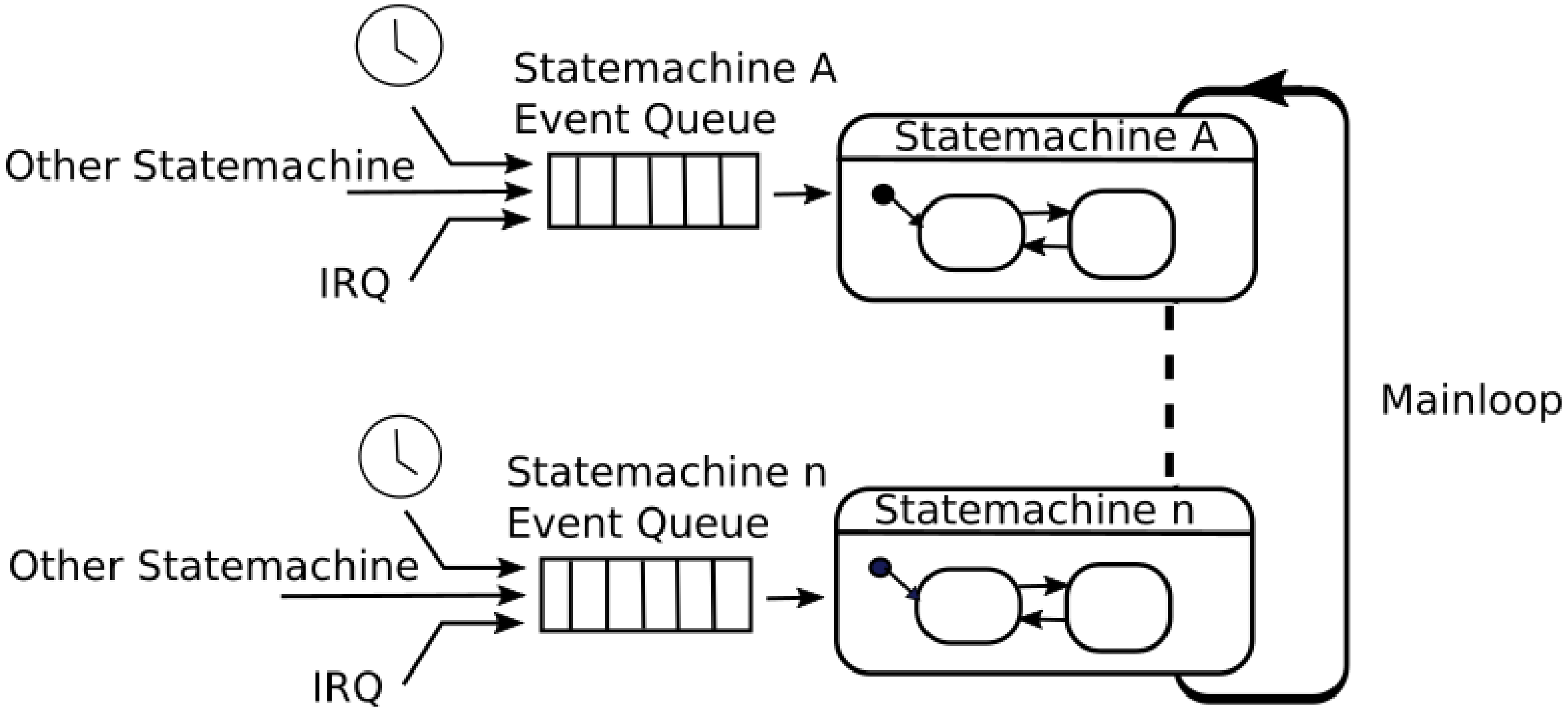
Generally, it has more states than Mealy	Generally, it has fewer states than Moore
Output does not react immediately to input change (one clock cycle later)	Outputs have immediate reaction to inputs
Output is placed on states	Output is placed on transitions
Easy to design	It is difficult to design

State Machines – Moore vs Mealy





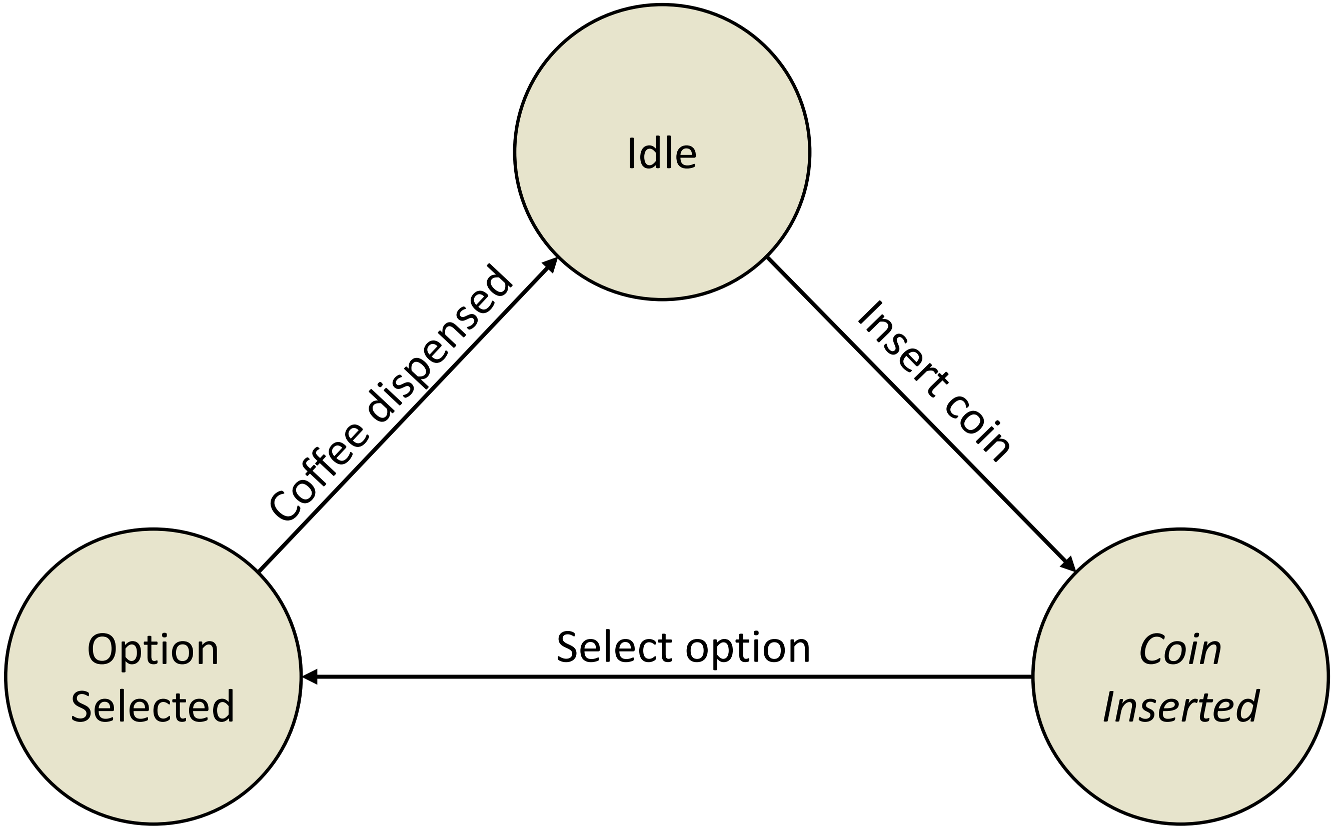
State Machines



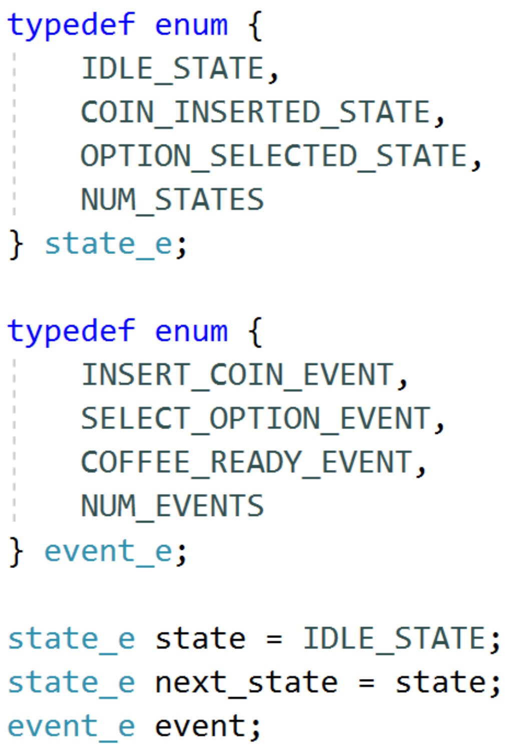
https://www.sinelabore.com/

Simple State Machine – Example

Coffee Machine

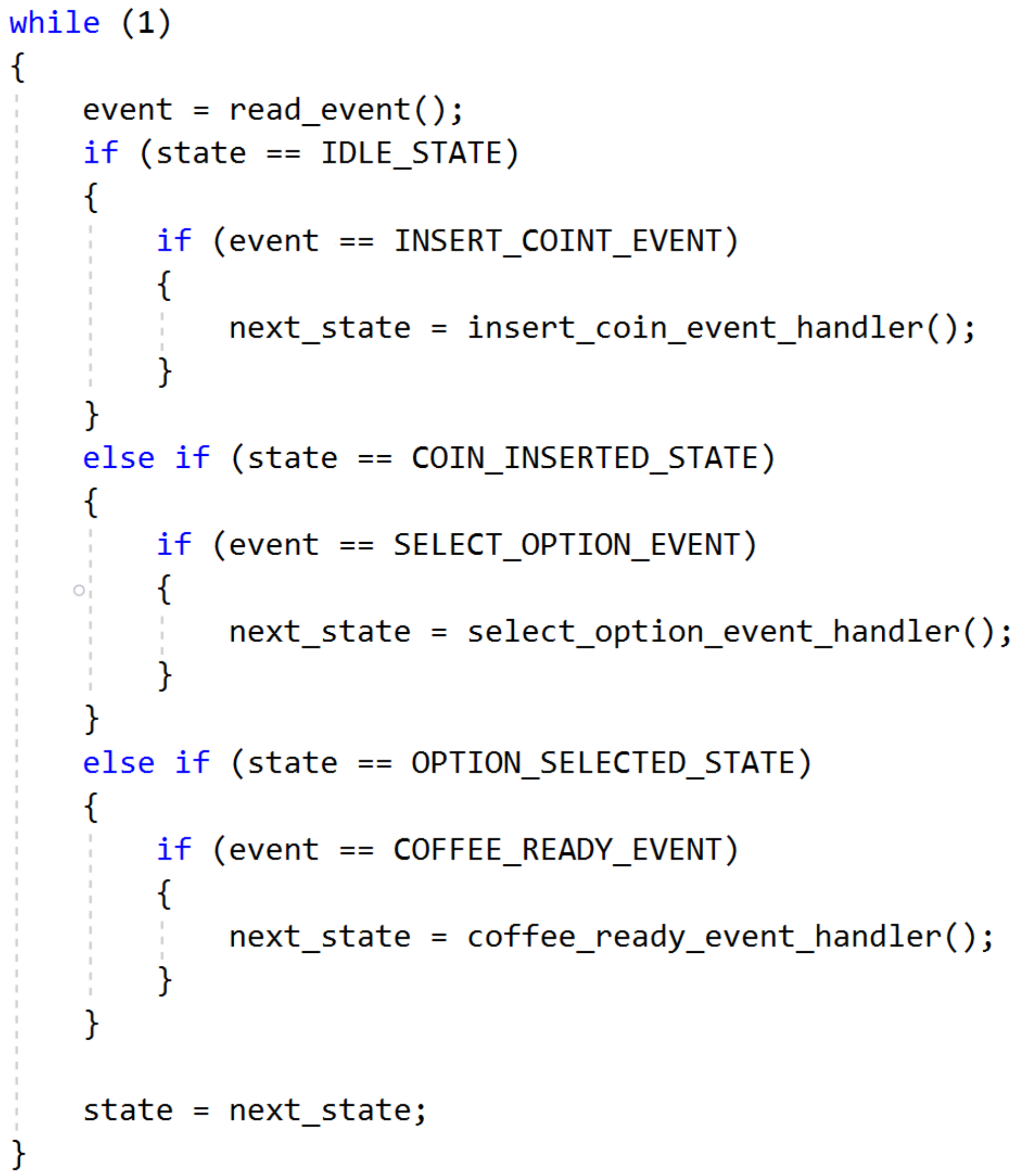


Simple State Machine – Example



```c
typedef enum {
    IDLE_STATE,
    COIN_INSERTED_STATE,
    OPTION_SELECTED_STATE,
    NUM_STATES
} state_e;

typedef enum {
    INSERT_COIN_EVENT,
    SELECT_OPTION_EVENT,
    COFFEE_READY_EVENT,
    NUM_EVENTS
} event_e;

state_e state = IDLE_STATE;
state_e next_state = state;
event_e event;
```

Simple State Machine – Example

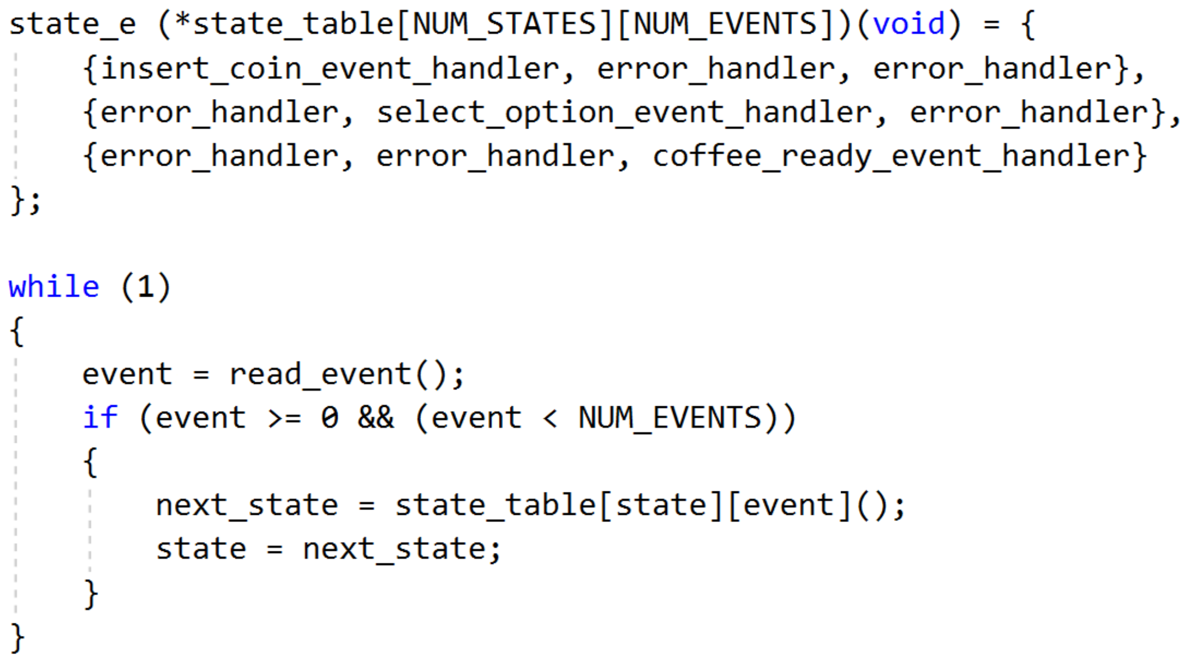


```
while (1)
{
    event = read_event();
    if (state == IDLE_STATE)
    {
        if (event == INSERT_COINT_EVENT)
        {
            next_state = insert_coin_event_handler();
        }
    }
    else if (state == COIN_INSERTED_STATE)
    {
        if (event == SELECT_OPTION_EVENT)
        {
            next_state = select_option_event_handler();
        }
    }
    else if (state == OPTION_SELECTED_STATE)
    {
        if (event == COFFEE_READY_EVENT)
        {
            next_state = coffee_ready_event_handler();
        }
    }

    state = next_state;
}
```

State Transition Matrix

STATE	Event_1	Event_2	Event_3	OUTPUT
State_1	State_2			OUT_1
State_2	State_1	State_5	State_3	OUT_2
State_3			State_4	OUT_3
State_4			State_5	OUT_4
State_5		State_2	State_2	OUT_5

Simple State Machine – Example



```
state_e (*state_table[NUM_STATES][NUM_EVENTS])(void) = {
    {insert_coin_event_handler, error_handler, error_handler},
    {error_handler, select_option_event_handler, error_handler},
    {error_handler, error_handler, coffee_ready_event_handler}
};

while (1)
{
    event = read_event();
    if (event >= 0 && (event < NUM_EVENTS))
    {
        next_state = state_table[state][event]();
        state = next_state;
    }
}
```

Example – Washer Machine

Example – Washer Machine

```
// Function declaration

// Variables

void main (void) {
        SystemInit();
        while(1) {
                Wash();  // State Machine for Washing
                Heat();   // State Machine for Heating
        }

}
```

Example – Washer Machine

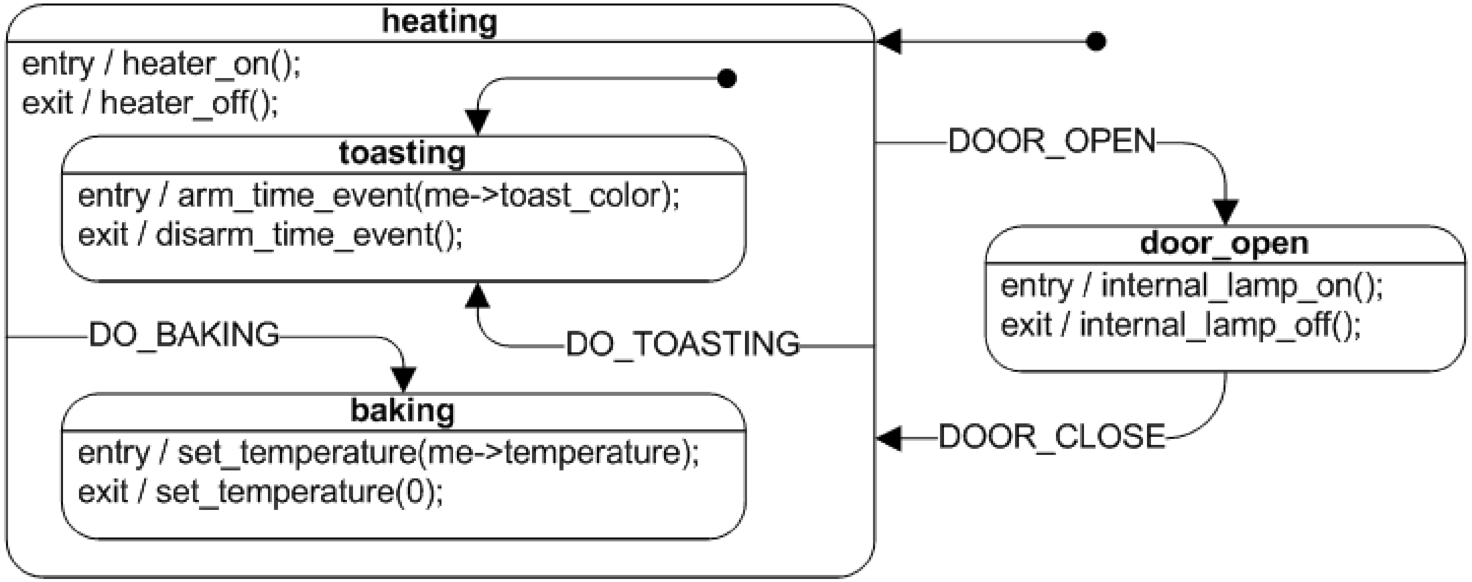
```
typedef enum {STATE1, STATE2, STATE3} state_ex_t;

typedef enum {s_OFF, s_FILL, s_WASH, s_EMPTY} state_washer_t;
typedef enum {s_OFF_H, s_WAIT1, s_HEAT, s_WAIT2} state_heat_t;

typedef enum {NONE, ONOFF, FILLED, EMPTIED} event_t;
```

Example – Washer Machine

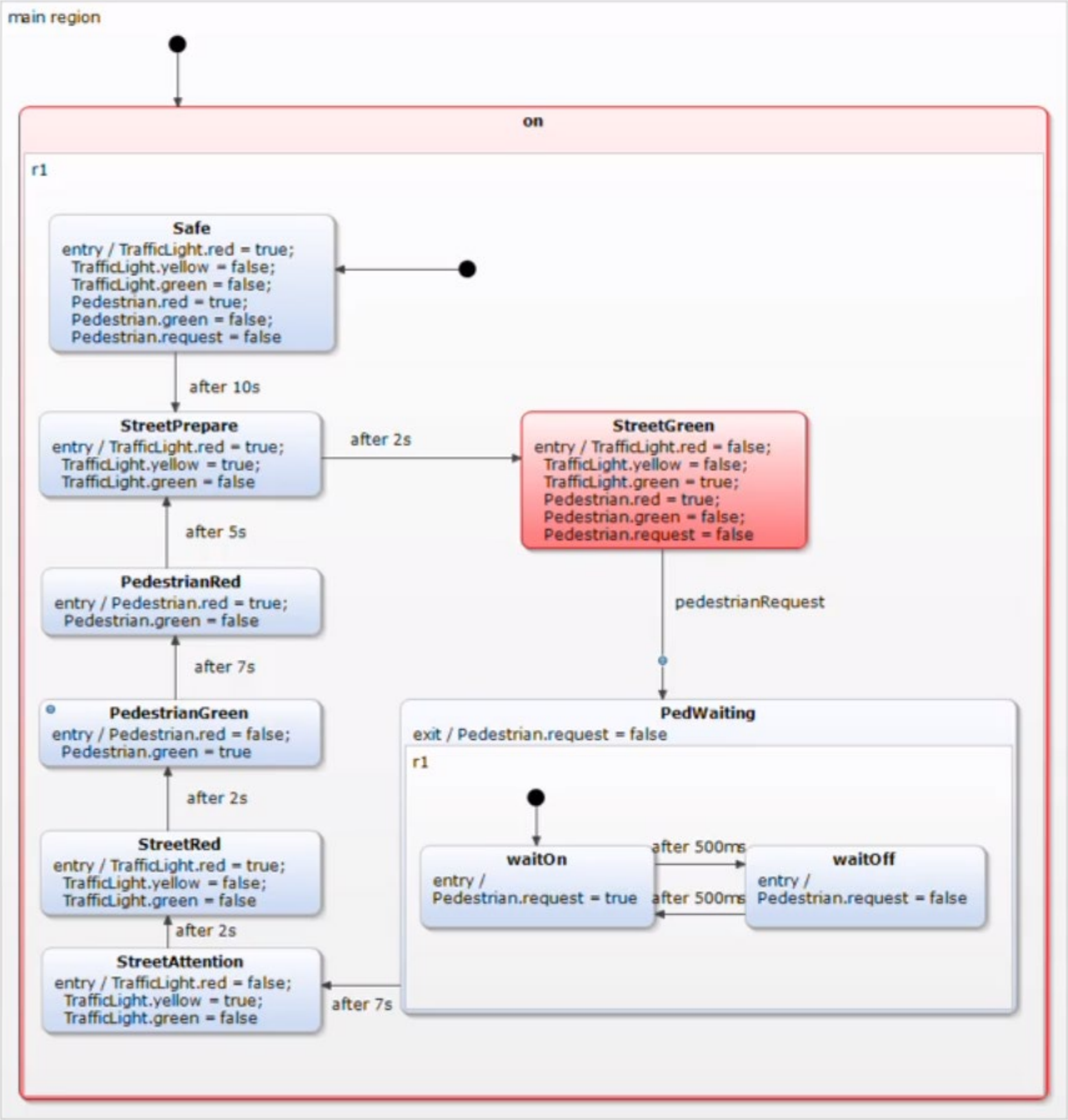
```
static state_washer_t  washState = s_OFF;

switch (washState) {

        case s_OFF:
                Do something
                if (event == ONOFF) {
                        washState = s_FILL;
                }
                break;


        case s_FILL:
                Do something
                if (event == FILLED) {
                        washState = s_WASH;
                }
                break;

        case s_WASH:
                Do something
                if (washing_process == READY) {
                        washState = s_EMPTY;
                }
                break;


        case s_EMPTY:
                Do something
                if (event == EMPTIED) {
                        washState = s_OFF;

                }
                break;


        default: washState = s_OFF;
                break;
}
```

NEVER A WHILE INSIDE A STATE MACHINE

Simple State Machine – Example


heating
entry / heater_on();
exit / heater_off();
toasting
entry / arm_time_event(me->toast_color);
exit / disarm_time_event();
DO_BAKING
DO_TOASTING
baking
entry / set_temperature(me->temperature);
exit / set_temperature(0);
DOOR_OPEN
door_open
entry / internal_lamp_on();
exit / internal_lamp_off();
DOOR_CLOSE

State Machine – Example



main region

on

r1

Safe
entry / TrafficLight.red = true;
TrafficLight.yellow = false;
TrafficLight.green = false;
Pedestrian.red = true;
Pedestrian.green = false;
Pedestrian.request = false

StreetPrepare
entry / TrafficLight.red = true;
TrafficLight.yellow = true;
TrafficLight.green = false

StreetGreen
entry / TrafficLight.red = false;
TrafficLight.yellow = false;
TrafficLight.green = true;
Pedestrian.red = true;
Pedestrian.green = false;
Pedestrian.request = false

PedestrianRed
entry / Pedestrian.red = true;
Pedestrian.green = false

PedestrianGreen
entry / Pedestrian.red = false;
Pedestrian.green = true

StreetRed
entry / TrafficLight.red = true;
TrafficLight.yellow = false;
TrafficLight.green = false

StreetAttention
entry / TrafficLight.red = false;
TrafficLight.yellow = true;
TrafficLight.green = false

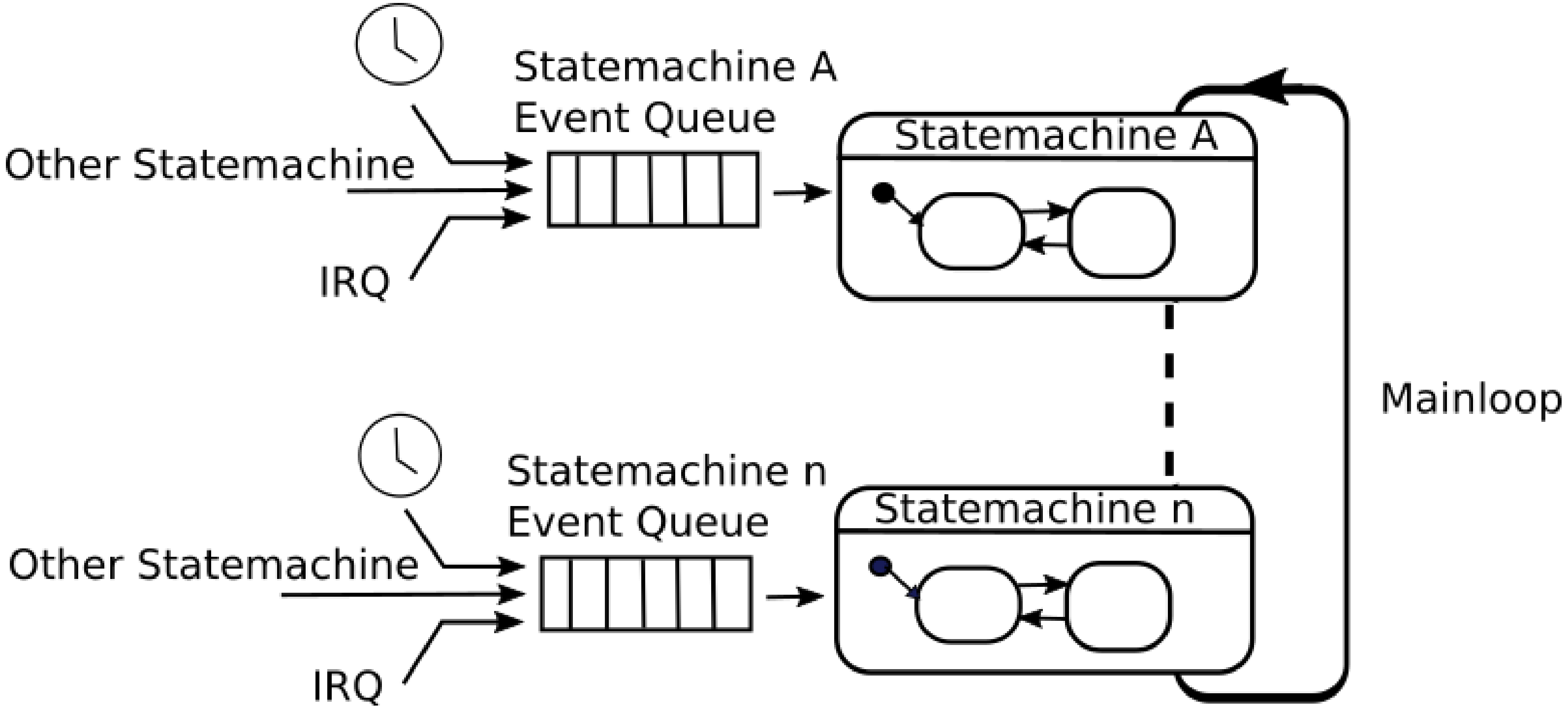
PedWaiting
exit / Pedestrian.request = false

r1

waitOn
entry /
Pedestrian.request = true

waitOff
entry /
Pedestrian.request = false

State Machines – Revisited



https://www.sinelabore.com/

Microcomputer Engineering

Questions?

Contact information

Andreas Axelsson

Email: andreas.axelsson@ju.se

Mobile: 0709-467760