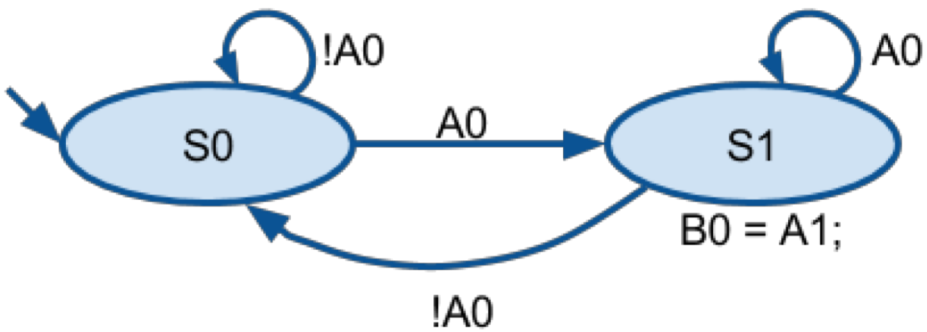
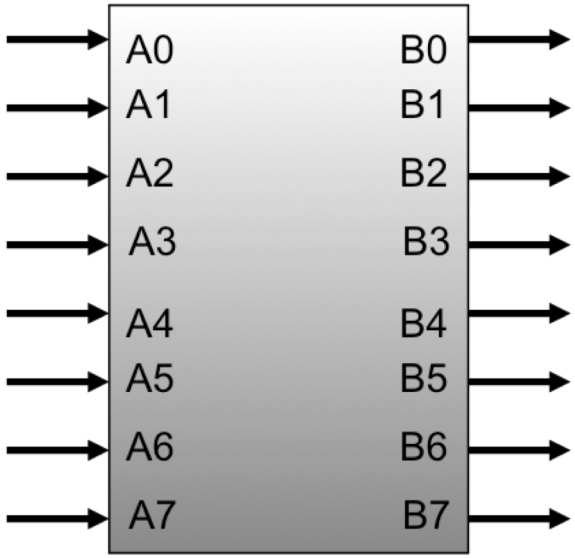
SynchSM Basics

Synchronous State Machine (SynchSM)

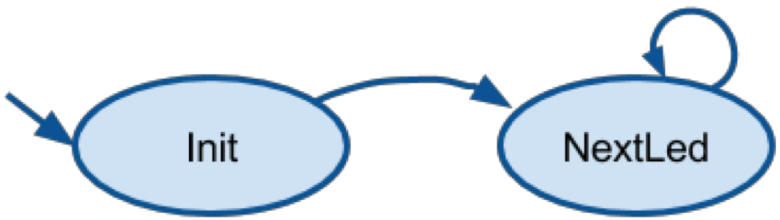


RIMS



SM

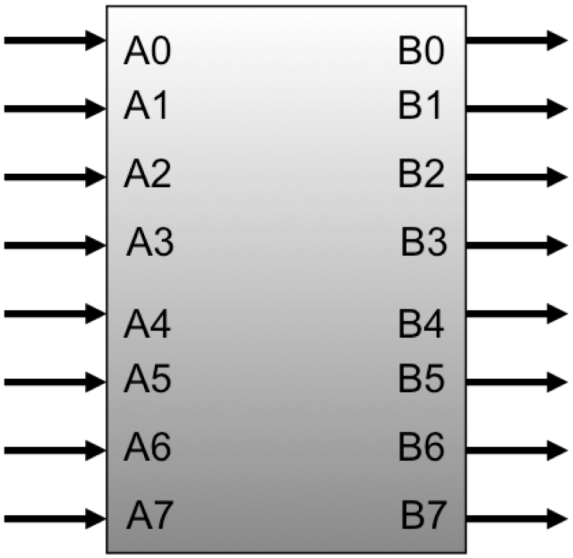
SequenceLEDs
unsigned char out;



B = out = 0x01;

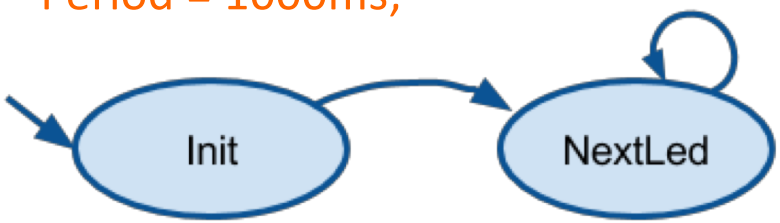
out = (0x08)
? 0x01
: out << 1;
B = out;

RIMS



SynchSM

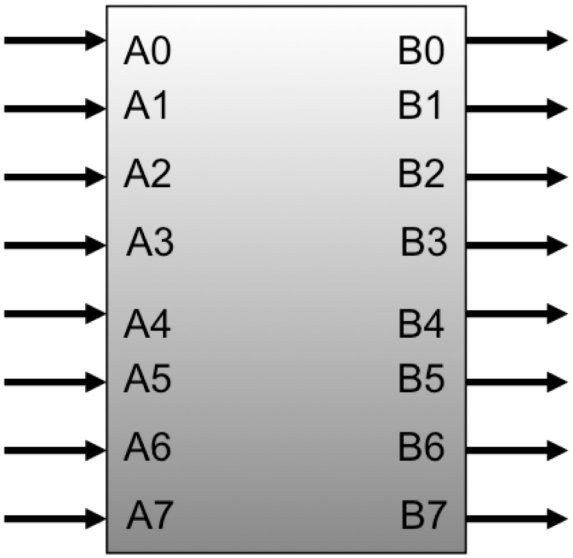
SequenceLEDs
unsigned char out;
Period = 1000ms;



B = out = 0x01;

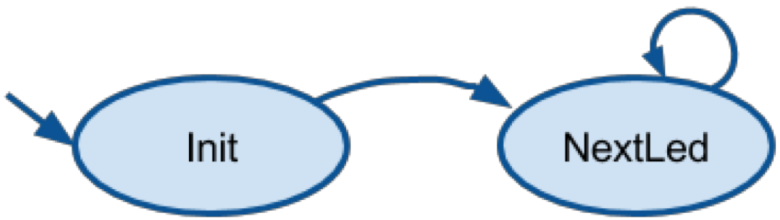
out = (0x08)
? 0x01
: out << 1;
B = out;

RIMS



SM

SequenceLEDs
unsigned char out;

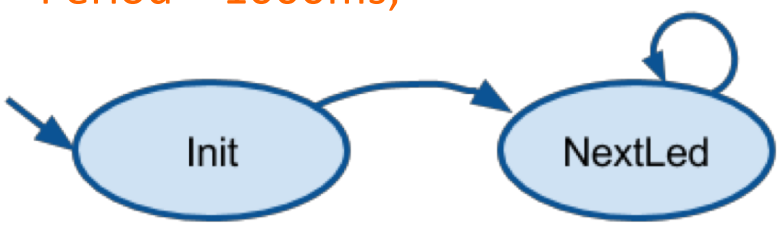


B = out = 0x01;

out = (0x08)
? 0x01
: out << 1;
B = out;

SynchSM

SequenceLEDs
unsigned char out;
Period = 1000ms;



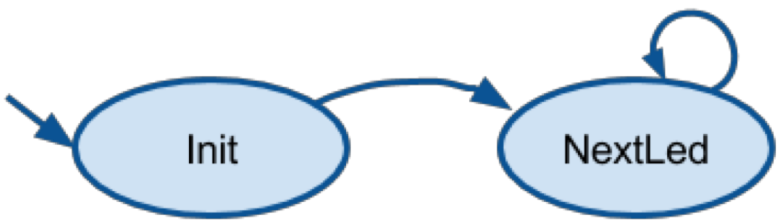
B = out = 0x01;

out = (0x08)
? 0x01
: out << 1;
B = out;

What's the Difference?

SM

SequenceLEDs
unsigned char out;

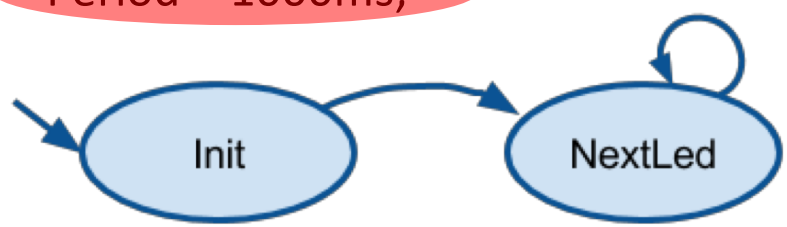


B = out = 0x01;

out = (0x08)
? 0x01
: out << 1;
B = out;

SynchSM

SequenceLEDs
unsigned char out;
Period = 1000ms;

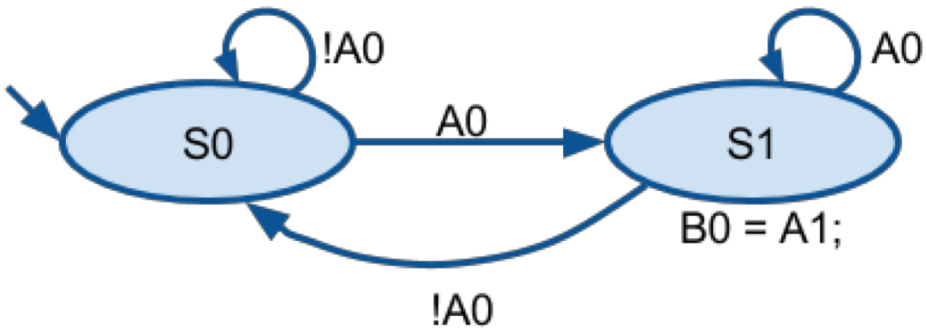


B = out = 0x01;

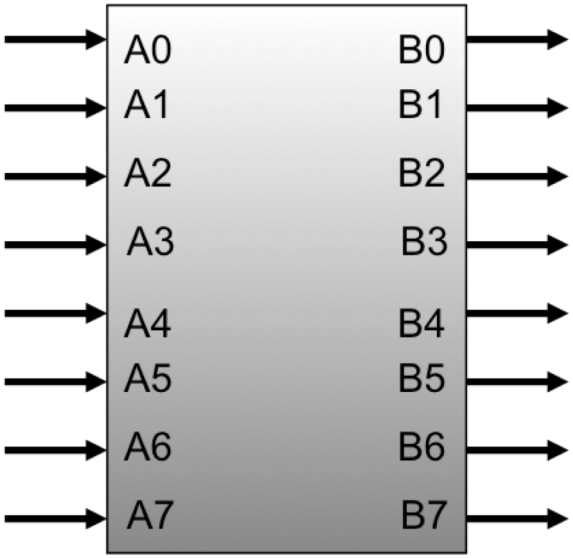
out = (0x08)
? 0x01
: out << 1;
B = out;

SynchSM Conversion to C

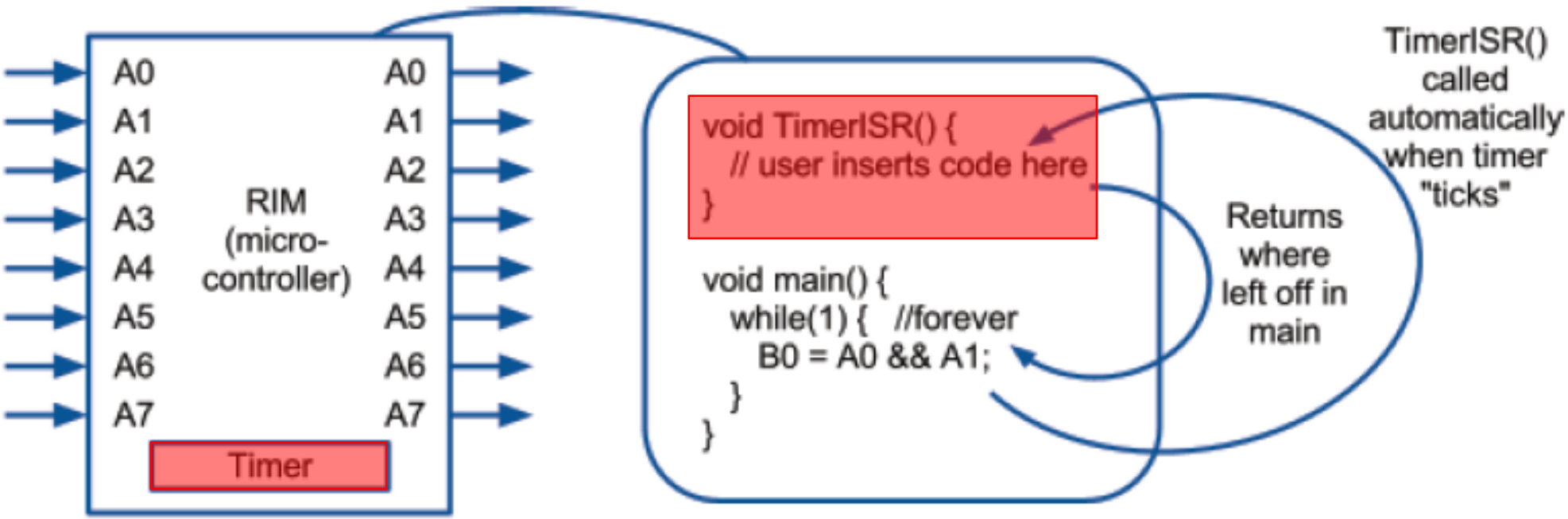
SynchSM



RIMS



Interrupt Service Routing (ISR)



Example

```
#include “RIMS.h”

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
	B = 0;                          //Initialize output
	TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
	TimerOn();                      // Turn timer on

	while (1) {
		B0 = !B0;               // Toggle B0
		while(!TimerFlag) {}    // Wait 1 sec
		TimerFlag = 0;

		// NOTE: This example just illustrates use of an ISR and flag
	}
}
```

Initialization

TimerFlag: 0

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
    B = 0;                      //Initialize output
    TimerSet(1000);             // Timer period = 1000 ms (1 sec)
    TimerOn();                  // Turn timer on

    while (1) {
        B0 = !B0;               // Toggle B0
        while(!TimerFlag) {}    // Wait 1 sec
        TimerFlag = 0;

        // NOTE: This example just illustrates use of an ISR and flag
    }
}
```

Execute C Code

TimerFlag: 0

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
    B = 0;                      //Initialize output
    TimerSet(1000);             // Timer period = 1000 ms (1 sec)
    TimerOn();                  // Turn timer on

    while (1) {
        B0 = !B0;               // Toggle B0
        while(!TimerFlag) {}    // Wait 1 sec
        TimerFlag = 0;

        // NOTE: This example just illustrates use of an ISR and flag
    }
}
```

Wait…

TimerFlag: 0

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
    B = 0;                      //Initialize output
    TimerSet(1000);             // Timer period = 1000 ms (1 sec)
    TimerOn();                  // Turn timer on

    while (1) {
        B0 = !B0;               // Toggle B0
        while(!TimerFlag) {}    // Wait 1 sec
        TimerFlag = 0;

        // NOTE: This example just illustrates use of an ISR and flag
    }
}
```

TimerISR Fires After 1 Second

TimerFlag: 1

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
	B = 0;                          //Initialize output
	TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
	TimerOn();                      // Turn timer on

	while (1) {
		B0 = !B0;               // Toggle B0
		while(!TimerFlag) {}    // Wait 1 sec
		TimerFlag = 0;

		// NOTE: This example just illustrates use of an ISR and flag
	}
}
```

Exit the While Loop

TimerFlag: 1

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
    B = 0;                      //Initialize output
    TimerSet(1000);             // Timer period = 1000 ms (1 sec)
    TimerOn();                  // Turn timer on

    while (1) {
        B0 = !B0;               // Toggle B0
        while(!TimerFlag) {}    // Wait 1 sec
        TimerFlag = 0;

        // NOTE: This example just illustrates use of an ISR and flag
    }
}
```

Reset the Timer Flag

TimerFlag: 0

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
	B = 0;                          //Initialize output
	TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
	TimerOn();                      // Turn timer on

	while (1) {
		B0 = !B0;               // Toggle B0
		while(!TimerFlag) {}    // Wait 1 sec
		TimerFlag = 0;

		// NOTE: This example just illustrates use of an ISR and flag
	}
}
```

Execute C Code

TimerFlag: 0

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
    B = 0;                      //Initialize output
    TimerSet(1000);             // Timer period = 1000 ms (1 sec)
    TimerOn();                  // Turn timer on

    while (1) {
        B0 = !B0;               // Toggle B0
        while(!TimerFlag) {}    // Wait 1 sec
        TimerFlag = 0;

        // NOTE: This example just illustrates use of an ISR and flag
    }
}
```

Wait…

TimerFlag: 0

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
    B = 0;                      //Initialize output
    TimerSet(1000);             // Timer period = 1000 ms (1 sec)
    TimerOn();                  // Turn timer on

    while (1) {
        B0 = !B0;               // Toggle B0
        while(!TimerFlag) {}    // Wait 1 sec
        TimerFlag = 0;

        // NOTE: This example just illustrates use of an ISR and flag
    }
}
```

Volatile Keyword

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
        B = 0;                          //Initialize output
        TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
        TimerOn();                      // Turn timer on

        while (1) {
                B0 = !B0;               // Toggle B0
                while(!TimerFlag) {}    // Wait 1 sec
                TimerFlag = 0;

                // NOTE: This example just illustrates use of an ISR and flag
        }
}
```

What happens if we omit "Volatile?"

```
#include "RIMS.h"

unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
        B = 0;                          //Initialize output
        TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
        TimerOn();                      // Turn timer on

        while (1) {
                B0 = !B0;               // Toggle B0
                while(!TimerFlag) {}    // Wait 1 sec
                TimerFlag = 0;

                // NOTE: This example just illustrates use of an ISR and flag
        }
}
```

Since the program does not call TimerISR(), the compiler (which does not understand the microcontroller OS/API) thinks it is dead code

What happens if we omit "Volatile?"

```
#include "RIMS.h"

unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
        B = 0;                          //Initialize output
        TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
        TimerOn();                      // Turn timer on

        while (1) {
                B0 = !B0;               // Toggle B0
                while(!TimerFlag) {}    // Wait 1 sec
                TimerFlag = 0;

                // NOTE: This example just illustrates use of an ISR and flag
        }
}
```

Eliminate Dead Code

Constant Propagation

Both assignments to TimerFlag assign the value 0, so the compiler applies <u>constant propagation</u>.

- The one use of TimerFlag is replaced with constant 0

```
#include "RIMS.h"

unsigned char TimerFlag = 0;


void main() {
        B = 0;                          //Initialize output
        TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
        TimerOn();                      // Turn timer on

        while (1) {
                B0 = !B0;       0       // Toggle B0
                while(!TimerFlag) {}    // Wait 1 sec
                TimerFlag = 0;

                // NOTE: This example just illustrates use of an ISR and flag
        }
}
```

Constant Propagation

```
#include "RIMS.h"

void main() {
        B = 0;                          //Initialize output
        TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
        TimerOn();                      // Turn timer on

        while (1) {
                B0 = !B0;               // Toggle B0
                while(!0) {}            // Wait 1 sec

                // NOTE: This example just illustrates use of an ISR and flag
        }
}
```

!0 = 1

A Change in Program Functionality

```
#include "RIMS.h"

void main() {
        B = 0;                          //Initialize output
        TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
        TimerOn();                      // Turn timer on

        while (1) {
                B0 = !B0;               // Toggle B0
                while(1) {}             // Wait 1 sec

                // NOTE: This example just illustrates use of an ISR and flag
        }
}
```

Not anymore…

Volatile Keyword

Please don't optimize me away, EVIL COMPILER!!!!!

```
#include "RIMS.h"

volatile unsigned char TimerFlag = 0;

void TimerISR() { TimerFlag = 1; }

void main() {
	B = 0;                          //Initialize output
	TimerSet(1000);                 // Timer period = 1000 ms (1 sec)
	TimerOn();                      // Turn timer on

	while (1) {
		B0 = !B0;               // Toggle B0
		while(!TimerFlag) {}    // Wait 1 sec
		TimerFlag = 0;

		// NOTE: This example just illustrates use of an ISR and flag
	}
}
```

SynchSM Template

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

volatile unsigned char TimerFlag = 0;
void TimerISR() { TimerFlag = 1; }
void Tick() {…}

void main() {
        B = 0x00;
        state = …;
        TimerSet(…);
        TimerOn();

         while(1) {
                Tick();
                while (!TimerFlag) {}
                TimerFlag = 0;
        }
}
```

SynchSM Template Summary

- Minimal changes to the SM Template
  - TimerFlag
  - TimerISR()
  - TimerSet(…)
  - TimerOn()
  - ‘volatile’ keyword in C

- No changes to the Tick() function