

Micro-controller Programming

Copyright (c) 2014 - 2020 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Based on

[https://en.wikiversity.org/wiki/
The_necessities_in_Microprocessor_Based_System_Design#Assembly_Programming](https://en.wikiversity.org/wiki/The_necessities_in_Microprocessor_Based_System_Design#Assembly_Programming)

ARM System-on-Chip Architecture, 2nd ed, Steve Furber

Introduction to ARM Cortex-M Microcontrollers
– Embedded Systems, Jonathan W. Valvano

Digital Design and Computer Architecture,
D. M. Harris and S. L. Harris

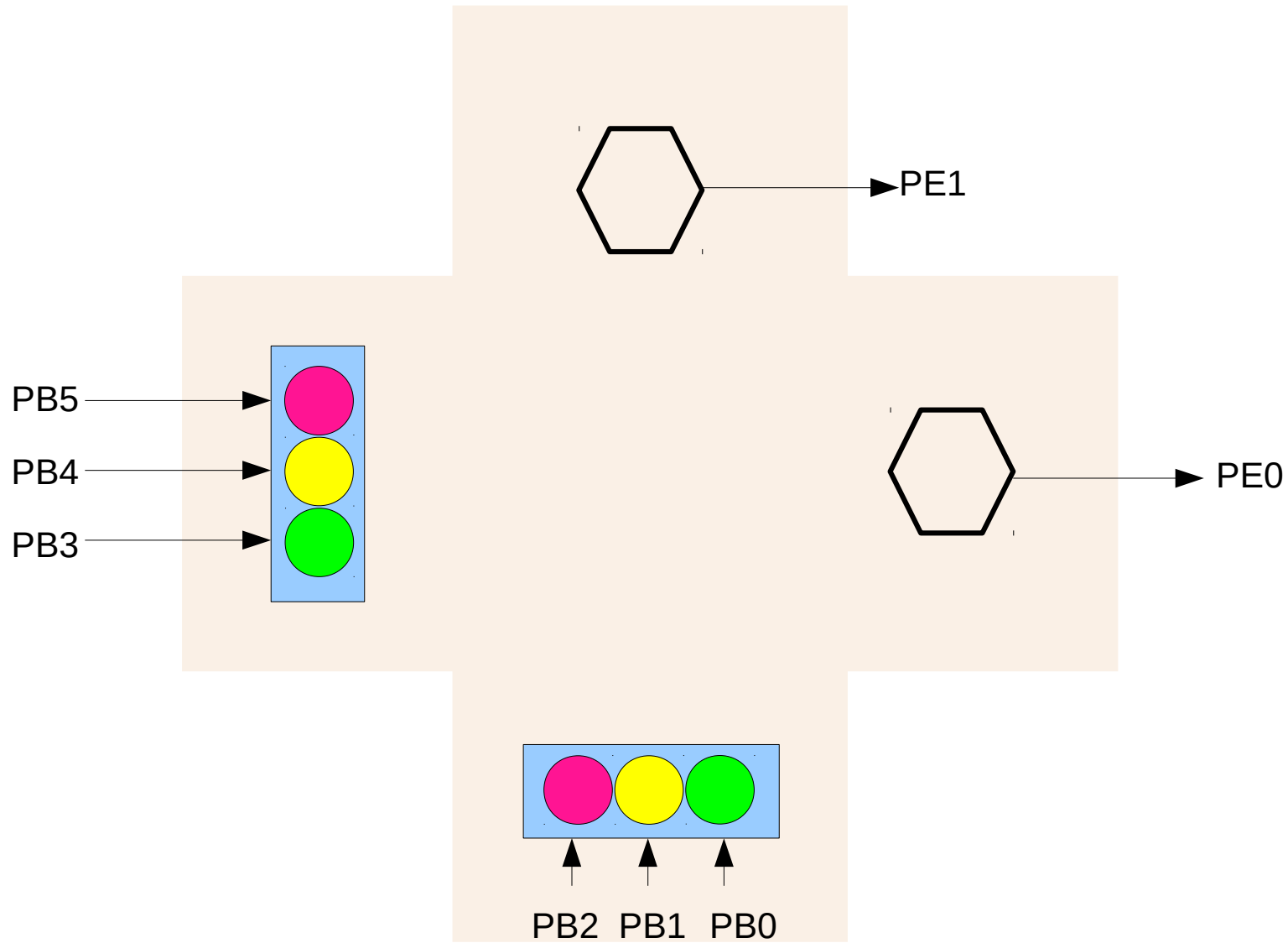
Computer Organization and Design ARM Edition:
The Hardware Software Interface
By David A. Patterson, John L. Hennessy

<https://thinkingeek.com/arm-assembler-raspberry-pi/>

Traffic Controller Rules

- (R1) If no cars are coming, stay in a **green** state, but which one doesn't matter. (00)
- (R2) To change from **green** to **red**, implement a **yellow** light of exactly **5** seconds
- (R3) **Green** lights will last at least **30** seconds
- (R4) If cars are only coming in one direction, move to and stay green in that direction. (10, 01)
- (R5) If cars are coming in both directions, cycle through all four states (11)

Sensor and Control Signals



Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

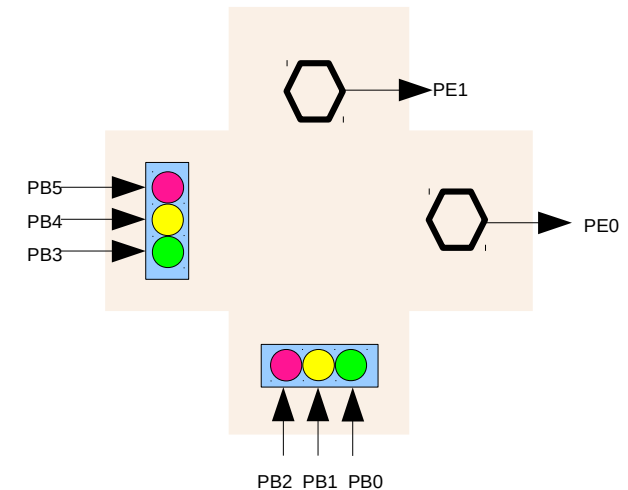
Inputs and Outputs Encoding

Inputs Encodings

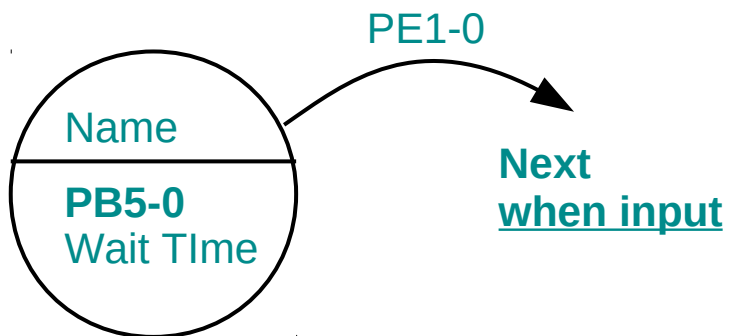
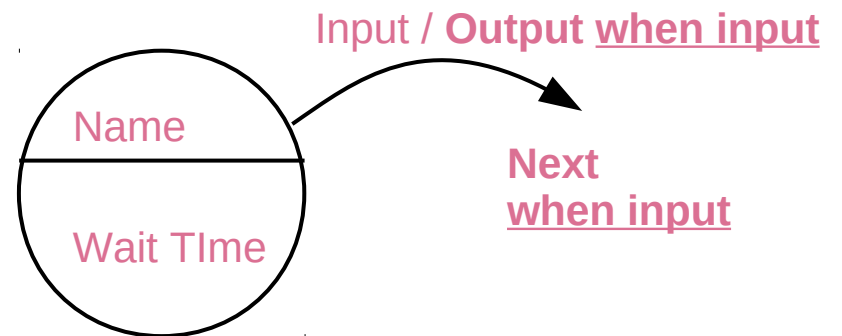
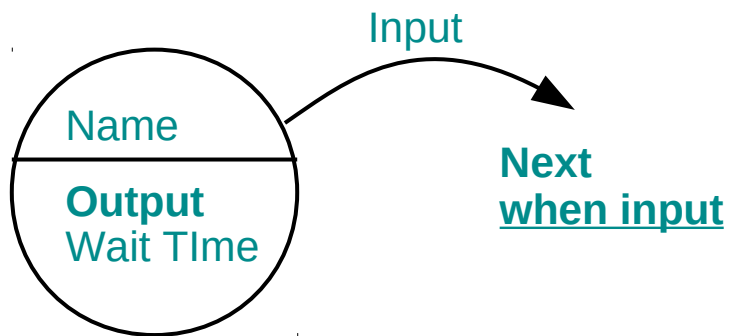
PE1 : indicates cars on the North road
PE0 : indicates cars on the East road

Output Encoding

PB5 : Red traffic lights on the East road
PB4 : Yellow traffic lights on the East road
PB3 : Green traffic lights on the East road
PB2 : Red traffic lights on the North road
PB1 : Yellow traffic lights on the North road
PB0 : Green traffic lights on the North road

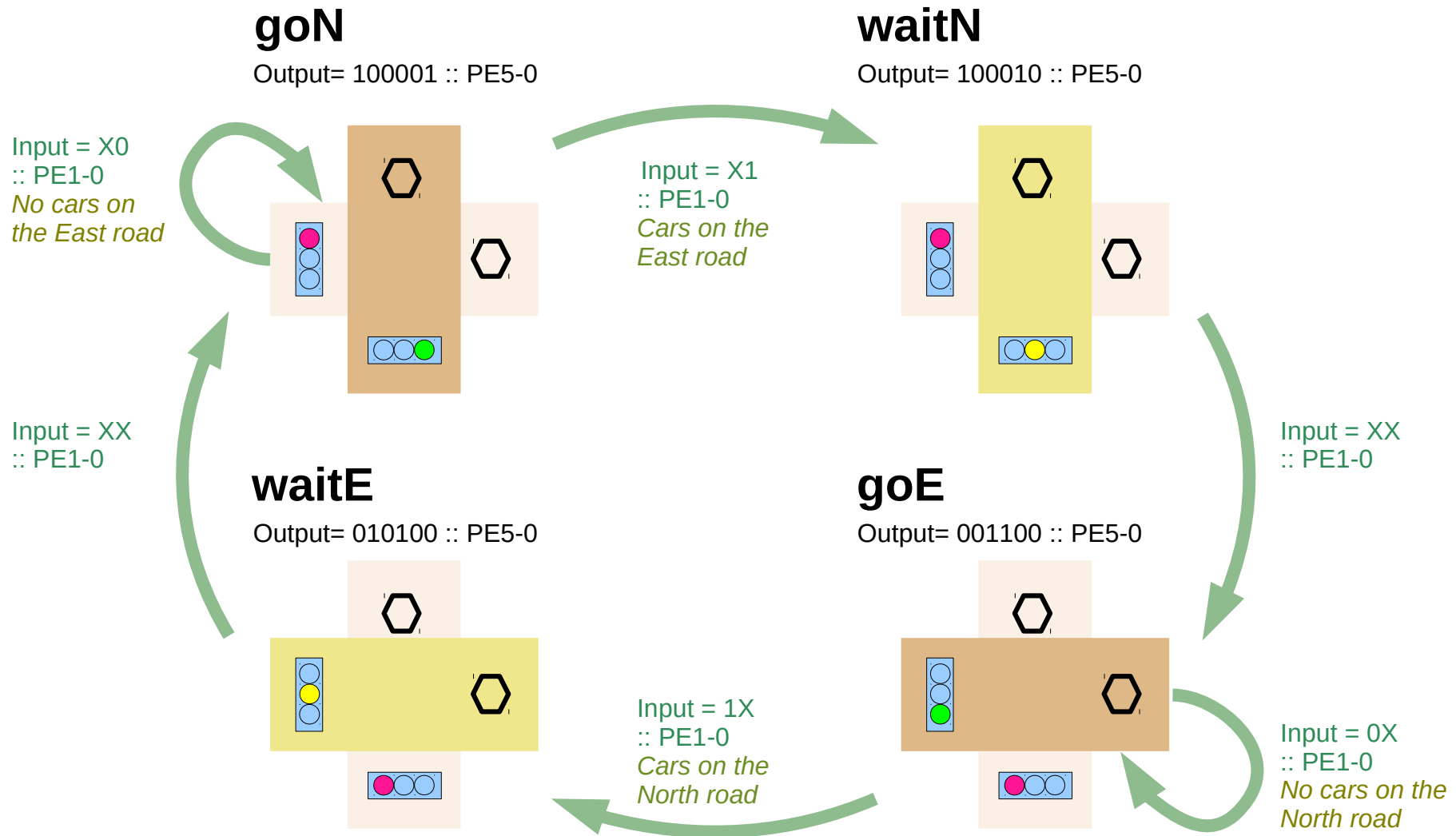


Moore FSM v.s. Mealy FSM



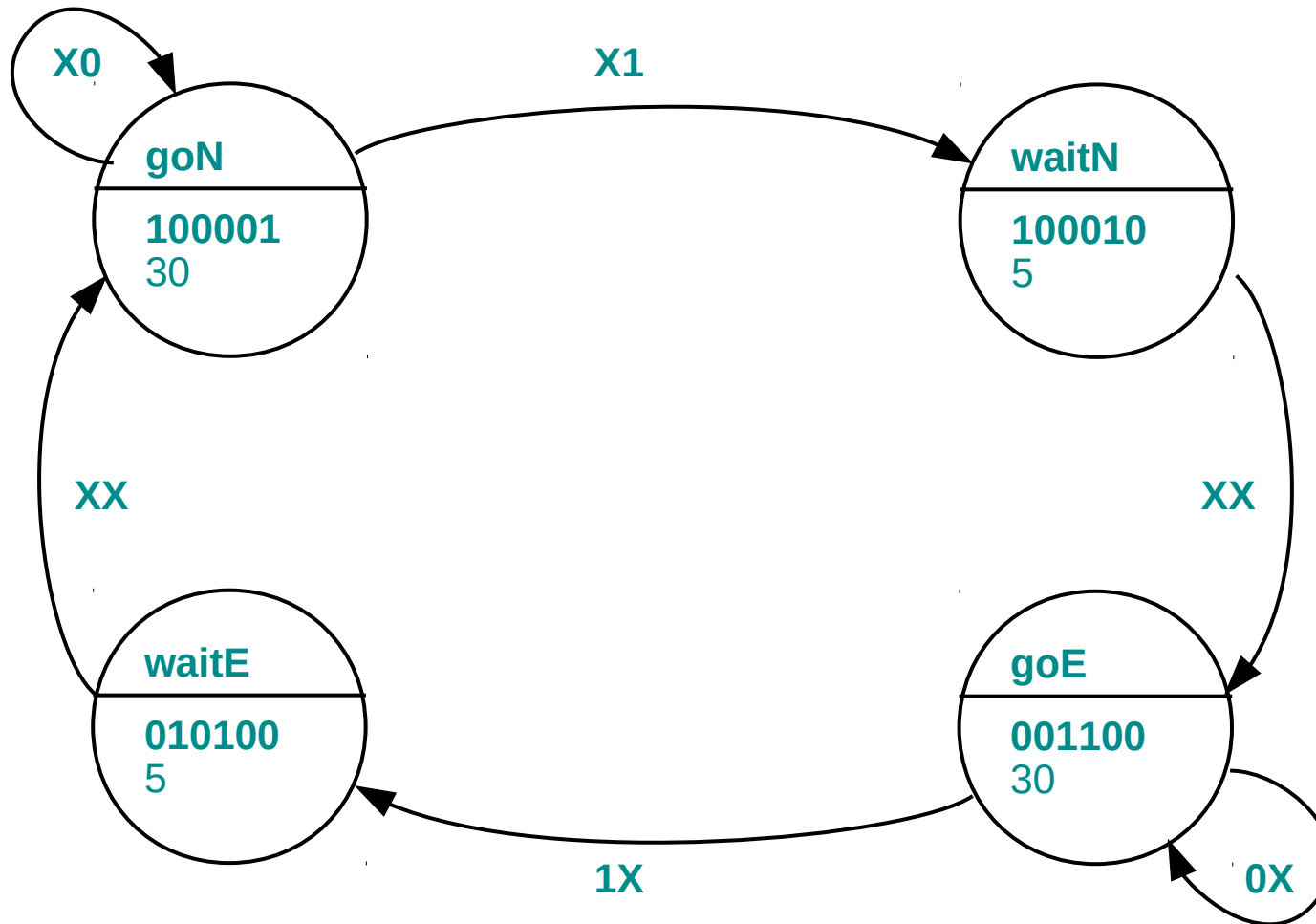
Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

State Transition – Moore FSM



Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano





State Transition Diagram – Moore FSM



Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

State Transition Table – Moore FSM

PE1-0 = 00 : no cars on either road
PE1-0 = 01 : cars on the East road
PE1-0 = 10 : cars on the North road
PE1-0 = 11 : cars on both roads

	PE1-0	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

Transition Conditions – Moore FSM

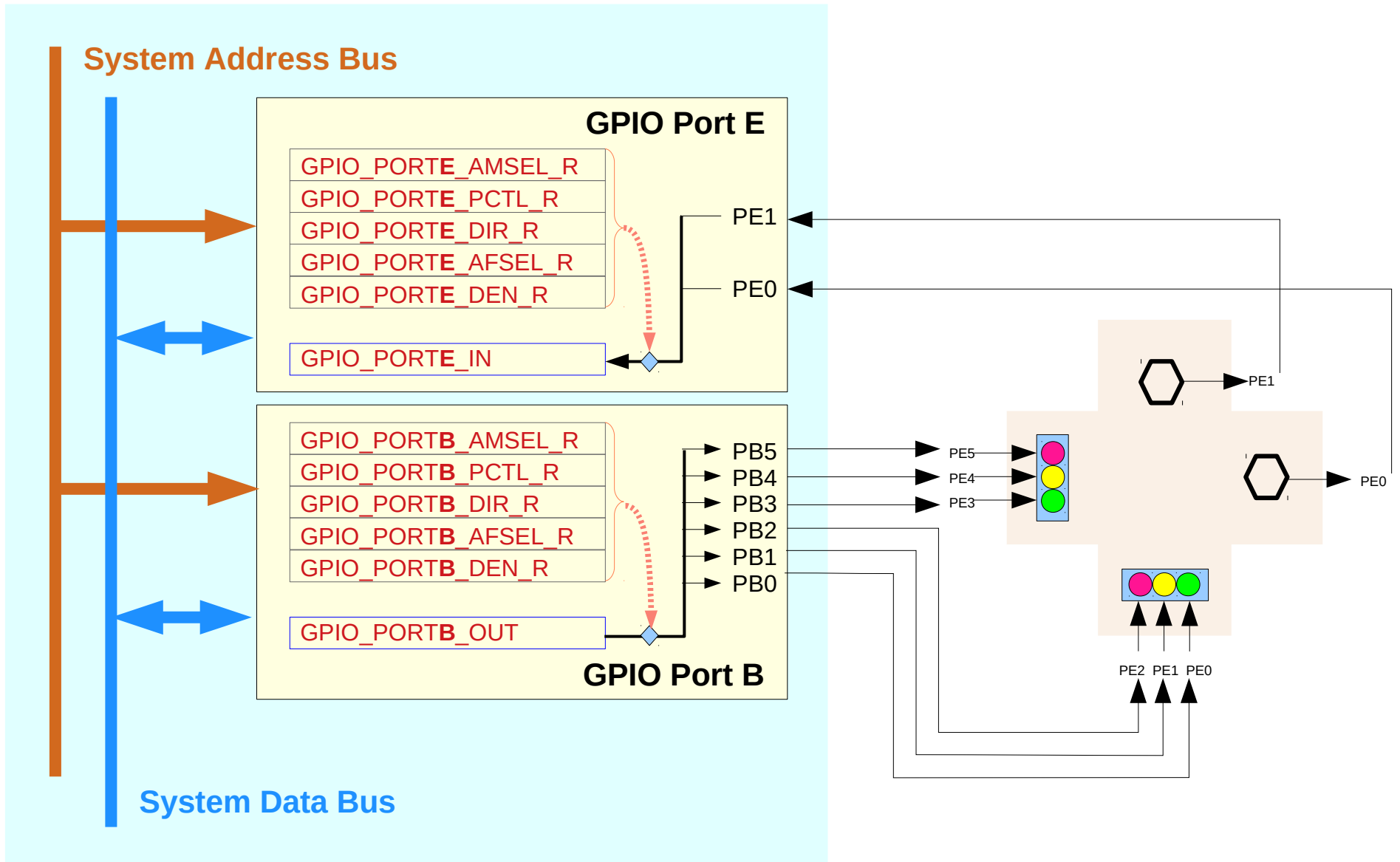
	PE1-0	00	01	10	11
goN	(100001, 30)	goN	waitN	goN	waitN
waitN	(100010, 5)	goE	goE	goE	goE
goE	(001100, 30)	goE	goE	waitE	waitE
waitE	(010100, 5)	goN	goN	goN	goN

goN	00	goN	(R1)
	01	waitN	(R4)
	10	goN	(R1)
	11	waitN	(R5)

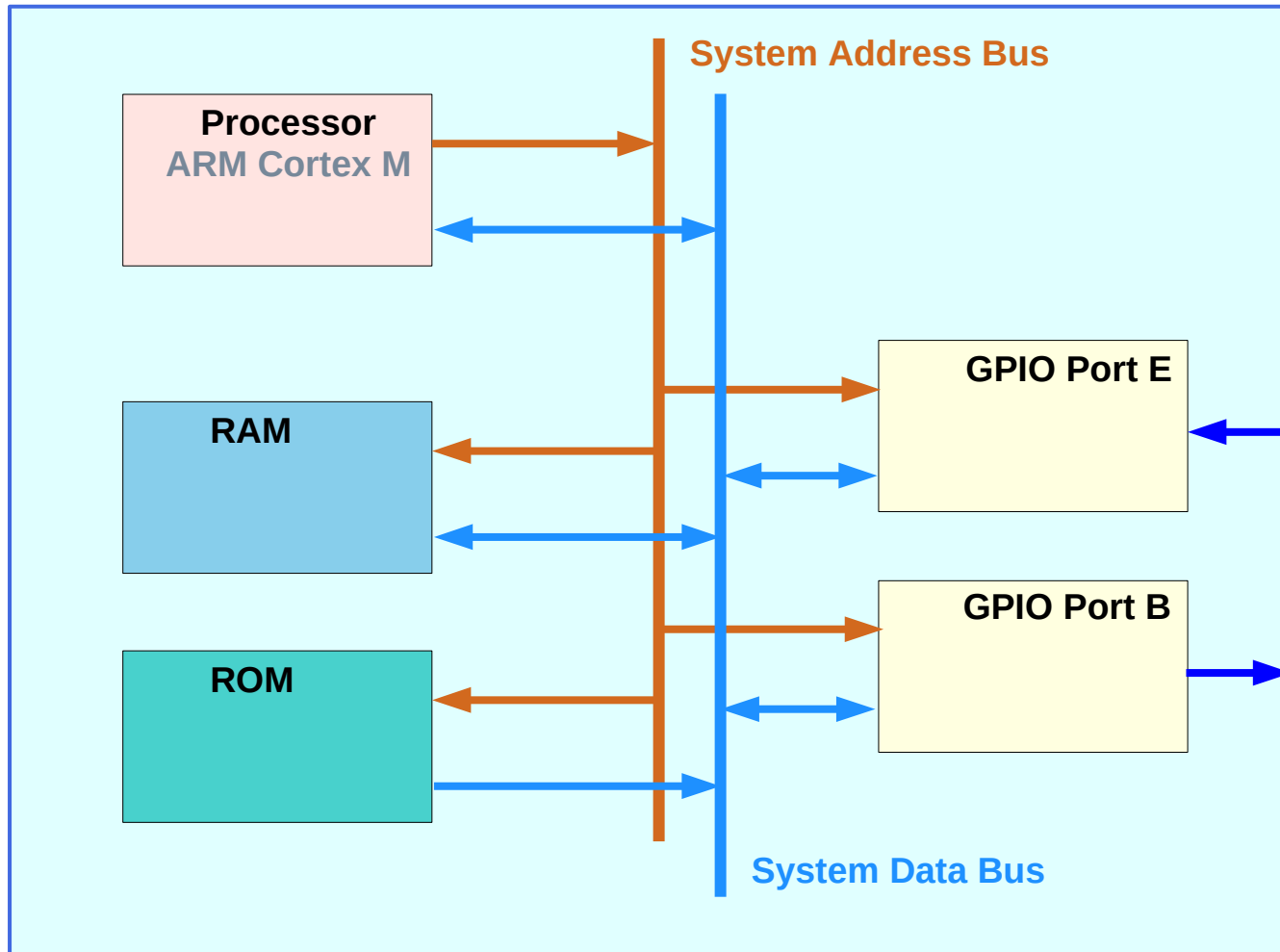
goE	00	goE	(R1)
	01	goE	(R4)
	10	waitE	(R4)
	11	waitE	(R5)

- (R1) If no cars are coming, stay in a **green** state, but which one doesn't matter. (00)
- (R2) To change from **green** to **red**, implement a **yellow** light of exactly **5** seconds
- (R3) **Green** lights will last at least **30** seconds
- (R4) If cars are only coming in one direction, move to and stay green in that direction. (10, 01)
- (R5) If cars are coming in both directions, cycle through all four states (11)

Some GPIO Control Registers

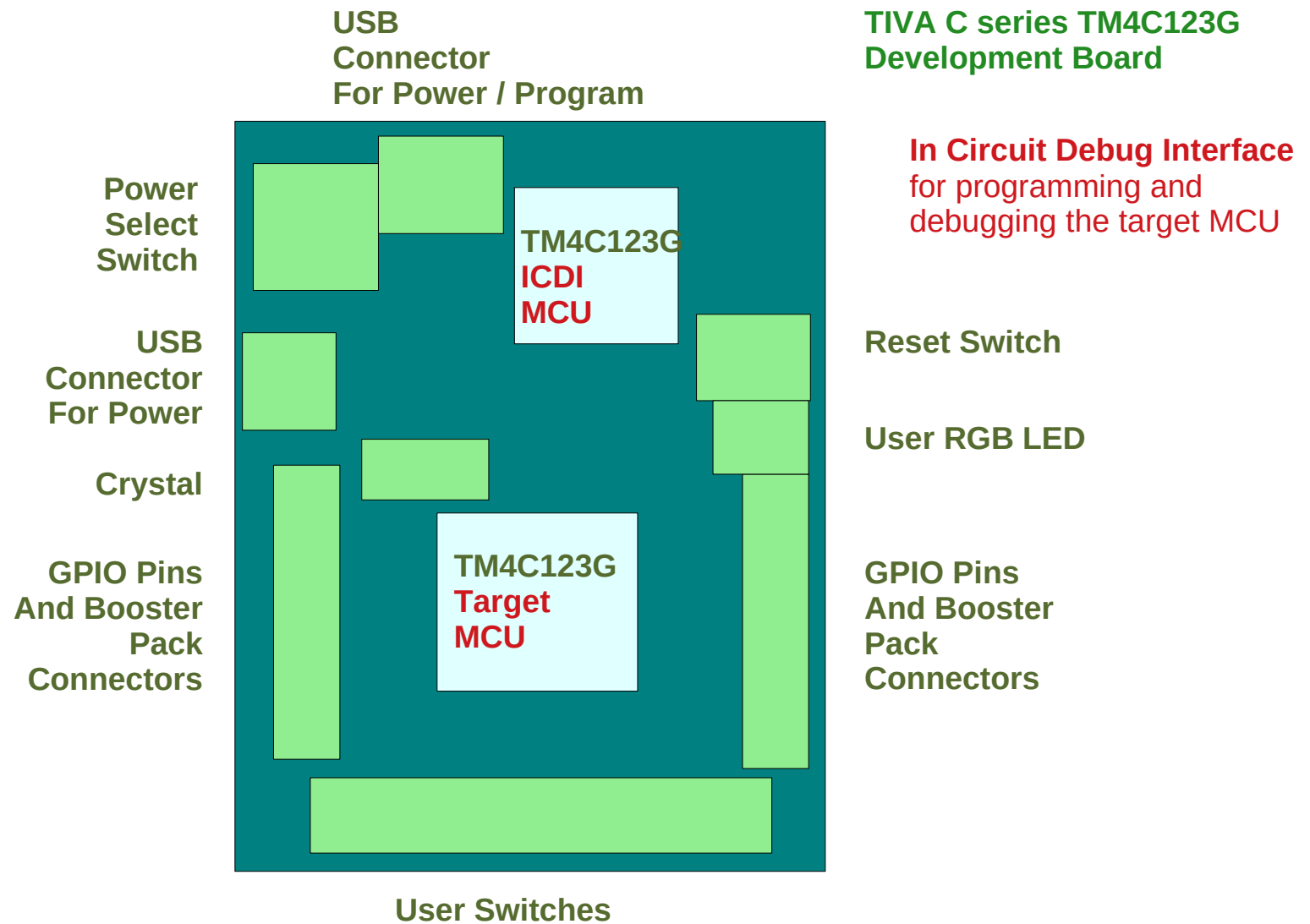


Simplified MicroController Architecture



MicroController
TI's TM4C123GH6PM

Simplified Development Board Architecture



<https://circuitdigest.com/microcontroller-projects/getting-started-with-tiva-c-series-tm4c123g-launchpad-from-texas-instruments>

Moore FSM – C code (1)

```
#define LIGHT *((volatile unsigned long *) 0x400050FC)
#define GPIO_PORTB_OUT *((volatile unsigned long *) 0x400050FC) // bits 5-0
#define GPIO_PORTB_DIR_R *((volatile unsigned long *) 0x40005400)
#define GPIO_PORTB_AFSEL_R *((volatile unsigned long *) 0x40005420)
#define GPIO_PORTB_DEN_R *((volatile unsigned long *) 0x4000551C)
#define GPIO_PORTB_AMSEL_R *((volatile unsigned long *) 0x40005528)
#define GPIO_PORTB_PCTL_R *((volatile unsigned long *) 0x4000552C)

#define SENSOR *((volatile unsigned long *) 0x4002400C)
#define GPIO_PORTE_IN *((volatile unsigned long *) 0x4002400C) // bits 1-0
#define GPIO_PORTE_DIR_R *((volatile unsigned long *) 0x40024400)
#define GPIO_PORTE_AFSEL_R *((volatile unsigned long *) 0x40024420)
#define GPIO_PORTE_DEN_R *((volatile unsigned long *) 0x4002451C)
#define GPIO_PORTE_AMSEL_R *((volatile unsigned long *) 0x40024528)
#define GPIO_PORTE_PCTL_R *((volatile unsigned long *) 0x4002452C)

#define SYSCTL_RCGC2_R *((volatile unsigned long *) 0x400FE108)
#define SYSCTL_RCGC2_GPIOE 0x00000010 // port E Clock Gating Control
#define SYSCTL_RCGC2_GPIOB 0x00000002 // port B Clock Gating Control
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – C code (2)

```
// Linked data structures
struct State {
    uint32_t      Out;
    uint32_t      Time;
    const struct State * Next[4];
};

Typedef 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}}, // 30 sec
    {0x22, 500,  {goE, goE,  goE,  goE}}, // 5 sec
    {0x0c, 3000, {goE, goE,  waitE, waitE}}, // 30 sec
    {0x14, 500,  {goN, goN,  goN,  goN}}   }; // 5 sec
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – C code (3)

```
int main (void) {  
    Styp *    Pt;                // state pointer  
    uint32_t  Input;  
  
    PLL_Init();                 // 50 MHz  
    SysTick_Init();  
    SYSCTL_RCGCGPIO_R |= 0x12; // 1) B E  
  
    while ((SYSCTI_PRGPIO_R & 0x12) != 0x12) { }; // ready?  
    // 2) no need to unlock  
}
```

Moore FSM – C code (4)

```
// 3) disable analog function on PE1-0
GPIO_PORTE_AMSEL_R &= ~0x03;
// 4) enable regular GPIO
GPIO_PORTE_PCTL_R &= ~0x000000FF;
// 5) inputs on PE1-0
GPIO_PORTE_DIR_R &= ~0x03;
// 6) regular function on PE1-0
GPIO_PORTE_AFSEL_R &= ~0x03;
// 7) enable digital on PE1-0
GPIO_PORTE_DEN_R |= 0x03;
```

```
GPIO_PORTE_AMSEL_R    &= ~0x03;
GPIO_PORTE_PCTL_R     &= ~0x000000FF;
GPIO_PORTE_DIR_R      &= ~0x03;
GPIO_PORTE_AFSEL_R    &= ~0x03;
GPIO_PORTE_DEN_R      |= 0x03;
```

Moore FSM – C code (5)

// 3) disable analog function on PE5-0

GPIO_PORTB_AMSEL_R &= ~0x3F;

// 4) enable regular GPIO

GPIO_PORTB_PCTL_R &= ~0x00FFFFFF;

// 5) inputs on PE1-0

GPIO_PORTB_DIR_R |= ~0x3F;

// 6) regular function on PE1-0

GPIO_PORTB_AFSEL_R &= ~0x3F;

// 7) enable digital on PE1-0

GPIO_PORTB_DEN_R |= 0x03;

```
GPIO_PORTB_AMSEL_R    &= ~0x3F;
GPIO_PORTB_PCTL_R     &= ~0x00FFFFFF;
GPIO_PORTB_DIR_R      |= ~0x3F;
GPIO_PORTB_AFSEL_R    &= ~0x3F;
GPIO_PORTB_DEN_R      |= 0x03;
```

Moore FSM – C code (6)

```
Pt = goN;
```

```
while (1) {  
    LIGHT = Pt->Out ;           // set lights  
    SysTick_Wait10min(Pt->Time);  
    Input = SENSOR;           // read sensors  
    Pt = Pt->Next[Input];       // state transition  
}  
}
```

```
#define LIGHT                (*((volatile unsigned long *) 0x400050FC))  
#define GPIO_PORTB_OUT      (*((volatile unsigned long *) 0x400050FC)) // bits 5-0  
#define SENSOR              (*((volatile unsigned long *) 0x4002400C))  
#define GPIO_PORTE_IN       (*((volatile unsigned long *) 0x4002400C)) // bits 1-0
```

Moore FSM – C code (7)

```
Pt = goN;

while (1) {
    LIGHT = Pt->Out ;           // set lights
    SysTick_Wait10min(Pt->Time);
    Input = SENSOR;           // read sensors
    Pt = Pt->Next[Input];     // state transition
}
}
```

<pre>// Linked data structures struct State { uint32_t Out; uint32_t Time; const struct State * Next[4]; }; Typedef struct State Styp; Styp * Pt; uint32_t Input;</pre>	<pre>#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 } }};</pre>
--	--

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – ARM assembly (1)

```
; Linked data structure
; Put in ROM
Out      EQU      0      ; offset for output
WAIT     EQU      4      ; offset for time
NEXT     EQU      8      ; offset for next

goN      10_0001 = 0x21
waitN    10_0010 = 0x22
goE      00_1100 = 0x0c
waitE    01_0100 = 0x14
```

```
goN      DCD      0x21      ; North green, East red
          DCD      3000      ; 30 sec
          DCD      goN, waitN, goN, waitN
```

```
waitN    DCD      0x22      ; North yello, East red
          DCD      500       ; 5 sec
          DCD      goE, goE, goE, goE
```

```
goE      DCD      0x0c      ; North red, East green
          DCD      3000      ; 30 sec
          DCD      goE, goE, waitE, waitE
```

```
waitE    DCD      0x14      ; North red, East yellow
          DCD      500       ; 5 sec
          DCD      goN, goN, goN, goN
```

Moore FSM – ARM assembly (2)

Start

```
BL    PLL_Init           ; 50 MHz clock
BL    SysTick_Init      ; enable SysTick


---


LDR   R1, =SYSCTL_RCGCGPIO_R
LDR   R0, [R1]
ORR   R0, R0, #0x12     ; activate B E
STR   R0, [R1]
NOP
NOP                       ; allow time to finish


---


```

Moore FSM – ARM assembly (3)

```
LDR    R1, =GPIO_PORTE_AMSEL_R
LDR    R0, [R1]
BIC    R0, R0, #0x03                ; no analog
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTE_PCTL_R
LDR    R0, [R1]
BIC    R0, R0, #0x000000FF        ; PE1-0
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTE_DIR_R
LDR    R0, [R1]
BIC    R0, R0, #0x03                ; PE1-0 input
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTE_AFSEL_R
LDR    R0, [R1]
BIC    R0, R0, #0x03                ; no alt function
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTE_DEN_R
LDR    R0, [R1]
ORR    R0, R0, #0x03                ; enable PE1-0
STR    R0, [R1]
```

Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – ARM assembly (4)

```
LDR    R1, =GPIO_PORTB_AMSEL_R
LDR    R0, [R1]
BIC    R0, R0, #0x3F                ; no analog
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTB_PCTL_R
LDR    R0, [R1]
BIC    R0, R0, #0x00FFFFFF         ; PB5-0
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTB_DIR_R
LDR    R0, [R1]
ORR    R0, R0, #0x3F                ; PB5-0 output
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTB_AFSEL_R
LDR    R0, [R1]
BIC    R0, R0, #0x3F                ; no alt function
STR    R0, [R1]
-----
LDR    R1, =GPIO_PORTB_DEN_R
LDR    R0, [R1]
ORR    R0, R0, #0x3F                ; enable PB5-0
STR    R0, [R1]
```

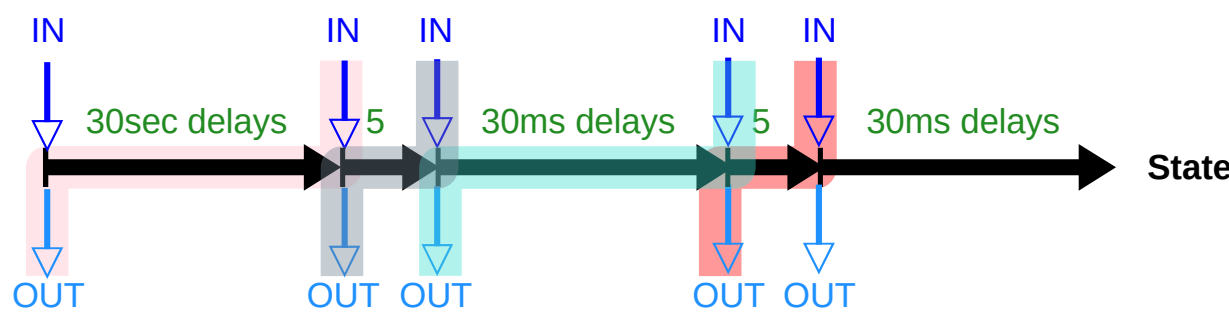
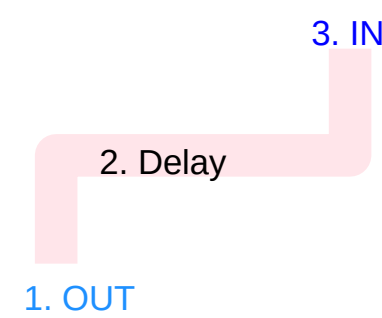
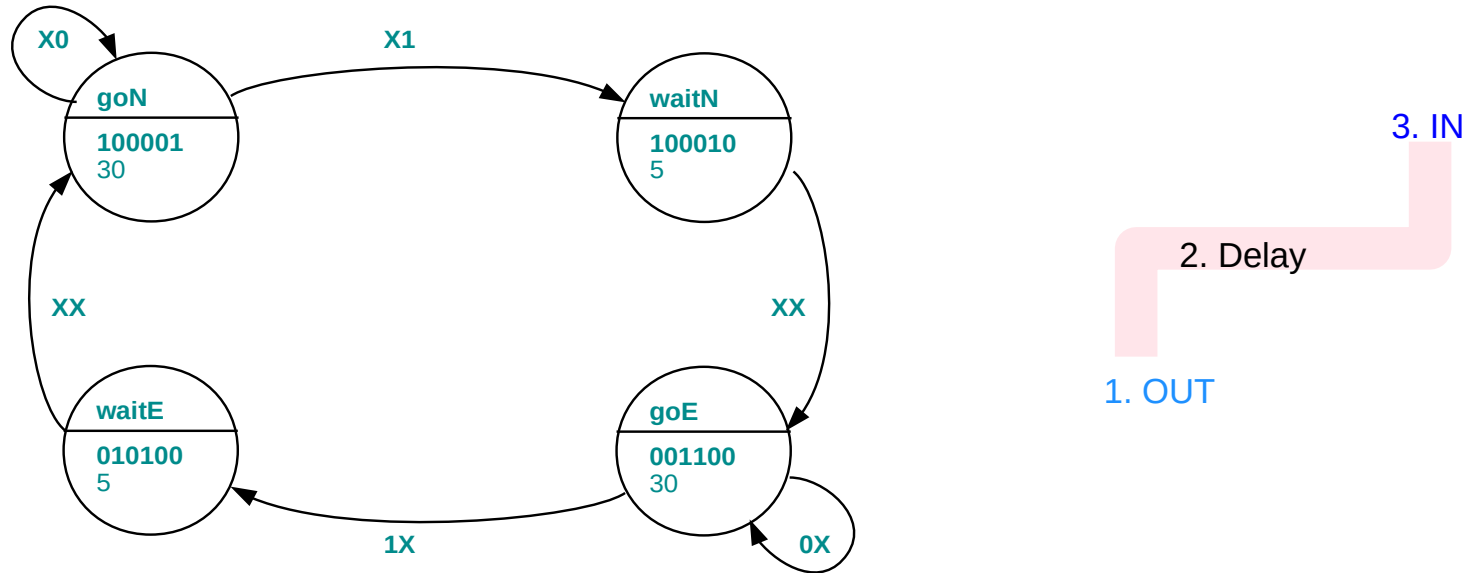
Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – ARM assembly (5)

```
FSM:
    LDR    R4, =goN           ; state pointer
    LDR    R5, =SENSOR       ; 0x4002400C
    LDR    R6, =LIGHT        ; 0x400050FC
    LDR    R0, [R4, #OUT]    ; output value
    STR    R0, [R6]          ; set lights
    LDR    R0, [R4, #WAIT]   ; time delay
    BL     SysTick_Wait10ms
    LDR    R0, [R5]          ; read input
    LSL    R0, R0, #2        ; 4 bytes/address
    ADD    R0, R0, #NEXT     ; 8, 12, 16, 20
    LDR    R4, [R4, R0]     ; go to next state
    B      FSM
```

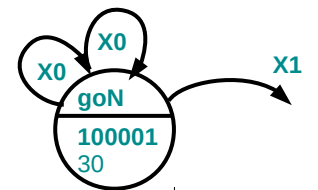
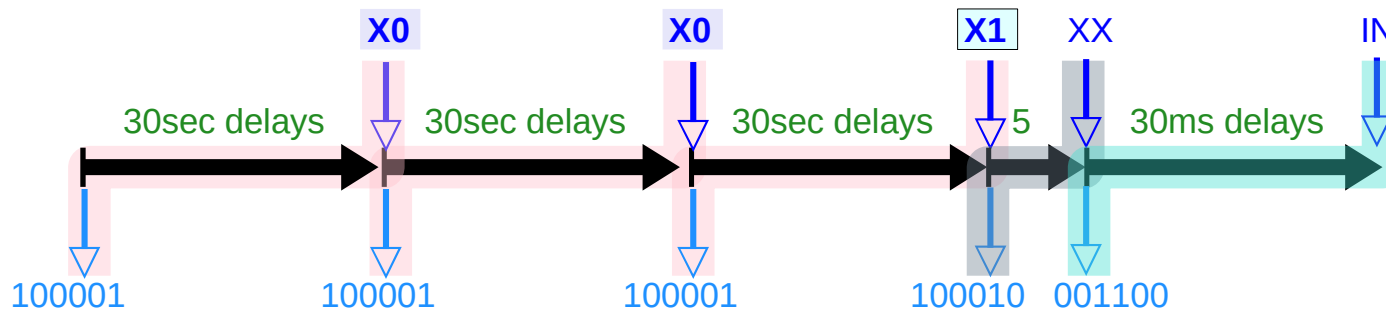
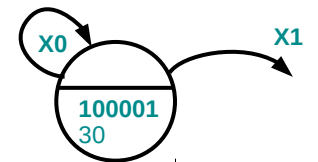
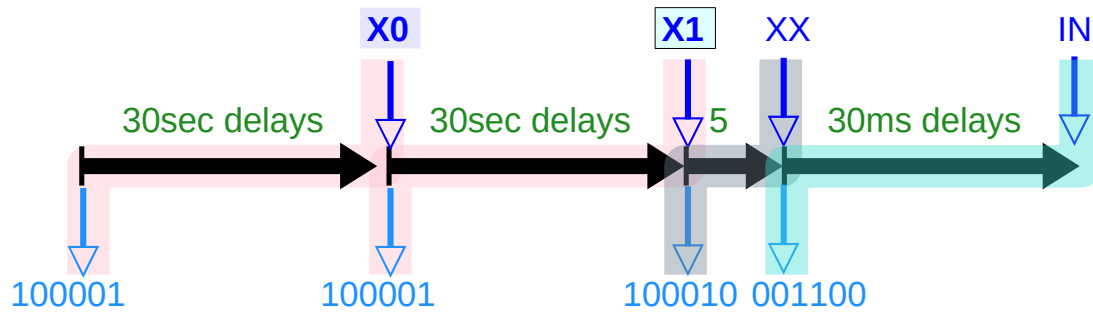
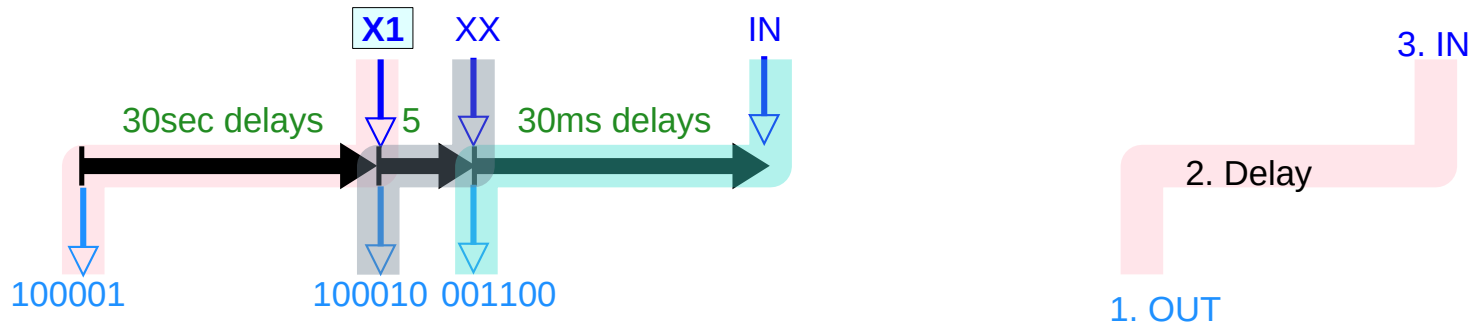
```
#define LIGHT          (*((volatile unsigned long *) 0x400050FC))
#define GPIO_PORTB_OUT  (*((volatile unsigned long *) 0x400050FC)) // bits 5-0
#define SENSOR         (*((volatile unsigned long *) 0x4002400C))
#define GPIO_PORTE_IN  (*((volatile unsigned long *) 0x4002400C)) // bits 1-0
```

Moore FSM – ARM assembly (6)



Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – ARM assembly (7)



Introduction to ARM Cortex-M Microcontrollers – Embedded Systems, Jonathan W. Valvano

Moore FSM – ARM assembly (8)

```

FSM:
    LDR    R4, =goN           ; state pointer
    LDR    R5, =SENSOR        ; 0x4002400C
    LDR    R6, =LIGHT         ; 0x400050FC
    LDR    R0, [R4, #OUT]     ; output value
    STR    R0, [R6]           ; set lights
    LDR    R0, [R4, #WAIT]    ; time delay
    BL     SysTick_Wait10ms
    LDR    R0, [R5]           ; read input
    LSL    R0, R0, #2         ; 4 bytes/address 0, 4, 12, 16
    ADD    R0, R0, #NEXT      ; 8, 12, 16, 20
    LDR    R4, [R4, R0]       ; go to next state
    B     FSM
  
```

```

Out EQU 0 ; offset for output
WAIT EQU 4 ; offset for time
NEXT EQU 8 ; offset for next
  
```

```

struct State {
    uint32_t Out;
    uint32_t Time;
    const struct State * Next[4];
};
  
```

R4 + #OUT	Out
R4 + #WAIT	Time
R4 + #NEXT	Next

Moore FSM – ARM assembly (9)

```

goN  DCD 0x21
      DCD 3000
      DCD goN, waitN, goN, waitN
waitN DCD 0x22
      DCD 500
      DCD goE, goE, goE, goE
goE   DCD 0x0c
      DCD 3000
      DCD goE, goE, waitE, waitE
waitE DCD 0x14
      DCD 500
      DCD goN, goN, goN, goN
    
```

goN	Out			
	Time			
	Next[0]	Next[1]	Next[2]	Nex[3]
waitN	Out			
	Time			
	Next[0]	Next[1]	Next[2]	Nex[3]
goE	Out			
	Time			
	Next[0]	Next[1]	Next[2]	Nex[3]
waitE	Out			
	Time			
	Next[0]	Next[1]	Next[2]	Nex[3]

```

struct State {
  uint32_t      Out;
  uint32_t      Time;
  const struct State * Next[4];
};
    
```

R4 = goN / waitN / goE / waitE

R4 + #OUT	Out	R4 + #0
R4 + #WAIT	Time	R4 + #4
R4 + #NEXT	Next[0]	R4 + #8

R4 + #NEXT	Next[0]	Next[1]	Next[2]	Nex[3]
	Input=00	Input=01	Input=10	Input=11
	4*R5=0	4*R5=4	4*R5=8	4*R5=12

Pointer access to an array

References

- [1] <ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf>
- [2] <https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf>