Reading the Encoder Switch

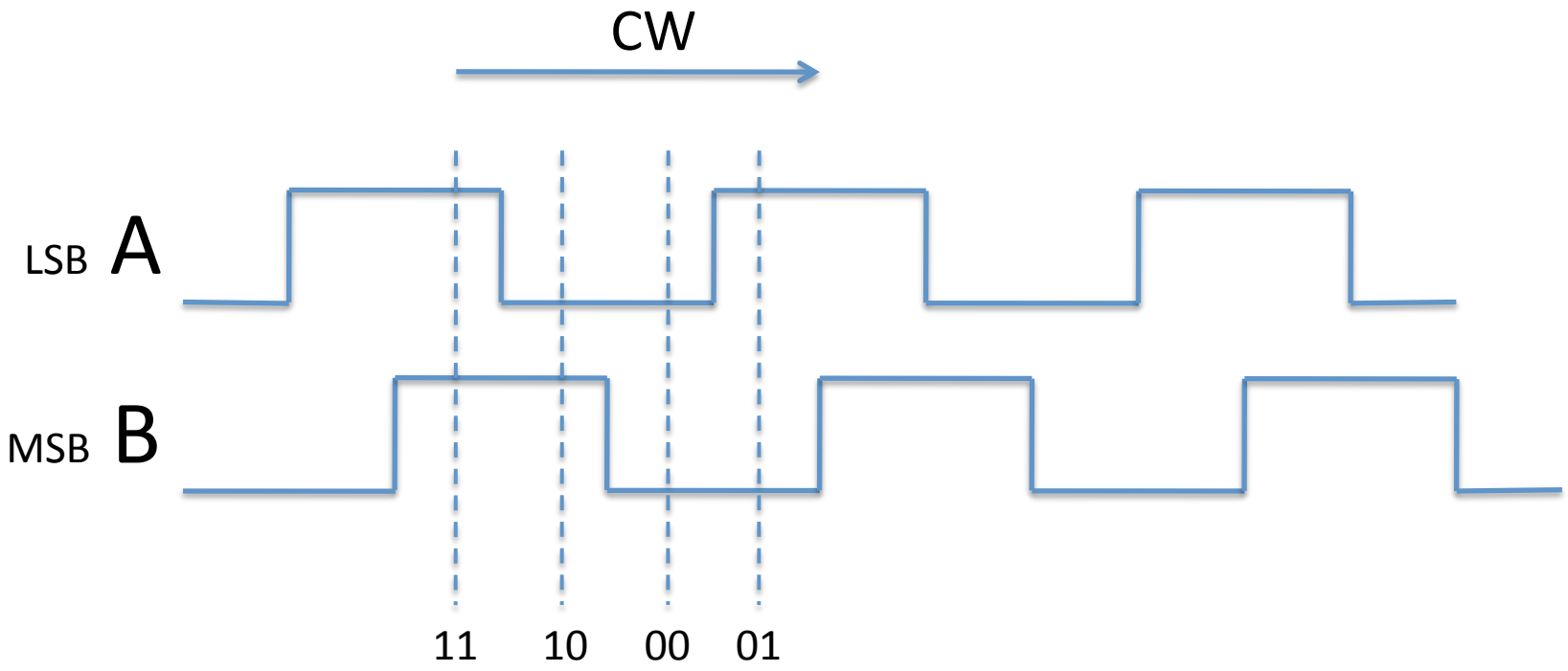
Three Methods

State Machine

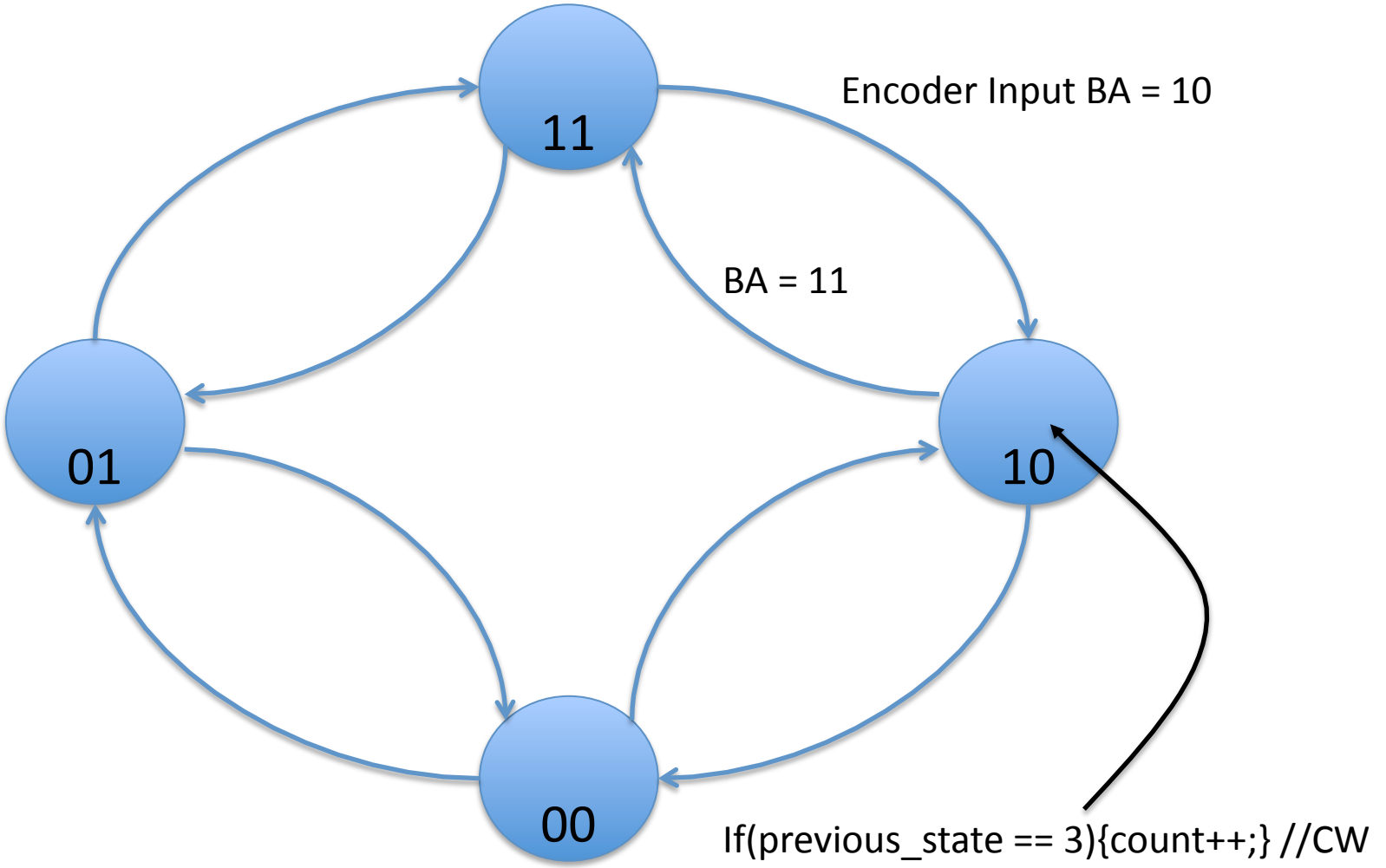
Table

A-Tracking

Encoder Switch Phase Diagram



State Machine



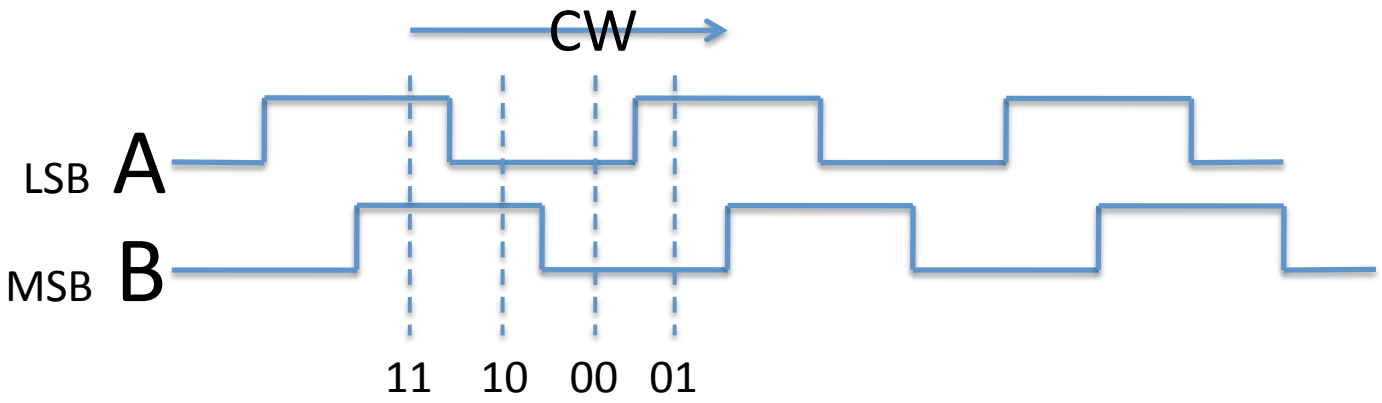
State Machine con't

- When reaching State 11:
- Look at count:
  - if positive → CW
  - If negative → CCW
- Execution Time: 3.318uS
- Code Size: 366 bytes
- Variable Size: 3 bytes

State Machine Code

```
switch(alarm_state2){

        case 0:
          if(past_astate2 == 2){acount2++;}    // CW
          if(past_astate2 == 1){acount2--;}     // CCW
          if(encoder2 == 1){alarm_state2 = 1;}
          if(encoder2 == 2){alarm_state2 = 2;}
        break;

        case 1:
          if(past_astate2 == 0){acount2++;}    // CW
          if(past_astate2 == 3){acount2--;}      // CCW
          if(encoder2 == 3){alarm_state2 = 3;}
          if(encoder2 == 0){alarm_state2 = 0;}
        break;

        case 2:
          if(past_astate2 == 3){acount2++;}    // CW
          if(past_astate2 == 0){acount2--;}     // CCW
          past_astate2 = alarm_state2;
          if(encoder2 == 0){alarm_state2 = 0;}
          if(encoder2 == 3){alarm_state2 = 3;}
        break;

        case 3:
          if(past_astate2 == 1){acount2++;}
          if(past_astate2 == 2){acount2--;}
          past_astate2 = alarm_state2;
          if((acount2 >= 1) && (acount2 <100)){
                time_alarm++;
                 if(time_alarm > 1439){time_alarm = 0;}}
        if((acount2 <= 0xFF) && (acount2 > 0x90)){
                time_alarm--;
                 if(time_alarm > 1439){time_alarm = 1439;}}
          acount2 = 0;
          if(encoder2 == 2){alarm_state2 = 2;}          // CW
          If(encoder2 == 1){alarm_state2 = 1;}          // CCW
          if(encoder2 == 0){alarm_state2 = 0;}
        break;

        default:
          alarm_state2 = 3;
          acount2 = 0;       // no action
 }      // switch

past_astate2 = alarm_state2;
```

Table Method



- Track Previous Phase and Current Phase
- Concatenate Previous & Current
- Example: Previous BA =01 & Current BA=11
- Index = 0111
- 4 bits! Build a table

Direction Table

Previous		Current		
B	A	B	A	Direction
0	0	0	0	NA
0	0	0	1	CW
0	0	1	0	CCW
0	0	1	1	NA
0	1	0	0	CCW
0	1	0	1	NA
0	1	1	0	NA
0	1	1	1	CW
1	0	0	0	CW
1	0	0	1	NA
1	0	1	0	NA
1	0	1	1	CCW
1	1	0	0	NA
1	1	0	1	CCW
1	1	1	0	CW
1	1	1	1	NA

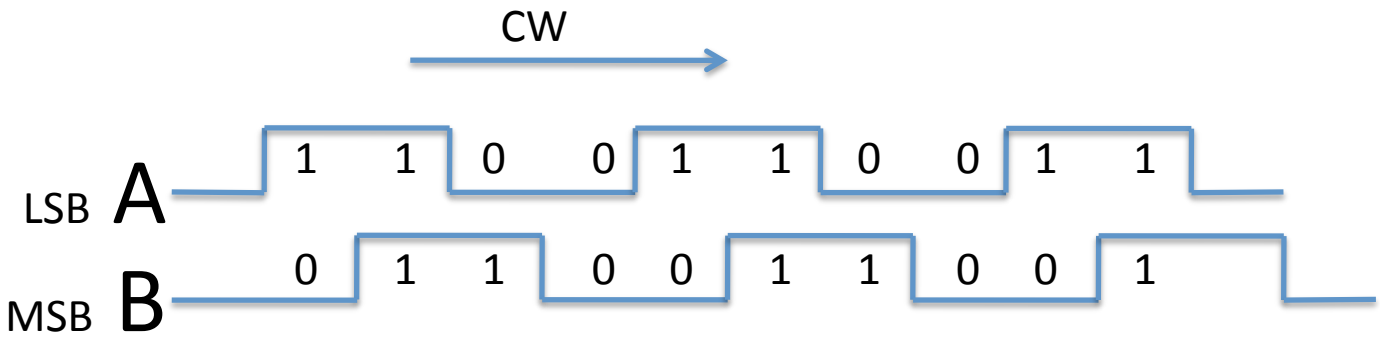
Table con’t

- When reaching State 11:
- Look at count:
  - if positive → CW
  - If negative → CCW
- Execution Time: 2.57uS
- Code Size: 142 bytes
- Variable Size: 18 bytes

Table Code

```
#define CW 1
#define CCW 2
        static uint8_t sw_table[] = {0, 1, 2, 0, 2, 0, 0, 1, 1, 0, 0, 2, 0, 2, 1, 0};
        uint8_t sw_index = 0;
        uint8_t direction = 0;
        static uint8_t acount2  = 0;
        static uint8_t previous_encoder2 = 0;

        sw_index = (previous_encoder2<<2) | encoder2;
        direction = sw_table[sw_index];
        if(direction == CW){acount2++;}
        if(direction == CCW){acount2--;}
        if(encoder2 ==3){
                if((acount2 > 1) && (acount2 < 100)){
                        time_alarm++;
                        if(time_alarm >1439){time_alarm = 0;}}
                if((acount2 <= 0xFF) && (acount2 > 0x90)){
                        time_alarm--;
                        if(time_alarm >1439){time_alarm = 1439;}}
        acount2 = 0;
        }
        previous_encoder2 = encoder2;
```

A Tracking



- Previous & Current of Track A
  - Example: PC = 11
- Four combinations: PC = 11, 10, 00, 01
- Determine PC for track A, look at track B
- Test for Direction

A Tracking con’t

- When reaching State 11:
- Look at count:
  - if positive → CW
  - If negative → CCW
- Execution Time: 3.134uS
- Code Size: 256 bytes
- Variable Size: 3 bytes

A Tracking Code

```
#define CW 1
#define CCW 2
        a_current = encoder2 & 0x01;
        b_current = (encoder2>>1) & 0x01;

        if(a_past == a_current){
        if((a_current == 1) && (b_past < b_current)){direction = CW;}
        if((a_current == 1) && (b_past > b_current)){direction = CCW;}
        if((a_current == 0) && (b_past > b_current)){direction = CW;}
        if((a_current == 0) && (b_past < b_current)){direction = CCW;}
        }
        if((a_past < a_current) && ((b_past | b_current) == 0)){direction = CW;}
        if((a_past < a_current) && ((b_past | b_current) == 1)){direction = CCW;}
        if((a_past > a_current) && ((b_past | b_current) == 1)){direction = CW;}
        if((a_past > a_current) && ((b_past | b_current) == 0)){direction = CCW;}

        //increment alarm count
        // test for over/under flows

        a_past = a_current;
        b_past = b_current;
```