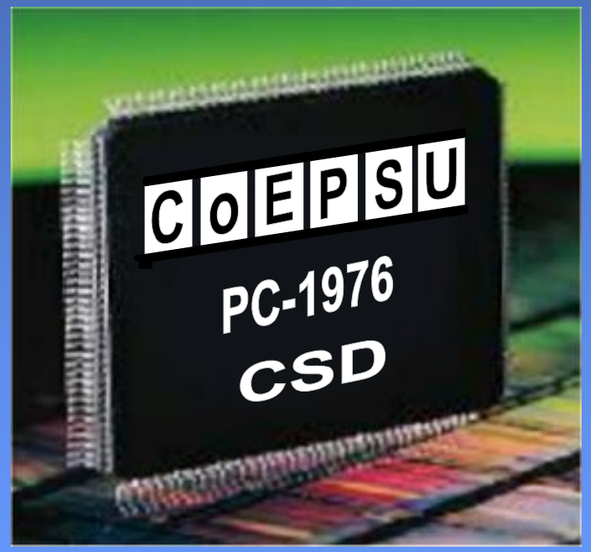
Department of Computer Engineering
Faculty of Engineering
Prince of Songkla University

240-319

Embedded System Developer Module



March 23, 2023

Associate Prof. Dr. Panyayot Chaikan
panyayot@coe.psu.ac.th

Chapter 5

การแปลงแอนะล็อกเป็นดิจิทัล

เนื้อหา

วงจรแปลงแอนะล็อกเป็นดิจิทัล (Analog-to-Digital Converter: ADC)

เรจิสเตอร์ควบคุมวงจรแปลงแอนะล็อกเป็นดิจิทัล

การแปลงแบบการประมาณสืบเนื่อง

การเขียนโปรแกรมควบคุมวงจรแปลงแอนะล็อกเป็นดิจิทัล

ตัวอย่างวงจรประยุกต์ใช้งานวงจรแปลงแอนะล็อกเป็นดิจิทัล

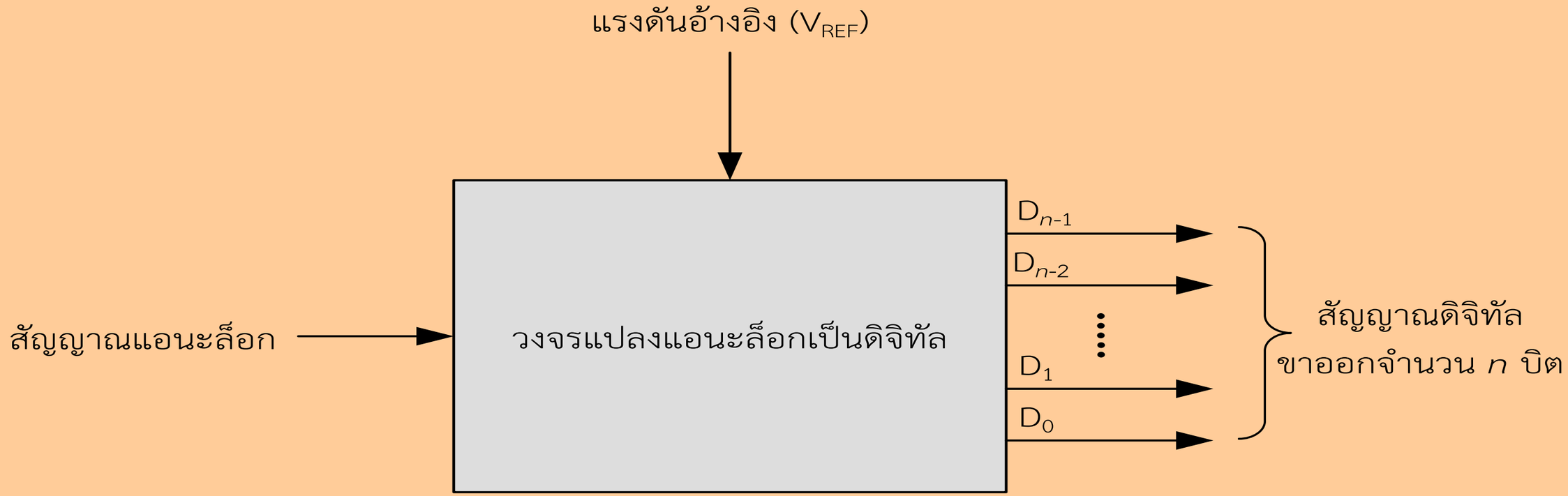
การสุ่มตัวอย่างสัญญาณแอนะล็อก

Chapter 5

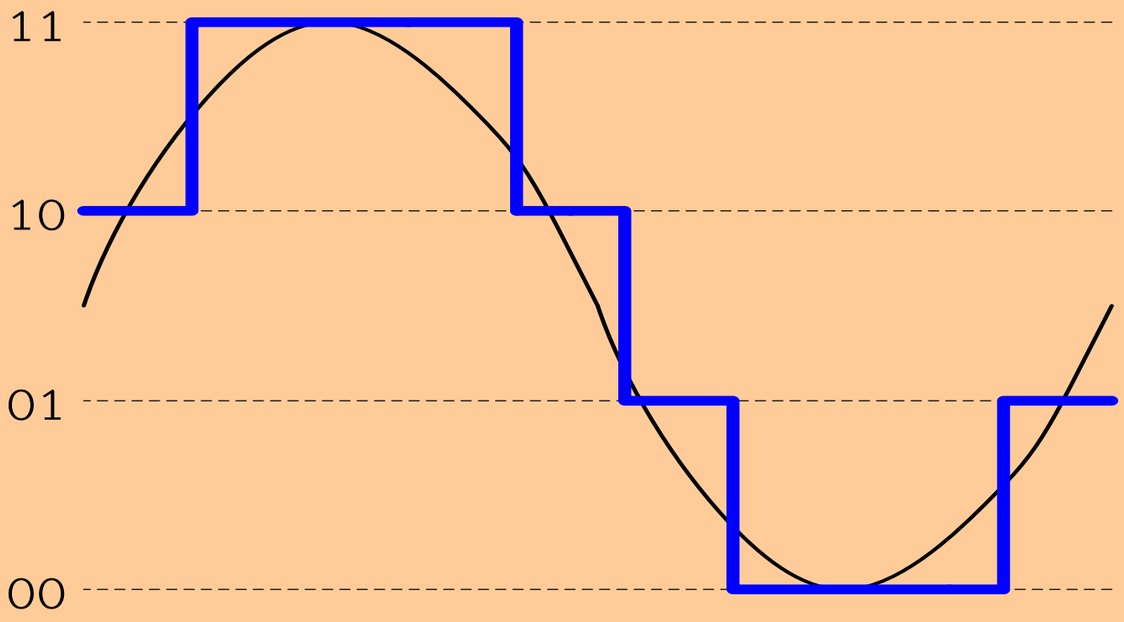
Analog-to-Digital Converter

ตอนที่ 1

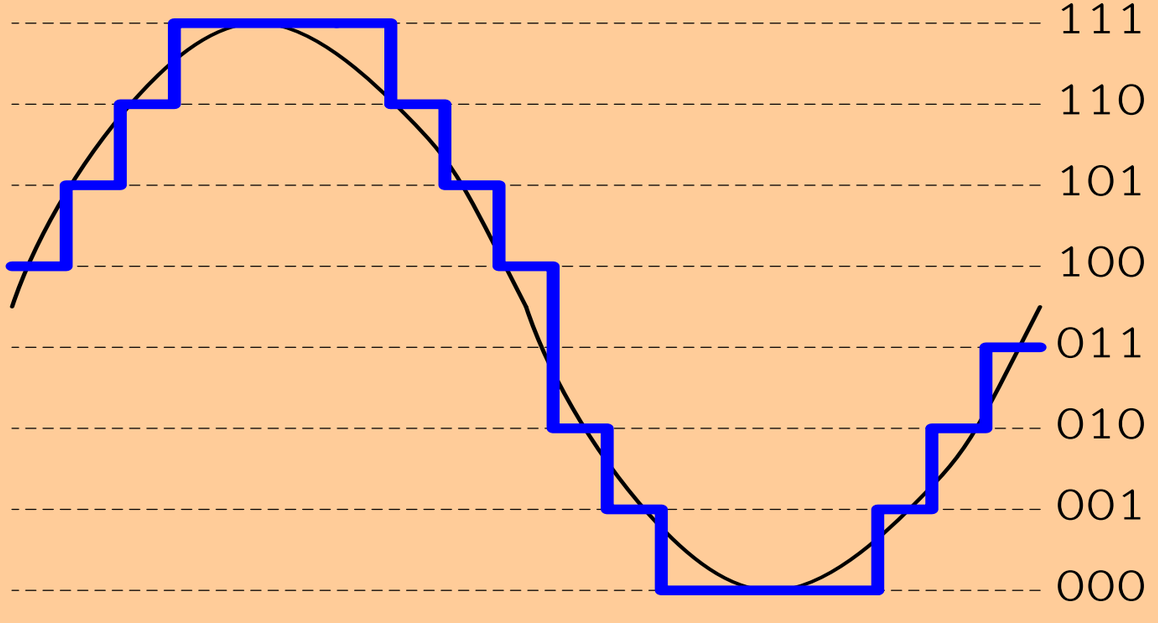
ADC:Analog to Digital Converter



ความละเอียดของการแปลง

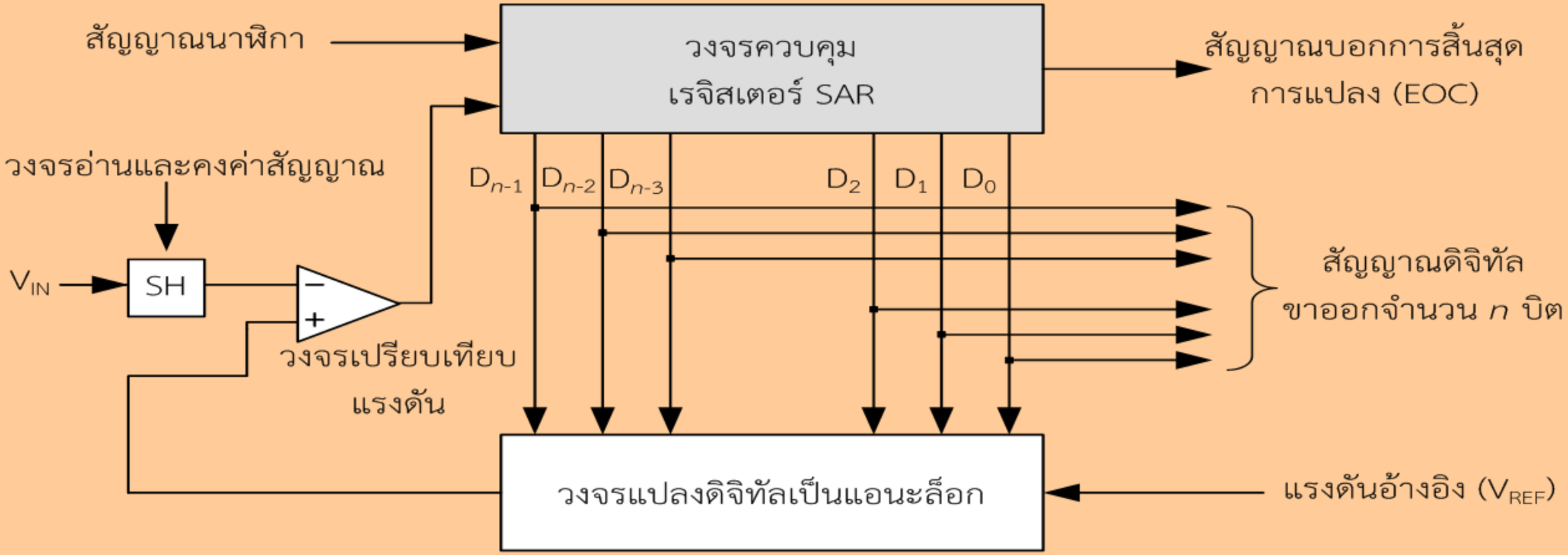


การแปลงที่ความละเอียด 2 บิต



การแปลงที่ความละเอียด 3 บิต

การแปลงแบบการประมาณสืบเนื่อง (SAR)



วงจรแปลงแอนะล็อกเป็นดิจิทัลใน AVR

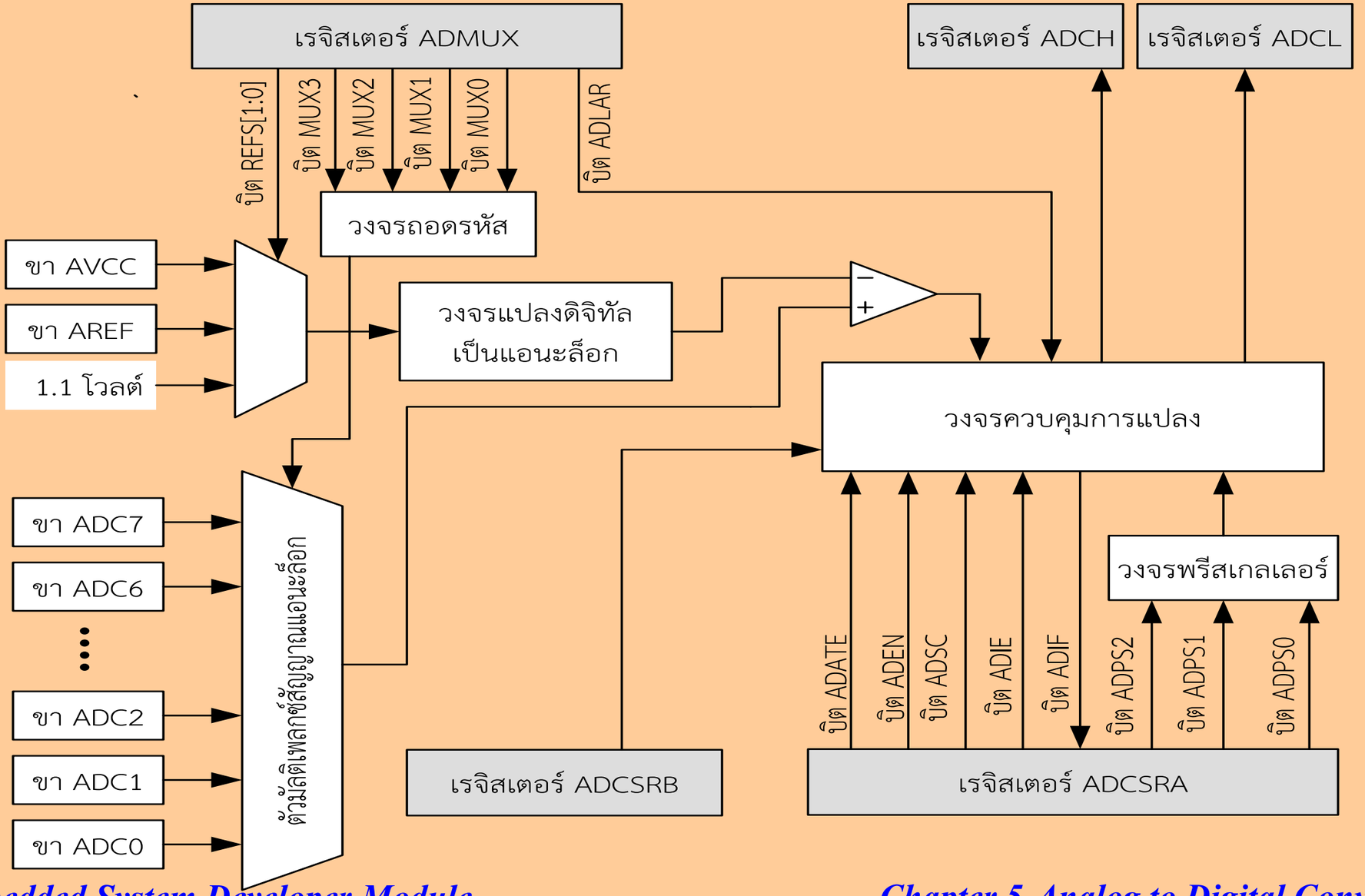
- ความละเอียดของวงจรแปลงขนาด 10 บิต
- ตัวแปลงแบบการประมาณสืบเนื่อง (Successive Approximation)

$$ADC = \frac{V_{IN} * 1024}{V_{REF}}$$

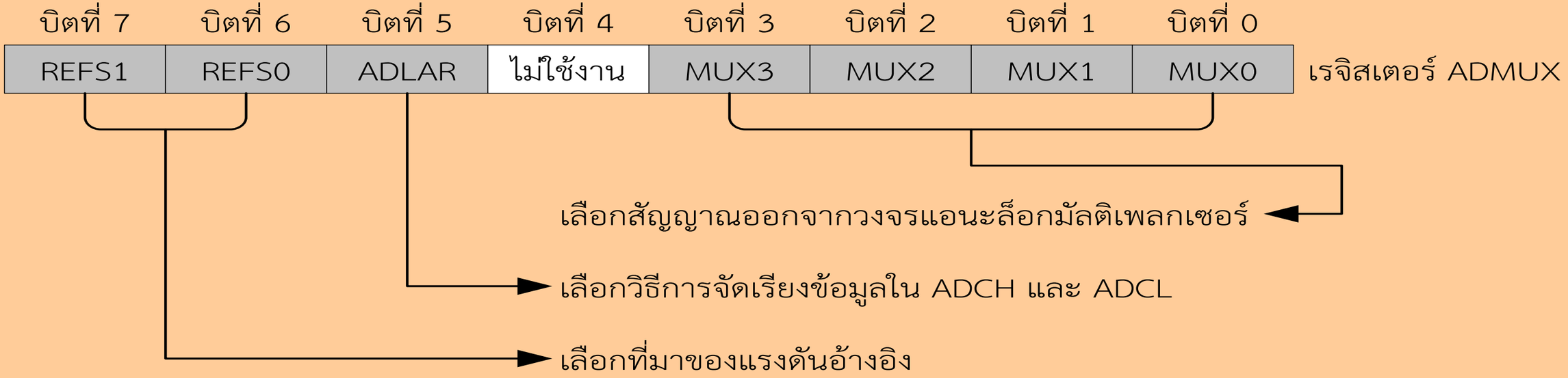
ตัวอย่าง

Vin	Vref	Digital Out
0.3	5	61
0.4	5	82
0.5	5	102
0.6	5	123
0.7	5	143
0.8	5	164
…	…	…
…	…	….
4.6	5	942
4.7	5	963
4.8	5	983
4.9	5	1004
5	5	1023

ADC of AVR


เรจิสเตอร์ ADMUX
เรจิสเตอร์ ADCH
เรจิสเตอร์ ADCL
บิต REFS[1:0]
บิต MUX3
บิต MUX2
บิต MUX1
บิต MUX0
บิต ADLAR
วงจรถอดรหัส
ขา AVCC
ขา AREF
1.1 โวลต์
วงจรแปลงดิจิทัล
เป็นแอนะล็อก
วงจรควบคุมการแปลง
ขา ADC7
ขา ADC6
ขา ADC2
ขา ADC1
ขา ADC0
ตัวมัลติเพลกซ์สัญญาณแอนะล็อก
วงจรพรีสเกลเลอร์
บิต ADATE
บิต ADEN
บิต ADSC
บิต ADIE
บิต ADIF
บิต ADPS2
บิต ADPS1
บิต ADPS0
เรจิสเตอร์ ADCSRB
เรจิสเตอร์ ADCSRA

ADMUX



บิตควบคุม		การเลือกค่าระดับแรงดันอ้างอิง
REFS1	REFS0	
0	0	รับจากขา AREF
0	1	รับจากขา AVCC
1	0	ถูกสงวนเอาไว้ มิให้ใช้งาน
1	1	รับจากค่าแรงดันอ้างอิงภายในค่า 1.1 โวลต์

วิธีการจัดเรียงข้อมูลใน ADCH และ ADCL

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0	
-	-	-	-	-	-	ADC9	ADC8	เรจิสเตอร์ ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	เรจิสเตอร์ ADCL

เมื่อ ADLAR=0

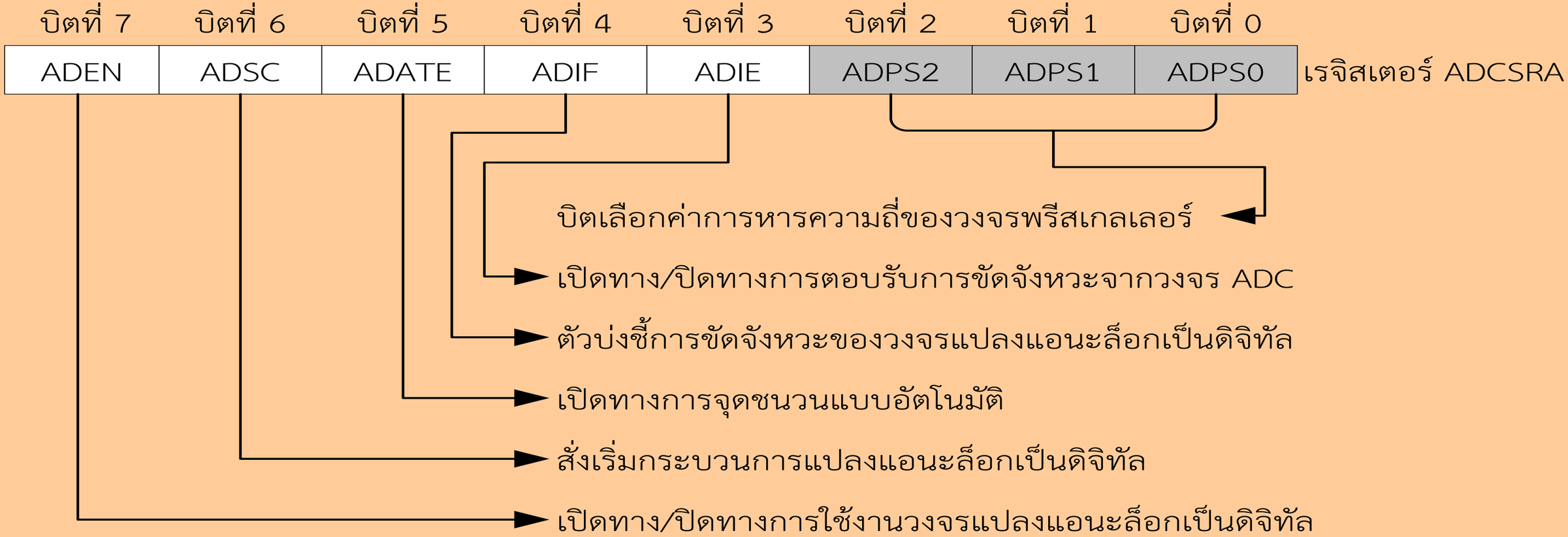
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	เรจิสเตอร์ ADCH
ADC1	ADC0	-	-	-	-	-	-	เรจิสเตอร์ ADCL

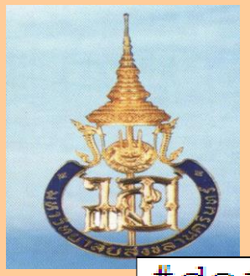
เมื่อ ADLAR=1

ADMUX

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8(1)
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

ADCSRA – ADC Control and Status Register A





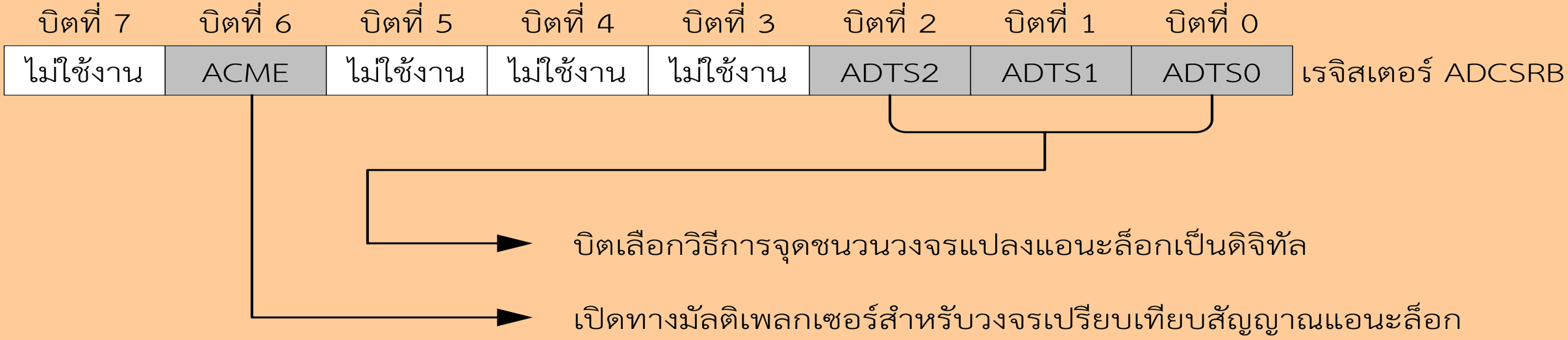
ค่าเวกเตอร์การขัดจังหวะในภาษาซี

```
#define INT0_vect              _VECTOR(1)    /* External Interrupt Request 0 */
#define INT1_vect              _VECTOR(2)    /* External Interrupt Request 1 */
#define PCINT0_vect            _VECTOR(3)    /* Pin Change Interrupt Request 0 */
#define PCINT1_vect            _VECTOR(4)    /* Pin Change Interrupt Request 1 */
#define PCINT2_vect            _VECTOR(5)    /* Pin Change Interrupt Request 2 */
#define WDT_vect               _VECTOR(6)    /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect      _VECTOR(7)    /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect      _VECTOR(8)    /* Timer/Counter2 Compare Match A */
#define TIMER2_OVF_vect        _VECTOR(9)    /* Timer/Counter2 Overflow */
#define TIMER1_CAPT_vect       _VECTOR(10)   /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect      _VECTOR(11)   /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect      _VECTOR(12)   /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect        _VECTOR(13)   /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect      _VECTOR(14)   /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect      _VECTOR(15)   /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect        _VECTOR(16)   /* Timer/Couner0 Overflow */
#define SPI_STC_vect           _VECTOR(17)   /* SPI Serial Transfer Complete */
#define USART_RX_vect          _VECTOR(18)   /* USART Rx Complete */
#define USART_UDRE_vect        _VECTOR(19)   /* USART, Data Register Empty */
#define USART_TX_vect          _VECTOR(20)   /* USART Tx Complete */
#define ADC_vect               _VECTOR(21)   /* ADC Conversion Complete */
#define EE_READY_vect          _VECTOR(22)   /* EEPROM Ready */
#define ANALOG_COMP_vect       _VECTOR(23)   /* Analog Comparator */
#define TWI_vect               _VECTOR(24)   /* Two-wire Serial Interface */
#define SPM_READY_vect         _VECTOR(25)   /* Store Program Memory Read */
```

การตั้งค่าใน ADPS[2:0]

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ADCSRB – ADC Control and Status Register B



วิธีการจุดชนวนวงจรแปลงแอนะล็อกเป็นดิจิทัล

ADTS2	ADTS1	ADTS0	แหล่งกำเนิดสัญญาณจุดชนวน
0	0	0	สั่งให้ตัวแปลงทำงานในแบบวิธีวิ่งอิสระ
0	0	1	วงจรเปรียบเทียบสัญญาณแอนะล็อก
0	1	0	จากขา INT0
0	1	1	วงจรนับ/จับเวลาหมายเลข 0 เมื่อค่าใน TCNT0 เท่ากับ OCR0A
1	0	0	วงจรนับ/จับเวลาหมายเลข 0 เมื่อตัวบ่งชี้ TOV0 มีค่าเป็นตรรกะสูง
1	0	1	วงจรนับ/จับเวลาหมายเลข 1 เมื่อค่าใน TCNT1 เท่ากับ OCR1B
1	1	0	วงจรนับ/จับเวลาหมายเลข 1 เมื่อตัวบ่งชี้ TOV1 มีค่าเป็นตรรกะสูง
1	1	1	หน่วยดักจับ (Input Capture Unit) ของวงจรนับ/จับเวลาหมายเลข 1

การเขียนโปรแกรมควบคุมวงจร ADC

- ขั้นที่ 1 – เลือกขารับเข้าสัญญาณแอนะล็อก และตั้งค่าตัวมัลติเพลกซ์สัญญาณแอนะล็อกให้รับค่าจากขาสัญญาณที่ถูกเลือกเอาไว้
- ขั้นที่ 2 – เลือกแบบวิธีการทำงานของวงจรแปลงแอนะล็อกเป็นดิจิทัล ซึ่งสามารถเลือกได้ 3 รูปแบบ
  - แบบวิธีแปลงครั้งเดียว
  - แบบวิธีวิ่งอิสระ
  - แบบวิธีจุดชนวน หากเลือกแบบวิธีจุดชนวนจะต้องมีการเลือกแหล่งกำเนิดสัญญาณจุดชนวนด้วย

การเขียนโปรแกรมควบคุมวงจร ADC

- ขั้นที่ 3 – เลือกวิธีการเรียงข้อมูลในเรจิสเตอร์ ADCH และ ADCL
- ขั้นที่ 4 – เลือกค่าตัวหารความถี่ของวงจรพรีสเกลเลอร์ โดยคำนึงจากความถี่ของสัญญาณแอนะล็อกที่รับเข้ามา
- ขั้นที่ 5 – เปิดทางการทำงานของวงจรแปลงแอนะล็อกเป็นดิจิทัล
- ขั้นที่ 6 – สั่งให้บิต ADSC เป็นตรรกะสูงเพื่อให้วงจรแปลงเริ่มการแปลงสัญญาณ
- ขั้นที่ 7 – เปิดทางการตอบรับการขัดจังหวะจากมอดูลแปลงแอนะล็อกเป็นดิจิทัล

Chapter 5

Analog-to-Digital Converter

ตอนที่ 2

แบบวิธีในการทำงานของวงจร ADC

- วงจรแปลงแอนะล็อกเป็นดิจิทัลของเอวีอาร์ สามารถเลือกแบบวิธีการทำงานได้ 3 รูปแบบ
  - แบบวิธีแปลงครั้งเดียว
  - แบบวิธีวิ่งอิสระ
  - แบบวิธีจุดชนวน

แบบวิธีแปลงครั้งเดียว (Single Conversion mode)

- ผู้ใช้จะต้องเขียนค่าตรรกะสูงลงสู่บิต ADSC (ADC Start Conversion)
- บิต ADSC จะคงค่าที่ตรรกะสูงจนกว่ากระบวนการแปลงสัญญาณจะเสร็จสิ้นแล้วจะกลับมาเป็นค่าตรรกะต่ำโดยอัตโนมัติ
- หากผู้ใช้ต้องการให้เกิดการแปลงสัญญาณในรอบถัดไป ก็เป็นหน้าที่ของผู้ใช้เองที่จะต้องสั่งให้เกิดการเขียนค่าตรรกะสูงลงสู่บิต ADSC ใหม่อีกครั้ง
- ใช้เวลาการแปลงเท่ากับ 13 รอบสัญญาณนาฬิกา
- ยกเว้นการทำงานในครั้งแรกภายหลังจากสั่งให้ตัวแปลงเริ่มทำงานจะใช้เวลาในการแปลงทั้งสิ้น 25 รอบสัญญาณนาฬิกา

แบบวิธีวิ่งอิสระ (Free Running mode)

- ตัวแปลงทำการแปลงสัญญาณในรอบถัดไปทันทีที่การแปลงในรอบที่ผ่านมาดำเนินการเสร็จสิ้น
- การแปลงแต่ละครั้งจะใช้สัญญาณนาฬิกาเท่ากับ 13 รอบ
- ยกเว้นครั้งแรกที่เริ่มทำงานจะใช้สัญญาณนาฬิกาเท่ากับ 25 รอบ

แบบวิธีจุดชนวน

- วงจรแปลงถูกสั่งให้เริ่มการแปลงโดยแหล่งกำเนิดสัญญาณจุดชนวนอื่น
- โดยผู้ใช้สามารถระบุแหล่งกำเนิดสัญญาณจุดชนวนได้
- ผู้ใช้ต้องทำการเลือกแหล่งกำเนิดสัญญาณจุดชนวนที่บิต ADTS[2:0] ของเรจิสเตอร์ ADCSRB
- ใช้สัญญาณนาฬิกาทั้งสิ้น 13.5 รอบ เมื่อมีสัญญาณจุดชนวนครั้งใหม่เข้ามาวงจรแปลงจึงจะเริ่มทำการแปลงในรอบใหม่ต่อไป

วิธีการจุดชนวนวงจรแปลงแอนะล็อกเป็นดิจิทัล

ADTS2	ADTS1	ADTS0	แหล่งกำเนิดสัญญาณจุดชนวน
0	0	0	สั่งให้ตัวแปลงทำงานในแบบวิธีวิ่งอิสระ
0	0	1	วงจรเปรียบเทียบสัญญาณแอนะล็อก
0	1	0	จากขา INT0
0	1	1	วงจรนับ/จับเวลาหมายเลข 0 เมื่อค่าใน TCNT0 เท่ากับ OCR0A
1	0	0	วงจรนับ/จับเวลาหมายเลข 0 เมื่อตัวบ่งชี้ TOV0 มีค่าเป็นตรรกะสูง
1	0	1	วงจรนับ/จับเวลาหมายเลข 1 เมื่อค่าใน TCNT1 เท่ากับ OCR1B
1	1	0	วงจรนับ/จับเวลาหมายเลข 1 เมื่อตัวบ่งชี้ TOV1 มีค่าเป็นตรรกะสูง
1	1	1	หน่วยดักจับ (Input Capture Unit) ของวงจรนับ/จับเวลาหมายเลข 1

ความเที่ยงตรงของการแปลงแอนะล็อกเป็นดิจิทัล

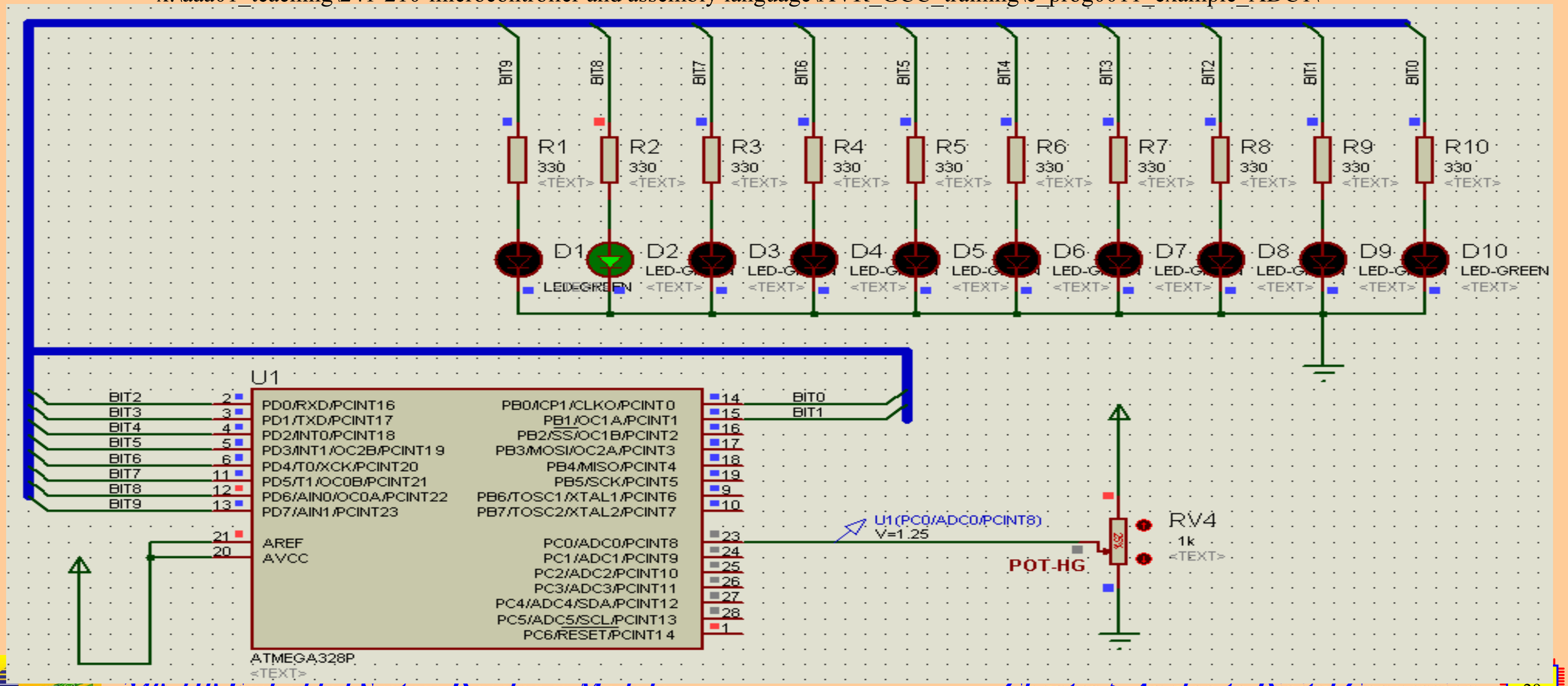
- ขึ้นอยู่กับปัจจัย 2 ประการ
  - ค่าความถี่ในการชักตัวอย่างสัญญาณ
  - ค่าความละเอียด (จำนวนบิต) ของวงจรแปลงสัญญาณแอนะล็อกเป็นดิจิทัล

การสุ่มสัญญาณแอนะล็อก

- ทฤษฎีการชักตัวอย่างของไนควิสต์และแชนนอน (Nyquist–Shannon sampling theorem)
- ไม่จำเป็นต้องบันทึกสัญญาณทั้งหมดเพื่อเก็บตัวอย่างของสัญญาณนั้น ๆ
- เราสามารถดึงเฉพาะบางส่วนของข้อมูลมาใช้เป็นตัวแทนของสัญญาณเดิมได้
- ความถี่ในการชักตัวอย่างสัญญาณจะต้องมากกว่า 2 เท่าของความถี่สูงสุดของสัญญาณรับเข้า

ตัวอย่างที่ 5.1

x:\aaa01_teaching\241-210-microcontroller and assembly language\AVR_GCC_training\c_prog0011_example_ADC1\



วิธีการจัดเรียงข้อมูลใน ADCH และ ADCL

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0	
-	-	-	-	-	-	ADC9	ADC8	เรจิสเตอร์ ADCH
ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	เรจิสเตอร์ ADCL

เมื่อ ADLAR=0

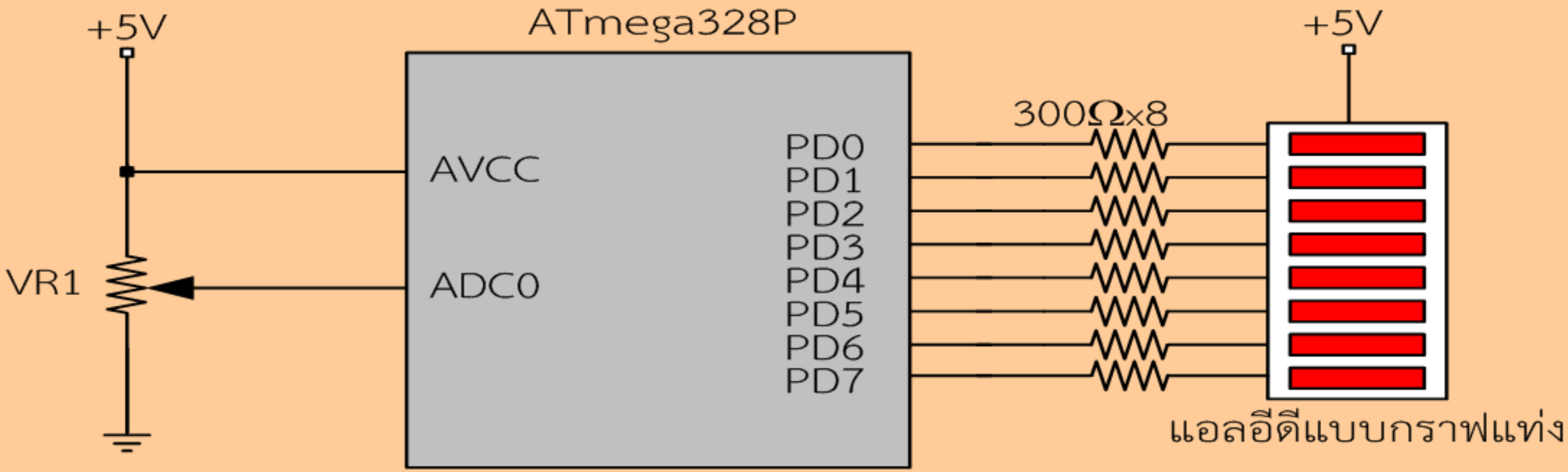
ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	เรจิสเตอร์ ADCH
ADC1	ADC0	-	-	-	-	-	-	เรจิสเตอร์ ADCL

เมื่อ ADLAR=1

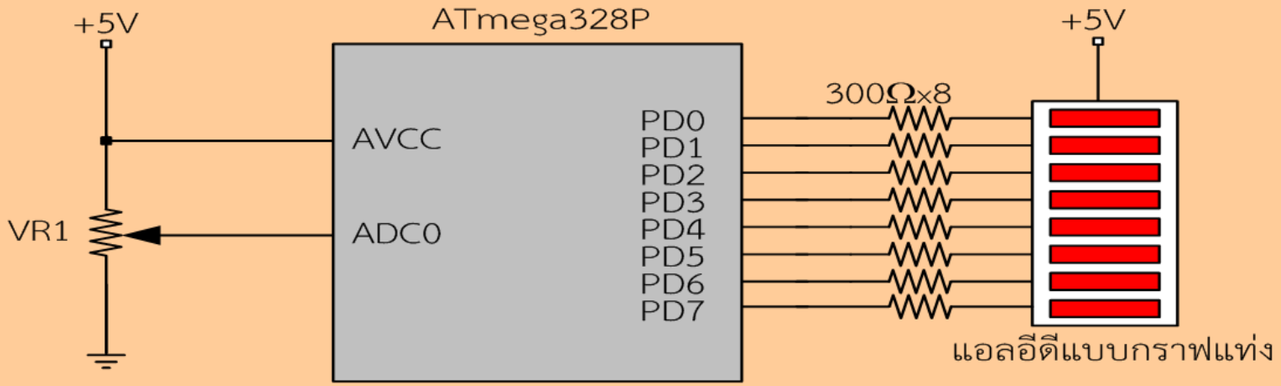
ตัวอย่างที่ 5.1

```
#include <avr/io.h>
#include <avr/interrupt.h>

void do_nothing(void)     {}

int main(void)
{
     DDRD = 0xFF;
     DDRB = 0xFF;

     //ADMUX[7:6]=00 ->using AREF pin
     //ADMUX[5]   = 1 ->ADLR=1
     //ADMUX[3:0]=0000->ADC0 pin //ADCSRA[7]=1    ->ADC enable
     ADMUX = 0b00100000;         //ADCSRA[6]=?    ->ADC start conversion
     while(1)                    //ADCSRA[5]=0    ->disable auto trigger
     {                           //ADCSRA[4]=0    ->ADC interrupt Flag
          ADCSRA = 0b11000111;   //ADCSRA[3]=0    ->ADC interrupt disable
                                 //ADCSRA[2:0]=111-> Division factor/128
          while (!(ADCSRA & (1<<ADIF)) )
               do_nothing();

          PORTD = ADCH;
          PORTB = (ADCL>>6);
     }
}
```

ตัวอย่างที่ 5.2


+5V
ATmega328P
+5V
300Ω×8
AVCC
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
VR1
ADC0
แอลอีดีแบบกราฟแท่ง

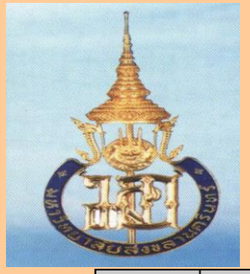
ตัวอย่างที่ 5.2



แรงดันขาเข้าที่ขา ADC0	จำนวนแอลอีดีที่สว่าง
$V_{IN} < 0.55$	ไม่มีแอลอีดีดวงใดสว่าง
$0.55 \le V_{IN} < 1.11$	1 ดวง
$1.11 \le V_{IN} < 1.67$	2 ดวง
$1.67 \le V_{IN} < 2.22$	3 ดวง
$2.22 \le V_{IN} < 2.78$	4 ดวง
$2.78 \le V_{IN} < 3.34$	5 ดวง
$3.34 \le V_{IN} < 3.90$	6 ดวง
$3.90 \le V_{IN} < 4.45$	7 ดวง
$4.45 \le V_{IN} < 5.0$	8 ดวง

ตัวอย่างที่ 5.2

- หากค่า VIN < 0.55	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 0-112
- หากอยู่ในช่วง 0.55 ≤ VIN < 1.11	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 113-227
- หากอยู่ในช่วง 1.11 ≤ VIN < 1.67	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 227-342
- หากอยู่ในช่วง 1.67 ≤ VIN < 2.22	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 342-455
- หากอยู่ในช่วง 2.22 ≤ VIN < 2.78	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 455-569
- หากอยู่ในช่วง 2.78 ≤ VIN < 3.34	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 569-684
- หากอยู่ในช่วง 3.34 ≤ VIN < 3.90	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 684-799
- หากอยู่ในช่วง 3.90 ≤ VIN < 4.45	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 799-911
- หากอยู่ในช่วง 4.45 ≤ VIN < 5.0	จะได้ค่าจากวงจรแปลงแอนะล็อกเป็นดิจิทัลในช่วง 911-1023

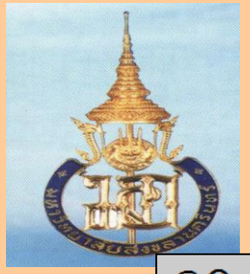


ตัวอย่างที่ 5.2

```
1   #include <avr/io.h>                    //เรียกใช้คลังโปรแกรม io.h
2   #define F_CPU 16000000UL               //กำหนดค่าความถี่ของตัวประมวลผลเท่ากับ 16 เมกะเฮิรทซ์
3   int main (void)                        //ฟังก์ชันหลักของโปรแกรม
4   {
5       uint16_t x;                        //ประกาศตัวแปร x สำหรับใช้รับค่าจากเรจิสเตอร์ ADC
6       DIDR0 = 0b00111111;                //ปิดทางวงจรบัฟเฟอร์สัญญาณดิจิทัลของขา ADC0-ADC5
7       DDRD = 0xFF;                       //ตั้งค่าทิศทางให้พอร์ต D ทำหน้าที่ส่งออกข้อมูล
8       DDRC = 0x00;                       //ตั้งค่าทิศทางให้พอร์ต C ทำหน้าที่รับเข้า
9       ADMUX = (1<<REFS0) | (0<<REFS1) | (0<<ADLAR); //ตั้งบิต ADLAR = 0 ในเรจิสเตอร์ ADMUX และ
10                                         //ค่า REFS[1:0] = 00₂ หมายถึงรับแรงดันอ้างอิงจากขา AVCC
11                                         //ADMUX[3:0]  = 000₂ หมายถึงรับสัญญาณแอนะล็อกจากขา ADC0
12      ADCSRA=  (1<<ADEN) | (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);//ตั้งค่าควบคุมในเรจิสเตอร์ ADCSRA
13                                         //โดยมีรายละเอียดบิตควบคุมดังนี้
14                                         //-ADEN=1 หมายถึงเปิดใช้งานวงจร ADC
15                                         //-ADPS[2:0] = 111₂ หมายถึงตั้งค่าตัวหารความถี่เท่ากับ 128
16                                         //- ADATE=0 หมายถึงตั้งให้ ADC ทำงานแบบวิธีแปลงครั้งเดียว
```

ตัวอย่างที่ 5.2

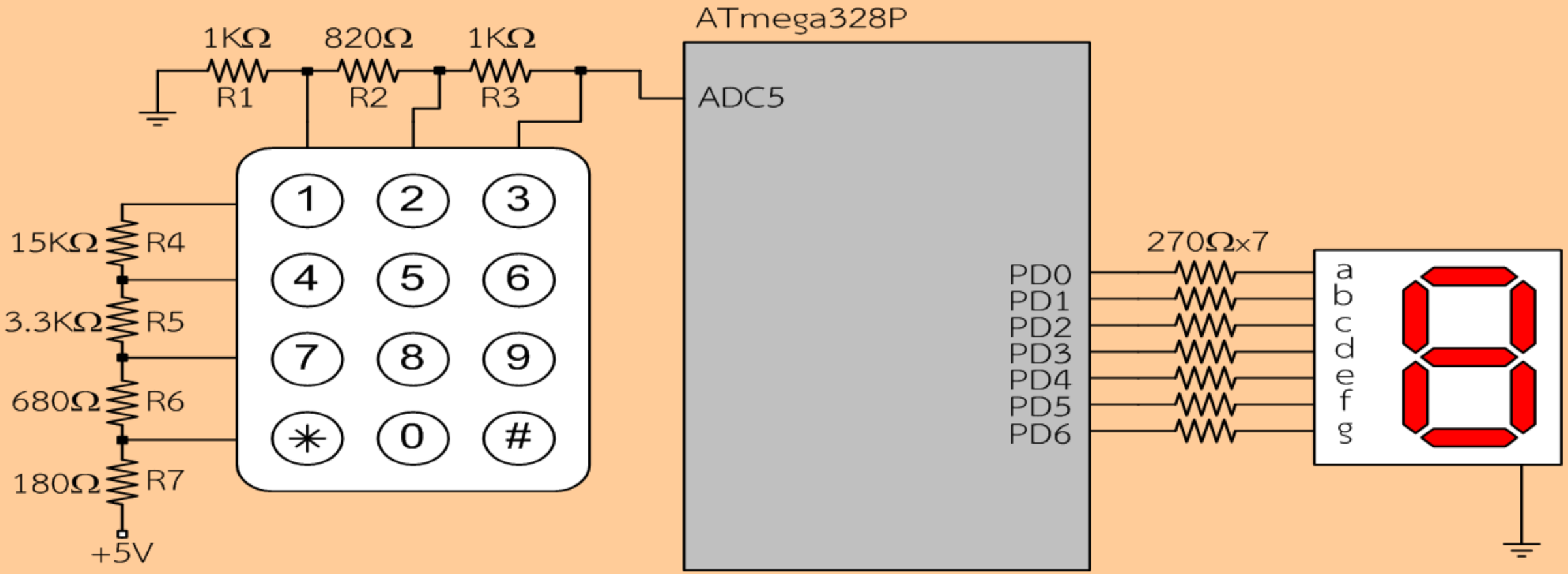
```
17      while(1)
18      {
19          ADCSRA |= (1<<ADSC);     //ตั้ง ADSC=1 หมายถึงสั่งให้วงจร ADC เริ่มต้นแปลงสัญญาณ
20          while (!(ADCSRA & (1<<ADIF)) )
21          ;                        //วนซ้ำจนกว่าการแปลงจะเสร็จสิ้น (เมื่อเสร็จ ADIF จะมีค่าตรรกะสูง)
22
23          x = ADC;                 //อ่านค่าจาก ADC ใส่ตัวแปร x
24          if (x<113)               //หาก x น้อยกว่า 113
25              PORTD = 0xFF;        //          สั่งให้แอลอีดีดับทุกดวง
26          else if (x<227)          //หาก 113 ≤ x < 227
27              PORTD = 0xFE;        //          สั่งให้แอลอีดีติดหนึ่งดวง
28          else if (x<342)          //หาก 227 ≤ x < 342
29              PORTD = 0xFC;        //          สั่งให้แอลอีดีติดสองดวง
```



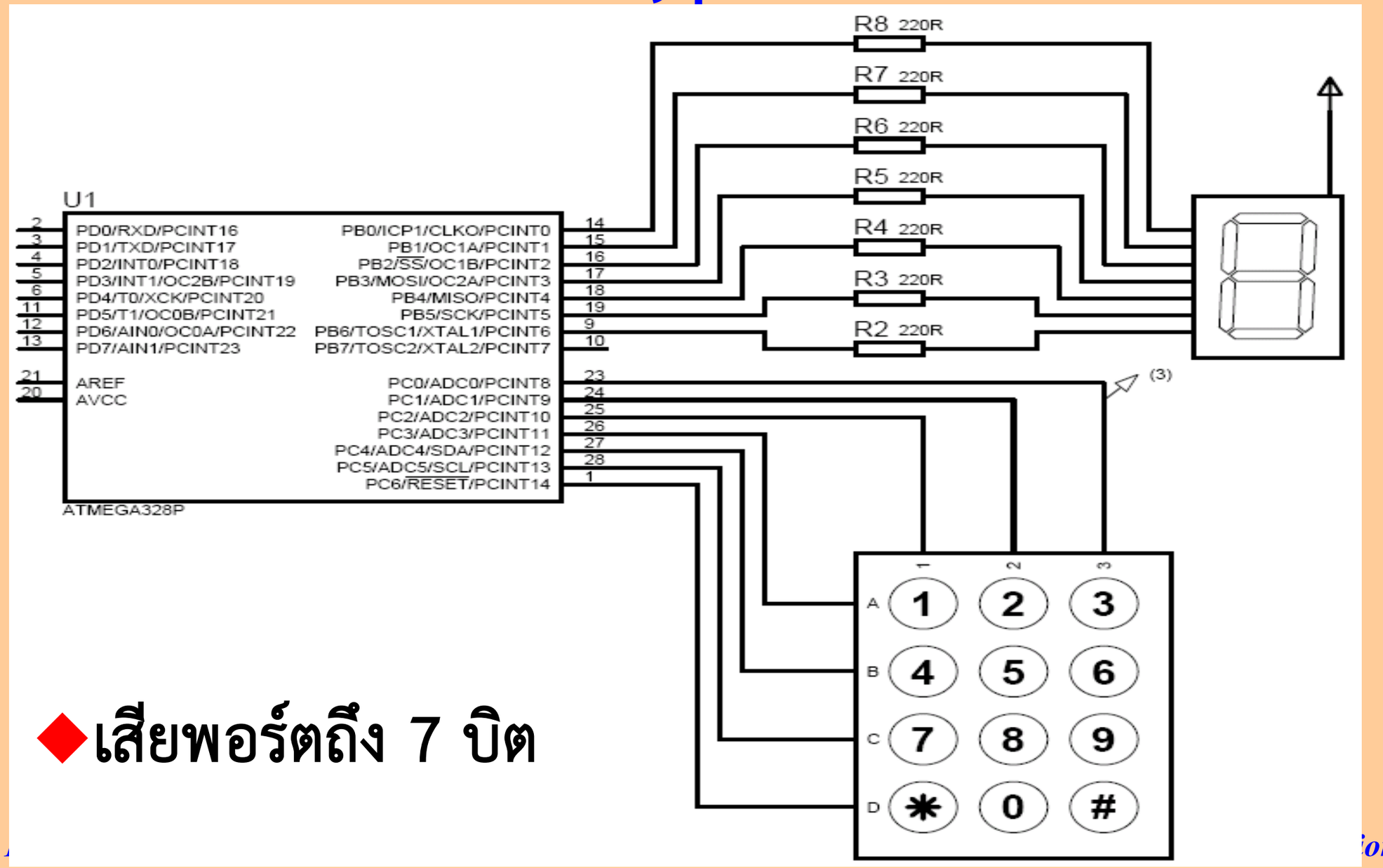
ตัวอย่างที่ 5.2

```
30          else if (x<455)             //หาก 342 ≤ x < 455
31              PORTD = 0xF8;           //          สั่งให้แอลอีดีติดสามดวง
32          else if (x<569)             //หาก 455 ≤ x < 569
33              PORTD = 0xF0;           //          สั่งให้แอลอีดีติดสี่ดวง
34          else if (x<684)             //หาก 569 ≤ x < 684
35              PORTD = 0xE0;           //          สั่งให้แอลอีดีติดห้าดวง
36          else if (x<799)             //หาก 684 ≤ x < 799
37              PORTD = 0xC0;           //          สั่งให้แอลอีดีติดหกดวง
38          else if (x<911)             //หาก 799 ≤ x < 911
39              PORTD = 0x80;           //          สั่งให้แอลอีดีติดเจ็ดดวง
40          else                        //หาก 911 ≤ x ≤ 1023
41              PORTD = 0x00;           //          สั่งให้แอลอีดีติดทุกดวง
42          ADCSRA |= (1<<ADIF);        //ลบล้างค่าในตัวบ่งชี้การขัดจังหวะจาก ADC
43      }
44  }
```

ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

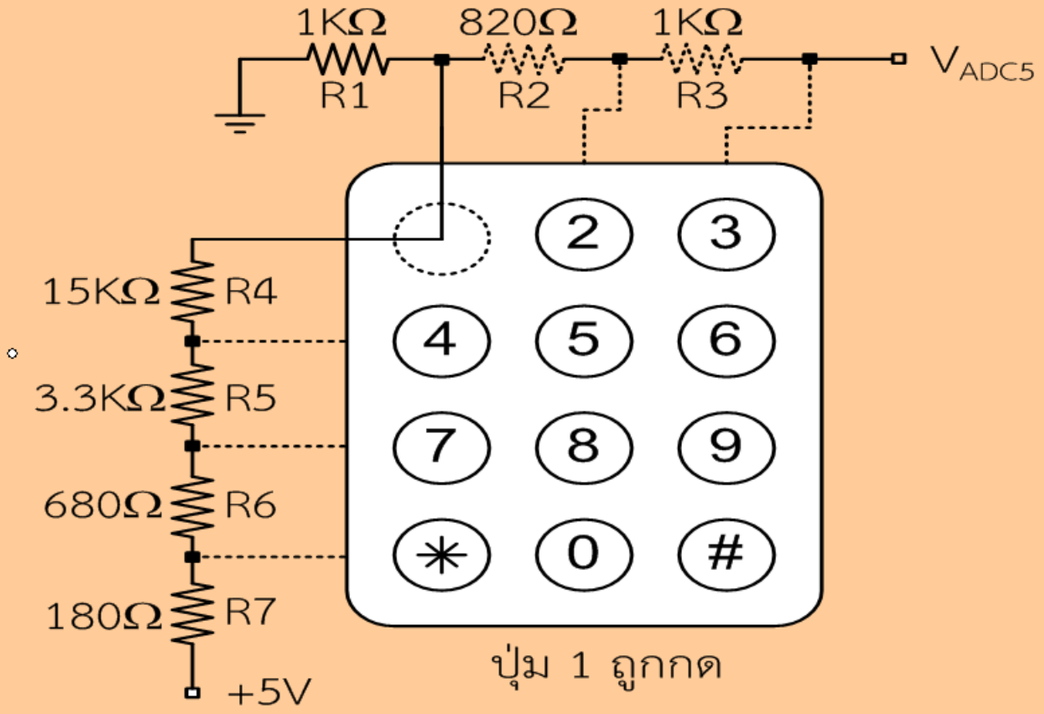


ตัวอย่างการต่อ Keypad ด้วยวิธีปกติ



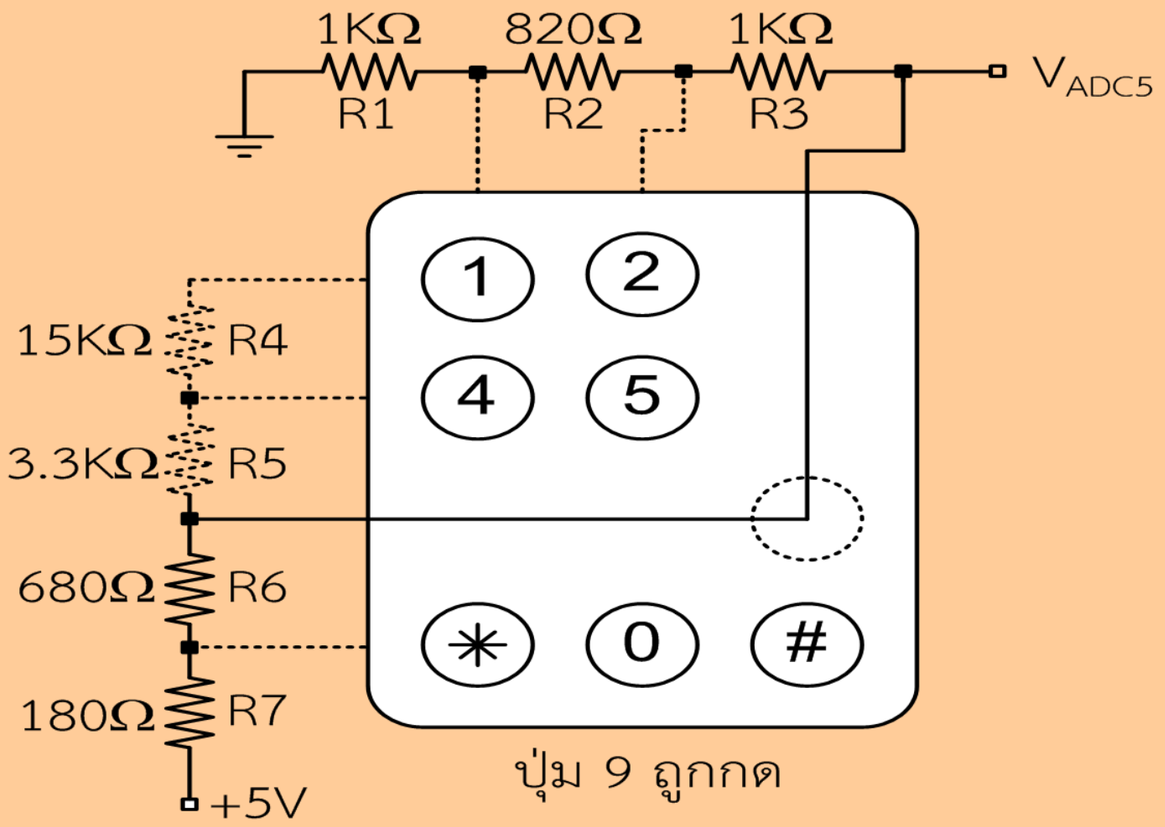
- เสียพอร์ตถึง 7 บิต

ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC



$$V_{ADC5} = \frac{5 * R1}{R1 + R4 + R5 + R6 + R7} = \frac{5 * 1000}{1000 + 15000 + 3300 + 680 + 180} = 0.248\ V$$

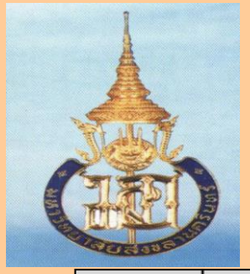
ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC



$$V_{ADC5} = \frac{5 * (R1 + R2 + R3)}{R1 + R2 + R3 + R6 + R7} = \frac{5 * (1000 + 820 + 1000)}{1000 + 820 + 1000 + 680 + 180} = 3.832\,V$$

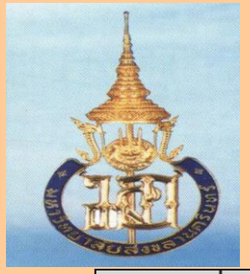
ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

ปุ่มที่ถูกกด	ค่าแรงดันที่ได้จากการคำนวณ	แรงดันน้อยสุดที่ยอมรับได้	แรงดันสูงสุดที่ยอมรับได้	ค่าดิจิทัลน้อยสุดที่ยอมรับได้	ค่าดิจิทัลสูงสุดที่ยอมรับได้
1	0.248	0.163	0.318	33	65
2	0.434	0.349	0.504	71	103
3	0.641	0.556	0.711	114	146
4	0.969	0.884	1.039	181	213
5	1.522	1.437	1.592	294	326
6	2.020	1.935	2.090	396	428
7	2.688	2.438	2.838	499	581
8	3.396	3.311	3.466	678	710
9	3.832	3.747	3.902	767	799
*	4.237	4.117	4.317	843	884
0	4.550	4.400	4.570	901	936
#	4.700	4.620	5.000	946	1023



ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

```
1  #include <avr/io.h>                    //เรียกใช้คลังโปรแกรม io.h
2  #define F_CPU 16000000UL               //กำหนดค่าความถี่ของตัวประมวลผลเท่ากับ 16 เมกะเฮิรทซ์
3  #include <avr/delay.h>                 //เรียกใช้คลังโปรแกรม delay.h
4  #define TURN_ALL_LED_OFF 0x00 //ตั้งค่าสำหรับใช้ปิดแอลอีดีทุกดวง
5  #define ADC5 5                         //อ่านสัญญาณแอนะล็อกจากขา ADC5
6  unsigned char TABLE7SEG[] =            //ตารางสำหรับเก็บรหัสแสดงผลแอลอีดีชนิด 7 ส่วน
7  {     0b00111111,                      //        รหัสของเลข 0
8        0b00000110,                      //        รหัสของเลข 1
9        0b01011011,                      //        รหัสของเลข 2
10       0b01001111,                      //        รหัสของเลข 3
11       0b01100110,                      //        รหัสของเลข 4
12       0b01101101,                      //        รหัสของเลข 5
13       0b01111101,                      //        รหัสของเลข 6
14       0b00000111,                      //        รหัสของเลข 7
```



ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

```
15        0b01111111,                    //        รหัสของเลข 8
16        0b01101111,                    //        รหัสของเลข 9
17        0b01110111,                    //        รหัสของเลข A
18        0b01111100,                    //        รหัสของเลข B
19        0b00111001,                    //        รหัสของเลข C
20        0b01011110,                    //        รหัสของเลข D
21        0b01111001,                    //        รหัสของเลข E
22        0b01110001,                    //        รหัสของเลข F
23        0b00000000 };                  // รหัสสำหรับสั่งให้แอลอีดีดับทุกดวง
24  int counter = 0;                     // how many times we have seen new value
25  int adcVal, prevADC, DECODED;        //ประกาศตัวแปร
26  void DISPLAY7segment(signed char a)  //ฟังก์ชันสำหรับใช้แสดงผลออกสู่แอลอีดีชนิด 7 ส่วน
27  {                                    //นำค่าพารามิเตอร์ที่รับเข้ามา เปิดตารางค้นหา
28        PORTD = TABLE7SEG[a];          //และนำค่าที่อ่านได้ออกแสดงผลที่พอร์ต D
29  }
```

ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

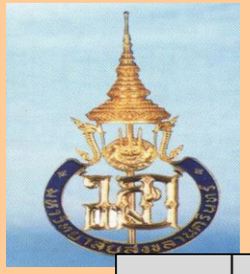
```
30  unsigned int ADC_read(unsigned char a) //ฟังก์ชันสำหรับอ่านค่าแรงดันจากตัวแปลงแอนะล็อกเป็นดิจิทัล
31  {                                      //โดยรับค่าพารามิเตอร์ a ซึ่งระบุว่าจะอ่านจากช่องสัญญาณใด
32      ADMUX&=  0xF0;                     //ลบล้างค่าในสี่บิตล่างของเรจิสเตอร์ ADMUX ให้กลายเป็นตรรกะต่ำ
33      ADMUX|=   a;                       //ตั้งให้ ADMUX บิตสี่บิตล่างให้เท่ากับ a
34      ADCSRA |= (1<<ADSC);               //สั่งให้วงจรแปลงเริ่มทำการแปลงสัญญาณ
35      while ( !(ADCSRA & (1<<ADIF)) );  //วนซ้ำจนกว่าตัวบ่งชี้ ADIF มีค่าเป็นตรรกะสูง
36      ADCSRA |= 1<<ADIF;                 //ลบล้างค่าในตัวบ่งชี้ ADIF ให้กลับมาเป็นตรรกะต่ำ
37      return ADC;                        //คืนค่าที่ได้จากการแปลงสู่โปรแกรมผู้เรียก
38  }
39  int main(void)
40  {
41      DDRC = 0x00;                       //input ADC5
42      DDRD = 0xFF;                       //port D is connected to 7-segment LED
43      DISPLAY7segment(TURN_ALL_LED_OFF);        //สั่งให้แอลอีดีดับทุกดวงเมื่อเริ่มเปิดเครื่อง
44      ADMUX = 0b01000101;                //เริ่มต้นให้เลือกค่าแรงดันอ้างอิงจากขา AVCC
45      ADCSRA = 0x87;                     //สั่งให้วงจรแปลงทำงานในแบบวิธีแปลงครั้งเดียว
46      DECODED = 0;                       //ตั้งค่าเริ่มต้นให้ตัวแปร DECODED เท่ากับศูนย์
```

ADMUX

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0	
REFS1	REFS0	ADLAR	ไม่ใช้งาน	MUX3	MUX2	MUX1	MUX0	เรจิสเตอร์ ADMUX

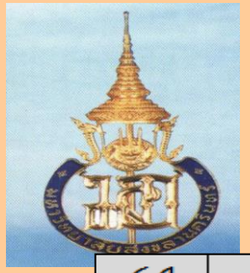
เลือกสัญญาณออกจากวงจรแอนะล็อกมัลติเพลกเซอร์

MUX3...0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	ADC8[1]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)



ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

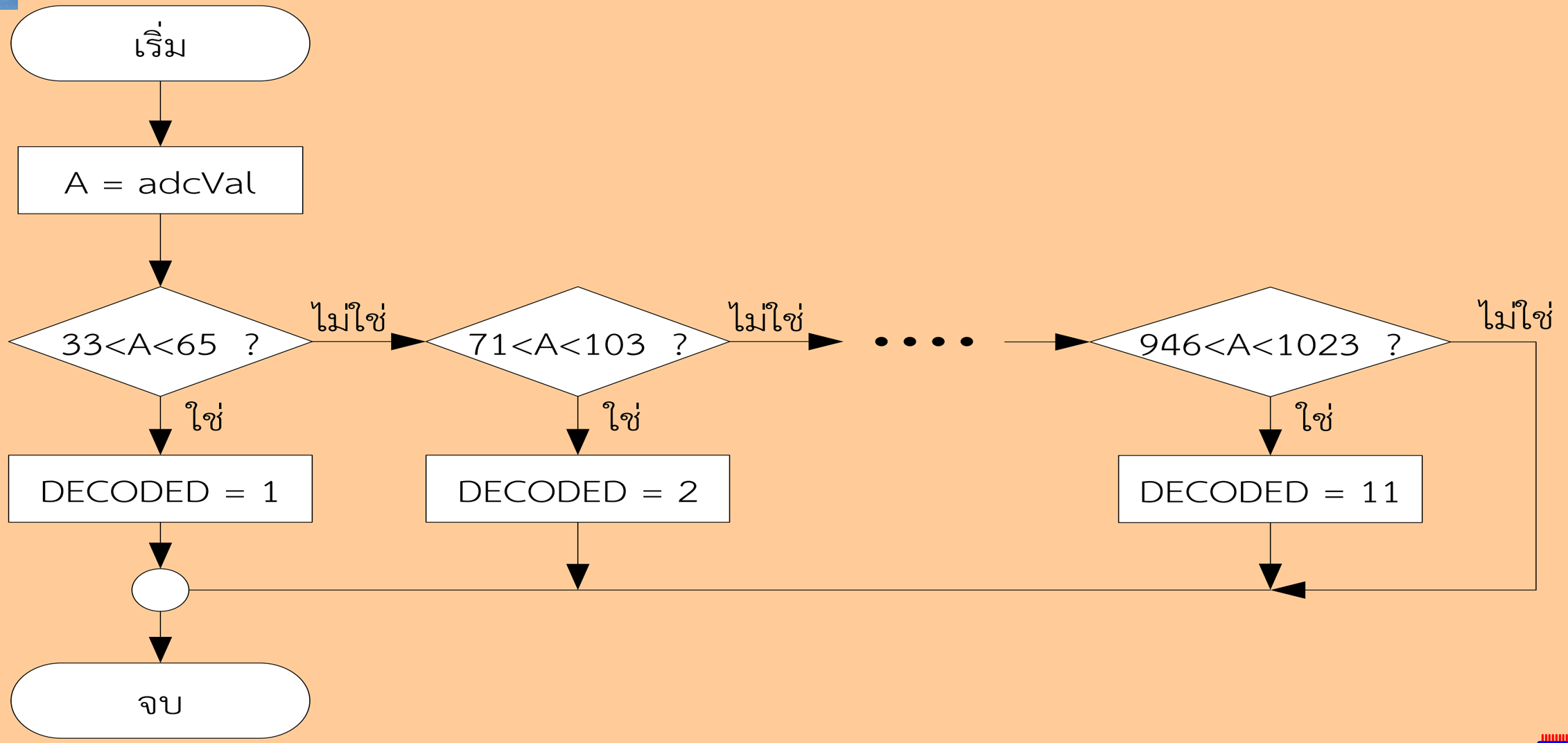
```
47      while(1)                        //วนซ้ำไม่รู้จบ
48      {
49          counter=0;                  //ตั้งค่าเริ่มต้นให้ตัวแปร counter เท่ากับศูนย์
50          prevADC = 0;                //ตั้งค่าเริ่มต้นให้ตัวแปร prevADC เท่ากับศูนย์
51           do                         //เริ่มต้นการวนซ้ำ
52           {                          //ตรวจสอบว่ามีการกดสวิตช์หรือไม่หากมีให้เพิ่มค่าใน counter
53              adcVal = ADC_read(ADC5);                //อ่านค่าแรงดันจากขา ADC5 ใส่ตัวแปร adcVal
54              if ((abs(adcVal-prevADC)<6)&& (adcVal >33))     //ตรวจว่ามีการกดสวิตช์หรือไม่
55                  counter++;          //เพิ่มค่าใน counter หากมีการกดสวิตช์
56              else
57                  counter =0;         //หากไม่มีการกดสวิตช์ให้ลบล้างค่าใน counter กลับมาเป็นศูนย์
58              _delay_ms(3);           //หน่วงเวลา 3 มิลลิวินาที
59              prevADC = adcVal;       //คัดลอกตัวแปร adcVal สู่ตัวแปร prevADC
60           } while (counter <20);     //วนซ้ำตราบเท่าที่ตัวแปร counter ยังน้อยกว่า 20
61           if ((adcVal>33 ) && (adcVal<65 )) DECODED = 1;             //ปุ่ม 1 ถูกกด
62           else if  ((adcVal>71 ) && (adcVal<103 )) DECODED = 2;      //ปุ่ม 2 ถูกกด
63           else if  ((adcVal>114) && (adcVal<146 )) DECODED = 3;      //ปุ่ม 3 ถูกกด
```



ตัวอย่างที่ 5.3 การต่อ Keypad โดยใช้ ADC

```
64              else if  ((adcVal>181) && (adcVal<213 )) DECODED = 4;      //ปุ่ม 4 ถูกกด
65              else if  ((adcVal>294) && (adcVal<326 )) DECODED = 5;      //ปุ่ม 5 ถูกกด
66              else if  ((adcVal>396) && (adcVal<428)) DECODED = 6;       //ปุ่ม 6 ถูกกด
67              else if  ((adcVal>499) && (adcVal<581)) DECODED = 7 ;      //ปุ่ม 7 ถูกกด
68              else if  ((adcVal>678) && (adcVal<710)) DECODED = 8 ;      //ปุ่ม 8 ถูกกด
69              else if  ((adcVal>767) && (adcVal<799)) DECODED = 9 ;      //ปุ่ม 9 ถูกกด
70              else if  ((adcVal>843) && (adcVal<884)) DECODED = 10;      //ปุ่ม * ถูกกด
71              else if  ((adcVal>901) && (adcVal<936)) DECODED = 0 ;      //ปุ่ม 0 ถูกกด
72              else if  ((adcVal>946) && (adcVal<1023)) DECODED = 11;     //ปุ่ม # ถูกกด
73              do                                                         //เริ่มต้นการวนซ้ำ
74              {
75                  adcVal = ADC_read(ADC5);                               //อ่านค่าแรงดันจากขา ADC5
76              } while(adcVal > 33);                                      //วนซ้ำจนกว่าปุ่มจะถูกปล่อย
77              DISPLAY7segment(DECODED);                                  //แสดงค่าที่ถอดรหัสได้สู่แอลอีดี
78          }
79  }
```

การถอดรหัสค่าในตัวแปร ADC



สรุป

- สถาปัตยกรรมเอวีอาร์ชุด ATmega มีวงจร ADC ความละเอียด 10 บิต ทำงานโดยใช้วิธีการแปลงแบบการประมาณสืบเนื่อง
- วงจร ADC ประกอบด้วยโหมดการทำงาน 3 รูปแบบ
  - แบบวิธีจุดชนวน
  - แบบวิธีวิ่งอิสระ
  - แบบวิธีการแปลงครั้งเดียว
- ค่าแรงดันอ้างอิงของวงจรแปลง กำหนดที่ขา AREF
- กำหนดค่าตัวหารความถี่ของวงจรพรีสเกลเลอร์ให้เหมาะสม

จบบทที่ 5