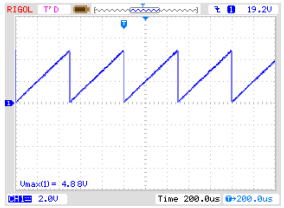


Arduino กับการเชื่อมต่อกับสัญญาณแอนะล็อก

รศ.ณรงค์ บวบทอง

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

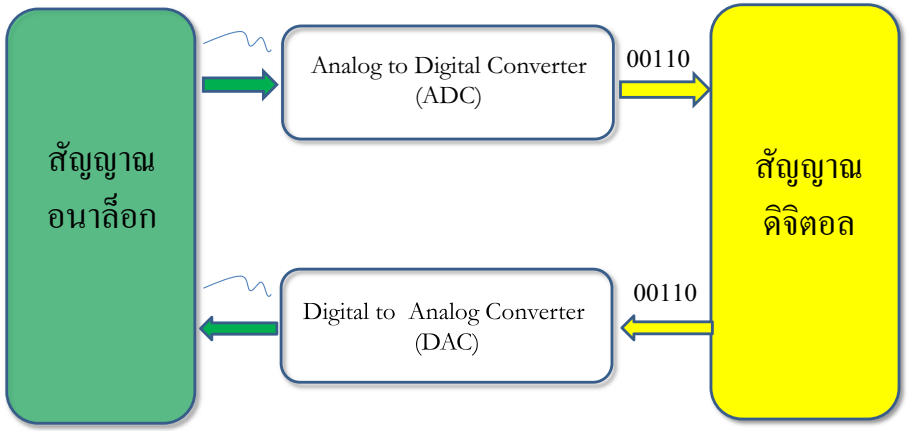
มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต

หัวข้อ

1. บทนำ
2. การแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (Digital to Analog Conversion)
3. การแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (Analog to Digital conversion)

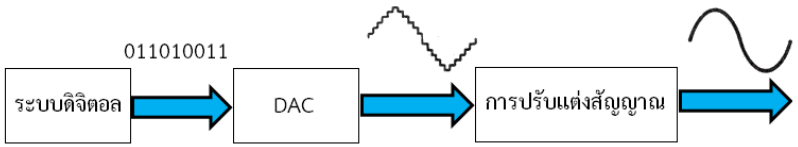
บทนำ

การแปลงสัญญาณ



วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณ อนาล็อก (Digital-to-Analog Converter หรือ D/A หรือ D2A)

หมายถึงวงจรที่เปลี่ยนข้อมูลทางดิจิตอลให้เป็นค่าอนาล็อก ค่าอนาลอกนี้อาจเป็นกระแสไฟฟ้าหรือแรงดันไฟฟ้าก็ได้

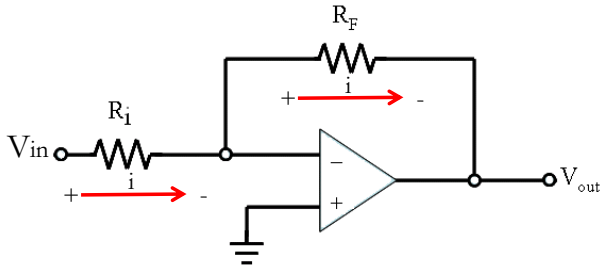


เทคนิคของ DAC

- Binary Weighted DAC
  - Uses switches & precision resistors
  - Difficult to achieve high density due to large resistance values
- R/2R Ladder
  - Most common method
  - Keeps resistance values low
  - Uses intricate interconnections

Binary Weighted DAC

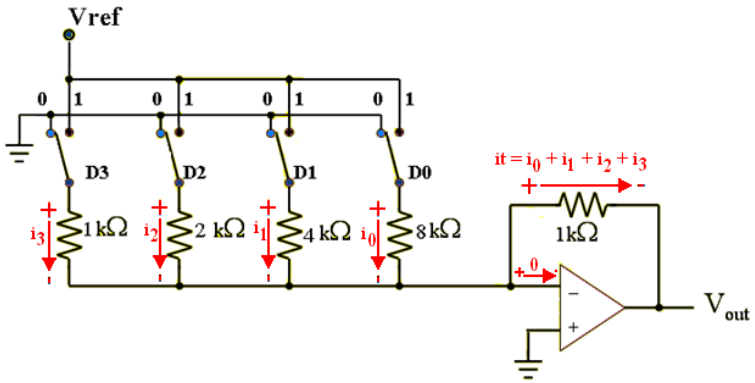
Op-Amp Review



For Negative Feedback,

1. Inputs impedance = Ri
2. Vout = $-i_x R_F = -(V_{in}/R_i) \times R_F$
3. Voltage gain is : A_v = Vout/Vin = $-R_f / R_i$

Binary Weighted DAC

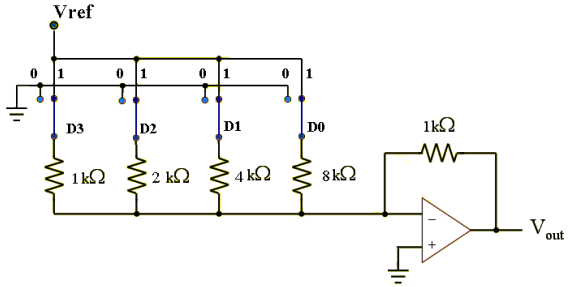


แรงดันเอาท์พุท

ต่ำสุดเกิดเมื่อสวิตซ์ D3- D0 อยู่ตำแหน่ง ‘0’ Vout = 0 โวลท์

สูงสุดเกิดเมื่อสวิตช์ D3 – D0 อยู่ตำแหน่ง ‘1’ Vout = -1k x (i3+i2+i1+i0)

Binary Weighted DAC



ถ้า D3- D0 อยู่ตำแหน่ง ‘1’ จะได้แรงดันเอาท์พุทสูงสุด

Vout = - Vref x Rf x(1/1k + 1/2k + 1/4k + 1/8k)

Vout = -Vref x 1 x(1/1 + 1/2 + 1/4 + 1/8)

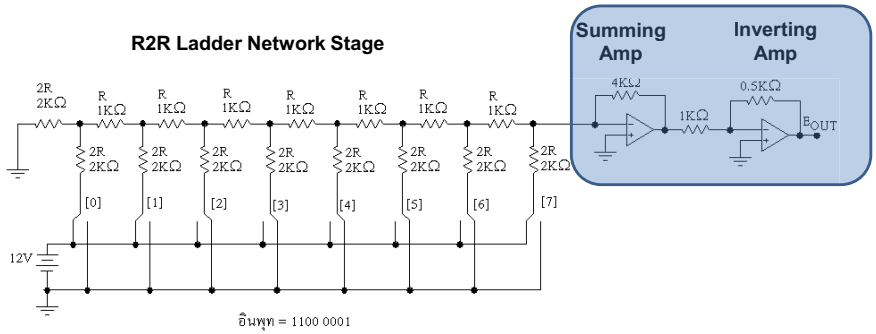
Vout = -Vref x (1 + .5 + .25 + .125)

Vout = -1.875 x Vref

เช่นถ้า Vref = +5V Vout เมื่ออินพุทเป็น ‘1’ หมดจะได้ -9.375 โวลท์

R2R Ladder DAC

R2R Ladder DAC



การวิเคราะห์วงจรโดยใช้ทฤษฎี Thevenin

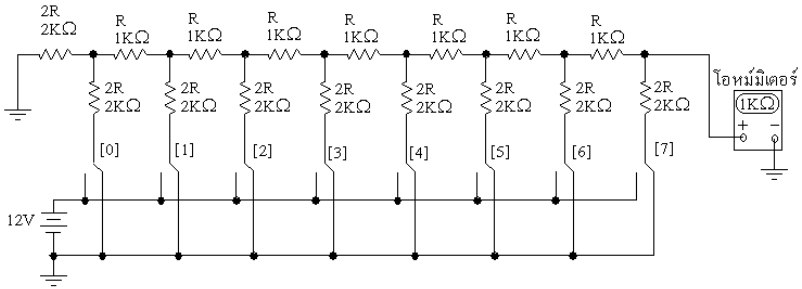
หา Rth ของ Thevenin

ที่ตำแหน่งบิตต่างๆ สามารถหา Rth ได้ดังนี้

บิต 0 Rth = 1 k, บิต 1 Rth = 1 k,บิต 2 Rth = 1 k,บิต 3 Rth = 1 k,บิต 4 Rth = 1 k,บิต 5 Rth = 1 k,

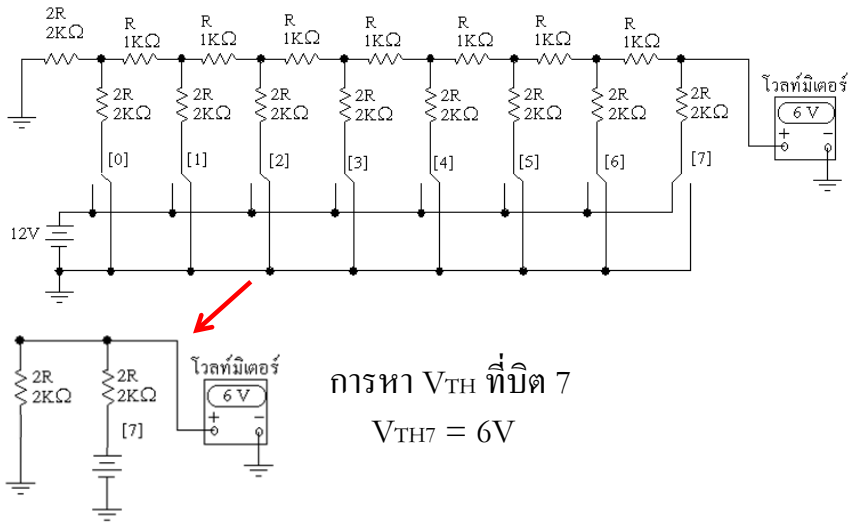
บิต 6 Rth = 1 k และ บิต 7 Rth = 1 k

วงจร Thevenin equivalent เพื่อหา Rth



Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
1 kΩ	1 kΩ	1 kΩ	1 kΩ	1 kΩ	1 kΩ	1 kΩ	1 kΩ

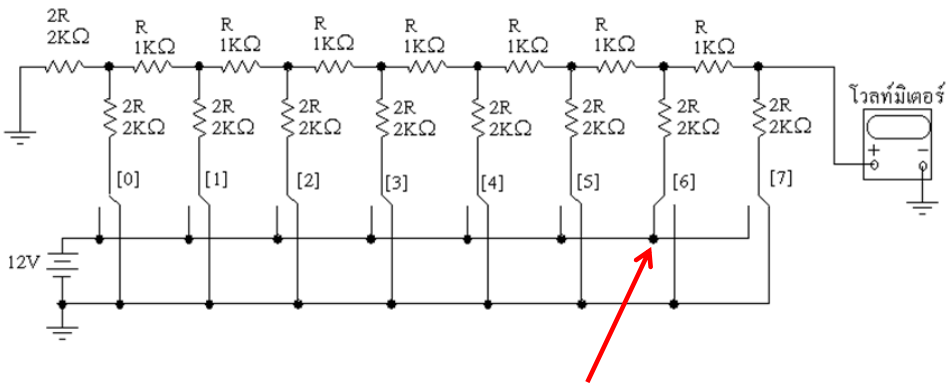
หา Thevenin Voltage ใน the R2R Ladder Network (Vth)



การหา V_{TH} ที่บิต 7

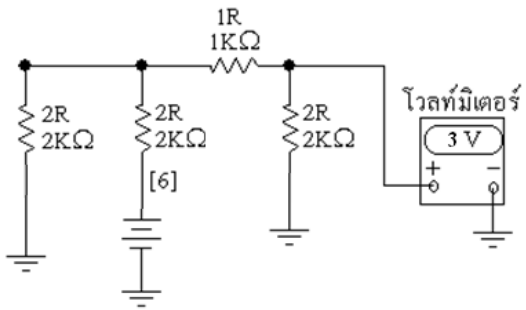
$V_{TH7} = 6V$

หา Thevenin Voltage บิตที่ 6 (Vth6)



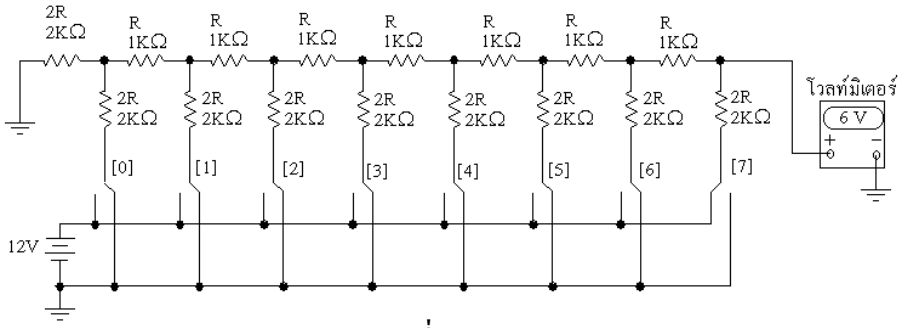
การหา V_{TH} ที่บิต 6

หา Thevenin Voltage บิตที่ 6 (Vth6)



การหา V_{TH} ที่บิต 6

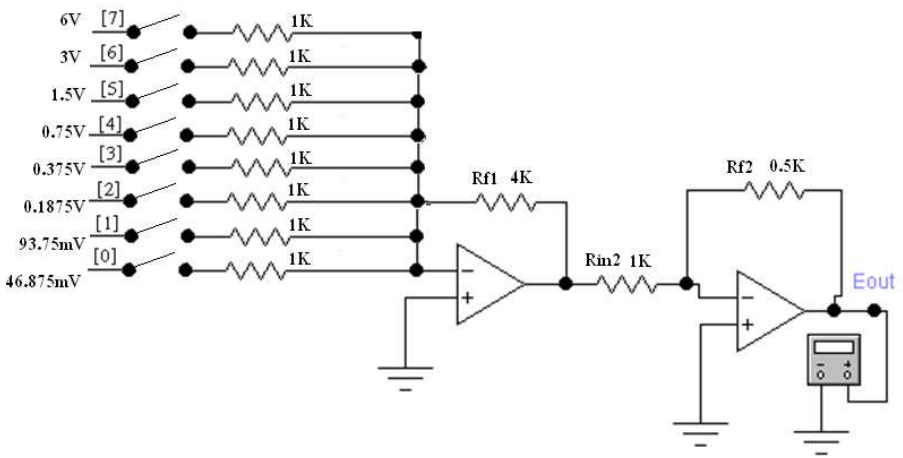
หา Thevenin Voltage ใน the R2R Ladder Network (Vth)



V_{TH} ที่บิตต่างๆ

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
46.875 m V	93.75 m V	0.1875 V	0.375 V	0.75 V	1.5 V	3 V	6 V

วงจรเสมือนเมื่อแทนด้วย Thevenin ของทุกบิต



จากตัวอย่าง สามารถหาค่า Analog Outputs

$$V_{OUT(MSB)} = \frac{V_{REF}}{2}\left(\frac{R_{F1}}{R_{TH}}\right)\left(\frac{R_{F2}}{R_{IN2}}\right)$$

$$V_{OUT(LSB)} = \frac{V_{REF}}{256}\left(\frac{R_{F1}}{R_{TH}}\right)\left(\frac{R_{F2}}{R_{IN2}}\right)$$

$$V_{OUT(AnyBit)} = \frac{V_{REF}}{2^{8-n}}\left(\frac{R_{F1}}{R_{TH}}\right)\left(\frac{R_{F2}}{R_{IN2}}\right)$$

$$V_{OUT(Total)} = V_{OUT(MSB)}\left(\frac{Input_{10}}{MSB_{10}}\right)$$

สรุปการคำนวณแบบไม่รวมอัตราขยาย

$$V_{OUT} = X_{10}\left(\frac{V_{ref}^{+}}{2^{n}}\right)$$

$$V_{OUT} = V_{min} + X_{10}\left(\frac{V_{max} - V_{min}}{2^{n}-1}\right)$$

$$\text{Resolution} = \left(\frac{V_{ref}^{+}}{2^{n}}\right)$$

$$\text{Resolution} = \left(\frac{V_{max} - V_{min}}{2^{n}-1}\right)$$

$$V_{OUT} = X_{10}\,\text{Resolution}$$

$$V_{OUT} = V_{min} + X_{10}\,\text{Resolution}$$

V_{OUT} = Output Voltage

V_{ref}^{+} = Voltage Reference +

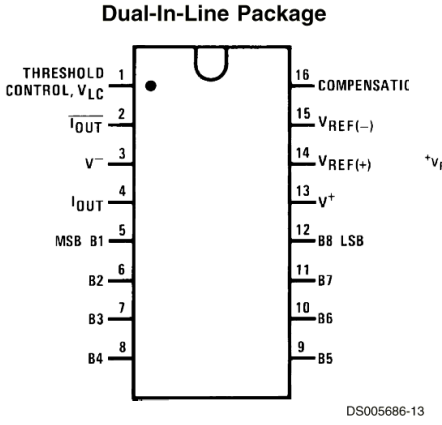
V_{min} = Minimum output voltage

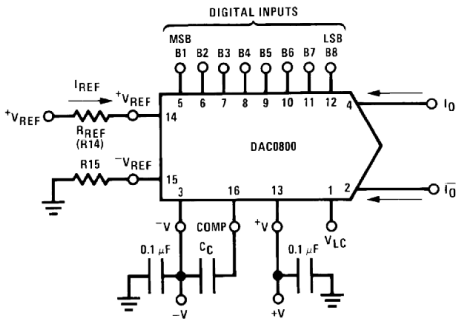
V_{max} = Maximum output voltage

X_{10} = Input value base 10

n = จำนวนบิต

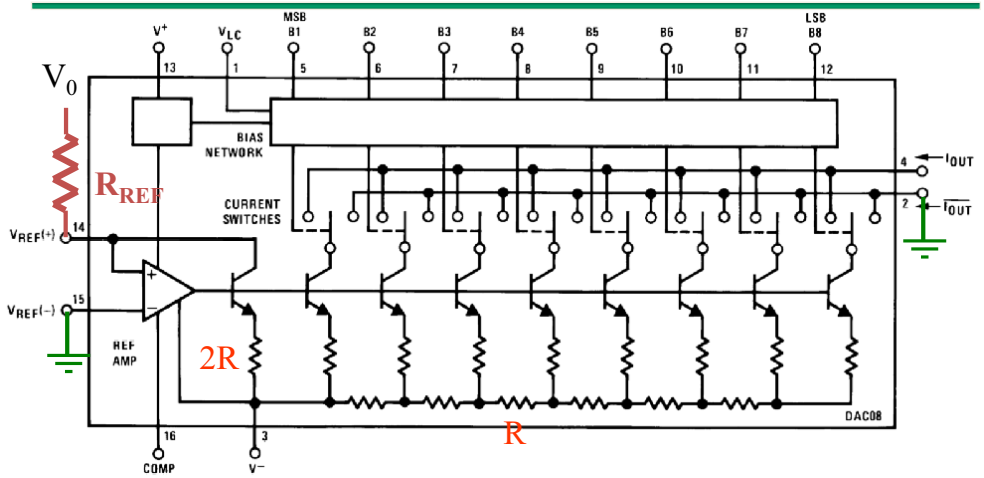
DAC0800 Pinout and Application





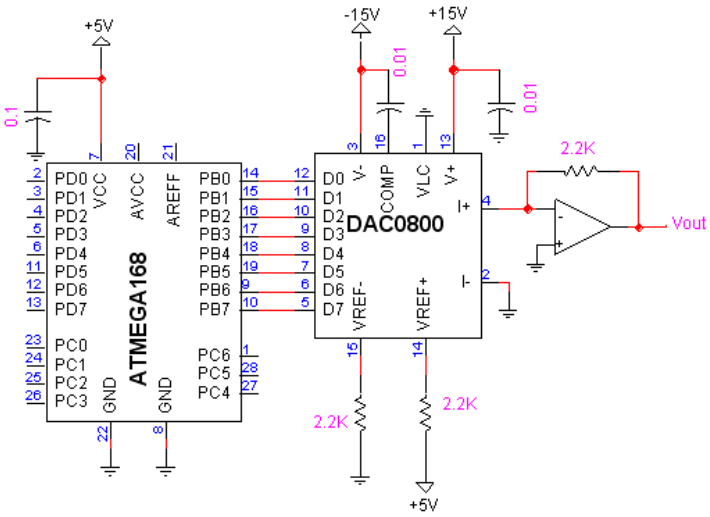
This is a current producing DAC, with the output determined by the digital code and by the input voltage V_{ref}. It is very fast (~100 ns), as with most DACs

DAC0800 Diagram

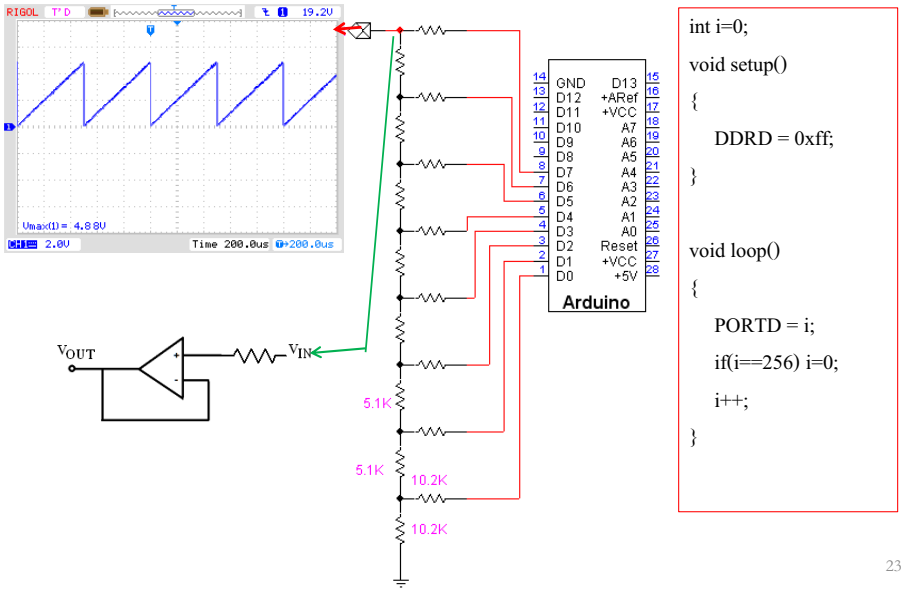


The current output makes it faster (no op amp to limit). It still has the R-2R ladder. To analyze this imagine this connection diagram. We will first focus on the REF AMP, then look at the currents in the R-2R ladder.

ATmega168 กับ DAC0800



Simple 8 bit DAC for the Arduino ATmega168



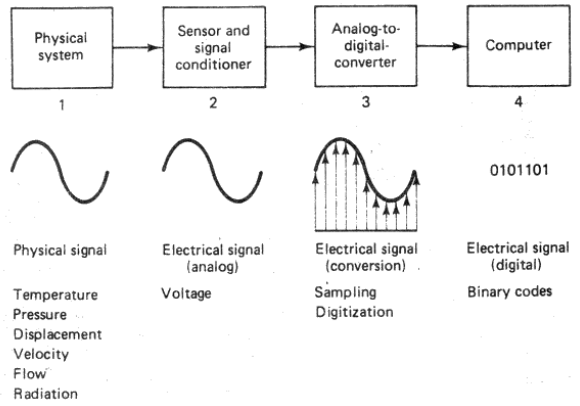

```
int i=0;
void setup()
{
    DDRD = 0xff;
}

void loop()
{
    PORTD = i;
    if(i==256) i=0;
    i++;
}
```

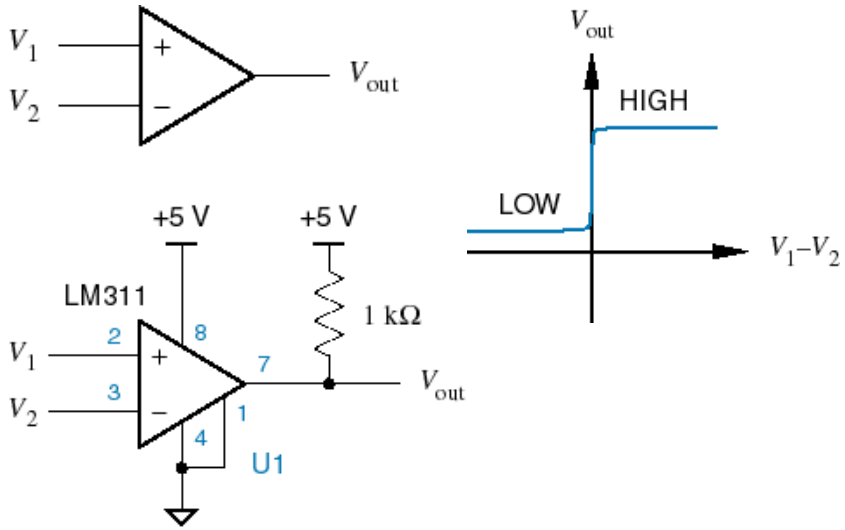

วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

(Analog-to-Digital Converter หรือ A/D หรือ A2D)

หมายถึงวงจรที่เปลี่ยนสัญญาณอนาลอกให้เป็นสัญญาณทางดิจิตอล



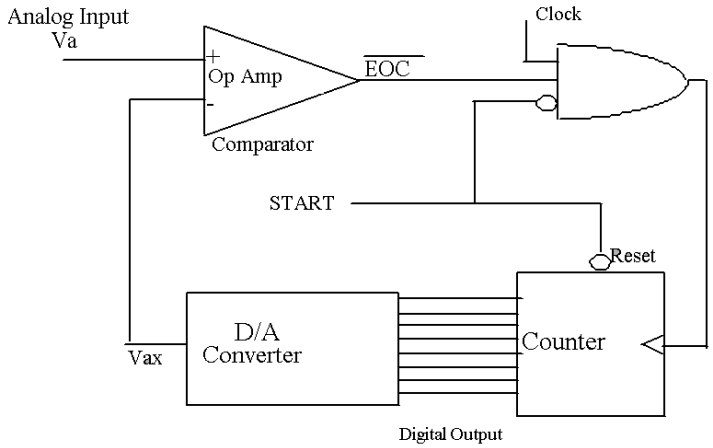
- Analog comparator = 1-bit A-to-D



ADC Techniques

- Digital Ramp ADC (Counter method)
- Successive Approximation
  - High Speed, Medium Resolution
- Parallel Comparator ("Flash")
  - Very high-speed conversion
- Dual Slope
  - Slow Speed, High Resolution

Digital Ramp ADC (Counter method)



•This simplest ADC uses a binary counter as the register

•A Start pulse resets the counter & disables the And gate

•With all 0s at its input, the DAC's output is VAX=0 volts

•Since VAX < VA, the op-amp EOC output will be High

•When Start returns Low, the AND gate is enabled.

•As the counter advances, the DAC output, VAX, increases one step at a time

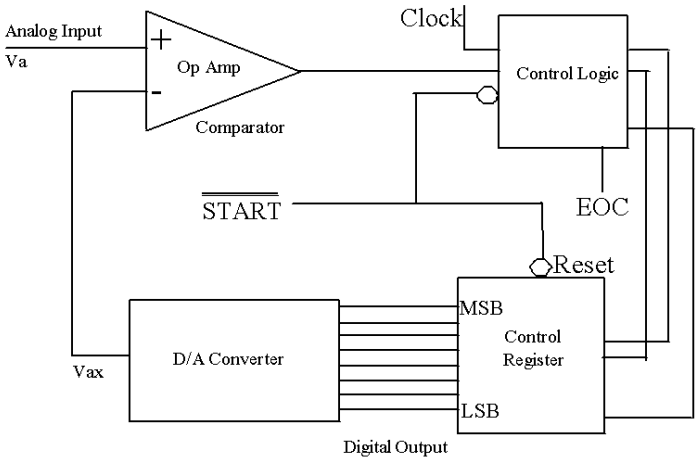
•This continues until VAX reaches a step that just exceeds

•VA by about VT. EOC is then Low disabling the AND.

•The A/D Conversion is now complete and the contents of the counter are the digital representation of VA.

•The digital data is lost at the next START pulse.

Successive Approximation ADC (SAC)

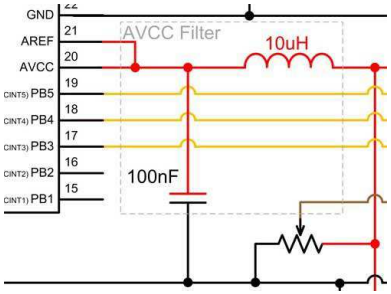


ADC ใน ATMega168

AVCC

Atmega168 มีขาแหล่งจ่ายแรงดันอยู่ 2 ขา คือ VCC และ AVCC.

AVCC เป็นแหล่งจ่ายสำหรับ PC0-PC5 เมื่อทำเป็นขาอินพุทสำหรับสัญญาณ analog แหล่งจ่าย AVCC นี้ต้องการความเสถียรมาก ดังนั้นจะใช้ low pass filter ซึ่งประกอบด้วย inductor และ capacitor.



AREF

ขา AREF ใช้เป็นขาสำหรับแรงดันอ้างอิง ที่ 100% (เมื่อสัญญาณ analog เท่ากับค่านี้ จะได้ค่าดิจิตอลเท่ากับ 1024)

Analogue Input

Atmega168 มีขาสำหรับสัญญาณ analog อยู่ 6 ขา PC0 to PC5. – ขาเหล่านี้เป็นได้ทั้ง digital I/O และ analogue input

Registers

Atmega168 มีรีจิสเตอร์ที่ใช้งานเกี่ยวกับการแปลงสัญญาณอนาลอกอยู่ 6 ตัว

Register	Description
• ADMUX	ADC Multiplexer Selection Register
• ADCSRA	ADC Control and Status Register A
• ADCSRB	ADC Control and Status Register B
• DIDR0	Digital Input Disable Register 0
• ADCL	ADC Data Register – Low
• ADCH	ADC Data Register – High

ADMUX

The ADMUX register allows you to control:

- The Reference Voltage
- Left adjustment of results (used for 8 bit results)
- Selection of input channel

ADMUX – ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 22-2. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 22-2. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

ADMUX : ADLAR และ MUX3:0

- **Bit 5 – ADLAR: ADC Left Adjust Result**

กำหนดผลลัพทธ์การแปลง

1 = left adjust

0 = right adjusted

รายละเอียดดู “ADCL and ADCH – The ADC Data Register”

- **Bits 3:0 – MUX3:0: Analog Channel Selection Bits**

เลือกสัญญาณอินพุทที่ต้องการแปลง

Table 22-3. Input Channel Selections

MUX3:0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	
1001	
1010	
1011	
1100	
1101	
1110	1.30V (V_{BG})
1111	0V (GND)

Atmega168

เลือก 1.1 V หรือ 0 V

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

1 = enables the ADC

0 = ADC is turned off ถ้า turnoff ในขณะแปลง การแปลงจะหยุด

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 6 – ADSC: ADC Start Conversion**

แบบ Single Conversion mode

ให้บิตนี้เป็น 1 = สั่งให้เริ่มแปลง

แบบ Free Running mode

ให้บิตนี้เป็น 1 = start the first conversion. The first conversion จะเริ่มหลังจาก ADC enabled

เมื่อแปลงเสร็จบิตนี้จะเป็น 0

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 5 – ADATE: ADC auto trigger enable**

1 = ทำงานแบบ Autotrig ADC จะเริ่มแปลงเมื่อ ขอบบวกของสัญญาณทริก โดยสัญญาณนี้เลือกจากบิต ADTS ใน ADCSRB

0 = ไม่ทำงานแบบ Auto trig

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

- **Bit 4 – ADIF: ADC Interrupt Flag**

ถูกทำให้เป็น 1 เมื่อแปลงเสร็จและข้อมูลถูกส่งไปที่ ADCL/ADCH

ใช้เป็นบิตบอกให้เกิดการอินเตอร์รัพท์ (บิต I ใน SREG ต้องเป็น 1 ด้วย)

บิตนี้จะถูก clear เมื่อการอินเตอร์รัพท์ได้รับการตอบสนอง

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

• Bit 3 – ADIE: ADC Interrupt Enable

1 = Enable

0 = Disable

When this bit is written to one and the I-bit in SREG is set, the ADC

Conversion Complete Interrupt

is activated.

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

• Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits

เลือกตัวหาร

Table 22-4. ADC Prescaler Selections

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

ADCL and ADCH – The ADC Data Register

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

ADEN = 1 ADC ทำงาน
ADATE = 0 ไม่ Auto trig
ADPS2-0 = 001 เลือก XTAL/8
ส่วนบิตอื่นๆ ให้เป็น 0 หมด

เขียนแบบที่ 1

```
ADCSRA = 0b10000001;
```

เขียนแบบที่ 2

```
ADCSRA = (1<<ADEN)|(0<<ADATE);
ADCSRA |= (0<<ADPS2)|(0<<ADPS1)|(1<<ADPS0);
```

ตัวอย่างโปรแกรมภาษา C

```
//----- Includes ------//
#include <avr/io.h>                 // AVR device-specific IO definitions
#include <avr/interrupt.h>          // Interrupt Service routine

#define F_CPU 8000000UL             // 8 MHz
#include <util/delay.h>         // header file implement simple delay loops

#include "lib_UART.c"                       // Use Module USART

#define Vadc       1024                     // 1024 (10-bit Resolution)

//----- delay_ms ------//
void delay_ms(uint16_t i)
{
   for (;i > 0; i--)
       _delay_ms(1);
}
```



```
int main(void)
{
    ADCSRA = (1<<ADEN)|(0<<ADATE); // ADC Enable & Auto Trigger Disable
    ADCSRA |= (0<<ADPS2)|(0<<ADPS1)|(1<<ADPS0); // XTAL/8
    ADCSRA =
    while(1)
    {       // Infinite loop; define here the
            //Aref,left adjust, select ADC0  (bit 2=0 bit 1 = 0 bit 0 = 0)
            ADMUX = 0b0000000;
            ADCSRA |= (1<<ADSC); // ADC Start Conversion
            while (!(ADCSRA &(1<<ADIF)));    // Wait Coversion completes
            adc0 = ADCW;                    // Read ADC
            _delay_ms(10);
    }
}
```


ฟังก์ชั่นเกี่ยวกับสัญญาณอนาล็อก ของ Arduino

- analogReference(type)
- analogRead()
- analogWrite() - PWM

analogReference(type)

Configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The options are:

- DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)
- INTERNAL: an built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328 and 2.56 volts on theATmega8 (*not available on the Arduino Mega*)
- INTERNAL1V1: a built-in 1.1V reference (*Arduino Mega only*)
- INTERNAL2V56: a built-in 2.56V reference (*Arduino Mega only*)
- EXTERNAL: the voltage applied to the AREF pin (**0 to 5V only**) is used as the reference.

analogRead()

Description

Reads the value from the specified analog pin. The Arduino board contains a 6 channel (8 channels on the Mini and Nano, 16 on the Mega), 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().

It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

Syntax

analogRead(pin)

ตัวอย่างโปรแกรมอ่านค่าอนาล็อกของ Arduino

```
int analogPin = 3;    // potentiometer wiper (middle terminal) connected
to analog pin 3       // outside leads to ground and +5V
int val = 0;          // variable to store the value read
void setup()
{
        Serial.begin(9600); // setup serial
}
void loop()
{
        val = analogRead(analogPin); // read the input pin
        Serial.println(val); // debug value
}
```


analogWrite() - PWM

analogWrite(pin, value)

Parameters

- pin: the pin to write to.

  กรณี atmega168 มี D3 D5 D6 D9 D10 และ D11

- value: the duty cycle: between 0 (always off) and 255 (always on).

ตัวอย่างสร้าง PWM ด้วย analogWrite

```
#define pwm_1 3
#define pwm_2 5
#define pwm_3 6
#define pwm_4 9
#define pwm_5 10
#define pwm_6 11

void setup()
{
 analogWrite(pwm_1, 10);
 analogWrite(pwm_2, 50);
 analogWrite(pwm_3, 100);
 analogWrite(pwm_4, 150);
 analogWrite(pwm_5, 200);
 analogWrite(pwm_6, 250);
}

void loop()
{

}
```