Chapter 13 (Mazidi): ADC, DAC and Sensor Interfacing

Outline:

- Data acquisition using ADC in HCS12
- Factors to consider in selecting an ADC chip.
- Programming the HCS12's ADC.
- Interfacing with a DAC (digital-to-analog converter) chip.

Analog to Digital Converter (ADC)

- **ADC devices**

  - widely used for data acquisition

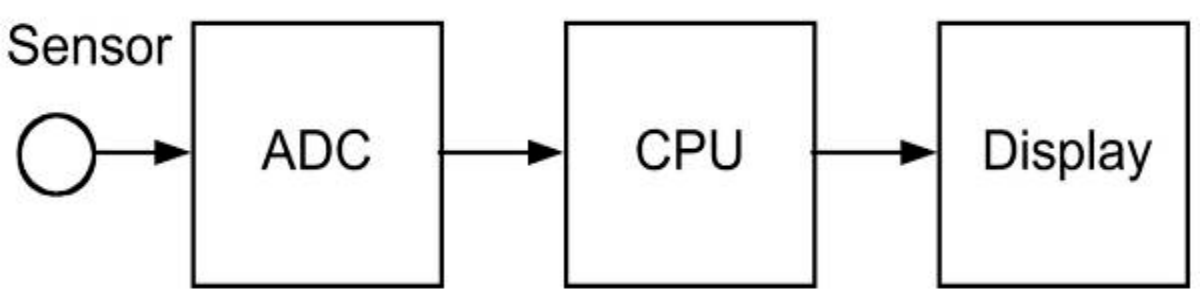
  - in the physical world everything is analog (continuous)

- Temperature, pressure (wind or liquid), humidity, and velocity are examples of physical quantities.

- Transducers/*sensors*:
  - temperature, velocity, light, and other natural quantities produce a voltage (or current) output

… ADC

Figure 13-1 Microcontroller Connection to Sensor via ADC



- ADC translates analog signals to digital numbers so the microcontroller can read and process them

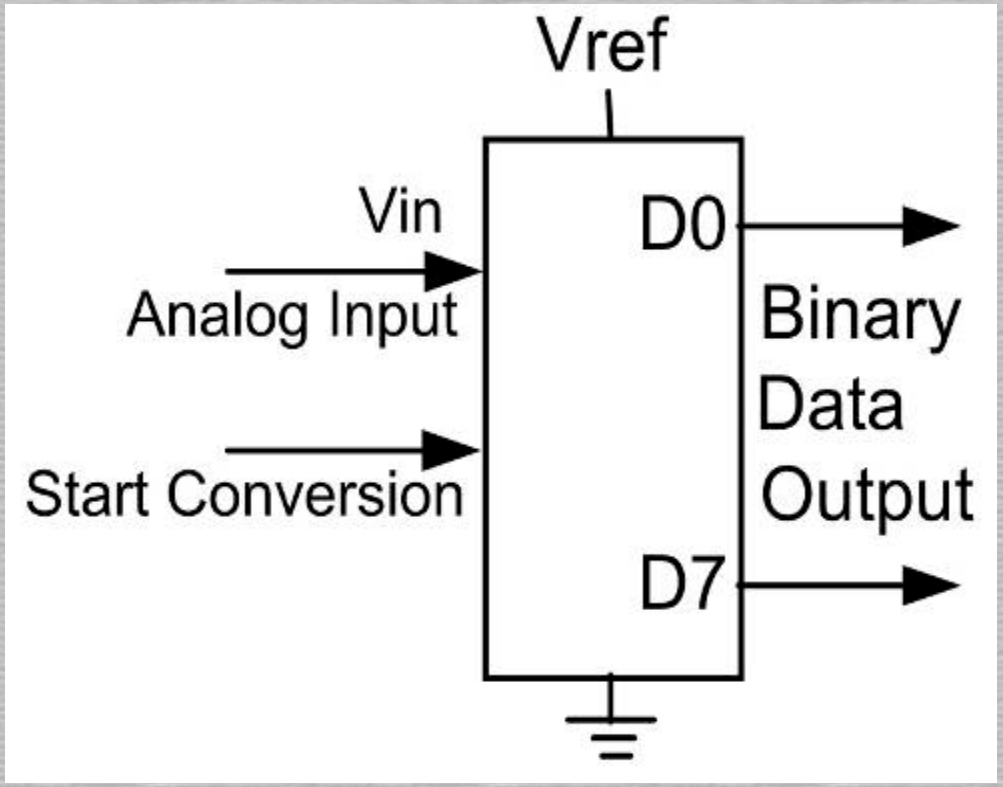


Figure 13-2 An 8-bit ADC Block Diagram

Major ADC Characteristics

- ***Resolution*** **-** ADCs have *n*-bit resolution, where *n* can be 8, 10, 12, 16, or even 24 bits.
  - higher-resolution → smaller **step size**
  - ***step size***: smallest change discerned by an ADC
- Step size can be controlled by V_{ref} (see previous page)

n-bit	Number of steps	Step size (mV)
8	256	5/256 = 19.53
10	1,024	5/1,024 = 4.88
12	4,096	5/4,096 = 1.2
16	65,536	5/65,536 = 0.076

Notes: V_{CC} = 5 V
Step size (resolution) is the smallest change that can be discerned by an ADC.

Resolution versus Step Size for ADC (Vref= 5 V)

… ADC Characteristics

- ***Conversion time***
  - time to convert analog input to a digital (binary) number.
- ***V_{ref}***
  - voltage connected to this pin, along with resolution of the ADC chip, dictate the step size
- *Example:* For an 8-bit ADC, step size is $V_{ref}/256$
- ***Digital data output* -** an 8-bit ADC has an 8-bit digital data output of D0–D7.

… ADC Characteristics

- ***Parallel versus serial ADC* -** parallel ADCs have 8 or more pins to bring out the binary data.

- Serial ADC: only one pin for data out
  - More time needed to get data in →CPU must get it one bit at a time, instead of a single read.

- Inside the serial ADC
    - parallel-in-serial-out shift register for sending data one bit at a time.

- Applications where space is a critical issue, using a large number of pins is not feasible.

  → serial ADC are becoming widely used

… ADC Characteristics

- ADC848 (left) is an example of a parallel ADC with 8 pins for the data output.
- MAX1112 (right) is an example of a serial ADC with a single pin for D_{out}.

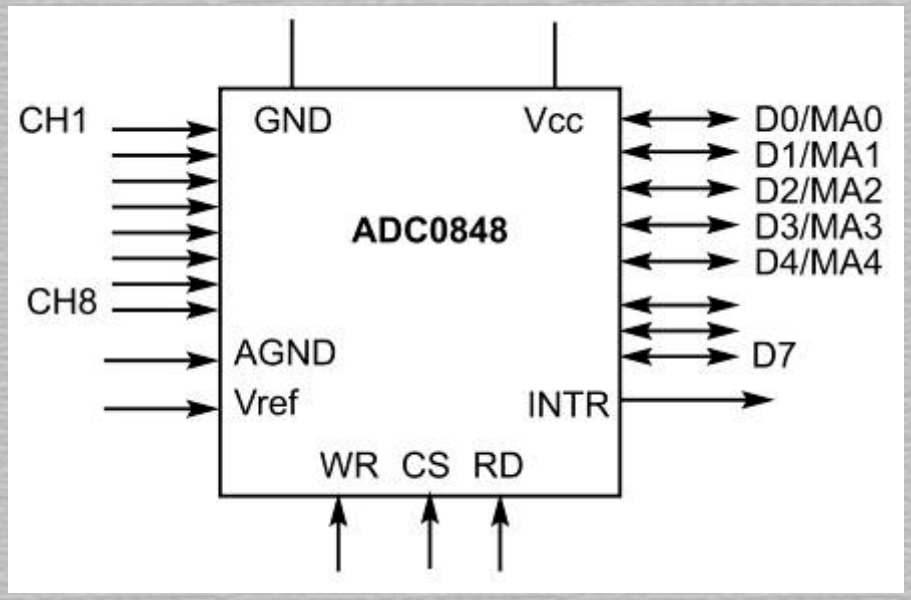


Figure 13-3 ADC0848 Parallel ADC Block Diagram

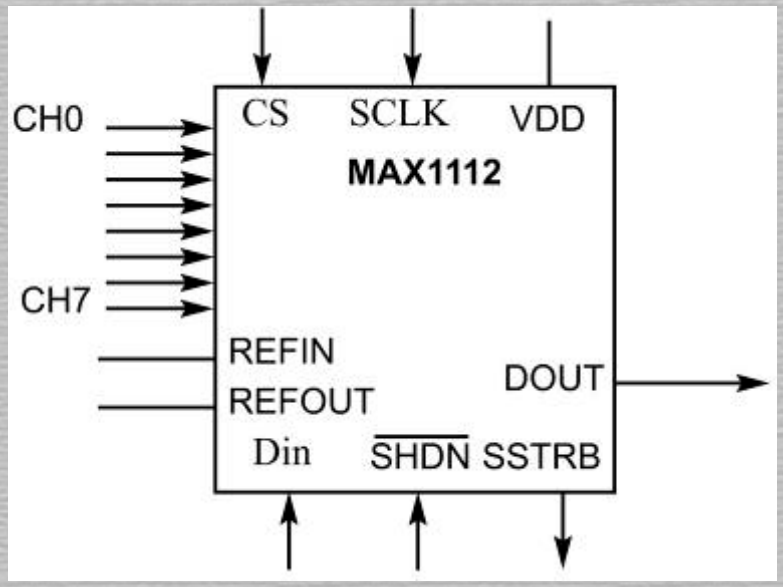


Figure 13-4 MAX1112 Serial ADC Block Diagram

… ADC Characteristics

- ***Analog input channels***
  - Many applications need more than one ADC.

- **Multiplexing** of analog inputs with 2, 4, 8, or 16 channels on a single chip is widely used.
  - as shown in the ADC848 and MAX1112

- ***Start conversion & end-of-conversion signals -*** an analog input channel and a single digital output register makes start conversion (SC) and end-of-conversion (EOC) signals necessary.

… ADC (HCS12)

- ***Successive Approximation ADC* -** a widely used method of converting analog input to digital output
- It has three main components
  - a) successive approximation register (SAR)
  - b) comparator
  - c) control unit.

- An advantage is that conversion time is fixed since it must go through all steps.

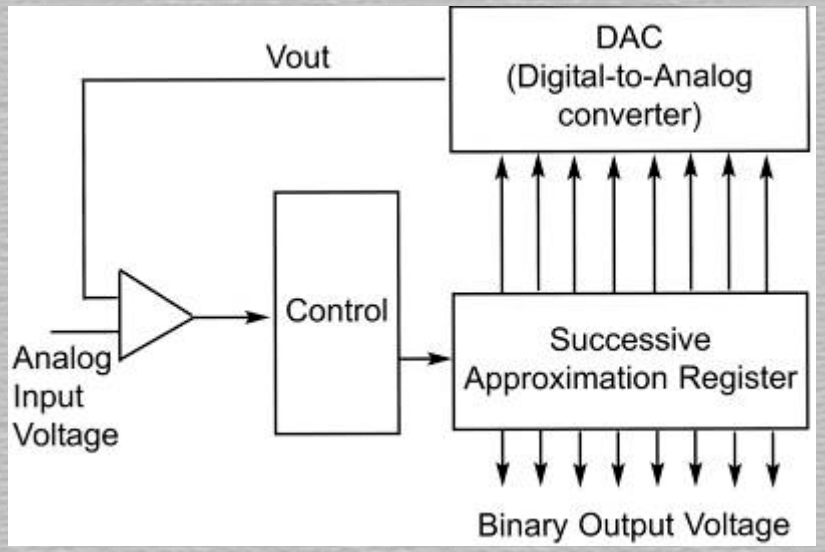
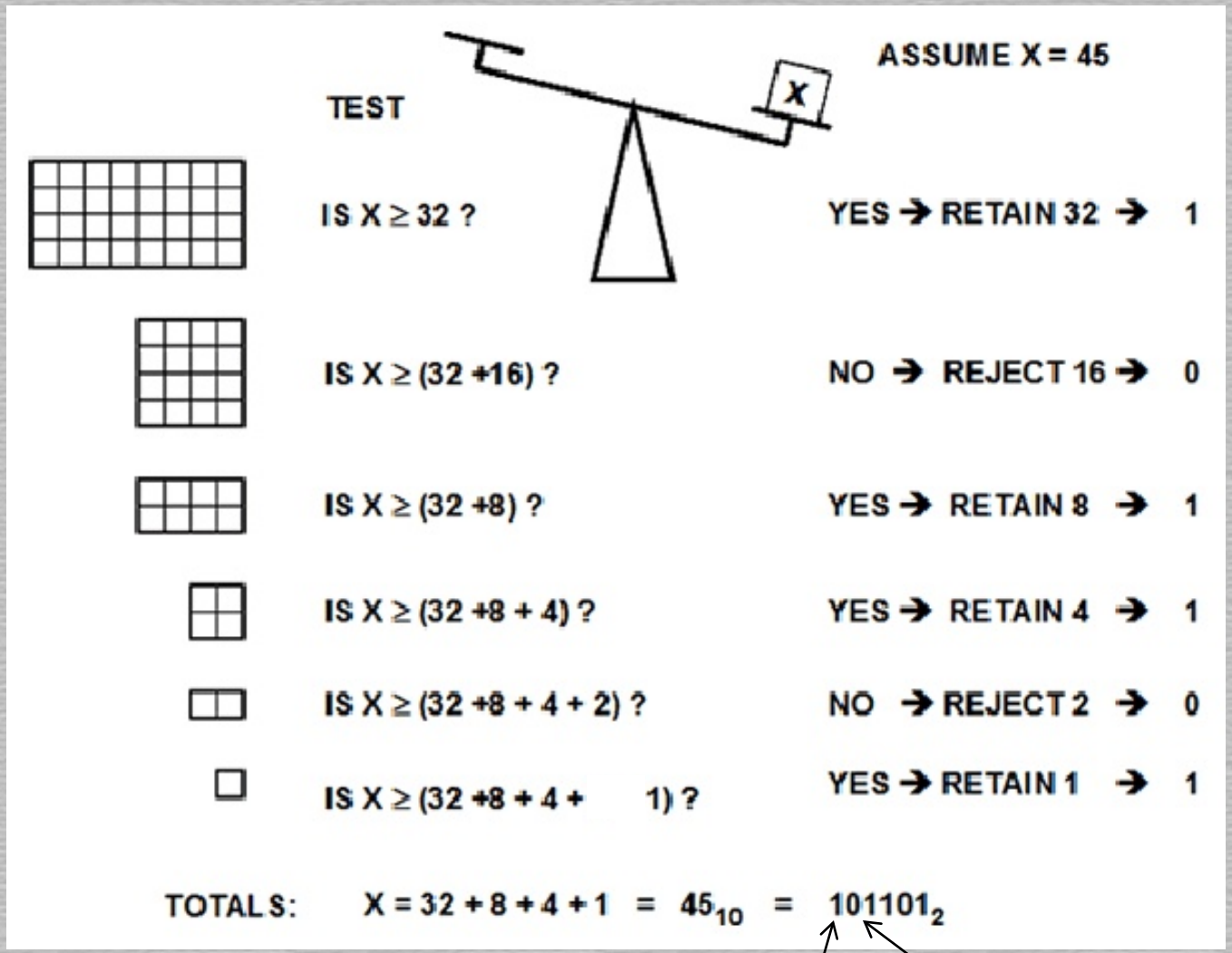


Figure 13-5 Successive Approximation ADC

How successive approximation works



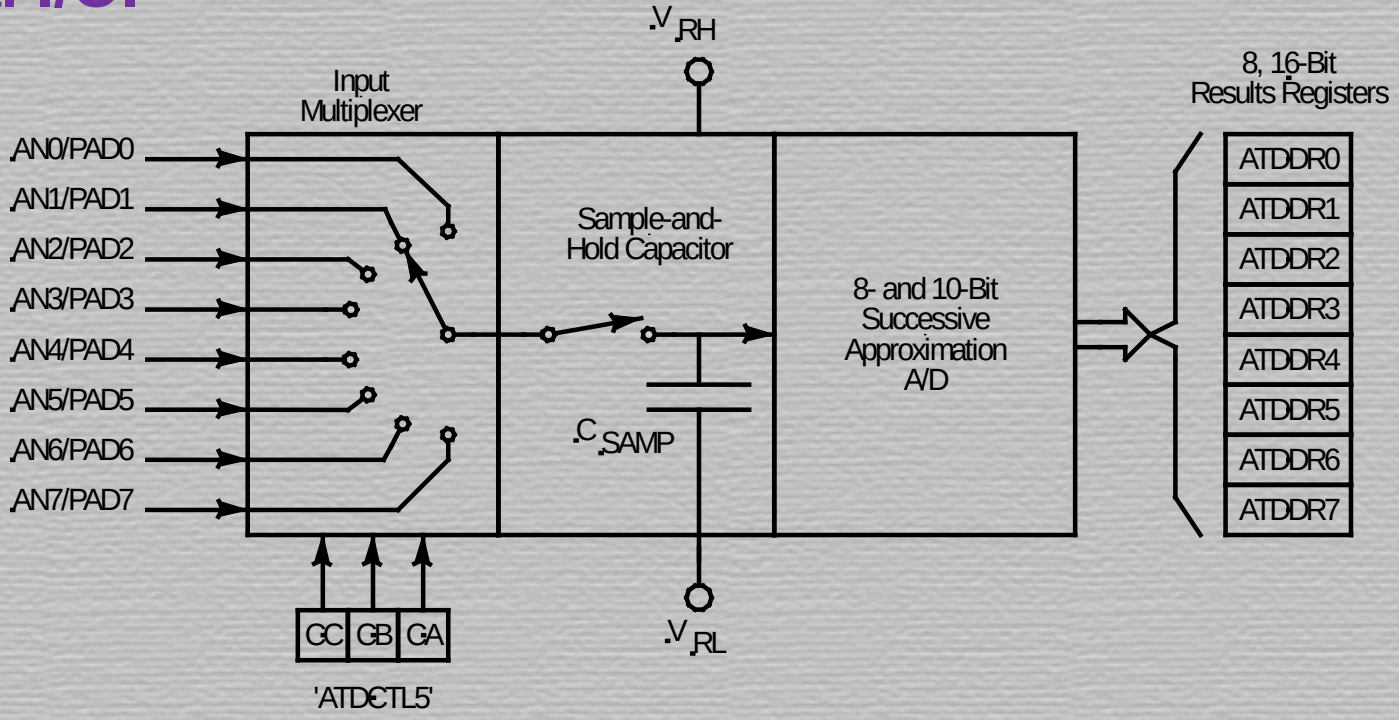
HCS12 ATD Programming

- The majority of HCS12 chips come with 8 channels of ADC.
- The Freescale HCS12 literature uses the ATD (analog-to-digital) designation instead of ADC (analog-to-digital converter).
- HCS12 chips have **two sets** of ADC called **ATD0 and ATD1**, **each have 8 channels of ADC**.

ADC in HCS12

- Two A/D converters in HCS12: ATD0 and ATD1.

- AN0 – AN7 may also be used as general purpose digital I/O pins and have another signal name PAD0 – PAD7.
- Cannot be used simultaneously as analog input and digital I/O.



… ADC Module (Freescale datasheet)

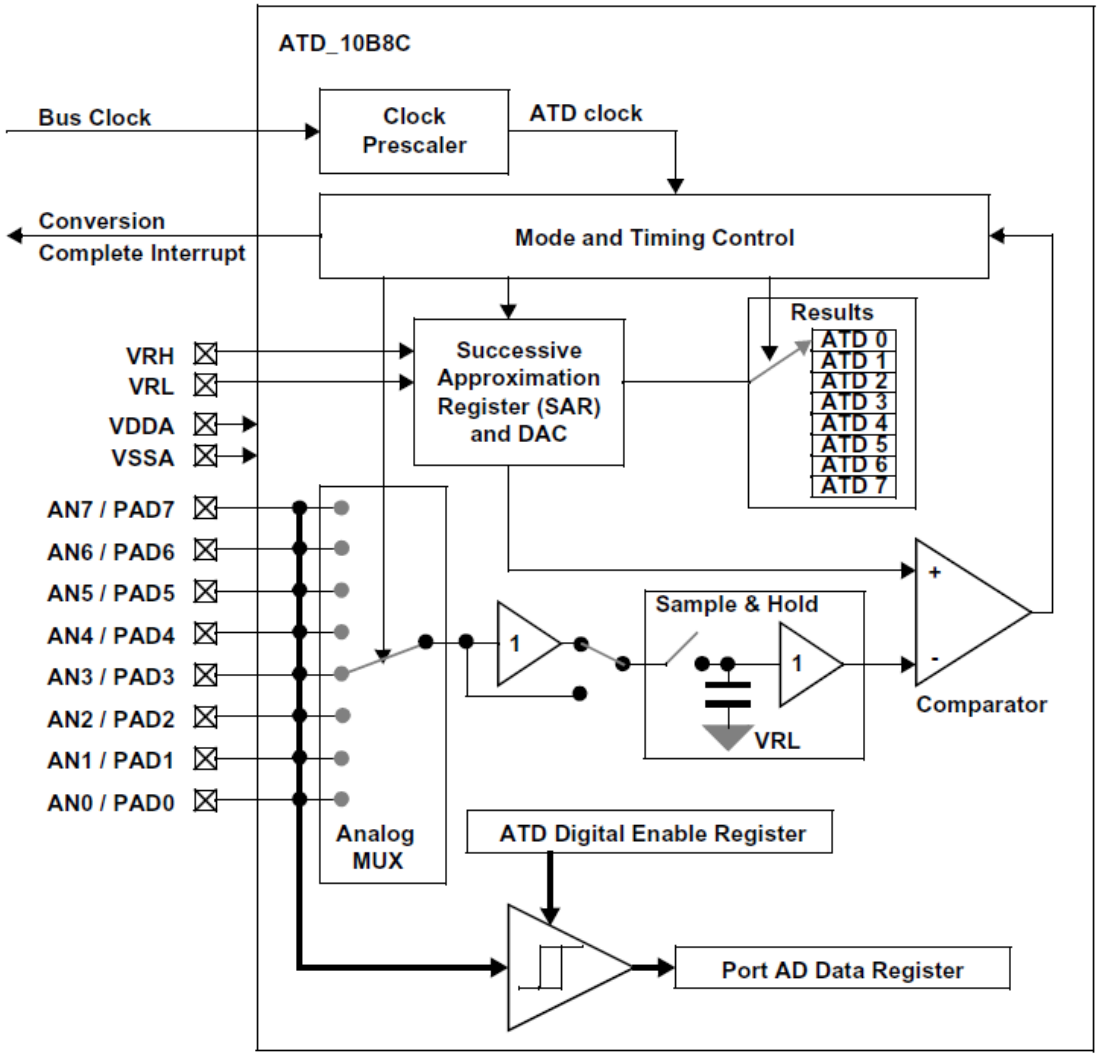


Figure 1. ATD_10B8C Block Diagram

Electrical model of an A/D input pin

(Freescale datasheet)

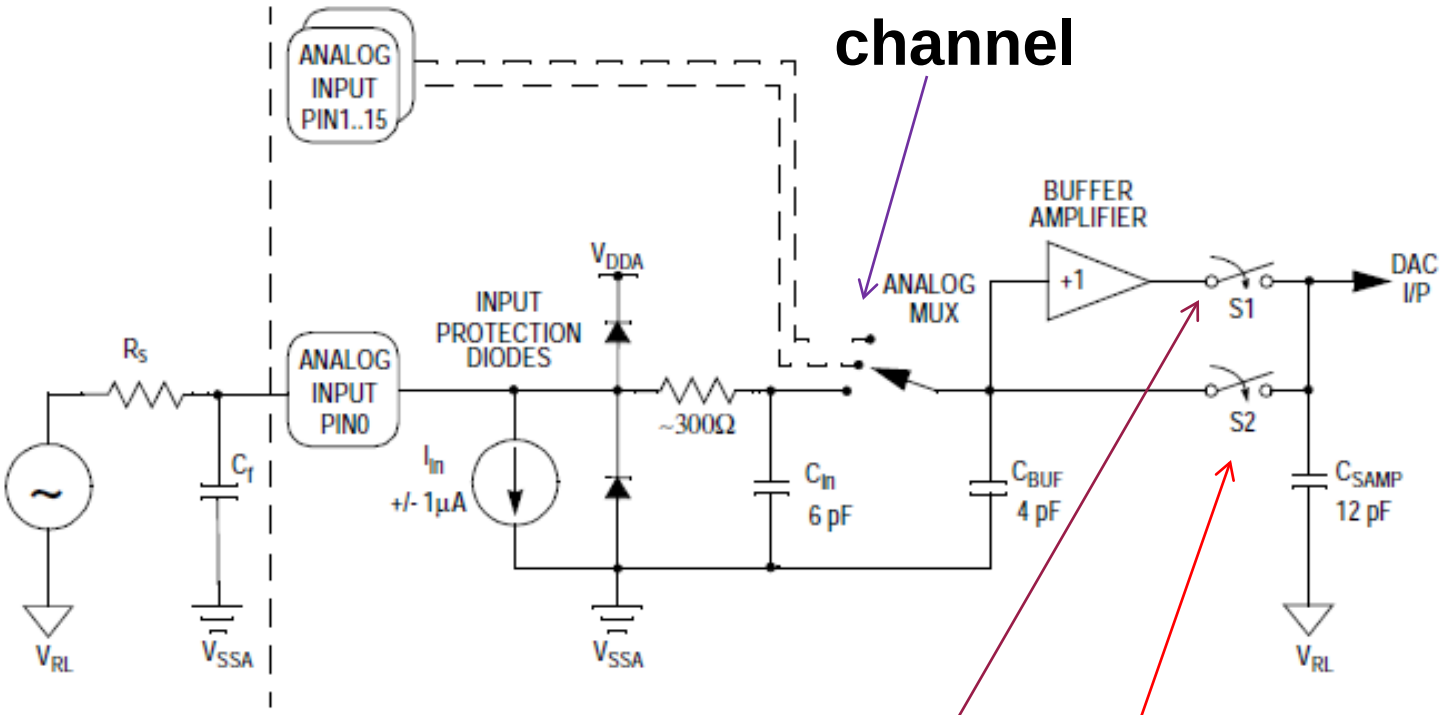


Figure 2. Electrical Model of an A/D Input Pin

ADC Control Registers:

Resolution: 8 or 10 bits, set by ATDCTL4 → successive approx. takes 8 or 10 A/D clocks

Sampling Time: Total of 4 to 18 A/D clock periods depending on resolution

Conversion time: 8 or 10 A/D clocks (successive approx)

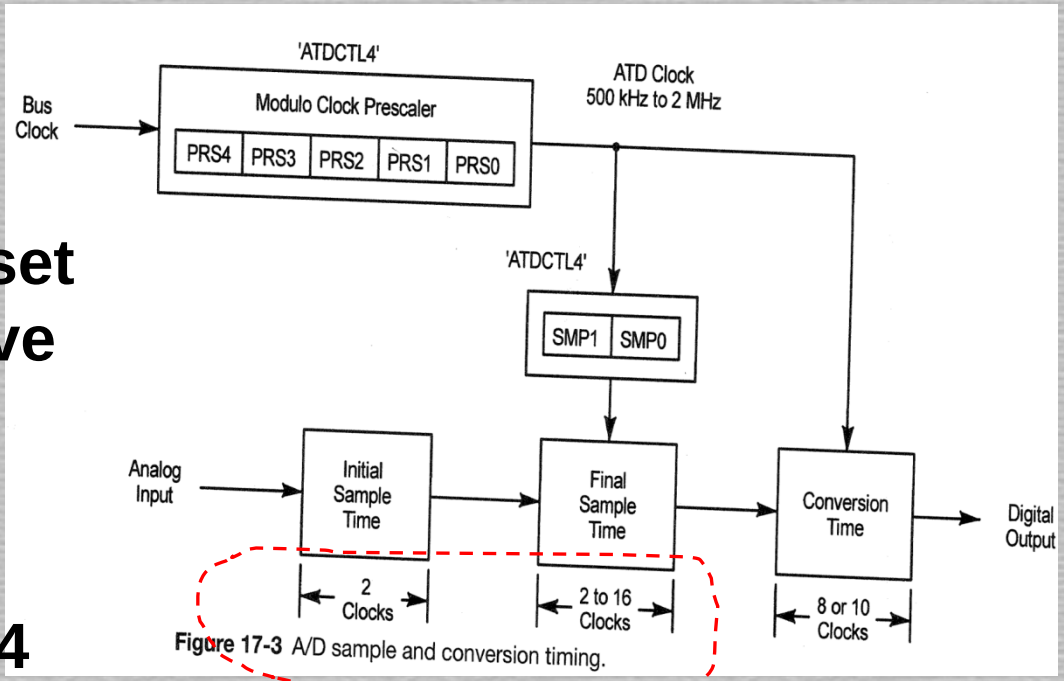


Figure 17-3 A/D sample and conversion timing.

Bus clock should be scaled to 500k-2MHz ATD clock

ADC Conversion Time

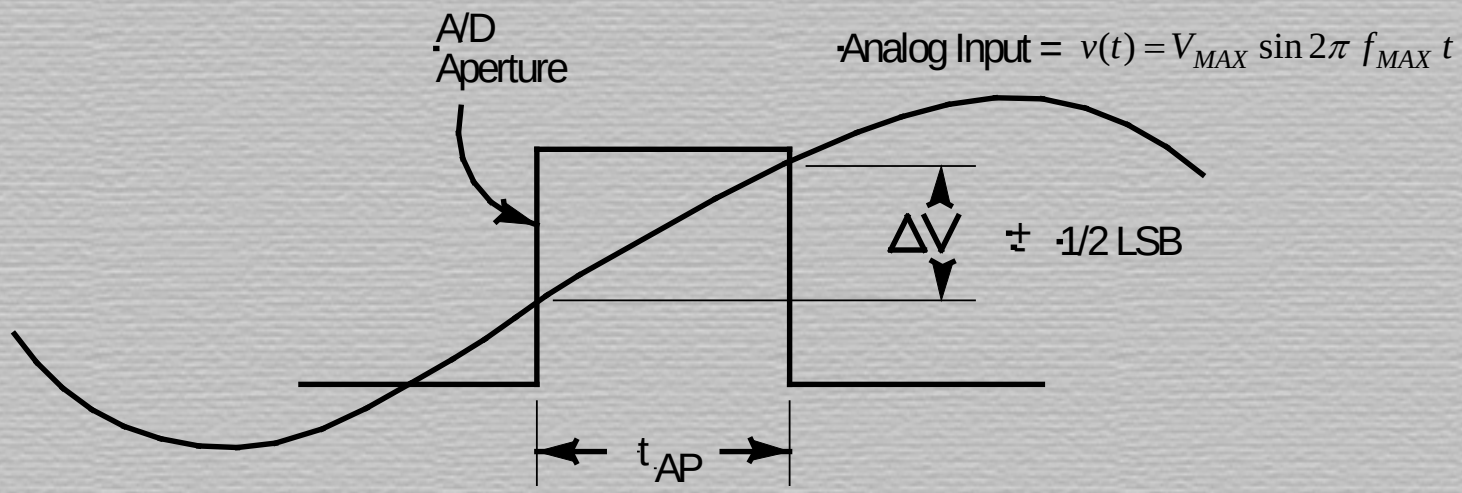
- The time the A/D takes to completely convert an analog value to a digital code.
  - Sets upper frequency limit that can be converted without aliasing.
  - E.g., conversion time=14 us (71kHz)

    → Upper freq limit set by Nyquist rate: 35.5kHz

- Shortest conversion time for an 8-bit conversion (when clock in 500ns = 1/(2MHz)):
  - (2 initial sample clocks) + (2 final sample clocks) + (8 conversion clocks) * 500 ns/clock = 6 μs.

ADC Aperture time error

- The time during which the A/D is sampling the signal (sample-time: during which a time-varying signal can change).
- Any change in the input signal during this time leads to an error called *aperture time error*.

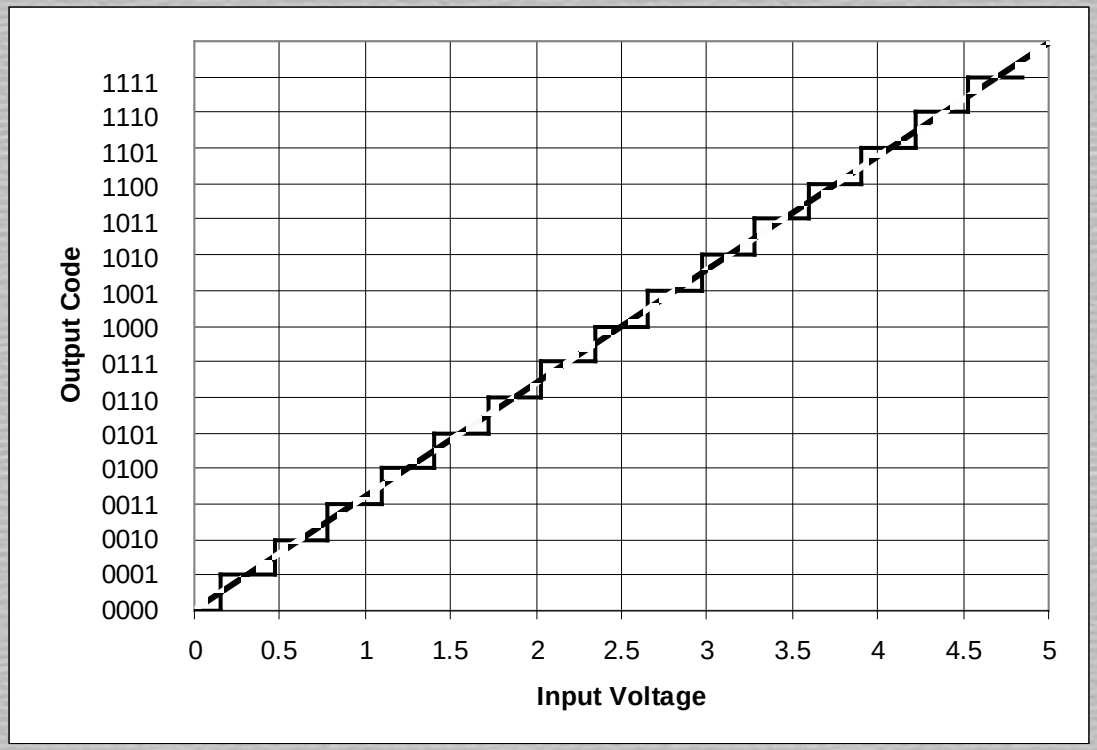


Max frequency for apperture error to be less than 1bit resolution:

f_max < 1/(2*pi*2^n*t_ap)

A/D transfer characteristic

- For an A/D with 8-bits and a 5V maximum (full scale) input, what is the smallest change in the input signal that can be detected?



- Answer: 5/256 = 0.0195 v

For above 4-bit A/D:

- What output code results from an input voltage of 3.1 V?

Other ATD control registers

- **ATDCTL2 register -** used to turn on the ATD using the **DPU** bit:
  - To reduce power consumption
  - A/D section is powered down upon reset.
  - Power up (turn on) the ADC section of the HCS12 with the ADPU bit of the ATDCTL2 register

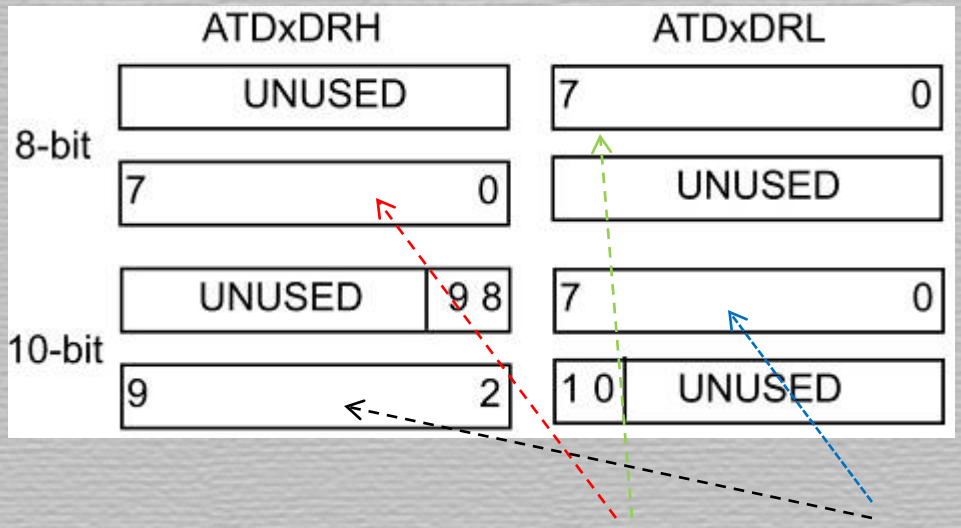
- Using options available in ATDCTL2, ATD conversion can be triggered by an external source.
  - an *interrupt to get data out when conversion is done*

... Other ATD control registers

- **ATDCTL3 register -** is used to select the number of conversions per sequence
  - **FIFO mode** allows us to store the consecutive conversions in consecutive registers and wrap around at the end.
  - **DSGN bits**
    - Result in signed or unsigned numbers:
    - Unsigned: 00 to $FF
    - Signed: $7F to $80 (+127 to –128)

… Other ATD registers

- **ATD result registers -** for each channel we have an ATD conversion result register called ATDDRHx:ATDDRLx
  - where x is 0, 1, 2, 3, etc.



Right- and Left-Justified Data for **8-bit** and **10-bit** Data

  - for 8-bit resolution only one of the ATDDRHigh or ATDDRLow registers is used
  - SRES8 bit in **ATDCTL4** and **ATDCTL5** bit DJM decide the resolution and format, respectively

... Other ATD registers

- **ATD status registers -** most important is ATDSTAT0.
- After the ATD converts the analog input, it places the result in the ATDDRH:ATDDRL registers and raises the flag SCF (sequence conversion flag).

SCF	0	ETORF	FIFOR	0	CC2	CC1	CC0

SCF D7 Sequence Conversion Complete flag
1 = Conversion is complete and the result is in ATDDRH:ATDDRL.
0 = Conversion is not completed.

ATDSTAT0 (A/D Status Register0) Register

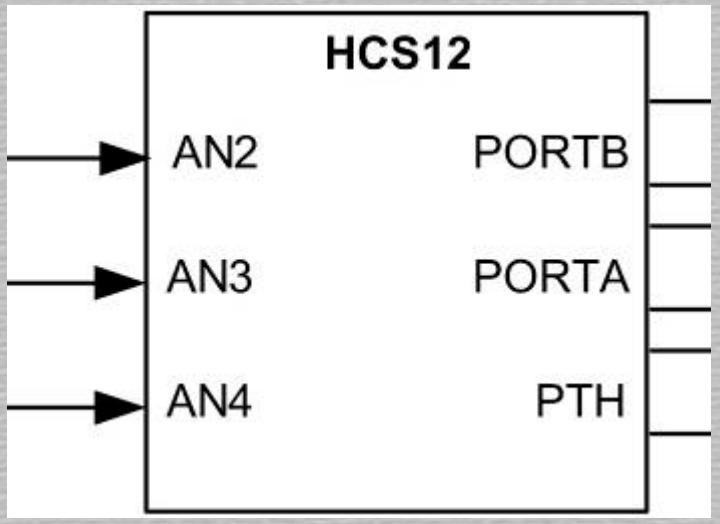
ATD Programming steps in HCS12 using polling

- To program the ATD converter:
  - turn on the ATD section of the HCS12
  - wait 20 μs for the ATD to get ready
  - use ATDCTL3 to select number of conversion sequences
  - use ATDCTL4 to select the conversion speed/resolution
  - ***** select channel & start conversion by writing to ATDCTL5
  - wait for the conversion to be completed by polling the SCF bit in ATDSTAT0
  - after SCF has gone HIGH, read ATDDRL and ATDDRH registers to get the digital data out
  - go back to step *****

… ATD Programming in the HCS12

- **Programming ATD for multiple input channels**

- **Use MULT bit option in the ATDxCTL5:**
  - If MULT = 0, only one channel is converted & the channel number is given by CC:CB:CA bits of the ATDxCTL5
  - If MULT = 1, we can convert two or more channels

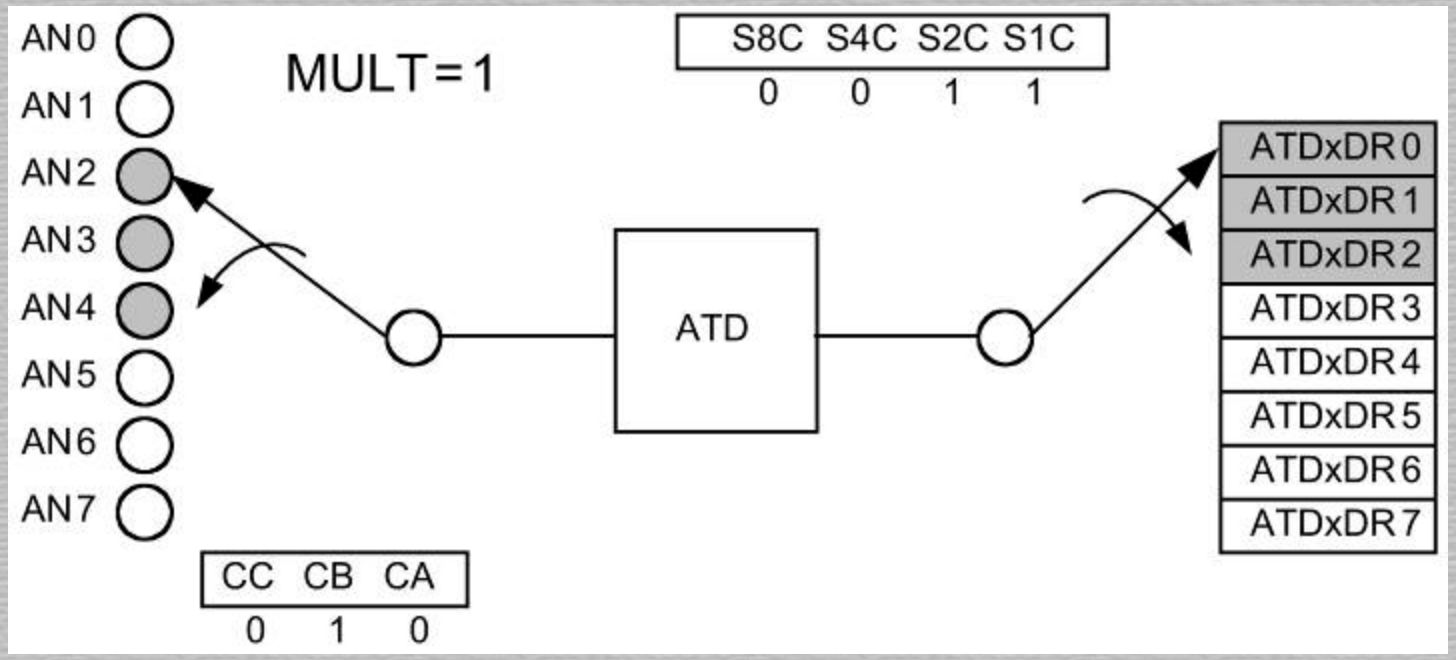


A/D Connection for multiple inputs

... ATD Programming in the HCS12

- To read channels 2, 3, and 4:

  ATDxCTL3 is set for sequence length = 3.



Three Consecutive Channels are Sampled, Once Each

Sample ADC programming

ATDCTL3: convert once

ATDCTL4: 8-bit, 16 clocks (speed), clock pre-scalar: divide by 8

0X82=1000 0010: Right justified, start converting channel 2

Program 13-1C: This program gets data from channel 2 (AN2) of ADC and displays the result on PORTB. This is the C version of Program 13-1.

```
//Program 13-1C
#include <mc9s12dp512.h>      /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dp512"
void DELAY20uS(void);
void TDELAY(void);
void main(void)
{
   DDRB = 0xFF;                //make PORTB an output
   ATD0CTL2 = 0x80;            //turn on ATD0
   DELAY20uS();          //20 microsec for ATD0 to get ready
   ATD0CTL3 = 0x08;
   ATD0CTL4 = 0xE3;
   for(;;)               //do it forever
     {
     ATD0CTL5 = 0x82;        //convert Channel 2 once (SCAN=0)
      //wait for conversion
     while(!(ATD0STAT0 & ATD0STAT0_SCF_MASK));
     PORTB = ATD0DR0L;         //display result on PORTB
     TDELAY();                 //wait so we can see the output
     }
}
void DELAY20uS(void)
  {
  int x = 40;
  while(x--);
  }
void TDELAY(void)
 {
  int x = 0;
  while(--x);
  }
```

SCF bit in ATDSTAT0 register indicates EOC

Power up

```
void wait20us (void);
void openAD0 (void)
{
        ATD0CTL2 = 0xE0;
        wait20us();
        ATD0CTL3 = 0x22;
        ATD0CTL4 = 0x05;
}
void wait20us (void)
{
        TSCR1    = 0x90;
        TSCR2    = 0;
        TIOS     |= OC0;
        TC0      = TCNT + 480;
        while(!(TFLG1 & C0F));
}
```

Write a subroutine to:

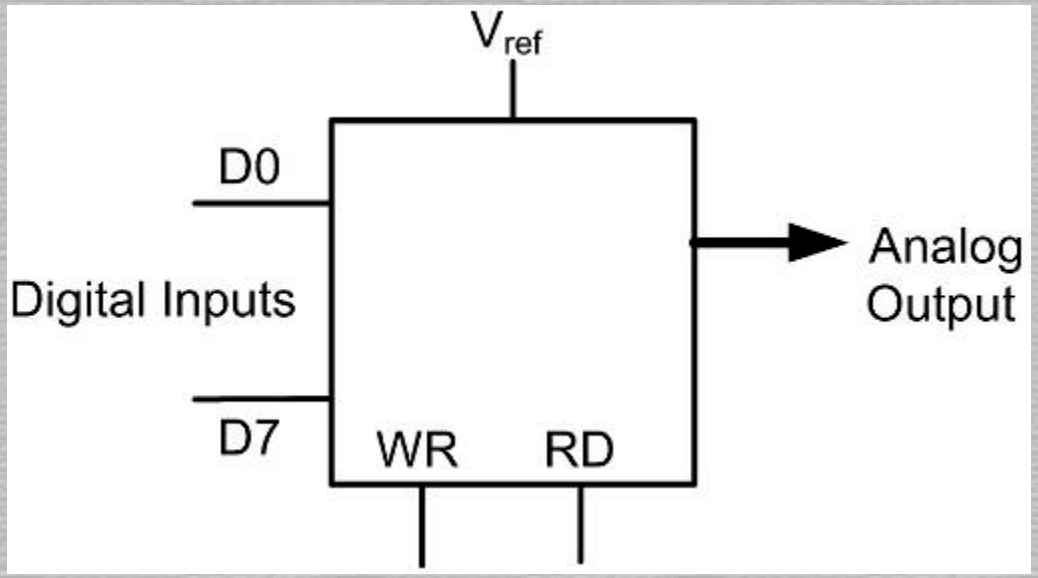
- Initialize ADC0
- Use channel 7
- Do 4 conversions (freeze during breakpoint)
- 10 bit operation
- 2MHz conversion on a 24MHz CPU clock (**0x05**: divide clock by 12)

```
int buff[20]; //collect 20 A/D values
void openAD0();

void main (void)
{
        int i;
        openAD0();
        for (i = 0; i < 5; i++) {
                ATD0CTL5 = 0x87;            /* start an A/D conversion */
                while (!(ATD0STAT0 & SCF)); /* wait for the A/D conversion to complete */
                buf[4*i + 0] = ATD0DR0;     /* save results right-justified */
                buf[4*i + 1] = ATD0DR1;
                buf[4*i + 2] = ATD0DR2;
                buf[4*i + 3] = ATD0DR3;
        }
….. Other
code…
}
```

DAC Interfacing

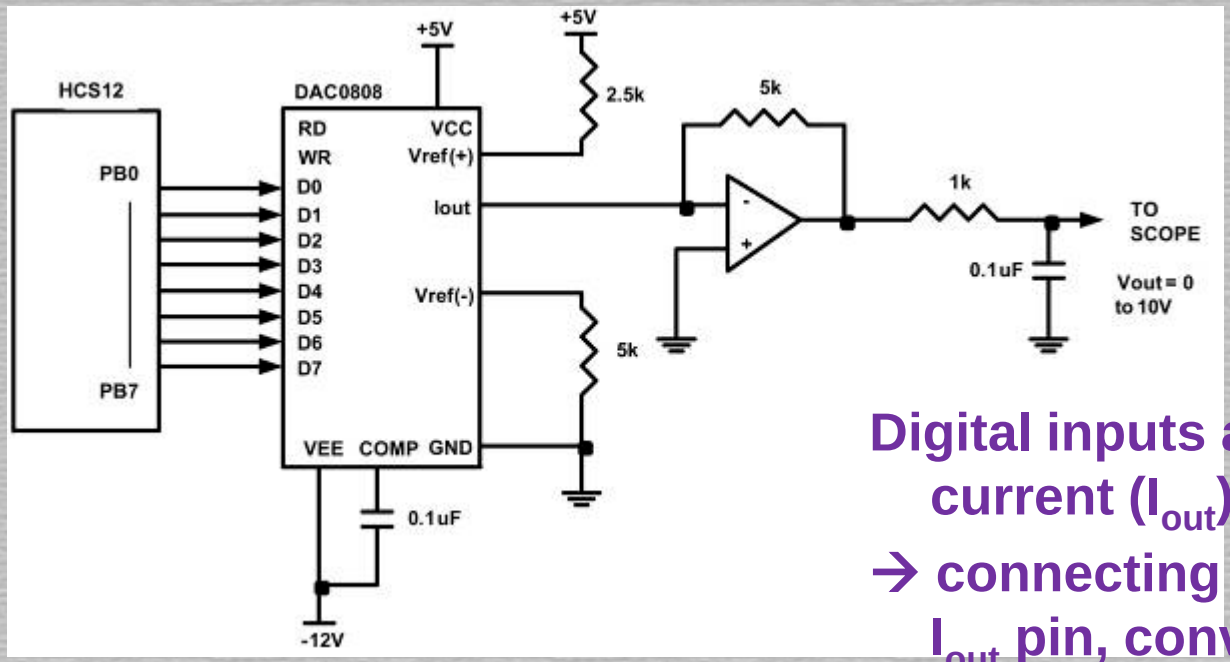
- **Digital-to-analog converter (DAC) -** The digital-to-analog converter (DAC) is a device widely used to convert digital pulses to analog signals.



DAC Block Diagram

... DAC Interfacing

- **DAC resolution** is a function of the number of binary inputs: Common ones are 8, 10, and 12 bits



HCS12 Connection to DAC0808

Digital inputs are converted into current (I_{out})

→ connecting a resistor to the I_{out} pin, converts the result to a voltage.

Q. How can we build a waveform generator using DAC?

Generating a sine wave: Using PORTB and DAC

```
#include <hidef.h>       /* common defines and macros */
#include <mc9s12dp512.h>      /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dp512"

unsigned char i;
const unsigned char TABLE[ 13] =
  {
  128,192,238,255,238,192,128,64,17,1,17,64,0
  };
void DELAY(void);
void main(void)
{
  DDRB = 0xFF; //PORTB as output

  PORTB = 0x0; //clear PORTB
  for(i=0;i<13;i++)
  {
     PORTB = TABLE[ i];
     DELAY();

  } /* wait forever */
}
```