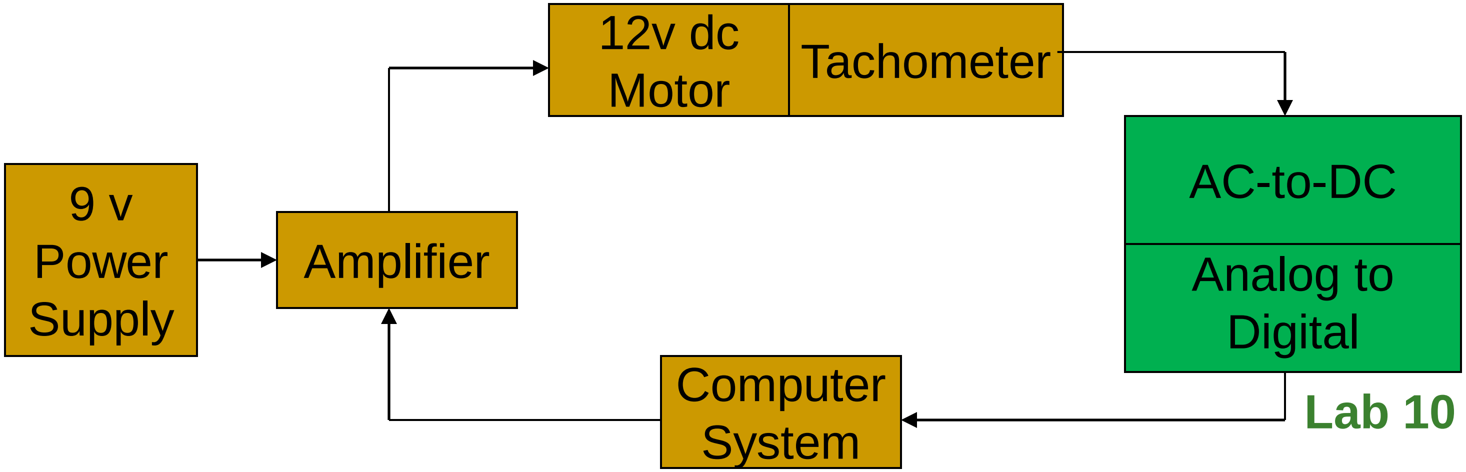
Lab 10. Speed Control of a D.C. motor

Speed Measurement:

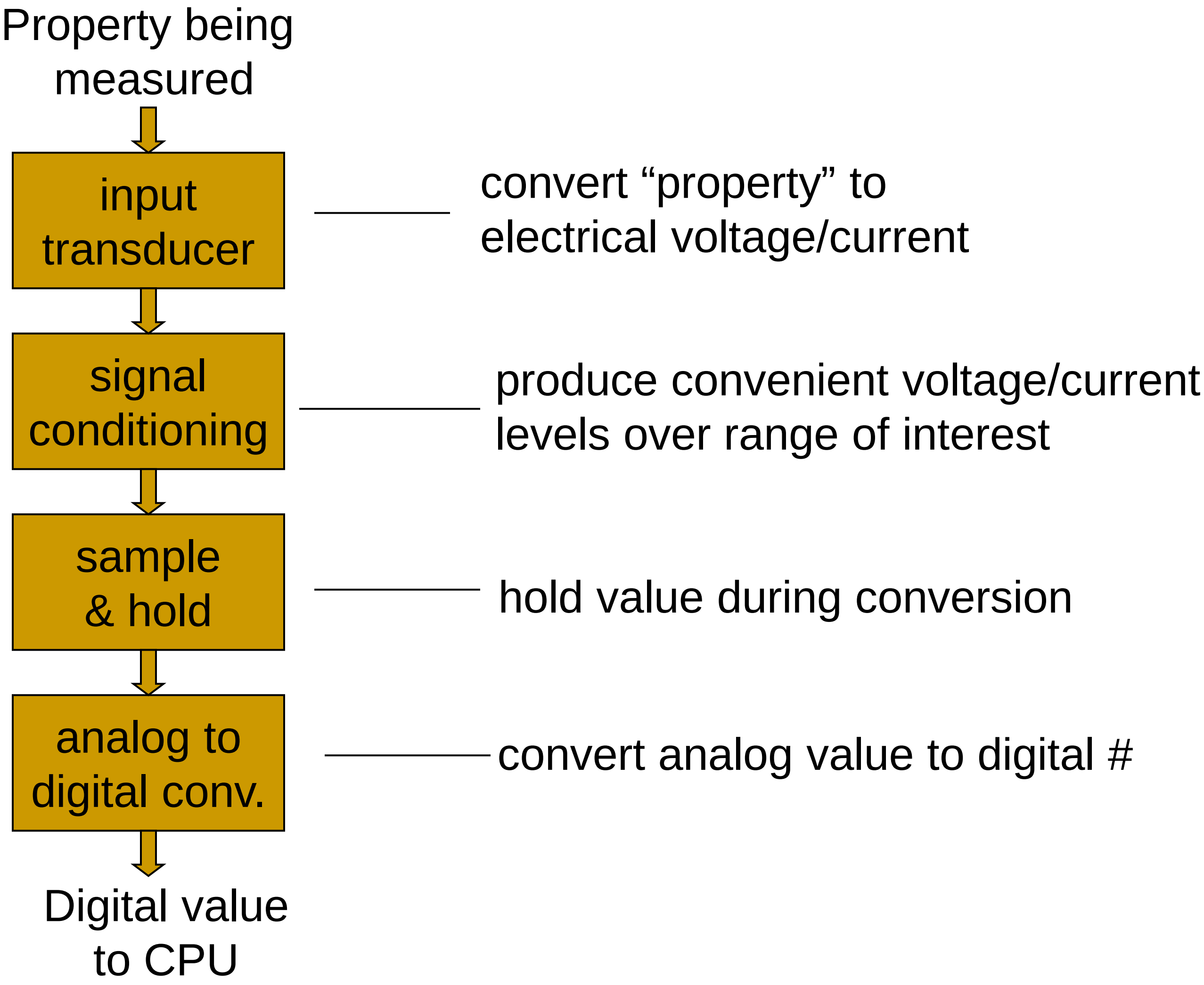
Tach Amplitude Method

Motor Speed Control Project

1. Generate PWM waveform
2. Amplify the waveform to drive the motor
3. Measure motor speed
4. Measure motor parameters
5. Control speed with a PID controller



Analog input subsystem



Property being measured

input transducer — convert “property” to electrical voltage/current

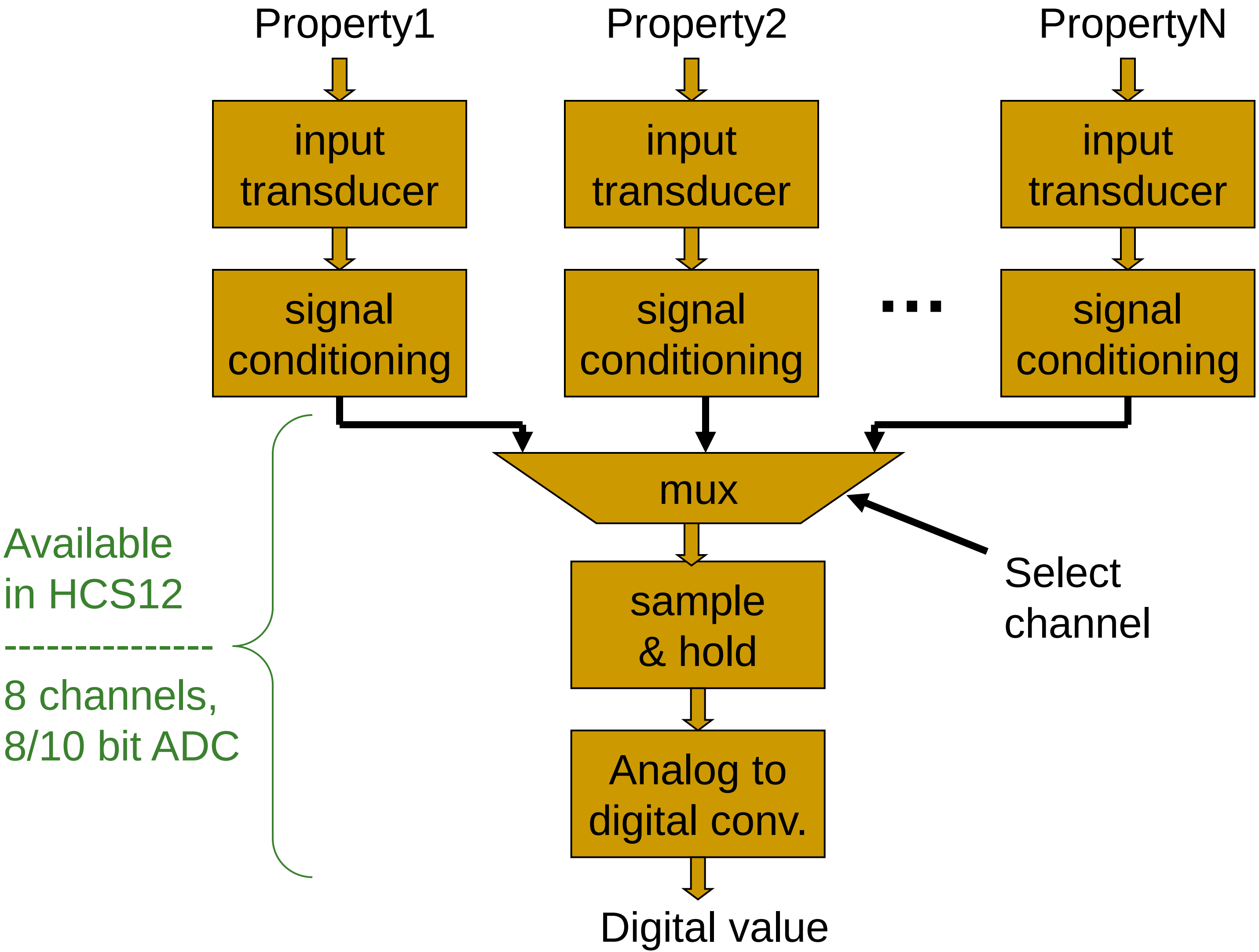
signal conditioning — produce convenient voltage/current levels over range of interest

sample & hold — hold value during conversion

analog to digital conv. — convert analog value to digital #

Digital value to CPU

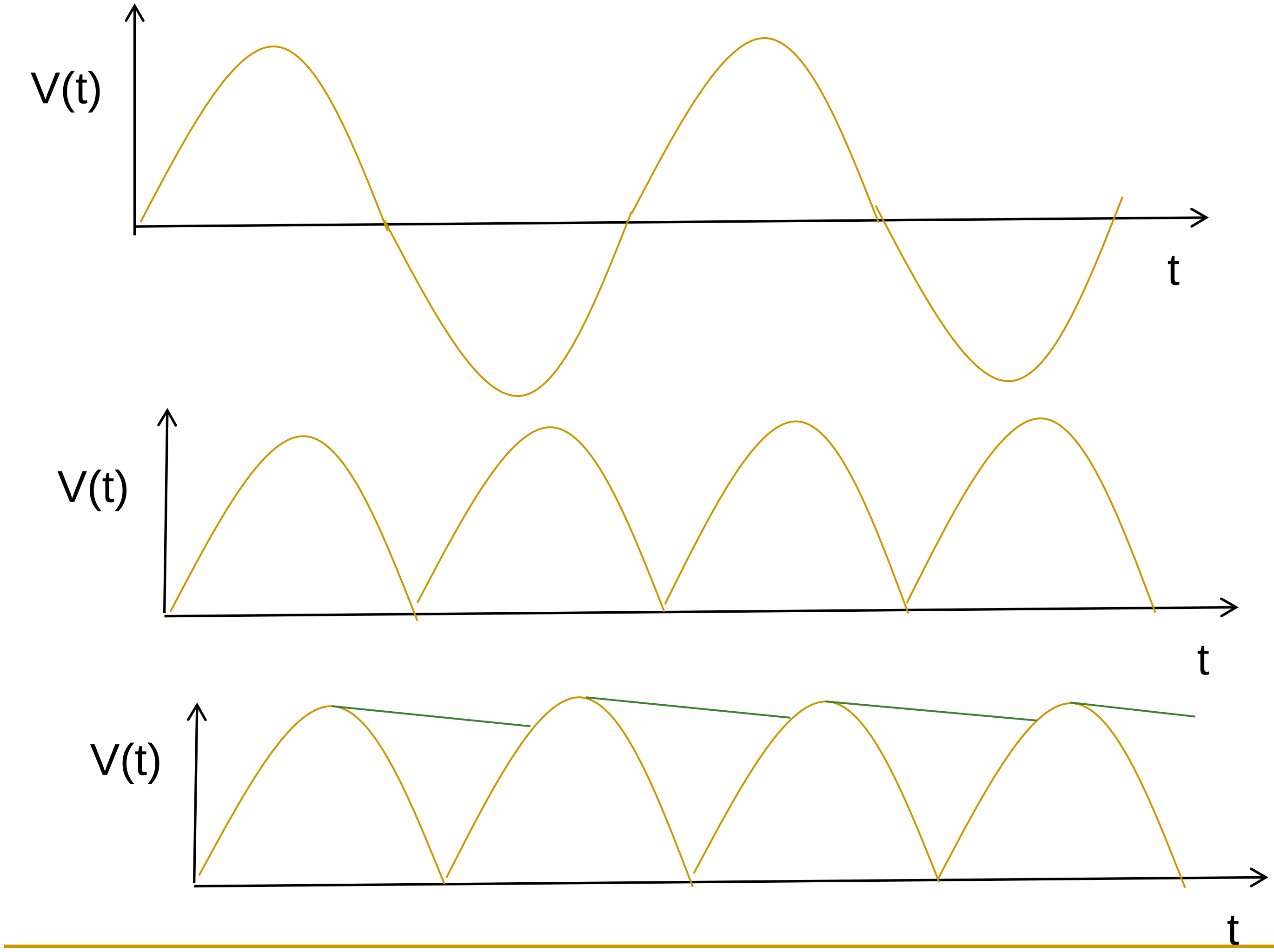
Multichannel analog input subsystem



Signal conditioning

- Produce noise-free signal over A/D converter input range
  - Amplify/attenuate voltage/current levels
  - Bias (shift levels to desired range)
  - Filter to remove noise
  - Isolation/protection (optical/transformer)
  - Common mode rejection for differential signals
  - Convert AC signal to DC for amplitude measurement (for this lab)

Example: AC to DC conversion



Unrectified

Full-wave
Rectified

Rectified &
Filtered

Analog to digital conversion

- Given: continuous-time electrical signal

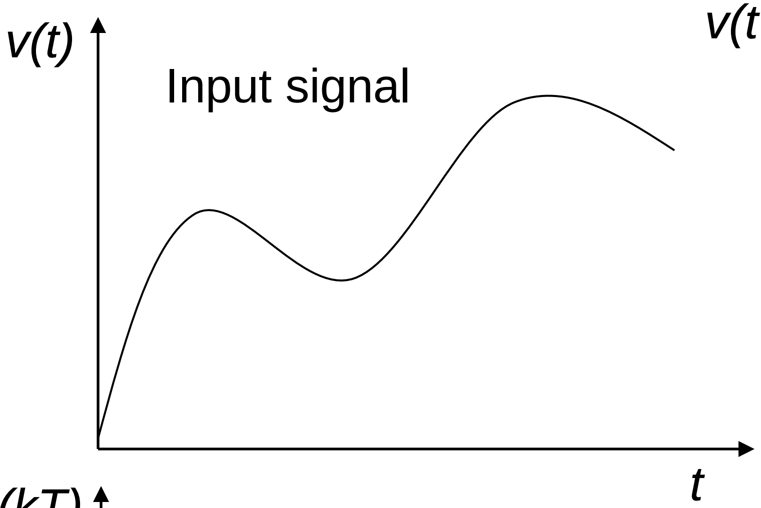
  v(t), t >=0

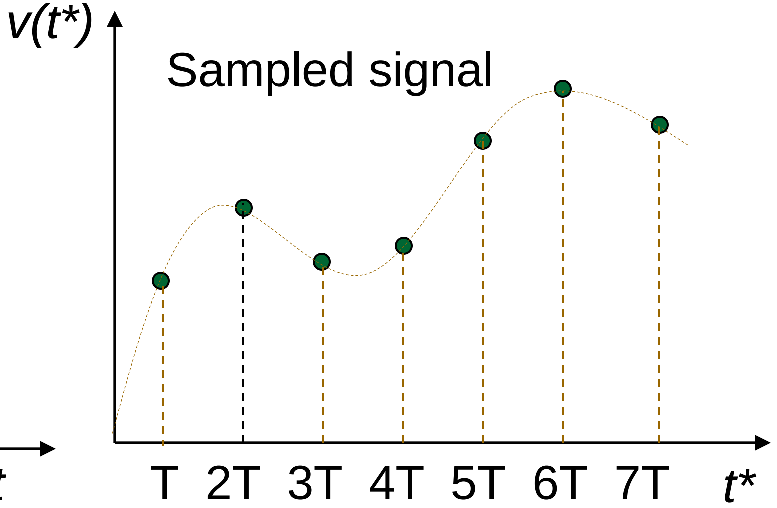
- Desired: sequence of discrete numeric values that represent the signal at selected sampling times :

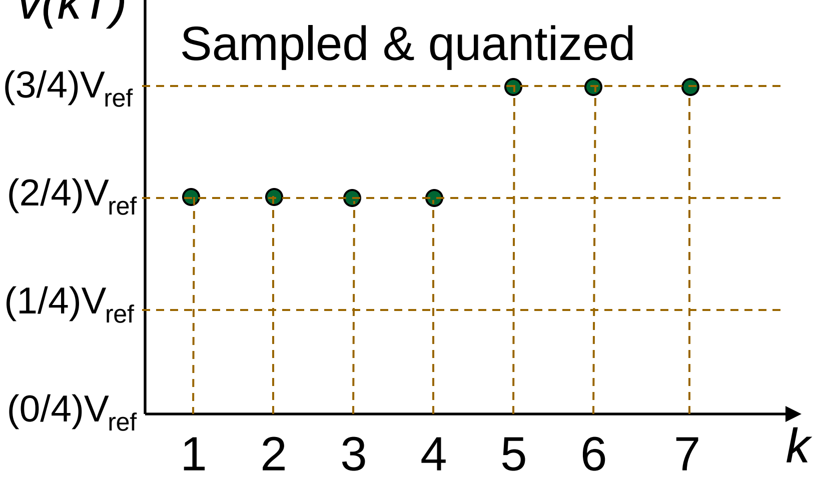
  v(0), v(T), v(2T),…v(nT)

  - T = “sampling time”: v(t) “sampled” every T seconds
  - n = sample number
  - v(nT) = value of v(t) measured at the n^{th} sample time and quantized to one of 2^k discrete levels

A/D conversion process





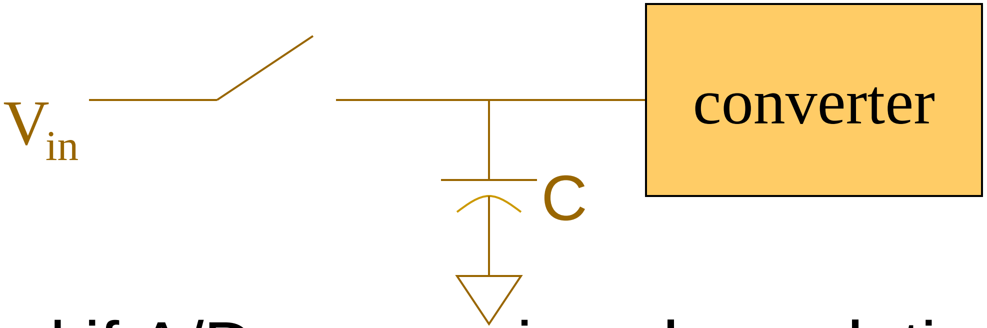


Sampled data sequence:

10, 10, 10, 10, 11, 11, 11

Binary values of n, where
$V(kT) = (n/4)V_{ref}$

Sample-and-hold



- Required if A/D conversion slow relative to frequency of signal:
  - Close switch to “sample” Vin (charge C to Vin)
    - Aperture (sampling) time = duration of switch closure
  - Open switch to “hold” Vin

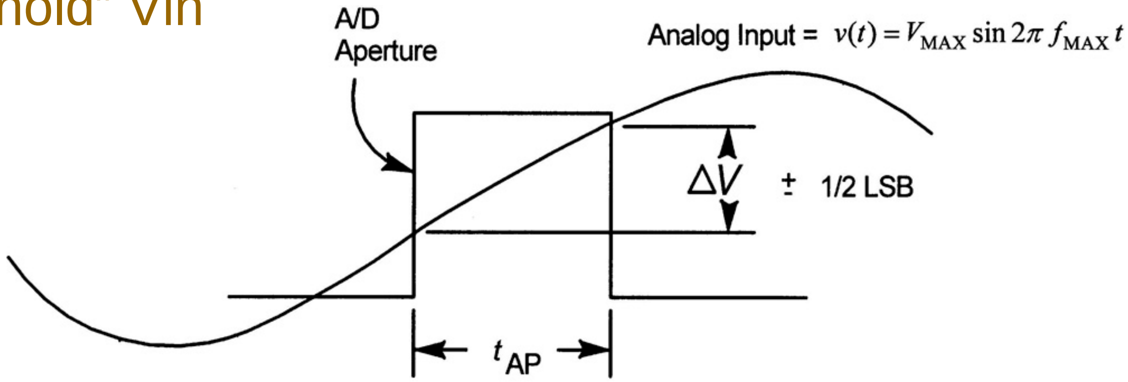

A/D
Aperture
Analog Input = $v(t) = V_{\text{MAX}} \sin 2\pi f_{\text{MAX}} t$
ΔV
± 1/2 LSB
t_{AP}

Figure 17-4 Aperture time error.

A/D conversion parameters

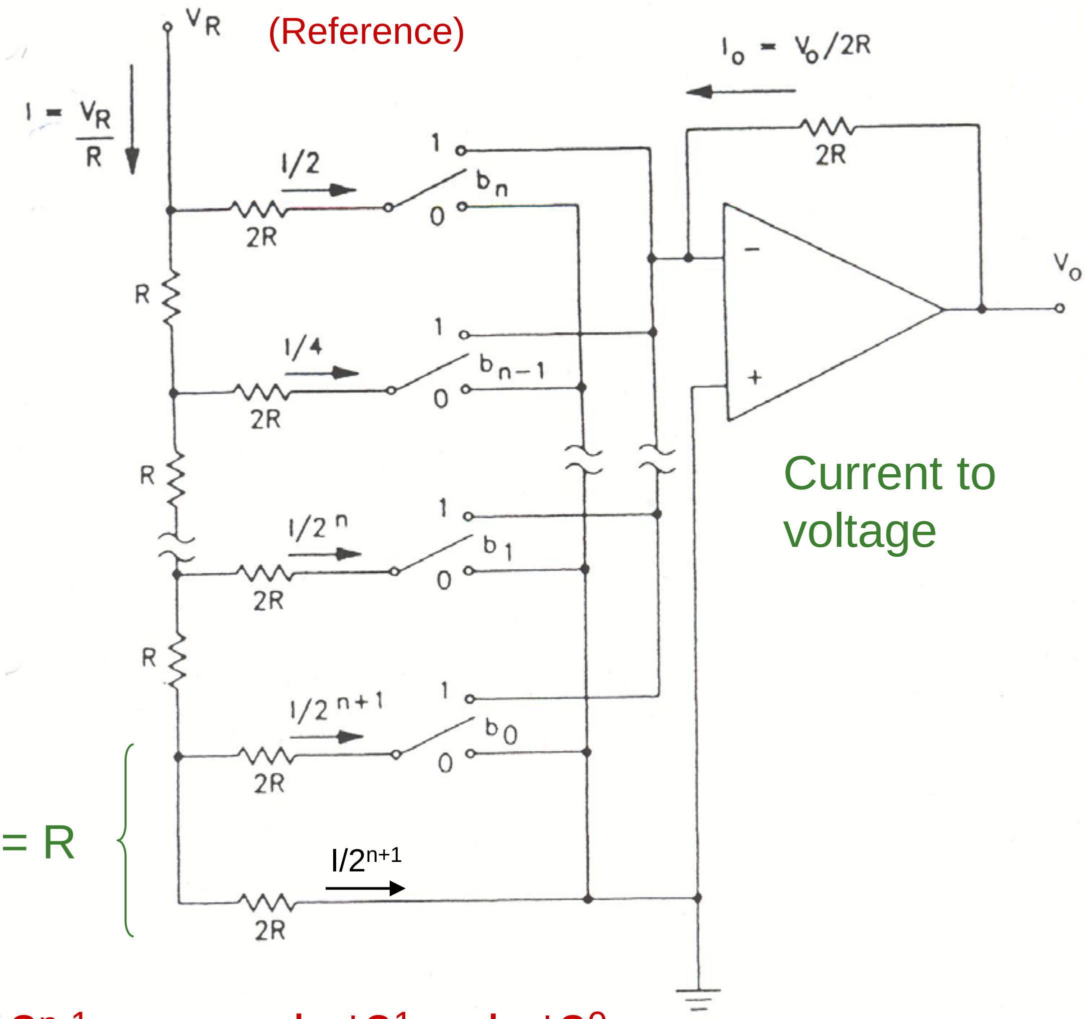
- Sampling rate, F (sampling interval T = 1/F)
  - Nyquist rate ≥ 2 x (highest frequency in the signal) to reproduce sampled signals
  - Examples:
    - CD-quality music sampled at 44.1KHz
      (ear can hear up to about 20-22KHz)
    - Voice in digital telephone sampled at 8KHz

- Precision (# bits in sample value)
  - k = # of bits used to represent samples
  - “precision”: each step represents $(1/2^k)*V_{range}$
  - “accuracy”: degree to which converter discerns proper level (error when rounding to nearest level)

Digital to analog conversion

R-2R Ladder Network

$$Vout = Vr\sum_{k=1}^{N} b_k\left(\frac{1}{2^k}\right)$$



(Reference)

Current to voltage

$I/2^{n+1}$

Equivalent resistance = R

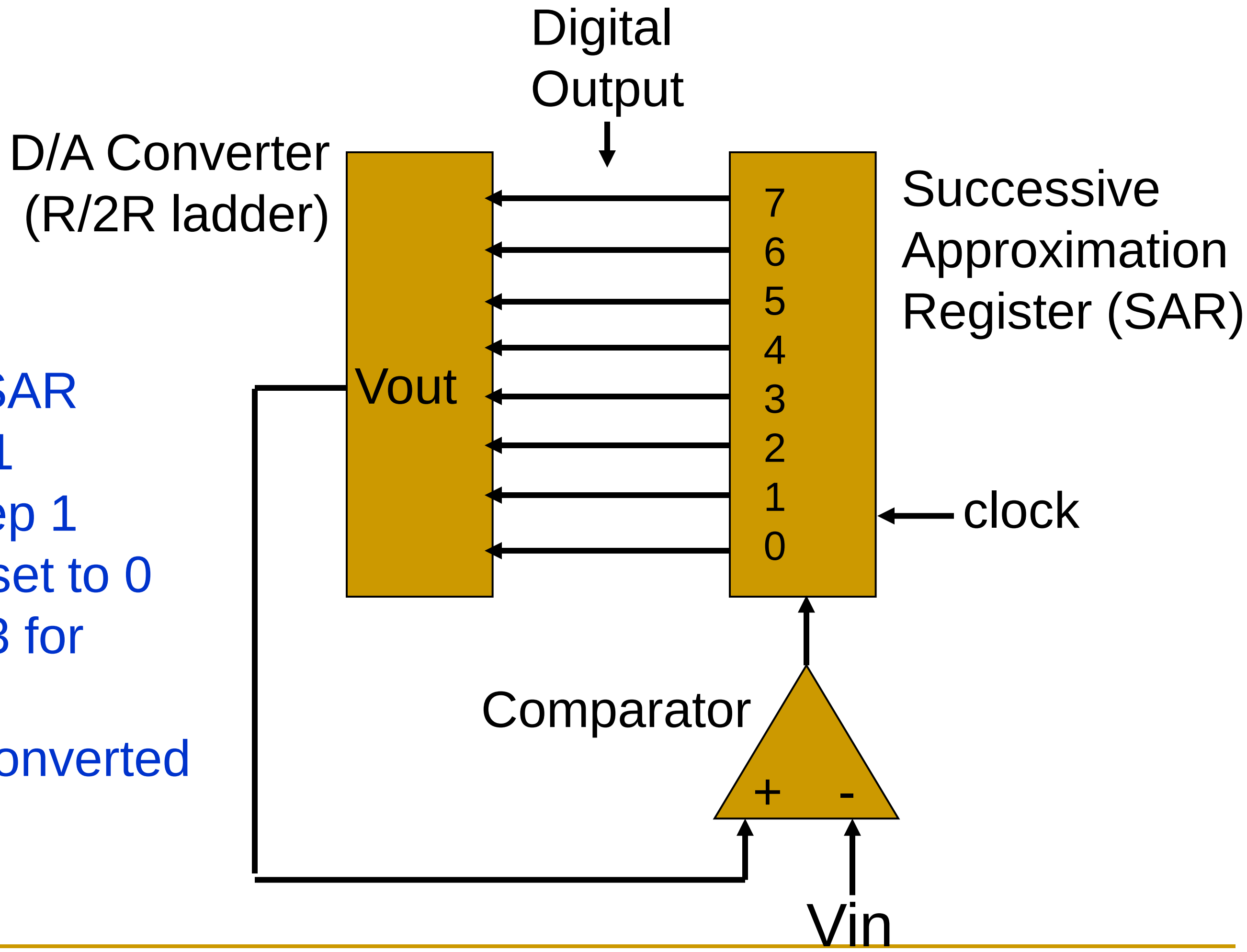
Equivalent resistance = R

number = $b_n*2^n + b_{n-1}*2^{n-1}$ + …. + b_1*2^1 + b_0*2^0

Conversion via “successive approximation”

- “Approximate” value by determining the most significant bit
- Refine the approximation by determining the next most significant bit
- Repeat until least significant bit has been determined
- Conversion time proportional to the number of bits

Successive approximation analog to digital converter



1. Clear all bits of SAR
2. Set SAR bit 7 = 1
3. If Vout < Vin, keep 1
   If Vout > Vin, reset to 0
4. Repeat steps 2-3 for
   bits 6-0.
5. SAR bits 7-0 = converted
   value

HCS12 successive approximation analog-to-digital converter

- 8 analog inputs on Port AD (pins AN7-AN0)
  - analog multiplexer selects one input for A/D
- reference voltage pins: $V_{RH} \leq 6$ volts, $V_{RL} \geq 0$ volts
  - $(V_{RH} – V_{RL}) \geq 2.5$ volts
- 8 or 10-bit value
  - ±1 LSB accuracy
- linear over entire range
- external sample & hold not required
  - S&H implemented as charge redistribution network
- programmable aperture & conversion times
- 8 result registers to store successive conversions

HCS12 analog-to-digital converter

PORTAD pins

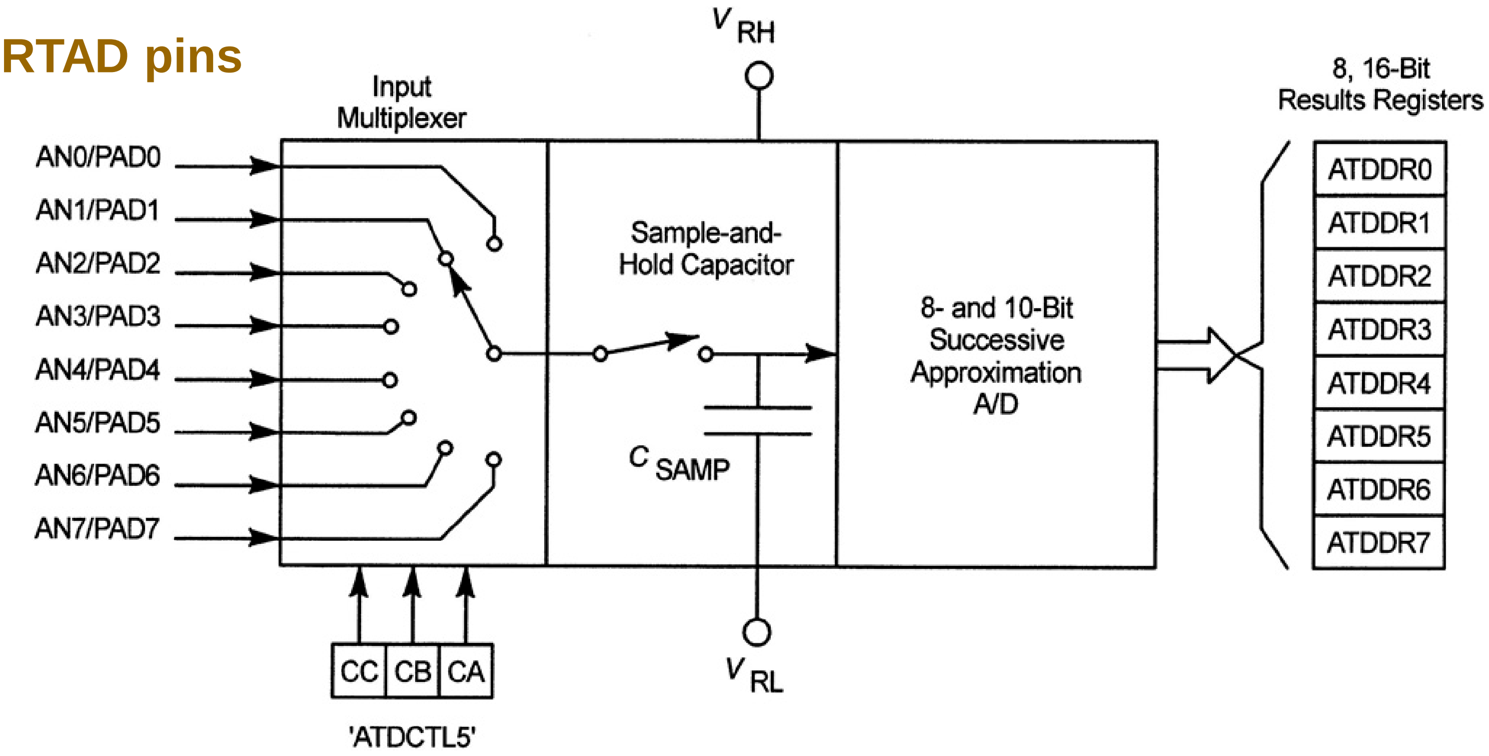


Figure 17-2 HCS12 analog-to-digital converter block diagram.

(Cady, Figure 17-2)

PORT AD pin electronics

- For analog input:
  - Disable digital input buffer in ATDDIEN register (default)
  - Clear bit in data direction register DDRAD (default)

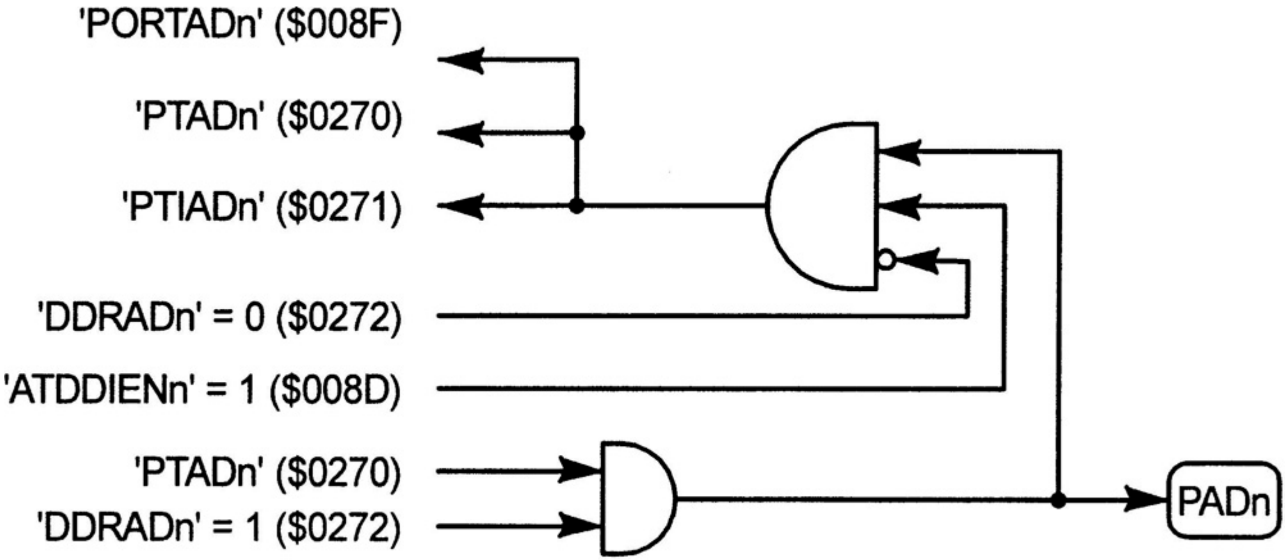


Figure 17-5 PORTAD and PTAD digital input and output.

ATD Result Registers

- Eight 16-bit result registers
  - Right or left-justified, signed or unsigned result
  - Addresses (high & low order bytes):
    - $0090 (ATDDR0H), $0091 (ATDDR0L),
      ….
      $009E (ATDDR7H), $009F (ATDDR7L)
    - Read with16-bit instruction, using address of high byte
- Right-justified result
  - Low 8 bits in ATDDRxL
  - If 10-bit result, bits 9-8 in ATDDRxH[1:0]
- Left-justified result
  - Upper 8 bits in ATDDRxH
  - If 10-bit result, bits 1-0 in ATDDRxL[7:6]

ATD Control Register 2

(ATDCTL2 - $00082)

7	6	5	4	3	2	1	0
ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF

ADPU – ATD power up

0 = power down(default), 1 = power up

*** Must power up & wait at least 20us before using the ATD

AFFC – ATD fast clear

0 = reset conversion flag by reading status reg & then result (default)

1 = reset conversion flag by accessing result reg (fast clear)

AWAI – ATD power down in WAIT mode

0 = continue to run (default), 1 = power down

ETRIGE – external trigger enable

0 = disable/default, 1 = enable

ETRIGLE, **ETRIGP** – external trigger level, polarity

00 = falling edge (default), 01 = rising edge, 10 = high level, 11 = low level

ASCIE, **ASCIF** – ATD sequence complete interrupt enable & flag

If ASCIE = 1. ASCIF same as SCF flag

ATD configuration & initialization

- Power up ATD before using
  - ATD control register ATDCTL2
- Select conversion length & FIFO mode
  - ATD control register ATDCTL3
- Select resolution, clock prescale & sample time
  - ATD control register ATDCTL4
- Select result format, scan mode, multichannel mode
  - ATD control register ATDCTL5
  - A/D converter starts on write to ATDCTL5
- Check conversion status via ATD status registers

ATD conversion sequencing

- Sequences of 1 to 8 conversions
  - Store in result registers, in order of register #
  - Length of sequence in ATDCTL3
- Scan vs. single sequence
  - Single = convert one sequence and stop
  - Scan = continuously repeat the sequence
- One vs. multiple channels
  - Sequence of samples of one channel
  - OR, Sequence of samples from sequential channel #s

ATD Control Register 3

(ATDCTL3 - $00083)

7	6	5	4	3	2	1	0
0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0

S8C,S4C,S2C,S1C: conversion sequence length

- 0000 – 8 conversions/sequence
- 0001 – 1 conversions/sequence
- 0010 – 2 conversions/sequence
- …
- 0111 – 7 conversions/sequence
- 1xxx – 8 conversions/sequence

FIFO: placement of results in registers

- 0 – each sequence to consecutive registers from 0 to sequence length
- 1 – 1st result of sequence to reg. following last reg. of previous sequence

FRZ1,FRZ0:

- 00 – Continue during “freeze mode” (default) – ex. During breakpoint
- 10 - Freeze conversion during “freeze mode”

FIFO vs. non-FIFO mode

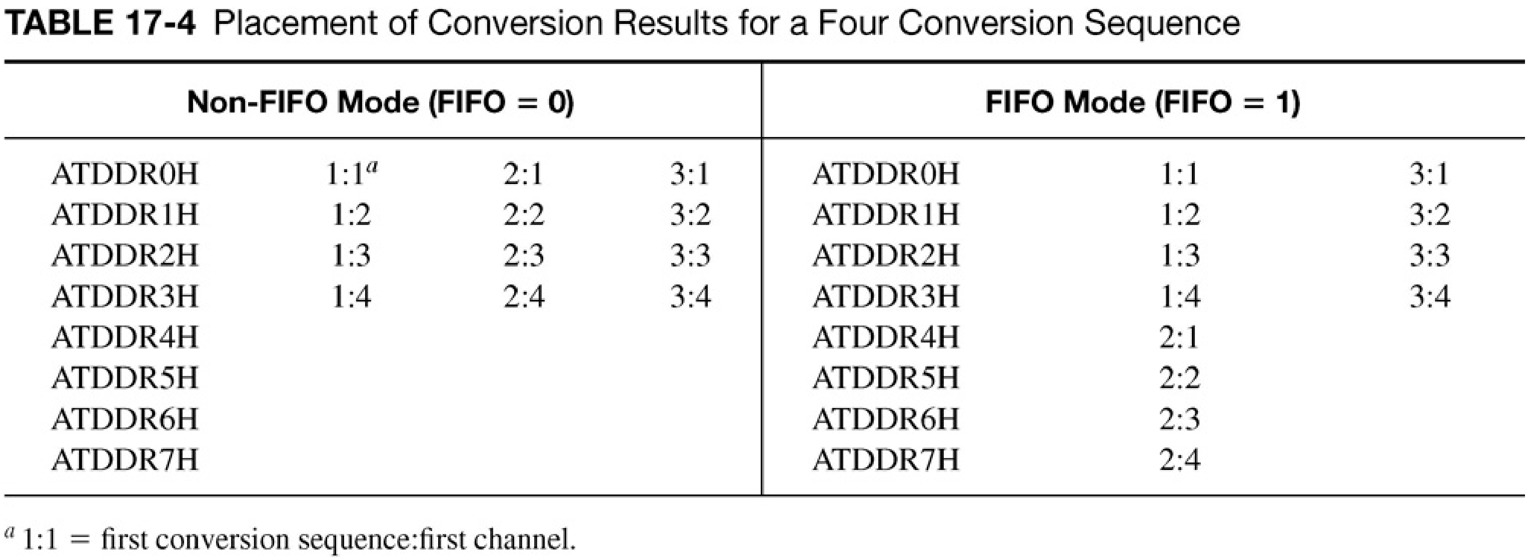


TABLE 17-4 Placement of Conversion Results for a Four Conversion Sequence

Non-FIFO Mode (FIFO = 0)				FIFO Mode (FIFO = 1)		
ATDDR0H	1:1[a]	2:1	3:1	ATDDR0H	1:1	3:1
ATDDR1H	1:2	2:2	3:2	ATDDR1H	1:2	3:2
ATDDR2H	1:3	2:3	3:3	ATDDR2H	1:3	3:3
ATDDR3H	1:4	2:4	3:4	ATDDR3H	1:4	3:4
ATDDR4H				ATDDR4H	2:1	
ATDDR5H				ATDDR5H	2:2	
ATDDR6H				ATDDR6H	2:3	
ATDDR7H				ATDDR7H	2:4	

[a] 1:1 = first conversion sequence:first channel.

(Cady, Table 17-4)

ATD Control Register 5
(ATDCTL5 - $00085)

7	6	5	4	3	2	1	0
DJM	DSGN	SCAN	MULT	0	CC	CB	CA

DJM: result register data justification

0 = left justify (default), 1 = right justify in result register

DSGN: result register data format

0 = unsigned (default), 1 = signed/2's complement

SCAN: continuous conversion sequence mode

0 = single conversion sequence & stop (default)
1 = continuous conversion sequences
(start new sequence when current one finishes)

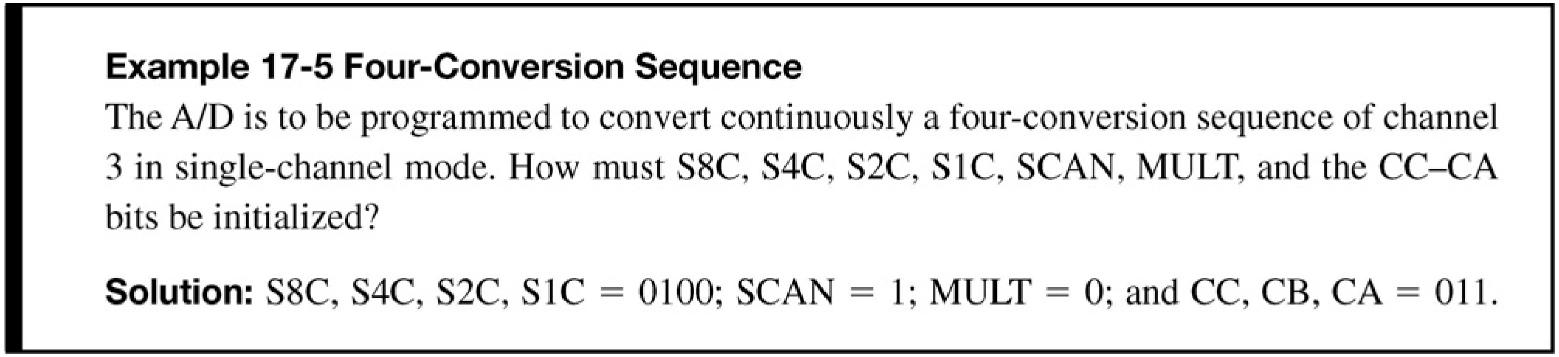
MULT: multichannel sample mode

0 = sample only one channel (default) – designated by CC,CB,CA
1 = sample across several channels, beginning at CC,CB,CA

CC,CB,CA: input multiplexer channel

7-0 => pins AN7-AN0 of PORTAD

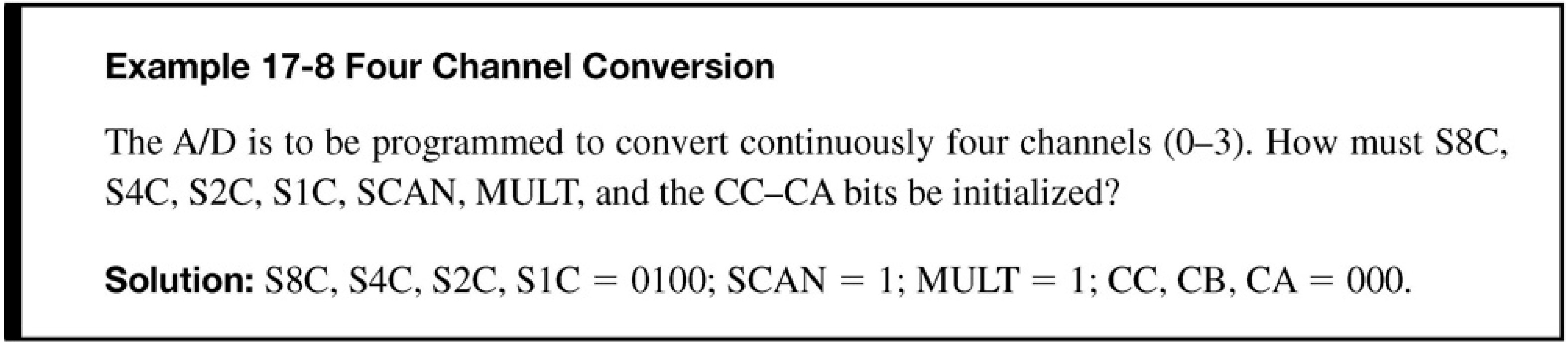
Examples



Example 17-5 Four-Conversion Sequence

The A/D is to be programmed to convert continuously a four-conversion sequence of channel 3 in single-channel mode. How must S8C, S4C, S2C, S1C, SCAN, MULT, and the CC–CA bits be initialized?

Solution: S8C, S4C, S2C, S1C = 0100; SCAN = 1; MULT = 0; and CC, CB, CA = 011.



Example 17-8 Four Channel Conversion

The A/D is to be programmed to convert continuously four channels (0–3). How must S8C, S4C, S2C, S1C, SCAN, MULT, and the CC–CA bits be initialized?

Solution: S8C, S4C, S2C, S1C = 0100; SCAN = 1; MULT = 1; CC, CB, CA = 000.

M68HC12 A/D sample and conversion timing

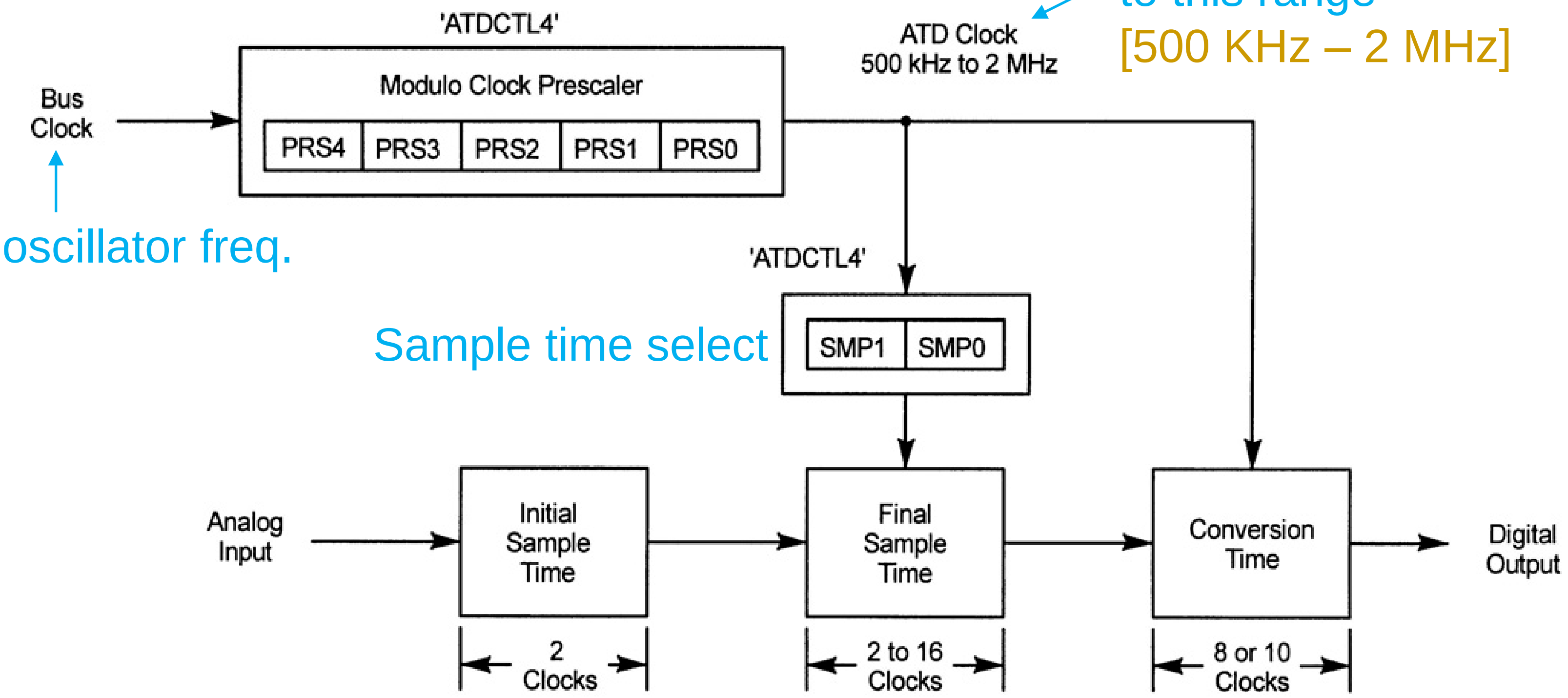


Figure 17-3 A/D sample and conversion timing.

Prescale down to this range [500 KHz – 2 MHz]

½ oscillator freq.

Sample time select

(Cady, Figure 17-3)

ATD Control Register 4
(ATDCTL4 - $00084)

7	6	5	4	3	2	1	0
SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0

SRES8 = A/D resolution

0 = 10-bit (default), 1 => 8-bit

- trade off conversion time for resolution

SMP1:SMP0 = Sample time (aperture) select

00 = 2 clocks (default), 01 = 4 clocks, 10 = 8 clocks, 11 = 16 clocks

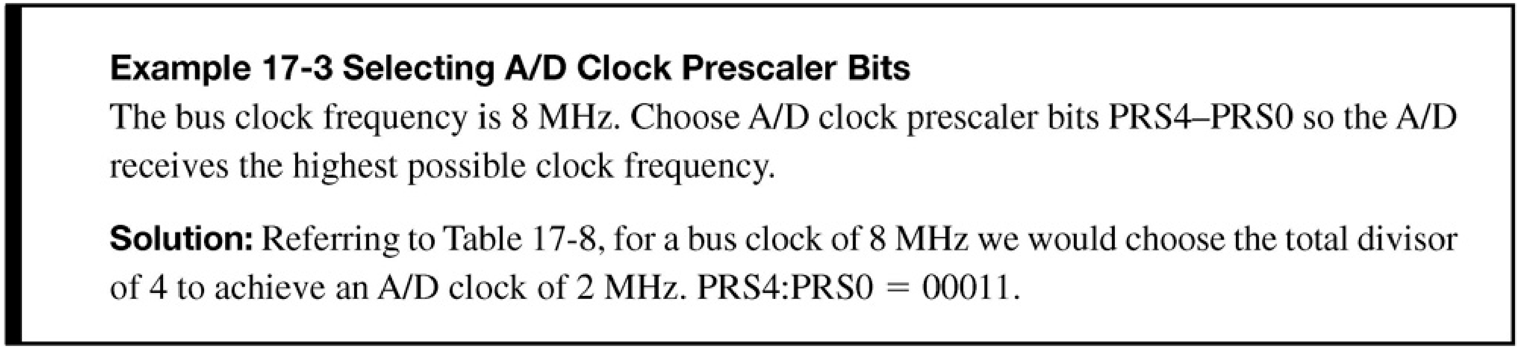
- use longer value for high impedance analog source

PRS4:PRS0 = ATD clock prescaler

Divide bus clock by factor of (n+1)x2 = 2, 4, 6, …,64

- A/D clock must be in the range 500KHz – 2 MHz
- select prescale factor to divide bus clock down to this range

Examples



Example 17-3 Selecting A/D Clock Prescaler Bits

The bus clock frequency is 8 MHz. Choose A/D clock prescaler bits PRS4–PRS0 so the A/D receives the highest possible clock frequency.

Solution: Referring to Table 17-8, for a bus clock of 8 MHz we would choose the total divisor of 4 to achieve an A/D clock of 2 MHz. PRS4:PRS0 = 00011.

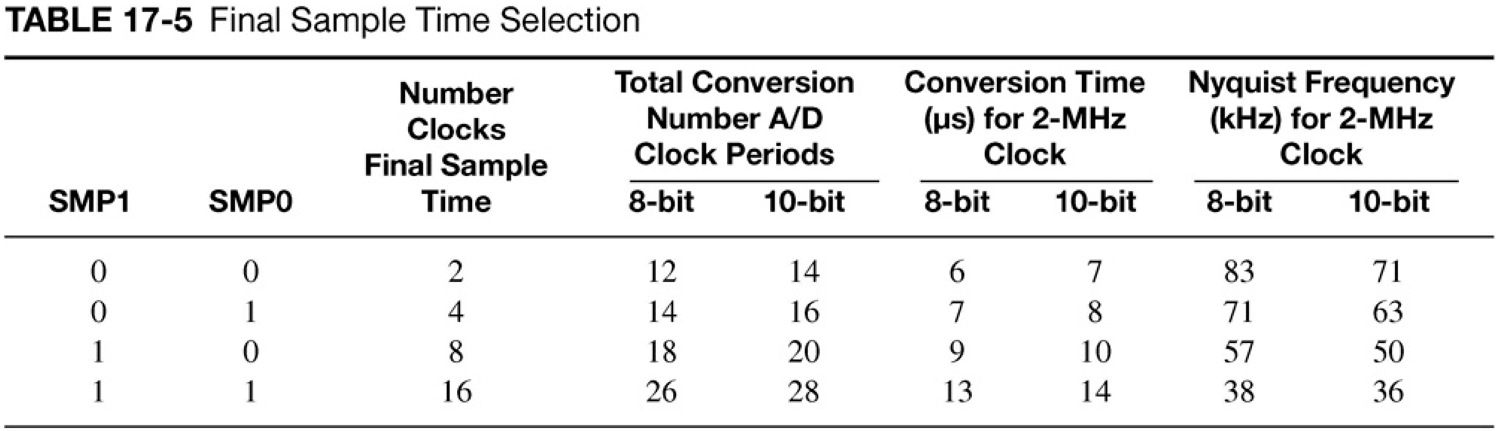


TABLE 17-5 Final Sample Time Selection

SMP1	SMP0	Number Clocks Final Sample Time	Total Conversion Number A/D Clock Periods		Conversion Time (μs) for 2-MHz Clock		Nyquist Frequency (kHz) for 2-MHz Clock	
			8-bit	10-bit	8-bit	10-bit	8-bit	10-bit
0	0	2	12	14	6	7	83	71
0	1	4	14	16	7	8	71	63
1	0	8	18	20	9	10	57	50
1	1	16	26	28	13	14	38	36

ATD status registers

(ATDSTAT0 - $0086)

7	6	5	4	3	2	1	0
SCF	0	ETORF	FIFOR	0	CC2	CC1	CC0

- SCF = sequence complete flag
  - Set when entire sequence complete
  - Interrupt on conversion complete, if enabled in ATDCTL2
  - Clear by writing 1 to SCF,
    or write to ATDCTL5 to start new conversion,
    or read a result register if AFFC=1 in ATDCTL2 (fast clear)
- CC2-CC0 = conversion counter
  - Points to result register to receive current conversion (0-7)
  - "Wraps" around in FIFO mode
- **FIFOR** – FIFO overrun flag (if using FIFO mode)
- **ETPRF** – External trigger overrun flag (if using external trigger)

ATD status register 1

(ATDSTAT1 = $0087)

7	6	5	4	3	2	1	0
CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0

- CCF7-CCF0 = conversion complete flags
  - CCFx sets when current conversion written to result register x (ATDDRx)
- CCFx clears on one of the following
  - Write to ATDCTL5 to start a new conversion
  - If AFFC=0 in ATDCTL2 (not fast clear)::
    - read ATDSTAT1, followed by reading ATDDRx
  - If AFFC=1 in ATDCTL2 (fast clear):
    - read ATDDRx

Example – convert input on AN1

```
        bset    ATDCTL2,ADPU            ;power up ATD
        ldaa    #20                     ;short delay > 20us
Delay:  nop
        dbne    Delay
        ;Normal flag clear, run in WAIT mode, no ext trigger, no interrupts
        movb    ATDCTL2,AFFC|AWAI|ETRIGE|ASCIE
        ;1 conversion/sequence, non-FIFO mode, run in freeze mode
        bclr    ATDCTL3,S8C|S4C|S2C|FIFO|FRZ1|FRZ0
        bset    ATDCTL3,S1C
        ;8-bit resolution, 16 clock sample time, prescale fbus by 4
        bclr    ATDCTL4,PRS4|PRS3|PRS2|PRS1
        bset    ATDCTL4,SRES8|PRS0|SMP1|SMP0
Main:   ;Start – right justify, unsigned, single conversion, one channel (AN1)
        movb    #10000001,ATDCTL5
Spin:   ;Wait for SCF=1 (conversion sequence complete)
        brclr   ATDSTAT0,SCF,Spin
        ;Read converted data
        ldab    ATDDR0L
        … (do whatever with the data)
        bra     Main
```

C example – initialize the ADC

```
void init_ATD () {
char i;
        /* Power up the ATD module */
        // ATDCTL2 = 0x80;
        ATDCTL2_ADPU   = 1;             //Activate power up
        ATDCTL2_AFFC    = 0;            //Normal flag clear (default)
        ATDCTL2_AWAI     = 0;           //Run in WAIT mode (default)
        ATDCTL2_ETRIGE = 0;             //No external trigger (default)
        ATDCTL2_ASCIE    = 0;           //No interrupts (default)
        /* Short delay (> 20us) for power up */
        for (i = 0; i < 20; i++);
        /* Set up ATDCTL3 for conversion sequence */
        //ATDCTL3 = 0x08 = (1<<3) | (0<<2) | 0
        ATDCTL3_S8C = 0;                //one conversion per sequence
        ATDCTL3_S4C = 0;
        ATDCTL3_S2C = 0;
        ATDCTL3_S1C = 1;
        ATDCTL3_FIFO = 0;               //result in consecutive registers (default)
        ATDCTL3_FRZ = 0;                //continue in freeze mode (default)
        /* Set up ATDCTL4 for conversion timing */
        //ATDCTL4 = 0xE1 = (1<<7) | (3<<5) | 1
        ATDCTL4_SRES8 = 1;              //8-bit resolution  (0 = 10-bit)
        ATDCTL4_SMP = 3;                //16-clock (max) sample time for accuracy
        ATDCTL4_PRS = 1;                //prescale fbus(4MHz) by 4 for range [2MHZ..500KHz]
}
```

C example, continued

```
/* Function to initiate conversion of pin AN1 and return converted data */
char sample_ATD () {
    // Right-justify, unsigned, single conversion, one channel, input AN1
    // ATDCTL5 = (1<<7) | (0<<6) | (0<<5) | (0<<4) | 1
    ATDCTL5 = 0x81;                  // write to ATDCTL5 starts conversion
    while (ATDSTAT0_SCF == 0);       // wait for conversion sequence complete
    return (ATDDR0L);                 // return value in (low) result register
}

void main(void) {
  char sample;

   init_ATD();                      // set up the ATD module
   for(;;) {                        // endless loop
      sample = sample_ATD();        // Initiate and return a sample
      /* use the data */
   }
}
```

Lab Procedure

- Design and incorporate a rectifier & filter into your circuit to convert the tachometer output to a DC voltage level
  - Model in PSPICE to verify design.
  - Measure tachometer signal and rectifier/filter output with o'scope.
  - Waveform generators in lab can be used to test the circuit without the motor, if desired.

- Modify software to trigger A/D conversion and read converted value
  - Consider averaging some # of samples
  - Use A/D sequencing to take multiple samples

- Measure: tachometer signal amplitude, rectifier/filter voltage output & ADC value for each of the 10 switch settings

- Plot:
  - ADC value vs tachometer output amplitude
  - ADC value vs. ATD input voltage (rectifier/filter output)
  - ADC value vs. PWM signal duty cycle